Tanzu Application Service for VMs v2.12

VMware Tanzu Application Service 2.12

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

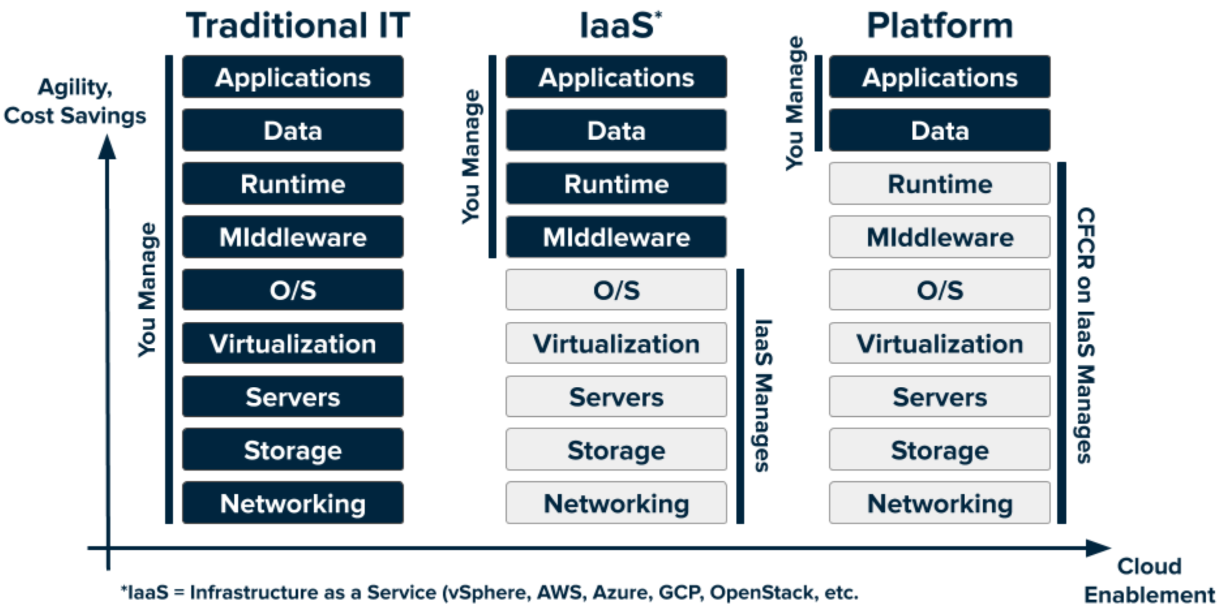
Contents

TAS for VMs Overview

This topic tells you about VMware Tanzu Application Service for VMs (TAS for VMs) and how it works with Ops Manager.

Cloud platforms let anyone deploy network apps or services and make them available to the world in a few minutes. When an app becomes popular, the cloud scales it to handle more traffic, replacing build-out and migration efforts that once took months with a few keystrokes. Cloud platforms enable you to focus exclusively on your apps and data without worrying about underlying infrastructure.

The following diagram shows the layers of a typical technology stack, and compares the traditional IT model to the cloud platform model:



See the following sections in this topic:

- About TAS for VMs
- How TAS for VMs Works
- How TAS for VMs Differs from Open Source Cloud Foundry

About TAS for VMs

This section describes why TAS for VMs is an industry-standard cloud platform.

Not all cloud platforms are created equal. Some have limited language and framework support, lack key app services, or restrict deployment to a single cloud.

As an industry-standard cloud platform, TAS for VMs offers the following:

- **Open source code**: The platform's openness and extensibility prevent its users from being locked into a single framework, set of app services, or cloud. For more information, see the Cloud Foundry project on GitHub.
- **Deployment automation**: Developers can deploy their apps to TAS for VMs using their existing tools and with zero modification to their code.
- **Flexible infrastructure**: You can deploy TAS for VMs to run your apps on your own computing infrastructure, or deploy on an IaaS like vSphere, AWS, Azure, GCP, or OpenStack.
- **Commercial options**: You can also use a PaaS deployed by a commercial TAS for VMs cloud provider. For more information, see Cloud Foundry Certified Platforms.
- **Community support**: A broad community contributes to and supports TAS for VMs. See Welcome to the Cloud Foundry Community.

TAS for VMs is ideal for anyone interested in removing the cost and complexity of configuring infrastructure for their apps.

How TAS for VMs Works

To flexibly serve and scale apps online, TAS for VMs has subsystems that perform specialized functions. The sections below describe how some of these main subsystems work.

Load balancing

This section describes how TAS for VMs handles load balancing.

Clouds balance their processing loads over multiple machines, optimizing for efficiency and resilience against point failure. A TAS for VMs installation accomplishes this using the following components:

- **BOSH** creates and deploys VMs on top of a physical computing infrastructure, and deploys and runs TAS for VMs on top of this cloud. To configure the deployment, BOSH follows a manifest document. For more information, see the BOSH documentation.
- **Cloud Controller** runs the apps and other processes on the cloud's VMs, balancing demand and managing app lifecycles. For more information, see Cloud Controller.
- The **Gorouter** routes incoming traffic from the world to the VMs that are running the apps that the traffic demands, usually working with a customer-provided load balancer. For more information, see TAS for VMs Routing Architecture.

Running apps

This section describes the VMs that run your apps in TAS for VMs and how the platform packages your apps to run on these VMs.

VMs in TAS for VMs

TAS for VMs designates the following types of VMs:

- **Component VMs** make up the platform's infrastructure.
- **Host VMs** host your apps for the outside world.

Within TAS for VMs, the Diego system distributes the hosted app load over all of the host VMs, and keeps it running and balanced through demand surges, outages, or other changes. Diego accomplishes this through an auction algorithm.

For more information, see Diego Components and Architecture.

Distributing apps

To meet demand, multiple host VMs run duplicate instances of the same app. This means that apps must be portable. TAS for VMs distributes app source code to VMs with everything the VMs need to compile and run the apps locally.

TAS for VMs includes the following with your app's source code:

- **Stack**: the operating system the app runs on.
- **Buildpack**: contains all languages, libraries, and services that the app uses.

Before sending an app to a VM, the Cloud Controller stages it for delivery by combining the stack, buildpack, and source code into a droplet that the VM can unpack, compile, and run. For simple, standalone apps with no dynamic pointers, the droplet can contain a pre-compiled executable instead of source code, language, and libraries.

For more information, see:

- Changing Stacks
- Buildpacks
- How Apps Are Staged

Organizing users and workspaces

This section describes how TAS for VMs organizes users and workspaces.

TAS for VMs manages user accounts through two User Account and Authentication (UAA) servers, which support access control as OAuth2 services and can store user information internally, or connect to external user stores through LDAP or SAML.

For more information, see User Account and Authentication (UAA) Server.

The following table describes what the two UAA servers do:

Server	Purpose
First UAA server	• Grants access to BOSH • Holds accounts for Ops Manager operators who deploy runtimes, services, and other software onto the BOSH layer directly

Second UAA server	- Controls access to the Cloud Controller - Defines user roles, such as admin, developer, or auditor, and grants them different sets of privileges to run TAS for VMs commands - Scopes the roles to separate, compartmentalized orgs and spaces within an installation to manage and track use

For more information, see Orgs, Spaces, Roles, and Permissions.

Storing resources

The following table describes where TAS for VMs stores resources:

Resource	Storage Location
- Source code - Buildpacks - Documentation - Custom configurations - Other platform resources	GitHub
- Large binary files - Droplets	Internal or external blobstore
- Internal component states - Other temporary information	MySQL

Communicating with components

This section describes how TAS for VMs components communicate with one another.

TAS for VMs components communicate in the following ways:

- By sending messages internally using HTTP and HTTPS protocols
- By sending NATS messages to each other directly

BOSH Director co-locates a BOSH DNS server on every deployed VM. All VMs keep up-to-date DNS records for all the other VMs in the same foundation. This enables service discovery between VMs.

BOSH DNS allows deployments to continue communicating with VMs even when the VMs' IP addresses change. It also provides client-side load balancing by randomly selecting a healthy VM when multiple VMs are available.

For more information about BOSH DNS, see Native DNS Support in the BOSH documentation.

Monitoring and Analyzing

This section describes logging in TAS for VMs.

TAS for VMs generates system logs from TAS for VMs components and app logs from hosted apps. As TAS for VMs runs, its component and host VMs generate logs and metrics. TAS for VMs apps also typically generate logs.

The following table describes where TAS for VMs sends logs:

Log Type	Destination
TAS for VMs component logs	Rsyslog agents
TAS for VMs component metrics	Loggregator
App logs	Loggregator

Component logs stream from rsyslog agents, and the cloud operator can configure them to stream out to a syslog drain.

The Loggregator system aggregates the component metrics and app logs into a structured, usable form, the Firehose. You can use all of the output of the Firehose, or direct the output to specific uses, such as monitoring system internals, triggering alerts, or analyzing user behavior, by applying nozzles.

For more information, see Loggregator Architecture.

Using services

This section describes how you can use services with your apps on TAS for VMs.

Typical apps depend on free or metered services such as databases or third-party APIs. To incorporate these into an app:

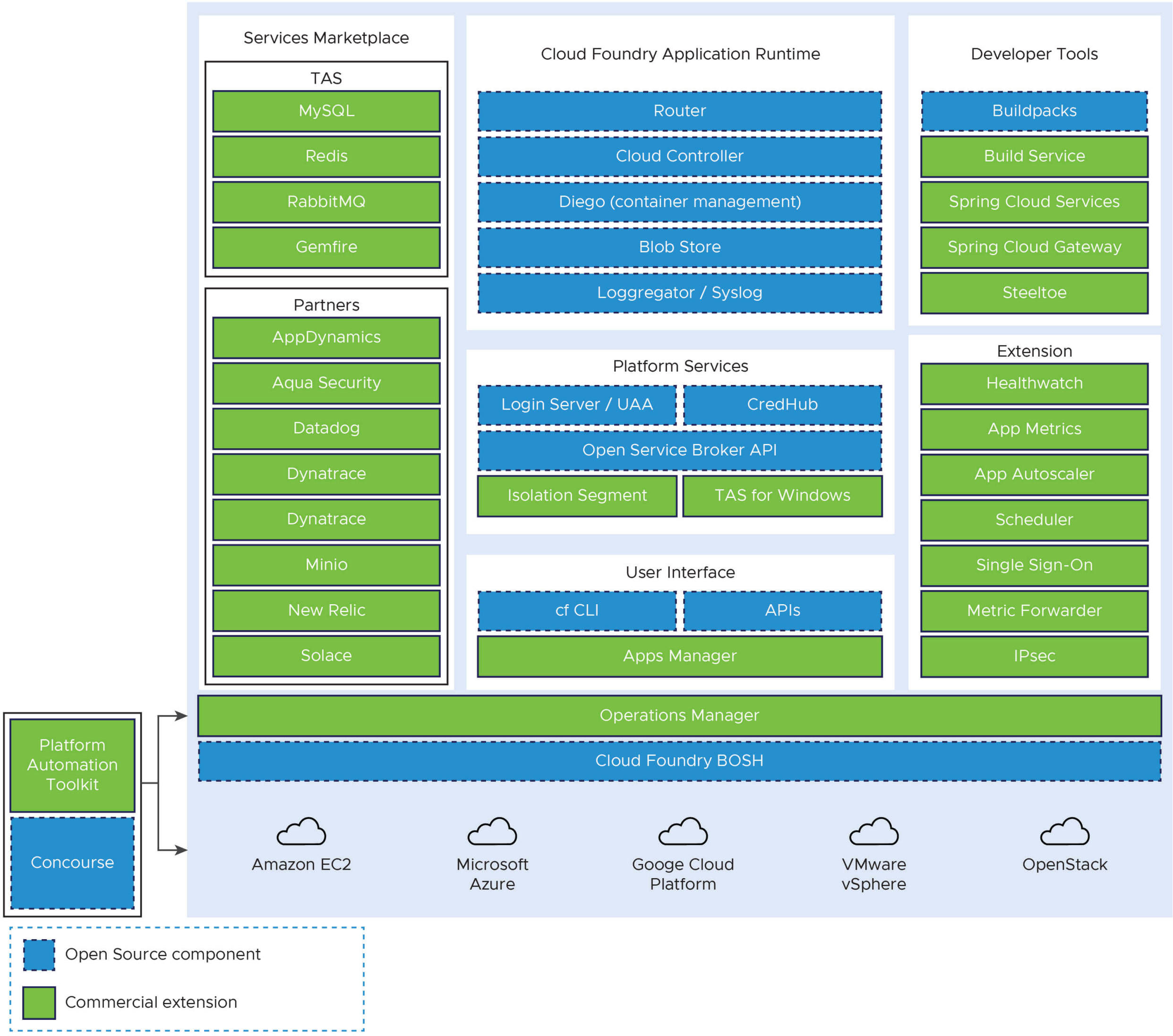
1. **Write a Service Broker**: This component manages the life cycle of the service. The Service Broker uses the Service Broker API to advertise a catalog of service offerings to TAS for VMs apps.
2. **Provision the Service Instance**: Create an instance of the service offering by sending a provision request to the Service Broker API.
3. **Enable apps to access the Service Instance**: Configure the TAS for VMs app to make calls to the Service Instance using the Service Broker API.

For more information, see Services.

How TAS for VMs Differs from Open Source Cloud Foundry

Open-source software provides the basis for TAS for VMs, which is the VMware distribution of Cloud Foundry Application Runtime software for hosting apps. VMware offers additional commercial features, enterprise services, support, documentation, certificates, and other value-adds.

The following diagram shows open source Cloud Foundry components in blue and commercial features available with TAS for VMs in green:



View a larger version of this diagram.

Release Notes

In this section:

- VMware Tanzu Application Service for VMs Release Notes
- VMware Tanzu Application Service for VMs [Windows] Release Notes
- Isolation Segment Release Notes

VMware Tanzu Application Service v2.12 Release Notes

⚠ **Caution: This release has been removed from General Support.**

Here you will find the release notes for VMware Tanzu Application Service for VMs (TAS for VMs) v2.12.

TAS for VMs is certified by the Cloud Foundry Foundation for 2023.

For more information about the Cloud Foundry Certified Provider Program, see How Do I Become a Certified Provider? on the Cloud Foundry website.

Because VMware uses the Percona Distribution for MySQL, expect a time lag between Oracle releasing a MySQL patch and VMware releasing TAS for VMs containing that patch.

Required Cloud Foundry Command-Line Interface (cf CLI) version: You must install cf CLI v7 or cf CLI v8 when upgrading to or using TAS for VMs v2.12. For more information, see Upgrading to cf CLI v7 and Upgrading to cf CLI v8.

Releases

2.12.25 - Withdrawn

⚠ **Caution: This release has been removed from VMware Tanzu Network due to a known issue with routing. For more information, see TAS and Isolation Segment with routing-release 0.259.0 fail to prune stale routes on gorouter.**

Release Date: 03/21/2023

Note: This version of TAS for VMs contains a known issue with Gorouter error handling for backend app requests. Failures that previously returned HTTP Status Codes 496, 499, 503, 525, or 526 may instead return 502. Additionally, stale routes may fail to be pruned properly, which could result in apps unexpectedly returning HTTP Status Codes 502.

- **[Security Fix]** Bump autoscaler to v249.1.3
- Bump binary-offline-buildpack to version `1.1.3`
- Bump cf-autoscaling to version `249.1.3`
- Bump cf-cli to version `1.44.0`
- Bump cf-networking to version `3.23.0`
- Bump cflinuxfs3 to version `0.356.0`
- Bump credhub to version `2.12.21`
- Bump diego to version `2.72.0`
- Bump dotnet-core-offline-buildpack to version `2.4.8`
- Bump garden-runc to version `1.25.0`
- Bump go-offline-buildpack to version `1.10.6`
- Bump mapfs to version `1.2.13`
- Bump nfs-volume to version `7.1.9`
- Bump nginx-offline-buildpack to version `1.2.1`
- Bump nodejs-offline-buildpack to version `1.8.6`
- Bump php-offline-buildpack to version `4.6.0`
- Bump python-offline-buildpack to version `1.8.7`
- Bump routing to version `0.259.0`
- Bump silk to version `3.23.0`
- Bump smb-volume to version `3.1.10`
- Bump staticfile-offline-buildpack to version `1.6.0`
- Bump uaa to version `74.5.64`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.448	
backup-and-restore-sdk	1.18.61	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>binary-offline-buildpack</td><td>1.1.3</td><td>▼ 1.1.3
* Updating github-config
* Uncached buildpack SHA256: 5523f4077d792b386671421f31899f409ea8ce5b8ae781cb4f84358cd138b5fd
* Uncached buildpack SHA256: 5dd66be045a50d1e4f1dc12e317e85add9fbe74734b3f9882a8a02578b66809f
* Uncached buildpack SHA256: 50836f839547c13d6ef435d87ce8e2b4863fdfe32343a26514d321cc8455d177
* Uncached buildpack SHA256: 1f72560521b626e5012934f39f62d31192f6bf8c3b6ec5a91f5216ff1e36b5ae</td></tr>
<tr><td>bosh-dns-aliases</td><td>0.0.4</td><td></td></tr>
<tr><td>bosh-system-metrics-forwarder</td><td>0.0.30</td><td></td></tr>
<tr><td>bpm</td><td>1.1.21</td><td></td></tr>
<tr><td>capi</td><td>1.117.12</td><td></td></tr>
<tr><td>cf-autoscaling</td><td>249.1.3</td><td>▼ v249.1.3
What's Changed
* Exclude SnakeYAML to avoid CVE-2022-1471 being flagged
* Update dependencies</td></tr>
<tr><td>cf-cli</td><td>1.44.0</td><td>▼ v1.44.0 ### This release contains the following versions of the CF CLI | Major version	Prior version	Current version
v8	**8.5.0**	[8.6.0](https://github.com/cloudfoundry/cli/releases/tag/v8.6.0)
v7	**7.5.0**	[7.6.0](https://github.com/cloudfoundry/cli/releases/tag/v7.6.0)
v6	**6.53.0**	**[6.53.0](https://github.com/cloudfoundry/cli/releases/tag/v6.53.0)** |</td></tr>
<tr><td>cf-networking</td><td>3.23.0</td><td></td></tr>
</table>

Component	Version	Release Notes
cflinuxfs3	0.356.0	

credhub	2.12.21	▼ 2.12.21
		`### Security Fixes`
		`- Bump various dependencies`
		`**Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.20...2.12.21`
		▼ 2.12.20
		`### Security Fixes`
		`- Bump various dependencies`

diego	2.72.0	

Component	Version	Release Notes
dotnet-core-offline-buildpack	2.4.8	▼ 2.4.8 * Add dotnet-runtime 6.0.14, remove dotnet-runtime 6.0.13 (#748) * Add dotnet-aspnetcore 6.0.14, remove dotnet-aspnetcore 6.0.13 (#747) * Add dotnet-runtime 7.0.3, remove dotnet-runtime 7.0.2 (#746) * Add dotnet-aspnetcore 7.0.3, remove dotnet-aspnetcore 7.0.2 (#745) * Add dotnet-sdk 7.0.200, remove dotnet-sdk 7.0.102 (#744) * Add dotnet-sdk 6.0.406, remove dotnet-sdk 6.0.405 (#742) * Add node 18.14.2, remove node 18.14.0 (#743) for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/184532347) * update libbuildpack-dynatrace (#691) * Removes compatibility table that only exists for brats tests and replaces it with simpler logic Packaged binaries: | name | version | cf_stacks | |-|-|-| | bower | 1.8.14 | cflinuxfs3, cflinuxfs4 | | dotnet-aspnetcore | 6.0.14 | cflinuxfs3, cflinuxfs4 | | dotnet-aspnetcore | 7.0.3 | cflinuxfs3, cflinuxfs4 | | dotnet-runtime | 6.0.14 | cflinuxfs3, cflinuxfs4 | | dotnet-runtime | 7.0.3 | cflinuxfs3, cflinuxfs4 | | dotnet-sdk | 6.0.406 | cflinuxfs3, cflinuxfs4 | | dotnet-sdk | 7.0.200 | cflinuxfs3, cflinuxfs4 | | libgdiplus | 6.1 | cflinuxfs3 | | libgdiplus | 6.1 | cflinuxfs4 | | libunwind | 1.6.2 | cflinuxfs3 | | libunwind | 1.6.2 | cflinuxfs4 | | node | 18.14.2 | cflinuxfs3 | | node | 18.14.2 | cflinuxfs4 | Default binary versions: | name | version | |-|-| | dotnet-runtime | 6.0.x | | dotnet-aspnetcore | 6.0.x | | dotnet-sdk | 6.0.x | | bower | 1.8.x | * Uncached buildpack SHA256: b08e323198ab104d7c906820e97c1365a37a7c99e7546e0bfb4c19517860f5ed * Uncached buildpack SHA256: be2a1d9e6a9dcd9759cc3c9bafde45ec79f1533fb92f328d259d7b2e044

Component	Version	Release Notes
		`bdc9d`
garden-runc	1.25.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>go-offline-buildpack</td><td>1.10.6</td><td>▼ 1.10.6 * Add go 1.20.1 for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/184573596) Packaged binaries: | name	version	cf_stacks
dep	0.5.4	cflinuxfs3
dep	0.5.4	cflinuxfs4
glide	0.13.3	cflinuxfs3
glide	0.13.3	cflinuxfs4
go	1.18.10	cflinuxfs3
go	1.18.10	cflinuxfs4
go	1.19.6	cflinuxfs3
go	1.19.6	cflinuxfs4
go	1.20.1	cflinuxfs3
go	1.20.1	cflinuxfs4
godep	80	cflinuxfs3
godep	80	cflinuxfs4 | Default binary versions: | name	version
-	-
go	1.18.x | * Uncached buildpack SHA256: e81fcaa4eb7142072cf5dbb5daa70c19a9ae17c51d54f1778f056ee7292872f2 * Uncached buildpack SHA256: 2475086b818de7270cbad0e2b24cfebe74bf41688159abaa746ef0f0ed32937d ▼ 1.10.5 * Add go 1.19.6, remove go 1.19.5 for stack(s) cflinuxfs4, cflinuxfs3 (https://www.pivotaltracker.com/story/show/184477448) * Bump github.com/Dynatrace/libbuildpack-dynatrace from 1.5.1 to 1.5.2 Packaged binaries: | name	version	cf_stacks
dep	0.5.4	cflinuxfs3
dep	0.5.4	cflinuxfs4
glide	0.13.3	cflinuxfs3
glide	0.13.3	cflinuxfs4
go	1.18.10	cflinuxfs3
go	1.18.10	cflinuxfs4
go	1.19.6	cflinuxfs3
go	1.19.6	cflinuxfs4
godep	80	cflinuxfs3
godep	80	cflinuxfs4 | Default binary versions: | name	version
-	-	</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>| go | 1.18.x |
* Uncached buildpack SHA256: 7b155c13e80e696732bb8c40da8862e5280b984a7457d0af70970180168c927f
* Uncached buildpack SHA256: 297b970a47e201e9151e216013a2b66b9365fda502ff3b079615f1070bf1c951</td></tr>
<tr><td>haproxy</td><td>11.10.2</td><td></td></tr>
<tr><td>java-offline-buildpack</td><td>4.54</td><td></td></tr>
<tr><td>log-cache</td><td>2.12.6</td><td></td></tr>
<tr><td>loggregator</td><td>106.7.5</td><td></td></tr>
<tr><td>loggregator-agent</td><td>6.5.8</td><td></td></tr>
<tr><td>mapfs</td><td>1.2.13</td><td>▼ v1.2.13
Changes
* Golang: Updated to v1.19.4 (#32)
* Golang: Updated to v1.19.5 (#37)
* Golang: Updated to v1.19.5 (#44)
* Golang: Updated to v1.20.1 (#48)
Dependencies
* **mapfs:** Updated to v`98da9f0`.
For more information, see [mapfs](https://github.com/cloudfoundry/mapfs).</td></tr>
<tr><td>metric-registrar</td><td>2.1.2</td><td></td></tr>
<tr><td>metrics-discovery</td><td>3.2.7</td><td></td></tr>
<tr><td>mysql-monitoring</td><td>9.23.0</td><td></td></tr>
<tr><td>nats</td><td>56.1.0</td><td></td></tr>
<tr><td>nfs-volume</td><td>7.1.9</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nginx-offline-buildpack</td><td>1.2.1</td><td>▼ 1.2.1
  * Bump github.com/Dynatrace/libbuildpack-dynatrace from 1.5.1 to 1.5.2 (#186)
  * Updating github-config
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | nginx | 1.22.1 | cflinuxfs3 |
  | nginx | 1.22.1 | cflinuxfs4 |
  | nginx | 1.23.3 | cflinuxfs3 |
  | nginx | 1.23.3 | cflinuxfs4 |
  | openresty | 1.13.6.2 | cflinuxfs3 |
  | openresty | 1.15.8.3 | cflinuxfs3 |
  | openresty | 1.17.8.2 | cflinuxfs3 |
  | openresty | 1.19.9.1 | cflinuxfs3 |
  | openresty | 1.19.9.1 | cflinuxfs4 |
  | openresty | 1.21.4.1 | cflinuxfs3 |
  | openresty | 1.21.4.1 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | nginx | 1.23.x |
  * Uncached buildpack SHA256: d181c97dfa97c652a33509a4fdf634efc5be33c67683d05fd5dbdc5c47aeabd5
  * Uncached buildpack SHA256: 38aff00f0ff17ae1884fce4efc03913ccd5407af507a253b573ad045275d3220</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nodejs-offline-buildpack</td><td>1.8.6</td><td>▼ 1.8.6
  * Add node 18.14.1, remove node 18.12.1 for stack(s) cflinuxfs4, cflinuxfs3
  * Add node 16.19.1, remove node 16.18.1 for stack(s) cflinuxfs3, cflinuxfs4
  * Add node 14.21.3, remove node 14.21.1 for stack(s) cflinuxfs3, cflinuxfs4
  * Update Node 16.x deprecation date (Nodejs update on SSL deprecation) [ref](https://nodejs.org/en/blog/vulnerability/february-2023-security-releases/)
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | node | 14.21.2 | cflinuxfs3 |
  | node | 14.21.2 | cflinuxfs4 |
  | node | 14.21.3 | cflinuxfs3 |
  | node | 14.21.3 | cflinuxfs4 |
  | node | 16.19.0 | cflinuxfs3 |
  | node | 16.19.0 | cflinuxfs4 |
  | node | 16.19.1 | cflinuxfs3 |
  | node | 16.19.1 | cflinuxfs4 |
  | node | 18.13.0 | cflinuxfs3 |
  | node | 18.13.0 | cflinuxfs4 |
  | node | 18.14.1 | cflinuxfs3 |
  | node | 18.14.1 | cflinuxfs4 |
  | yarn | 1.22.19 | cflinuxfs3, cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | node | 18.x |
  * Uncached buildpack SHA256: e8f42ddddcf355c9b37597944b498cf33f624d59852ba686a97a774e881a424b
  * Uncached buildpack SHA256: c6bee0e5213870daaa2e3c2db51e2a3bf56b14bf2014c72b8cb524432d31a642</td></tr>
<tr><td>notifications</td><td>62</td><td></td></tr>
<tr><td>notifications-ui</td><td>40</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>php-offline-buildpack</td><td>4.6.0</td><td>▼ 4.6.0
<pre>
  * Add composer 2.5.4, remove composer 2.5.2 for stack(s) cflinuxfs4, cflinuxfs3
  * Bump newrelic to 10.6.0.318
  * Add appdynamics 23.2.0-684, remove appdynamics 22.12.1-677 for stack(s) cflinuxfs4, cflinuxfs3
  * Add cflinuxfs4 stack support
  Packaged binaries:
name	version	cf_stacks	modules
appdynamics	23.2.0-684	cflinuxfs3, cflinuxfs4	
composer	2.5.4	cflinuxfs3, cflinuxfs4	
httpd	2.4.55	cflinuxfs3	
httpd	2.4.55	cflinuxfs4	
newrelic	10.6.0.318	cflinuxfs3, cflinuxfs4	
nginx	1.22.1	cflinuxfs3	
nginx	1.22.1	cflinuxfs4	
nginx	1.23.3	cflinuxfs3	
nginx	1.23.3	cflinuxfs4	
php	8.0.27	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.0.28	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.15	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, sn
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>mp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
php	8.1.15	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.16	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.16	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.2.2	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaml, zip, zlib
php	8.2.2	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, g</td></tr>
</table>

Component	Version	Release Notes
		d, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaml, zip, zlib | | php | 8.2.3 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaml, zip, zlib | | php | 8.2.3 | cflinuxfs4 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaml, zip, zlib | | python | 2.7.18 | cflinuxfs4 | | Default binary versions: | name | version | |-|-| | php | 8.1.16 | | httpd | 2.4.55 | | newrelic | 10.6.0.318 | | nginx | 1.23.3 | | composer | 2.5.4 | * Uncached buildpack SHA256: 2fdb877dd287ec73c3ae9f43a7fbcf9c725f6d3f6ded535358d8020224885d1a * Uncached buildpack SHA256: 9e7052e5e046d1e4e0cefa70f1862d1918d5a570509203ee2ff3d0f8c8359fb0 ▼ 4.5.3 * Add php 8.1.16, remove php 8.1.14 for stack(s) cflinuxfs4, cflinuxfs3 * Add php 8.2.3, remove php 8.2.1 for stack(s) cflinuxfs3, cflinuxfs4 * Add php 8.0.28, remove php 8.0.26

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
  for stack(s) cflinuxfs3
  Packaged binaries:
name	version	cf_stacks	modules
appdynamics	22.12.1-677	cflinuxfs3, cflinuxfs4	
composer	2.5.2	cflinuxfs3, cflinuxfs4	
httpd	2.4.55	cflinuxfs3	
httpd	2.4.55	cflinuxfs4	
newrelic	9.20.0.310	cflinuxfs3	
nginx	1.22.1	cflinuxfs3	
nginx	1.22.1	cflinuxfs4	
nginx	1.23.3	cflinuxfs3	
nginx	1.23.3	cflinuxfs4	
php	8.0.27	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.0.28	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.15	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.15	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oau
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>th, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
php	8.1.16	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.16	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.2.2	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaml, zip, zlib
php	8.2.2	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebu</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
g, xsl, yaml, zip, zlib |
  | php | 8.2.3 | cflinuxfs3 | amqp, apcu, bz
2, curl, dba, enchant, exif, fileinfo, ftp, g
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  | php | 8.2.3 | cflinuxfs4 | amqp, apcu, bz
2, curl, dba, enchant, exif, fileinfo, ftp, g
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  Default binary versions:
name	version
php	8.1.16
httpd	2.4.55
newrelic	9.20.0.310
nginx	1.23.3
composer	2.5.2
  * Uncached buildpack SHA256: 8a871fa365577d
f16af144fd407db7741f441d51ed82af19f74c841c1c9
d243c
</pre></td></tr>
<tr><td>push-apps-manager-release</td><td>675.0.5</td><td></td></tr>
<tr><td>push-usage-service-release</td><td>674.0.30</td><td></td></tr>
<tr><td>pxc</td><td>0.50.0</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>python-offline-buildpack</td><td>1.8.7</td><td>▼ 1.8.7 * Bump github.com/Dynatrace/libbuildpack-dynatrace to v1.5.2 (#701) Packaged binaries: | name	version	cf_stacks
libffi	3.2.1	cflinuxfs3, cflinuxfs4
libmemcache	1.0.18	cflinuxfs3, cflinuxfs4
miniconda3-py39	4.12.0	cflinuxfs3, cflinuxfs4
pip	23.0	cflinuxfs3, cflinuxfs4
pipenv	2023.2.4	cflinuxfs3
pipenv	2023.2.4	cflinuxfs4
python	3.7.16	cflinuxfs3
python	3.7.16	cflinuxfs4
python	3.8.16	cflinuxfs3
python	3.8.16	cflinuxfs4
python	3.9.16	cflinuxfs3
python	3.9.16	cflinuxfs4
python	3.10.9	cflinuxfs3
python	3.10.9	cflinuxfs4
python	3.11.1	cflinuxfs3
python	3.11.1	cflinuxfs4
setuptools	67.1.0	cflinuxfs3, cflinuxfs4 | Default binary versions: | name	version
-	-
python	3.10.x	 * Uncached buildpack SHA256: 24eb453c383476da9683905d2439fdd29c774689cb51580714f89d9ac558875b * Uncached buildpack SHA256: daaa857bc79eb56c3b39101b520a59d060f492318b1b99e58f98d0458d368c4f</td></tr>
<tr><td>r-offline-buildpack</td><td>1.2.0</td><td></td></tr>
</table>

Component	Version	Release Notes

routing	0.259.0	▼ v0.259.0
		## Changes
		- No changes from last version.
		- Fixing CI so that artifacts are generated correctly for github release.
		## Built with go 1.20.1
		Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.258.0...v0.259.0
		## Resources
		- [Download release v0.259.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.259.0).
		▼ v0.258.0
		## Changes
		- Update healthchecker to [0.4.0](https://github.com/cloudfoundry/healthchecker-release/releases/tag/v0.4.0)
		- Increase startup delay default to 30 seconds [PR](https://github.com/cloudfoundry/healthchecker-release/pull/2)
		- Upgrade golang to 1.20.1
		## Bosh Job Spec changes:
		```diff
		diff --git a/jobs/acceptance_tests/spec b/jobs/acceptance_tests/spec
		index 65bf4c30..6a73b9ae 100644
		--- a/jobs/acceptance_tests/spec
		+++ b/jobs/acceptance_tests/spec
		@@ -7,7 +7,7 @@ templates:
		bpm.yml.erb: config/bpm.yml
		packages:
		- - golang-1.19-linux
		+ - golang-1.20-linux
		- acceptance_tests
		- rtr
		- cf-cli-6-linux
		diff --git a/jobs/smoke_tests/spec b/jobs/smoke_tests/spec
		index b16357ed..0426dc99 100644
		--- a/jobs/smoke_tests/spec
		+++ b/jobs/smoke_tests/spec
		@@ -7,7 +7,7 @@ templates:
		bpm.yml.erb: config/bpm.yml
		packages:
		- - golang-1.19-linux
		+ - golang-1.20-linux
		- acceptance_tests
		- cf-cli-6-linux
		```
		## Built with go 1.20.1
		Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.257.0...v

Component	Version	Release Notes
		0.258.0 ## Resources - [Download release v0.258.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.258.0).
ruby-offline-buildpack	1.9.2	
silk	3.23.0	
smb-volume	3.1.10	▼ v3.1.10 ## Changes * Backfill property tests for force_noserverino (#103) ## Dependencies * **smbbroker:** Updated to v`17e471d`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker). * **smbdriver:** Updated to v`fcb9ca4`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).
smoke-tests	4.8.2	

Component	Version	Release Notes
staticfile-offline-buildpack	1.6.0	▼ 1.6.0

```
  * Run dynatrace API fixture using cflinuxfs3 (until go bp is pubic available on cflinuxfs4) (#348)
  * Rebuild nginx 1.23.3 (#347)
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/184035115)
  * Rebuild nginx 1.22.1 (#346)
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183579473)
  * Add support for cflinuxfs4 stack
  * Bump github.com/onsi/gomega from 1.27.0 to 1.27.1
  Bumps [github.com/onsi/gomega](https://github.com/onsi/gomega) from 1.27.0 to 1.27.1.
  - [Release notes](https://github.com/onsi/gomega/releases)
  - [Changelog](https://github.com/onsi/gomega/blob/master/CHANGELOG.md)
  ---
  updated-dependencies:
  - dependency-name: github.com/onsi/gomega
  ...
  * Bump golang.org/x/net from 0.6.0 to 0.7.0
  Bumps [golang.org/x/net](https://github.com/golang/net) from 0.6.0 to 0.7.0.
  - [Release notes](https://github.com/golang/net/releases)
  ---
  updated-dependencies:
  - dependency-name: golang.org/x/net
  ...
  * Updating github-config (#343 & #345)
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | nginx | 1.22.1 | cflinuxfs3 |
  | nginx | 1.22.1 | cflinuxfs4 |
  | nginx | 1.23.3 | cflinuxfs3 |
  | nginx | 1.23.3 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | nginx | 1.23.x |
  * Uncached buildpack SHA256: d2e7f7b08b6500c9cb445fc69a47f35ca7dd97745da15377ee6350af3d2fb9db
  * Uncached buildpack SHA256: e76bd0cfac7b170889830b0139bae30b6335ec84258576ee43ddc36d3257f9d4
```

▼ 1.5.39

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td> * update libbuildpack-dynatrace to 1.5.2
  * Bump github.com/onsi/gomega from 1.26.0 to 1.27.0
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | nginx | 1.22.1 | cflinuxfs3 |
  | nginx | 1.23.3 | cflinuxfs3 |
  Default binary versions:
  | name | version |
  |-|-|
  | nginx | 1.23.x |
  * Uncached buildpack SHA256: f2a894e0e2bca06ca77f2fb62a673dde86497ffd2246c40ea62ed2ec98bda772</td></tr>
<tr><td>statsd-injector</td><td>1.11.28</td><td></td></tr>
<tr><td>syslog</td><td>11.8.8</td><td></td></tr>
<tr><td>system-metrics-scraper</td><td>3.3.6</td><td></td></tr>
</table>

Component	Version	Release Notes
uaa	74.5.64	▼ v74.5.64 ## What's Changed * Bump github.com/cloudfoundry/bosh-utils from 0.0.350 to 0.0.352 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/510 * Bump rspec-mocks from 3.12.2 to 3.12.3 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/497 * Bump github.com/cloudfoundry/bosh-utils from 0.0.352 to 0.0.355 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/526 * Bump rspec-core from 3.12.0 to 3.12.1 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/509 * Bump zeitwerk from 2.6.6 to 2.6.7 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/518 * Bump github.com/onsi/gomega from 1.25.0 to 1.27.1 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/525 * Bump tzinfo from 2.0.5 to 2.0.6 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/503 * Bump activesupport from 6.1.7 to 6.1.7.2 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/505 * Bump github.com/cloudfoundry/bosh-utils from 0.0.355 to 0.0.356 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/530 **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.63...v74.5.64

2.12.24

Release Date: 02/28/2023

- Bump binary-offline-buildpack to version `1.1.2`
- Bump bosh-system-metrics-forwarder to version `0.0.30`
- Bump cf-autoscaling to version `249.1.2`
- Bump cf-networking to version `3.22.0`
- Bump cflinuxfs3 to version `0.352.0`
- Bump credhub to version `2.12.19`
- Bump dotnet-core-offline-buildpack to version `2.4.7`

- Bump garden-runc to version `1.23.0`
- Bump go-offline-buildpack to version `1.10.4`
- Bump log-cache to version `2.12.6`
- Bump loggregator-agent to version `6.5.8`
- Bump metric-registrar to version `2.1.2`
- Bump metrics-discovery to version `3.2.7`
- Bump nats to version `56.1.0`
- Bump nginx-offline-buildpack to version `1.2.0`
- Bump php-offline-buildpack to version `4.5.2`
- Bump python-offline-buildpack to version `1.8.6`
- Bump routing to version `0.257.0`
- Bump ruby-offline-buildpack to version `1.9.2`
- Bump silk to version `3.22.0`
- Bump smb-volume to version `3.1.9`
- Bump staticfile-offline-buildpack to version `1.5.38`
- Bump statsd-injector to version `1.11.28`
- Bump syslog to version `11.8.8`
- Bump system-metrics-scraper to version `3.3.6`
- Bump uaa to version `74.5.63`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.418	
backup-and-restore-sdk	1.18.61	

binary-offline-buildpack	1.1.2	▼ 1.1.2
		`  * Update libbuildpack`
		`  * Updating github-config`
		`  * Bump github.com/onsi/gomega from 1.25.0 to 1.26.0`
		`  * Uncached buildpack SHA256: bf9e70d06ccacf2086a942bce347ef8f30bafc4a5a09875044a8cb41817a240e`
		`  * Uncached buildpack SHA256: c92b3c994e8cfb97d80acd598685155a9ff3ac53903722e42aa5d8fb11686d5a`
		`  * Uncached buildpack SHA256: b8e510b406c92f27ec9520737dfbea429641200d3781b54572551c251ec0136d`
		`  * Uncached buildpack SHA256: 011c2255794a5f1ecb132ff477ffd41ddaba070558895321c508cfdddd3578f0`

Component	Version	Release Notes
bosh-dns-aliases	0.0.4	

bosh-system-metrics-forwarder	0.0.30	▼ v0.0.30
		`* update golang to 1.20.1`
		`**Full Changelog**: https://github.com/cloudfoundry/bosh-system-metrics-forwarder-release/compare/v0.0.29...v0.0.30`
		▼ v0.0.29
		`## What's Changed`
		`* update to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/bosh-system-metrics-forwarder-release/pull/12`
		`**Full Changelog**: https://github.com/cloudfoundry/bosh-system-metrics-forwarder-release/compare/v0.0.28...v0.0.29`

bpm	1.1.21	
capi	1.117.12	

cf-autoscaling	249.1.2	▼ v249.1.2
		`## What's Changed`
		`* update dependencies`
		`* Upgrade to go 1.20.1`

cf-cli	1.41.0	
cf-networking	3.22.0	
cflinuxfs3	0.352.0	

Component	Version	Release Notes
credhub	2.12.19	▼ 2.12.19 Security Fixes Bump various dependencies ## What's Changed * Feat: Add github actions to job specs by @peterhaochen47 in https://github.com/pivotal/credhub-release/pull/93 * Build(deps): Bump rspec-expectations from 3.12.0 to 3.12.1 in /spec by @peterhaochen47 in https://github.com/pivotal/credhub-release/pull/94 * Feat: Add github actions to run lint by @peterhaochen47 in https://github.com/pivotal/credhub-release/pull/95 * Build(deps): Bump parser from 3.1.3.0 to 3.2.0.0 in /spec by @dependabot in https://github.com/pivotal/credhub-release/pull/99 * Build(deps): Bump rspec-expectations from 3.12.1 to 3.12.2 in /spec by @dependabot in https://github.com/pivotal/credhub-release/pull/100 * Build(deps): Bump rspec-mocks from 3.12.1 to 3.12.2 in /spec by @dependabot in https://github.com/pivotal/credhub-release/pull/101 **Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.18...2.12.19
diego	2.71.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>dotnet-core-offline-buildpack</td><td>2.4.7</td><td>▼ 2.4.7
<pre>
  * Add node 18.14.0, remove node 18.13.0
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/184393261)
  Packaged binaries:
name	version	cf_stacks
bower	1.8.14	cflinuxfs3, cflinuxfs4
dotnet-aspnetcore	6.0.13	cflinuxfs3, cflinuxfs4
dotnet-aspnetcore	7.0.2	cflinuxfs3, cflinuxfs4
dotnet-runtime	6.0.13	cflinuxfs3, cflinuxfs4
dotnet-runtime	7.0.2	cflinuxfs3, cflinuxfs4
dotnet-sdk	6.0.405	cflinuxfs3, cflinuxfs4
dotnet-sdk	7.0.102	cflinuxfs3, cflinuxfs4
libgdiplus	6.1	cflinuxfs3
libgdiplus	6.1	cflinuxfs4
libunwind	1.6.2	cflinuxfs3
libunwind	1.6.2	cflinuxfs4
node	18.14.0	cflinuxfs3
node	18.14.0	cflinuxfs4
Default binary versions:
name	version
-	-
dotnet-runtime	6.0.x
dotnet-aspnetcore	6.0.x
dotnet-sdk	6.0.x
bower	1.8.x
  * Uncached buildpack SHA256: 3e358562aea02c009a323c2764a93b69b2d6cbbb7509284e578272a19e3e27fa
  * Uncached buildpack SHA256: 58b211da1c57e091e382cbba889a4dac650de6a2d9d463921af1fae12dfb8acd
</pre></td></tr>
<tr><td>garden-runc</td><td>1.23.0</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>go-offline-buildpack</td><td>1.10.4</td><td>▼ 1.10.4 * Update libbuildpack Packaged binaries: | name	version	cf_stacks
dep	0.5.4	cflinuxfs3
dep	0.5.4	cflinuxfs4
glide	0.13.3	cflinuxfs3
glide	0.13.3	cflinuxfs4
go	1.18.10	cflinuxfs3
go	1.18.10	cflinuxfs4
go	1.19.5	cflinuxfs3
go	1.19.5	cflinuxfs4
godep	80	cflinuxfs3
godep	80	cflinuxfs4 | Default binary versions: | name	version
-	-
go	1.18.x	 * Uncached buildpack SHA256: 56fac929ff6dcdb164233d2e945beb2525ef48a91c60f85dec991ebe92443b1b * Uncached buildpack SHA256: 3648319f545e416a6b7dc552cff8e8711901ab31271eee811a9269e0497b186f</td></tr>
<tr><td>haproxy</td><td>11.10.2</td><td></td></tr>
<tr><td>java-offline-buildpack</td><td>4.54</td><td></td></tr>
<tr><td>log-cache</td><td>2.12.6</td><td>▼ v2.12.6
* update to golang 1.20.1
Full Changelog: https://github.com/cloudfoundry/log-cache-release/compare/v2.12.5...v2.12.6
▼ v2.12.5
What's Changed
* update dependencies
* Update to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/log-cache-release/pull/162
Full Changelog: https://github.com/cloudfoundry/log-cache-release/compare/v2.12.4...v2.12.5</td></tr>
<tr><td>loggregator</td><td>106.7.5</td><td></td></tr>
</table>

Component	Version	Release Notes

loggregator-agent	6.5.8	▼ v6.5.8
		`## What's Changed`
		`* update dependencies`
		`* Upgrade to go 1.20.1 by @rroberts2222 in https://github.com/cloudfoundry/loggregator-agent-release/pull/224`
		`**Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.7...v6.5.8`

mapfs	1.2.12	

metric-registrar	2.1.2	▼ v2.1.2
		`## What's Changed`
		`* update to golang 1.20.1`
		`* update dependencies`

metrics-discovery	3.2.7	▼ v3.2.7
		`* update golang to 1.20.1`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.6...v3.2.7###`
		▼ v3.2.6
		`## What's Changed`
		`* Upgrade to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/104`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.5...v3.2.6`
		▼ v3.2.5
		`## What's Changed`
		`* Update dependencies`
		`* Expire individual metrics by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/103`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.4...v3.2.5`

mysql-monitoring	9.23.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nats</td><td>56.1.0</td><td></td></tr>
<tr><td>nfs-volume</td><td>7.1.8</td><td></td></tr>
<tr><td>nginx-offline-buildpack</td><td>1.2.0</td><td>▼ 1.2.0
<pre>
  * Add support for cflinuxfs4 stack
  Packaged binaries:
name	version	cf_stacks
nginx	1.22.1	cflinuxfs3
nginx	1.22.1	cflinuxfs4
nginx	1.23.3	cflinuxfs3
nginx	1.23.3	cflinuxfs4
openresty	1.13.6.2	cflinuxfs3
openresty	1.15.8.3	cflinuxfs3
openresty	1.17.8.2	cflinuxfs3
openresty	1.19.9.1	cflinuxfs3
openresty	1.19.9.1	cflinuxfs4
openresty	1.21.4.1	cflinuxfs3
openresty	1.21.4.1	cflinuxfs4
Default binary versions:
name	version
-	-
nginx	1.23.x
  * Uncached buildpack SHA256: ab7cd71dc8ff38fdb93558b7536ca03e41baf191fe72e1980f43d61a82378dc8
  * Uncached buildpack SHA256: 1ba096283d893a6040b3733cac0e90f1f87f6de0593b3f01a81183725aeab08a
</pre></td></tr>
<tr><td>nodejs-offline-buildpack</td><td>1.8.4</td><td></td></tr>
<tr><td>notifications</td><td>62</td><td></td></tr>
<tr><td>notifications-ui</td><td>40</td><td></td></tr>
</table>

Component	Version	Release Notes
php-offline-buildpack	4.5.2	▼ 4.5.2

```
  * Update buildpack-packager to 2.3.22
  * Update php default to latest 8.1.x versio
n (#807)
  * Updating version for php for 8.2.X (#805)
  * Add php 8.2.2
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/
184381976)
  * Add php 8.1.15, remove php 8.1.13
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/
184388841)
  * Add nginx 1.22.1, remove nginx 1.22.0
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183579473)
  * Add composer 2.5.2, remove composer 2.5.1
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
184394919)
  Packaged binaries:
  | name | version | cf_stacks | modules |
  |-|-|-|-|
  | appdynamics | 22.12.1-677 | cflinuxfs3, c
flinuxfs4 |  |
  | composer | 2.5.2 | cflinuxfs3, cflinuxfs4
|  |
  | httpd | 2.4.55 | cflinuxfs3 |  |
  | httpd | 2.4.55 | cflinuxfs4 |  |
  | newrelic | 9.20.0.310 | cflinuxfs3 |  |
  | nginx | 1.22.1 | cflinuxfs3 |  |
  | nginx | 1.22.1 | cflinuxfs4 |  |
  | nginx | 1.23.3 | cflinuxfs3 |  |
  | nginx | 1.23.3 | cflinuxfs4 |  |
  | php | 8.0.26 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sql
srv, pgsql, phalcon, phpiredis, pspell, psr,
rdkafka, readline, redis, shmop, snmp, soap,
sockets, sodium, solr, sqlsrv, ssh2, stomp, s
ysvmsg, sysvsem, sysvshm, tideways_xhprof, ti
dy, xdebug, xsl, yaf, yaml, zip, zlib |
  | php | 8.0.27 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sql
srv, pgsql, phalcon, phpiredis, pspell, psr,
rdkafka, readline, redis, shmop, snmp, soap,
```

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
php	8.1.14	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.14	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.15	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.15	cflinuxfs4	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.2.1	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, g</td></tr>
</table>

Component	Version	Release Notes

```
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  | php | 8.2.1 | cflinuxfs4 | amqp, apcu, bz
2, curl, dba, enchant, exif, fileinfo, ftp, g
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  | php | 8.2.2 | cflinuxfs3 | amqp, apcu, bz
2, curl, dba, enchant, exif, fileinfo, ftp, g
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  | php | 8.2.2 | cflinuxfs4 | amqp, apcu, bz
2, curl, dba, enchant, exif, fileinfo, ftp, g
d, gettext, gmp, igbinary, imagick, imap, lda
p, lzf, mailparse, maxminddb, mbstring, memca
ched, mongodb, msgpack, mysqli, oauth, opcach
e, openssl, pcntl, pdo, pdo_firebird, pdo_mys
ql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqls
rv, pgsql, phpiredis, pspell, psr, rdkafka, r
eadline, redis, shmop, snmp, soap, sockets, s
odium, solr, sqlsrv, ssh2, stomp, sysvmsg, sy
svsem, sysvshm, tideways_xhprof, tidy, xdebu
g, xsl, yaml, zip, zlib |
  Default binary versions:
  | name | version |
  |-|-|
  | php | 8.1.14 |
  | httpd | 2.4.55 |
  | newrelic | 9.20.0.310 |
  | nginx | 1.23.3 |
  | composer | 2.5.2 |
  * Uncached buildpack SHA256: 17c3a0b8f2fb84
b8e473e67168e138c2537ceb9b1abc66497704d00589d
```

Component	Version	Release Notes
		`9f725`
push-apps-manager-release	675.0.5	
push-usage-service-release	674.0.30	
pxc	0.50.0	

Component	Version	Release Notes

python-offline-buildpack	1.8.6	▼ 1.8.6
		`* Add special case for CFLAGS dir when using Python 3.7 (#703)`
		`* Update libbuildpack`
		`* Add pipenv 2023.2.4, remove pipenv 2022.12.19 (#694)`
		`for stack(s) cflinuxfs3, cflinuxfs4`
		`(https://www.pivotaltracker.com/story/show/184394842)`
		`* Add setuptools 67.1.0, remove setuptools 67.0.0 (#691)`
		`for stack(s) cflinuxfs4, cflinuxfs3`
		`(https://www.pivotaltracker.com/story/show/184372013)`
		`Packaged binaries:`
		`| name | version | cf_stacks |`
		`|-|-|-|`
		`| libffi | 3.2.1 | cflinuxfs3, cflinuxfs4 |`
		`| libmemcache | 1.0.18 | cflinuxfs3, cflinuxfs4 |`
		`| miniconda3-py39 | 4.12.0 | cflinuxfs3, cflinuxfs4 |`
		`| pip | 23.0 | cflinuxfs3, cflinuxfs4 |`
		`| pipenv | 2023.2.4 | cflinuxfs3 |`
		`| pipenv | 2023.2.4 | cflinuxfs4 |`
		`| python | 3.7.16 | cflinuxfs3 |`
		`| python | 3.7.16 | cflinuxfs4 |`
		`| python | 3.8.16 | cflinuxfs3 |`
		`| python | 3.8.16 | cflinuxfs4 |`
		`| python | 3.9.16 | cflinuxfs3 |`
		`| python | 3.9.16 | cflinuxfs4 |`
		`| python | 3.10.9 | cflinuxfs3 |`
		`| python | 3.10.9 | cflinuxfs4 |`
		`| python | 3.11.1 | cflinuxfs3 |`
		`| python | 3.11.1 | cflinuxfs4 |`
		`| setuptools | 67.1.0 | cflinuxfs3, cflinuxfs4 |`
		`Default binary versions:`
		`| name | version |`
		`|-|-|`
		`| python | 3.10.x |`
		`* Uncached buildpack SHA256: 3d3452fd1bc7b353f6c592117057866aa7bab77f4801492575f2afdd92c17730`
		`* Uncached buildpack SHA256: 18bedcabd2d61758f6bd76a6012ea4b6f015f0f248e07f0f95a0fdf2fd520cf7`
		▼ 1.8.5
		`* Add setuptools 67.0.0, remove setuptools 66.1.0 for stack(s) cflinuxfs4, cflinuxfs3`
		`* Add pip 23.0, remove pip 22.3.1 for stack(s) cflinuxfs4, cflinuxfs3`
		`* Add setuptools 67.0.0, remove setuptools`

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
65.6.3 for stack(s) cflinuxfs4, cflinuxfs3
  * Add pipenv 2022.12.19, remove pipenv 2022.11.30 for stack(s) cflinuxfs4, cflinuxfs3
  * Don't fall back to unvendored installs when vendored install fails (#638)
  Packaged binaries:
name	version	cf_stacks
libffi	3.2.1	cflinuxfs3, cflinuxfs4
libmemcache	1.0.18	cflinuxfs3, cflinuxfs4
miniconda3-py39	4.12.0	cflinuxfs3, cflinuxfs4
pip	23.0	cflinuxfs3, cflinuxfs4
pipenv	2022.12.19	cflinuxfs3
pipenv	2022.12.19	cflinuxfs4
python	3.7.16	cflinuxfs3
python	3.7.16	cflinuxfs4
python	3.8.16	cflinuxfs3
python	3.8.16	cflinuxfs4
python	3.9.16	cflinuxfs3
python	3.9.16	cflinuxfs4
python	3.10.9	cflinuxfs3
python	3.10.9	cflinuxfs4
python	3.11.1	cflinuxfs3
python	3.11.1	cflinuxfs4
setuptools	67.0.0	cflinuxfs3, cflinuxfs4
Default binary versions:
name	version
-	-
python	3.10.x
  * Uncached buildpack SHA256: eca1a38041bb6af7dcfac536fae13727247ab0ec88d66d7cb4ba8e444e252d7a
  * Uncached buildpack SHA256: 90269d1cde16fd530786e7c6e9120799c69a41c6e9de32d94ba04266fba56416
</pre></td></tr>
<tr><td>r-offline-buildpack</td><td>1.2.0</td><td></td></tr>
</table>

Component	Version	Release Notes

routing	0.257.0	▼ v0.257.0
		`## Changes`
		`- Bumped to build with golang 1.19.6`
		`##    Built with go 1.19.6`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.256.0...v0.257.0`
		`## Resources`
		`- [Download release v0.257.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.257.0).`
		▼ v0.256.0
		`## Changes`
		`- Update healthchecker in release to stable version`
		`##    Built with go 1.19.5`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.255.0...v0.256.0`
		`## Resources`
		`- [Download release v0.256.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.256.0).`

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>ruby-offline-buildpack</td><td>1.9.2</td><td>▼ 1.9.2 * Add node 18.14.0, remove node 18.13.0 for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/184393261) * Add jruby 9.4.1.0, remove jruby 9.4.0.0 for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/184417188) * Add jruby 9.3.10.0, remove jruby 9.3.9.0 for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/184375182) * Add rubygems 3.4.6, remove rubygems 3.4.4 (#703) for stack(s) cflinuxfs4, cflinuxfs3 (https://www.pivotaltracker.com/story/show/184361007) * Add bundler 2.4.6, remove bundler 2.4.4 (#702) for stack(s) cflinuxfs4, cflinuxfs3 (https://www.pivotaltracker.com/story/show/184360955) Packaged binaries: | name	version	cf_stacks
bundler	2.4.6	cflinuxfs3, cflinuxfs4
jruby	9.3.10.0	cflinuxfs3
jruby	9.3.10.0	cflinuxfs4
jruby	9.4.1.0	cflinuxfs3
jruby	9.4.1.0	cflinuxfs4
node	18.14.0	cflinuxfs3
node	18.14.0	cflinuxfs4
openjdk1.8-latest	1.8.0	cflinuxfs3, cflinuxfs4
ruby	2.7.6	cflinuxfs3
ruby	2.7.7	cflinuxfs3
ruby	3.0.4	cflinuxfs3
ruby	3.0.5	cflinuxfs3
ruby	3.1.2	cflinuxfs3
ruby	3.1.3	cflinuxfs3
ruby	3.1.3	cflinuxfs4
ruby	3.2.0	cflinuxfs3
ruby	3.2.0	cflinuxfs4
rubygems	3.4.6	cflinuxfs3, cflinuxfs4
yarn	1.22.19	cflinuxfs3, cflinuxfs4 | Default binary versions: | name	version
-	-
ruby	3.1.x	 * Uncached buildpack SHA256: c6ec574f34b409fde8a42562c478695c53677d2fcbd9fb3a45de8e12022da3f9 * Uncached buildpack SHA256: 02f519cd7818d2</td></tr>
</table>

Component	Version	Release Notes
		e268fb2356f207fb08cdd42b7373da54ad0219ef292ad69c93
silk	3.22.0	
smb-volume	3.1.9	▼ v3.1.9 ## Changes * Add force_noserverino property in smbdriver job (#102) ## Dependencies * **bosh-template:** Updated to v2.4.0. For more information, see [bosh-template](https://github.com/cloudfoundry/bosh). * **smbdriver:** Updated to v`adc77c7`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver). ▼ v3.1.8 ## Dependencies * **smbdriver:** Updated to v`6cc617a`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver). ▼ v3.1.7 ## Changes * Golang: Updated to v1.19.4 (#76) ## Dependencies * **rspec:** Updated to v3.12.0. For more information, see [rspec](https://github.com/rspec/rspec-metagem). * **smbbroker:** Updated to v`114bb05`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker). * **smbdriver:** Updated to v`f0b92e3`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).
smoke-tests	4.8.2	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.38</td><td>▼ 1.5.38 * Update libbuildpack * Updating github-config (#331) Packaged binaries: | name	version	cf_stacks
nginx	1.22.1	cflinuxfs3
nginx	1.23.3	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x	 * Uncached buildpack SHA256: d5e09a925a64324fe955509f792005da50626e553f03f4025064a9f7d3a9dc14</td></tr>
<tr><td>statsd-injector</td><td>1.11.28</td><td>▼ v1.11.28
* update to golang 1.20.1
Full Changelog: https://github.com/cloudfoundry/statsd-injector-release/compare/v1.11.27...v1.11.28
▼ v1.11.27
What's Changed
* update dependencies
* Upgrade to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/statsd-injector-release/pull/70
Full Changelog: https://github.com/cloudfoundry/statsd-injector-release/compare/v1.11.26...v1.11.27</td></tr>
</table>

Component	Version	Release Notes
syslog	11.8.8	▼ v11.8.8 * update to golang 1.20.1 **Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.7...v11.8.8 ▼ v11.8.7 ## What's Changed * use go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/syslog-release/pull/117 **Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.6...v11.8.7
system-metrics-scraper	3.3.6	▼ v3.3.6 * update to golang 1.20.1 **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.3.5...v3.3.6 ▼ v3.3.5 ## What's Changed * update dependencies * Upgrade to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/96 **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.3.4...v3.3.5
uaa	74.5.63	▼ v74.5.63 ### Dependency bumps - Various dependency bumps ### Security fixes - Add `X-Content-Type-Options=nosniff` header to UAA's static resource endpoint (`/resources/*`) to protect against content sniffing exploits. **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.62...v74.5.63

v2.12.23

Release Date: 02/09/2023

- **[Bug Fix]** Allows docker app workloads without a `sh` binary in the docker image to execute properly.
- **[Bug Fix]** [Cloud Controller] Truncate large errors from Service Brokers
- Bump backup-and-restore-sdk to version `1.18.61`
- Bump capi to version `1.117.12`
- Bump java-offline-buildpack to version `4.54`
- Bump loggregator-agent to version `6.5.7`
- Bump metric-registrar to version `2.1.1`
- Bump push-usage-service-release to version `674.0.30`
- Bump r-offline-buildpack to version `1.2.0`
- Bump routing to version `0.255.0`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.401	

Component	Version	Release Notes
backup-and-restore-sdk	1.18.61	▼ v1.18.61 ## Changes * Bump mysql from 5.7.39 to 5.7.40 (#849) * Bump ncurses from 6.3 to 6.4 (#847) * Download gnu-keyring on every execution (#840) * Fix gpg verify command (#846) * Golang: Updated to v1.19.5 (#834) * Update autobump-ncurses.sh (#839) ## Dependencies * **aws-sdk-go:** Updated to v1.44.180. For more information, see [aws-sdk-go](https://github.com/aws/aws-sdk-go). * **bosh-backup-and-restore:** Updated to v1.9.39. For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore). * **gomega:** Updated to v1.25.0. For more information, see [gomega](https://github.com/onsi/gomega). * **net:** Updated to v0.5.0. For more information, see [net](https://github.com/golang/net). * **oauth2:** Updated to v0.4.0. For more information, see [oauth2](https://github.com/golang/oauth2). * **api:** Updated to v0.107.0. For more information, see [api](https://github.com/googleapis/google-api-go-client).
binary-offline-buildpack	1.1.0	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.28	
bpm	1.1.21	
capi	1.117.12	
cf-autoscaling	249.1.1	
cf-cli	1.41.0	
cf-networking	3.19.0	
cflinuxfs3	0.350.0	
credhub	2.12.18	
diego	2.71.0	
dotnet-core-offline-buildpack	2.4.5	
garden-runc	1.22.9	

Component	Version	Release Notes
go-offline-buildpack	1.10.2	
haproxy	11.10.2	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>java-offline-buildpack</td><td>4.54</td><td>▼ 4.54 I'm pleased to announce the release of the java-buildpack, version 4.54. It primarily includes new OpenJDK versions, which are based on the Oracle Java Quarterly Updates for Jan 2023. This release includes the following: * A fix for offline packaging for the cflinuxfs4/jammy platform (#986) * Support for SPlunk Otel Java Framework (#968 - thanks to @breedx-splk) * A change to use the default-repository-root URI for the Apache Skywalking Framework (#983 - thanks to @mayrstefan) For a more detailed look at the changes in 4.54, please take a look at the [commit log](https://github.com/cloudfoundry/java-buildpack/compare/v4.53...v4.54). The packaged version of the buildpack, suitable for use with create-buildpack and update-buildpack, can be found attached to this release. | Dependency	Version	CVEs	Release Notes
AppDynamics Agent	`22.9.2_34409`		[Release Notes](https://docs.appdynamics.com/4.5.x/en/product-and-release-announcements/release-notes/language-agent-notes/java-agent-notes)
Azure Application Insights Agent	`2.6.4`		[Release Notes](https://github.com/Microsoft/ApplicationInsights-Java/releases)
CA Introscope APM Framework	`22.11.0_11`		
Client Certificate Mapper	`1.11.0_RELEASE`	Included inline above	Included inline above
Container Security Provider	`1.19.0_RELEASE`	Included inline above	Included inline above
Contrast Security Agent	`3.18.1`		[Release Notes](https://docs.contrastsecurity.com/en/java-agent-release-notes.html)
Datadog APM Javaagent	`1.5.0`		[Release Notes](https://github.com/DataDog/dd-trace-java/releases)
Elastic APM Agent	`1.35.0`		[Release Notes](https://www.elastic.co/guide/en/apm/agent/java/current/release-notes.html)
Gemalto Luna Security Provider	`7.4.0`		[Release Notes](https://www.thalesdocs.com/gphsm/luna/7/docs/network/Content/CRN/Luna/CRN_Luna.htm)
Geode Tomcat Session Store	`1.12.5`		|</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
Google Stackdriver Debugger	`2.32.0`		[Release Notes](https://cloud.google.com/debugger/docs/release-notes)
Google Stackdriver Profiler	`0.1.0`		[Release Notes](https://cloud.google.com/profiler/docs/release-notes)
Groovy	`2.5.20`		[Release Notes](http://www.groovy-lang.org/releases.html)
JaCoCo Agent	`0.8.8`		[Release Notes](https://github.com/jacoco/jacoco/releases)
Java Memory Assistant Agent	`0.5.0`		
Java Memory Assistant Clean Up	`0.1.0`		
JProfiler Profiler	`12.0.4`		[ChangeLog](https://www.ej-technologies.com/download/jprofiler/changelog.html)
JRebel Agent	`2022.4.2`		[ChangeLog](https://www.jrebel.com/products/jrebel/changelog)
jvmkill Agent	`1.17.0_RELEASE`	Included inline above	Included inline above
MariaDB JDBC Driver	`2.7.2`		[Release Notes](https://mariadb.com/kb/en/mariadb-connector-j-274-release-notes/)
Memory Calculator	`3.13.0_RELEASE`	Included inline above	Included inline above
Metric Writer	`3.5.0_RELEASE`	Included inline above	Included inline above
New Relic Agent	`7.11.1`		[Release Notes](https://docs.newrelic.com/docs/release-notes/agent-release-notes/java-release-notes/)
OpenJDK JRE 11	`11.0.18_10`	[Risk Matrix](https://www.oracle.com/security-alerts/cpuoct2022.html#AppendixJAVA)	[Release Notes](https://bell-sw.com/pages/liberica-release-notes-11.0.18/)
OpenJDK JRE 17	`17.0.6_10`	[Risk Matrix](https://www.oracle.com/security-alerts/cpuoct2022.html#AppendixJAVA)	[Release Notes](https://bell-sw.com/pages/liberica-release-notes-17.0.6/)
OpenJDK JRE 8	`1.8.0_362`	[Risk Matrix](https://www.oracle.com/security-alerts/cpuoct2022.html#AppendixJAVA)	[Release Notes](https://bell-sw.com/pages/liberica-release-notes-8u362/)
PostgreSQL JDBC Driver	`42.5.1`		[ChangeLog](https://jdbc.postgresql.org/documentation/changelog.html)
Redis Session Store	`1.3.6_RELEASE`	Included inline above	Included inline above
Riverbed Appinternals Agent	`11.8.5_BL527`		
SeaLights Agent	`4.0.2204`		
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
SkyWalking	`8.9.0`		[ChangeLog](https://github.com/apache/skywalking/tree/master/changes)
Splunk OpenTelemetry Java Agent	`1.20.0`		[Release Notes](https://github.com/signalfx/splunk-otel-java/releases)
Spring Auto-reconfiguration	`2.12.0_RELEASE`	Included inline above	Included inline above
Spring Boot CLI	`2.7.8`		
Spring Boot Container Customizer	`2.6.0_RELEASE`	Included inline above	Included inline above
Tomcat	`9.0.71`	[Security](https://tomcat.apache.org/security-9.html)	[ChangeLog](https://tomcat.apache.org/tomcat-9.0-doc/changelog.html)
Tomcat Access Logging Support	`3.4.0_RELEASE`	Included inline above	Included inline above
Tomcat Lifecycle Support	`3.4.0_RELEASE`	Included inline above	Included inline above
Tomcat Logging Support	`3.4.0_RELEASE`	Included inline above	Included inline above
YourKit Profiler	`2022.9.177`		[Release Notes](https://www.yourkit.com/download/yjp_2022_3_builds.jsp)
</pre></td></tr>
<tr><td>log-cache</td><td>2.12.4</td><td></td></tr>
<tr><td>loggregator</td><td>106.7.5</td><td></td></tr>
<tr><td>loggregator-agent</td><td>6.5.7</td><td>▼ v6.5.7<pre>
  ## What's Changed
  * Sanitize ProcID in syslog messages so messages with utf-8 in the source_type are not dropped by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/202
  * Update dependencies
  Full Changelog: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.6...v6.5.7
</pre></td></tr>
<tr><td>mapfs</td><td>1.2.12</td><td></td></tr>
</table>

Component	Version	Release Notes
metric-registrar	2.1.1	▼ v2.1.1 ## What's Changed * Bump dependencies **Full Changelog**: https://github.com/pivotal-cf/metric-registrar-release/compare/v2.1.0...v2.1.1
metrics-discovery	3.2.4	
mysql-monitoring	9.23.0	
nats	56	
nfs-volume	7.1.8	
nginx-offline-buildpack	1.1.45	
nodejs-offline-buildpack	1.8.4	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.5.0	
push-apps-manager-release	675.0.5	
push-usage-service-release	674.0.30	▼ v674.0.30 ## Dependencies * **app-usage-service:** Updated to v`c107e47`. For more information, see [app-usage-service](https://github.com/pivotal-cf/app-usage-service).
pxc	0.50.0	
python-offline-buildpack	1.8.4	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>r-offline-buildpack</td><td>1.2.0</td><td>▼ 1.2.0 * Add cflinuxfs4 stack support for R 4.2.X * Rebuild r 4.2.1 for stack(s) cflinuxfs3, cflinuxfs4 with dependencies for stack cflinuxfs3: forecast 8.18, plumber 1.2.1, rserve 1.8.10, shiny 1.7.3 with dependencies for stack cflinuxfs4: forecast 8.18, plumber 1.2.1, rserve 1.8.10, shiny 1.7.3 * Rebuild r 4.2.2 for stack(s) cflinuxfs4, cflinuxfs3 with dependencies for stack cflinuxfs4: forecast 8.20, plumber 1.2.1, rserve 1.8.11, shiny 1.7.4 with dependencies for stack cflinuxfs3: forecast 8.20, plumber 1.2.1, rserve 1.8.11, shiny 1.7.4 (https://www.pivotaltracker.com/story/show/184170168) Packaged binaries: | name	version	cf_stacks	modules
r	3.6.2	cflinuxfs3	forecast, plumber, rserve, shiny
r	3.6.3	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.1	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.1	cflinuxfs4	forecast, plumber, rserve, shiny
r	4.2.2	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.2	cflinuxfs4	forecast, plumber, rserve, shiny | * Uncached buildpack SHA256: c0f65e6cc63b505c497fa3f4f12df7707570535082796cec43c01b825a630755 * Uncached buildpack SHA256: bf3db369196f4a562e75d5cdb760f5a41a6fcc3d5100968804d45b4cd8e8c7f4</td></tr>
</table>

Component	Version	Release Notes

routing	0.255.0	▼ v0.255.0
		`[Upgrade healthchecker in release](https://github.com/cloudfoundry/routing-release/commit/ddb43e9e746b009d0ea6e6cf8cf8e7eb059ffafc). In order to limit the scope of packages brought in with the introduction of http healthchecker, we migrated the healthchecker package out of cf-networking-helpers into its own release.`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.254.0...v0.255.0`
		`Built with go 1.19.5`

ruby-offline-buildpack	1.9.0	
silk	3.19.0	
smb-volume	3.1.6	
smoke-tests	4.8.2	
staticfile-offline-buildpack	1.5.36	
statsd-injector	1.11.26	
syslog	11.8.6	
system-metrics-scraper	3.3.4	
uaa	74.5.62	

v2.12.22

Release Date: 01/30/2023

- Bump backup-and-restore-sdk to version `1.18.60`
- Bump cf-autoscaling to version `249.1.1`
- Bump cf-networking to version `3.19.0`
- Bump cflinuxfs3 to version `0.350.0`
- Bump garden-runc to version `1.22.9`
- Bump metric-registrar to version `2.1.0`
- Bump nats to version `56`
- Bump routing to version `0.254.0`
- Bump ruby-offline-buildpack to version `1.9.0`
- Bump silk to version `3.19.0`
- Bump uaa to version `74.5.62`

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>ubuntu-xenial stemcell</td><td>621.376</td><td></td></tr>
<tr><td>backup-and-restore-sdk</td><td>1.18.60</td><td>▼ v1.18.60
<pre>
  ## Changes
  * Support postgres 15 (#785)
  * Update vendored package golang-1-linux (#800)
  * chore: remove usage of ioutil (#809)
  ## Dependencies
  * **storage:** Updated to v1.28.1.
For more information, see [storage](https://github.com/googleapis/google-cloud-go).
  * **aws-sdk-go:** Updated to v1.44.157.
For more information, see [aws-sdk-go](https://github.com/aws/aws-sdk-go).
  * **mysql:** Updated to v1.7.0.
For more information, see [mysql](https://github.com/go-sql-driver/mysql).
  * **gomega:** Updated to v1.24.2.
For more information, see [gomega](https://github.com/onsi/gomega).
  * **net:** Updated to v0.4.0.
For more information, see [net](https://github.com/golang/net).
  * **api:** Updated to v0.104.0.
For more information, see [api](https://github.com/googleapis/google-api-go-client).
</pre></td></tr>
<tr><td>binary-offline-buildpack</td><td>1.1.0</td><td></td></tr>
<tr><td>bosh-dns-aliases</td><td>0.0.4</td><td></td></tr>
<tr><td>bosh-system-metrics-forwarder</td><td>0.0.28</td><td></td></tr>
<tr><td>bpm</td><td>1.1.21</td><td></td></tr>
<tr><td>capi</td><td>1.117.11</td><td></td></tr>
</table>

Component	Version	Release Notes
cf-autoscaling	249.1.1	▼ v249.1.1 ## What's Changed * Bump golang to 1.19.5 * Bump `org.apache.tomcat.embed:tomcat-embed-websocket` to 9.0.70 to address [CVE-2022-45143](https://nvd.nist.gov/vuln/detail/CVE-2022-45143) **Full Changelog**: https://github.com/pivotal-cf/cf-autoscaling-release/compare/v249.1.0...v249.1.1 ▼ v249.1.0 * update dependencies. * add cflinux stack as a bosh property
cf-cli	1.41.0	
cf-networking	3.19.0	
cflinuxfs3	0.350.0	
credhub	2.12.18	
diego	2.71.0	
dotnet-core-offline-buildpack	2.4.5	
garden-runc	1.22.9	
go-offline-buildpack	1.10.2	
haproxy	11.10.2	
java-offline-buildpack	4.53	
log-cache	2.12.4	
loggregator	106.7.5	
loggregator-agent	6.5.6	
mapfs	1.2.12	
metric-registrar	2.1.0	▼ v2.1.0 ## What's Changed * update dependencies * add cflinux stack as a bosh property
metrics-discovery	3.2.4	
mysql-monitoring	9.23.0	

Component	Version	Release Notes
nats	56	
nfs-volume	7.1.8	
nginx-offline-buildpack	1.1.45	
nodejs-offline-buildpack	1.8.4	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.5.0	
push-apps-manager-release	675.0.5	
push-usage-service-release	674.0.29	
pxc	0.50.0	
python-offline-buildpack	1.8.4	
r-offline-buildpack	1.1.36	

routing	0.254.0	▼ v0.254.0
		`Built with go 1.19.5`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.253.0...v0.254.0`
		▼ v0.253.0
		`## What's Changed`
		`* Specs to make maxRetries configurable for endpoints and route-services by @domdom82 in https://github.com/cloudfoundry/routing-release/pull/298`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.252.0...v0.253.0`

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>ruby-offline-buildpack</td><td>1.9.0</td><td>▼ 1.9.0
<pre>
  * Add support for cflinuxfs4 stack
  * Update default ruby version to 3.1
  * Remove deprecated bundler 1.x version line (#650)
  * supply: remove ruby.exe symlinking
  This seems to be from a heroku era effort to support windows.
  * openjdk8: update dependency
  Packaged binaries:
name	version	cf_stacks
bundler	2.3.26	cflinuxfs3, cflinuxfs4
jruby	9.3.9.0	cflinuxfs3
jruby	9.3.9.0	cflinuxfs4
jruby	9.4.0.0	cflinuxfs3
jruby	9.4.0.0	cflinuxfs4
node	18.12.1	cflinuxfs3
node	18.12.1	cflinuxfs4
openjdk1.8-latest	1.8.0	cflinuxfs3, cflinuxfs4
ruby	2.7.6	cflinuxfs3
ruby	2.7.7	cflinuxfs3
ruby	3.0.4	cflinuxfs3
ruby	3.0.5	cflinuxfs3
ruby	3.1.2	cflinuxfs3
ruby	3.1.3	cflinuxfs3
ruby	3.1.3	cflinuxfs4
rubygems	3.3.26	cflinuxfs3, cflinuxfs4
yarn	1.22.19	cflinuxfs3, cflinuxfs4
Default binary versions:
name	version
-	-
ruby	3.1.x
  * Uncached buildpack SHA256: c0ecb20c3135d572deb13cccbd2b7a8065b2b355609d2ab5d4bb71cc5963c0b8
  * Uncached buildpack SHA256: ab48ac40a7fc03dd1511508f8aaafd788af5bf9f764cdd3127e40a6e12f84cd2
</pre></td></tr>
<tr><td>silk</td><td>3.19.0</td><td></td></tr>
<tr><td>smb-volume</td><td>3.1.6</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.8.2</td><td></td></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.36</td><td></td></tr>
<tr><td>statsd-injector</td><td>1.11.26</td><td></td></tr>
<tr><td>syslog</td><td>11.8.6</td><td></td></tr>
</table>

Component	Version	Release Notes
system-metrics-scraper	3.3.4	

uaa	74.5.62	▼ v74.5.62
		`### Dependency bumps`
		`- Various dependency bumps`
		`**Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.61...v74.5.62`

v2.12.21

Release Date: 01/17/2023

- **[Bug Fix]** Fix issues with secure scraping with rolling deploys and with ncp
- Bump capi to version `1.117.11`
- Bump cf-networking to version `3.17.0`
- Bump cflinuxfs3 to version `0.347.0`
- Bump credhub to version `2.12.18`
- Bump diego to version `2.71.0`
- Bump dotnet-core-offline-buildpack to version `2.4.5`
- Bump garden-runc to version `1.22.7`
- Bump loggregator-agent to version `6.5.6`
- Bump metric-registrar to version `2.0.1`
- Bump nats to version `54`
- Bump nodejs-offline-buildpack to version `1.8.4`
- Bump php-offline-buildpack to version `4.5.0`
- Bump pxc to version `0.50.0`
- Bump python-offline-buildpack to version `1.8.4`
- Bump r-offline-buildpack to version `1.1.36`
- Bump routing to version `0.252.0`
- Bump silk to version `3.17.0`
- Bump smoke-tests to version `4.8.2`
- Bump staticfile-offline-buildpack to version `1.5.36`
- Bump uaa to version `74.5.61`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.364	

Component	Version	Release Notes
backup-and-restore-sdk	1.18.59	
binary-offline-buildpack	1.1.0	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.28	
bpm	1.1.21	
capi	1.117.11	
cf-autoscaling	249.0.25	
cf-cli	1.41.0	
cf-networking	3.17.0	
cflinuxfs3	0.347.0	

credhub	2.12.18	▼ 2.12.18
		`### Security Fixes`
		`- Bump various dependencies`
		▼ 2.12.17
		`### Security Fixes`
		`- Bump various dependencies`

diego	2.71.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>dotnet-core-offline-buildpack</td><td>2.4.5</td><td>

▼ 2.4.5

```
  * Deprecate .Net (aspnetcore, runtime and sdk)  3.x
  * Add dotnet-aspnetcore 6.0.12, remove dotnet-aspnetcore 6.0.11 for stack(s) cflinuxfs4, cflinuxfs3
  * Add dotnet-runtime 6.0.12, remove dotnet-runtime 6.0.11 for stack(s) cflinuxfs4, cflinuxfs3
  * Add dotnet-sdk 6.0.404, remove dotnet-sdk 6.0.403 for stack(s) cflinuxfs4, cflinuxfs3
  * Add dotnet-aspnetcore 7.0.1, remove dotnet-aspnetcore 7.0.0 for stack(s) cflinuxfs4, cflinuxfs3
  * Add dotnet-runtime 7.0.1, remove dotnet-runtime 7.0.0 for stack(s) cflinuxfs4, cflinuxfs3
  * Add dotnet-sdk 7.0.101, remove dotnet-sdk 7.0.100 for stack(s) cflinuxfs4, cflinuxfs3
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | bower | 1.8.14 | cflinuxfs3, cflinuxfs4 |
  | dotnet-aspnetcore | 6.0.12 | cflinuxfs3, cflinuxfs4 |
  | dotnet-aspnetcore | 7.0.1 | cflinuxfs3, cflinuxfs4 |
  | dotnet-runtime | 6.0.12 | cflinuxfs3, cflinuxfs4 |
  | dotnet-runtime | 7.0.1 | cflinuxfs3, cflinuxfs4 |
  | dotnet-sdk | 6.0.404 | cflinuxfs3, cflinuxfs4 |
  | dotnet-sdk | 7.0.101 | cflinuxfs3, cflinuxfs4 |
  | libgdiplus | 6.1 | cflinuxfs3 |
  | libgdiplus | 6.1 | cflinuxfs4 |
  | libunwind | 1.6.2 | cflinuxfs3 |
  | libunwind | 1.6.2 | cflinuxfs4 |
  | node | 18.12.1 | cflinuxfs3 |
  | node | 18.12.1 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | dotnet-runtime | 6.0.x |
  | dotnet-aspnetcore | 6.0.x |
  | dotnet-sdk | 6.0.x |
  | bower | 1.8.x |
  * Uncached buildpack SHA256: a5a6b1c0e919042835e7e1a2a444608c67023a5c675947171dfcbac4ed909fc5
  * Uncached buildpack SHA256: 3da2519e1db72575729d450dc0609dd58b219e381265ce2eb139a08183c
```

</td></tr>
</table>

Component	Version	Release Notes
		`a249d`
garden-runc	1.22.7	
go-offline-buildpack	1.10.2	
haproxy	11.10.2	
java-offline-buildpack	4.53	
log-cache	2.12.4	
loggregator	106.7.5	
loggregator-agent	6.5.6	▼ v6.5.6 ## What's Changed * fix scraping with non-positive intervals to preserve non-scraping behavior by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/174 * updated some dependencies. **Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.5...v6.5.6
mapfs	1.2.12	
metric-registrar	2.0.1	▼ v2.0.1 - bump-golang to v0.114.0 for golang 1.19.4 - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src/tools - remove unneeded appinfo getter (#91) ▼ v2.0.0 ## What's Changed * Add property to correctly handle nsxt/ncp when turned on * Add a property to allow setting a maximum number of metrics per instance * Fix bug: Rolling deploys should no longer cause metrics to stop scraping for secure endpoints.
metrics-discovery	3.2.4	
mysql-monitoring	9.23.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nats</td><td>54</td><td></td></tr>
<tr><td>nfs-volume</td><td>7.1.8</td><td></td></tr>
<tr><td>nginx-offline-buildpack</td><td>1.1.45</td><td></td></tr>
<tr><td>nodejs-offline-buildpack</td><td>1.8.4</td><td>▼ 1.8.4

  * Bumps default node version to 18
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | node | 14.20.1 | cflinuxfs3 |
  | node | 14.20.1 | cflinuxfs4 |
  | node | 14.21.1 | cflinuxfs3 |
  | node | 14.21.1 | cflinuxfs4 |
  | node | 16.18.0 | cflinuxfs3 |
  | node | 16.18.0 | cflinuxfs4 |
  | node | 16.18.1 | cflinuxfs3 |
  | node | 16.18.1 | cflinuxfs4 |
  | node | 18.10.0 | cflinuxfs3 |
  | node | 18.10.0 | cflinuxfs4 |
  | node | 18.12.1 | cflinuxfs3 |
  | node | 18.12.1 | cflinuxfs4 |
  | yarn | 1.22.19 | cflinuxfs3, cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | node | 18.x |
  * Uncached buildpack SHA256: 1f18bab4acde32b9d919a938e863492a5bda090ac200a54aba68cb429f308423
  * Uncached buildpack SHA256: 7d1371c250ab2af98b0be2375f6e578a6fb1790c432ca6d6b64ed0465cf84a6e</td></tr>
<tr><td>notifications</td><td>62</td><td></td></tr>
<tr><td>notifications-ui</td><td>40</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>php-offline-buildpack</td><td>4.5.0</td><td>▼ 4.5.0 * Deprecate PHP 7.4.x * Remove CAAPM Agent not supported by PHP 8 (https://github.com/cloudfoundry/php-buildpack/issues/746) * Update default PHP Version to 8.1.13 * Add appdynamics 22.12.0-667, remove appdynamics 22.10.0-627 for stack(s) cflinuxfs3 * Add nginx 1.23.3, remove nginx 1.23.2 for stack(s) cflinuxfs3 * Add php 8.0.26, remove php 8.0.24 for stack(s) cflinuxfs3 * Add php 8.1.13, remove php 8.1.11 for stack(s) cflinuxfs3 Packaged binaries: | name	version	cf_stacks	modules
appdynamics	22.12.0-667	cflinuxfs3	
composer	2.4.4	cflinuxfs3	
httpd	2.4.54	cflinuxfs3	
newrelic	9.20.0.310	cflinuxfs3	
nginx	1.22.0	cflinuxfs3	
nginx	1.23.3	cflinuxfs3	
php	8.0.25	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.0.26	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib
php	8.1.12	cflinuxfs3	amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, psp</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>ell, psr, rdkafka, readline, redis, shmop, sn
mp, soap, sockets, sodium, solr, sqlsrv, ssh
2, stomp, sysvmsg, sysvsem, sysvshm, tideways
_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, z
lib |
  | php | 8.1.13 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, io
ncube, ldap, lzf, mailparse, maxminddb, mbstr
ing, memcached, mongodb, msgpack, mysqli, oau
th, opcache, openssl, pcntl, pdo, pdo_firebir
d, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlit
e, pdo_sqlsrv, pgsql, phalcon, phpiredis, psp
ell, psr, rdkafka, readline, redis, shmop, sn
mp, soap, sockets, sodium, solr, sqlsrv, ssh
2, stomp, sysvmsg, sysvsem, sysvshm, tideways
_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, z
lib |
  Default binary versions:
  | name | version |
  |-|-|
  | php | 8.1.13 |
  | httpd | 2.4.54 |
  | newrelic | 9.20.0.310 |
  | nginx | 1.23.3 |
  | composer | 2.4.4 |
  * Uncached buildpack SHA256: a6fe40a80c5c3d
a8f9cb43127981b12150d0868437b3e2a3cd300b588c1
dacd8</td></tr>
<tr><td>push-apps-manager-release</td><td>675.0.5</td><td></td></tr>
<tr><td>push-usage-service-release</td><td>674.0.29</td><td></td></tr>
<tr><td>pxc</td><td>0.50.0</td><td></td></tr>
</table>

Component	Version	Release Notes
python-offline-buildpack	1.8.4	▼ 1.8.4

```
  * Add python 3.11.1, remove python 3.11.0 (#665)
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183976131)
  * Add python 3.10.9, remove python 3.10.8
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/183976637)
  * Add python 3.9.16, remove python 3.9.15 (#663)
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183975707)
  * Add python 3.8.16, remove python 3.8.15 (#660)
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183975586)
  * Add python 3.7.16, remove python 3.7.15
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183975932)
  * Add pipenv 2022.11.30, remove pipenv 2022.11.5
  for stack(s) cflinuxfs4, cflinuxfs3
  * Add setuptools 65.6.3, remove setuptools 65.5.1
  for stack(s) cflinuxfs4, cflinuxfs3
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | libffi | 3.2.1 | cflinuxfs3, cflinuxfs4 |
  | libmemcache | 1.0.18 | cflinuxfs3, cflinuxfs4 |
  | miniconda3-py39 | 4.12.0 | cflinuxfs3, cflinuxfs4 |
  | pip | 22.3.1 | cflinuxfs3, cflinuxfs4 |
  | pipenv | 2022.11.30 | cflinuxfs3 |
  | pipenv | 2022.11.30 | cflinuxfs4 |
  | python | 3.7.16 | cflinuxfs3 |
  | python | 3.7.16 | cflinuxfs4 |
  | python | 3.8.16 | cflinuxfs3 |
  | python | 3.8.16 | cflinuxfs4 |
  | python | 3.9.16 | cflinuxfs3 |
  | python | 3.9.16 | cflinuxfs4 |
  | python | 3.10.9 | cflinuxfs3 |
  | python | 3.10.9 | cflinuxfs4 |
  | python | 3.11.1 | cflinuxfs3 |
  | python | 3.11.1 | cflinuxfs4 |
  | setuptools | 65.6.3 | cflinuxfs3, cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
```

Component	Version	Release Notes
		| python | 3.10.x | * Uncached buildpack SHA256: 544f05133e30c82a490a59d68f27551e6370ef67df403a30c96496a490157a78 * Uncached buildpack SHA256: 52120d40c3bd47904ae109e76849fd2a4a9829e36b0a9fda30b690a2ae884c0b
r-offline-buildpack	1.1.36	▼ 1.1.36 * Rebuild r 4.2.2 for stack(s) cflinuxfs3 with dependencies for stack cflinuxfs3: forecast 8.19, plumber 1.2.1, rserve 1.8.11, shiny 1.7.3 (https://www.pivotaltracker.com/story/show/183897155) Packaged binaries: | name | version | cf_stacks | modules | |-|-|-|-| | r | 3.6.2 | cflinuxfs3 | forecast, plumber, rserve, shiny | | r | 3.6.3 | cflinuxfs3 | forecast, plumber, rserve, shiny | | r | 4.2.1 | cflinuxfs3 | forecast, plumber, rserve, shiny | | r | 4.2.2 | cflinuxfs3 | forecast, plumber, rserve, shiny | * Uncached buildpack SHA256: 0158a04a0597a9d22a319790ab1ed99422a406dc7a20f52102b4b2725cef740a
routing	0.252.0	▼ v0.252.0 ## What's Changed - Improve random source for least connection pool to be thread safe. Thanks Daniel Lynch! **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.251.0...v0.252.0
ruby-offline-buildpack	1.8.60	
silk	3.17.0	
smb-volume	3.1.6	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>smoke-tests</td><td>4.8.2</td><td>▼ 4.8.2
Port assets/ruby_simple to Ruby 3</td></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.36</td><td>▼ 1.5.36 * Handle comma separated X-Forwarded-Proto (https://github.com/cloudfoundry/staticfile-buildpack/pull/318) Packaged binaries: | name	version	cf_stacks
nginx	1.22.1	cflinuxfs3
nginx	1.23.2	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x	 * Uncached buildpack SHA256: cfc15d472d16a03a2691b497b5387da9e18be1493cd89a9be5becdb1d63754b3</td></tr>
<tr><td>statsd-injector</td><td>1.11.26</td><td></td></tr>
<tr><td>syslog</td><td>11.8.6</td><td></td></tr>
<tr><td>system-metrics-scraper</td><td>3.3.4</td><td></td></tr>
<tr><td>uaa</td><td>74.5.61</td><td>▼ v74.5.61
Dependency bumps
- Various dependency bumps</td></tr>
</table>

v2.12.20

Release Date: 12/15/2022

- **[Security Fix]** Fix CVE-2022-31733: Unsecured Application Port
- **[Bug Fix]** service_credential_bindings can bind inactive service plans in capi v3
- Bump backup-and-restore-sdk to version `1.18.59`
- Bump binary-offline-buildpack to version `1.1.0`
- Bump bosh-system-metrics-forwarder to version `0.0.28`
- Bump bpm to version `1.1.21`
- Bump capi to version `1.117.10`
- Bump cf-autoscaling to version `249.0.25`

- Bump cf-networking to version `3.16.0`
- Bump cflinuxfs3 to version `0.345.0`
- Bump credhub to version `2.12.16`
- Bump diego to version `2.70.0`
- Bump dotnet-core-offline-buildpack to version `2.4.4`
- Bump go-offline-buildpack to version `1.10.2`
- Bump log-cache to version `2.12.4`
- Bump loggregator to version `106.7.5`
- Bump loggregator-agent to version `6.5.5`
- Bump metric-registrar to version `1.2.14`
- Bump metrics-discovery to version `3.2.4`
- Bump mysql-monitoring to version `9.23.0`
- Bump nfs-volume to version `7.1.8`
- Bump php-offline-buildpack to version `4.4.68`
- Bump push-usage-service-release to version `674.0.29`
- Bump pxc to version `0.49.0`
- Bump python-offline-buildpack to version `1.8.3`
- Bump r-offline-buildpack to version `1.1.35`
- Bump routing to version `0.251.0`
- Bump ruby-offline-buildpack to version `1.8.60`
- Bump silk to version `3.16.0`
- Bump statsd-injector to version `1.11.26`
- Bump syslog to version `11.8.6`
- Bump system-metrics-scraper to version `3.3.4`
- Bump uaa to version `74.5.60`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.364	

Component	Version	Release Notes
backup-and-restore-sdk	1.18.59	▼ v1.18.59 ## Changes * Bump mariadb from 10.6.10 to 10.6.11 (#778) ## Dependencies * **aws-sdk-go:** Updated to v1.44.150. For more information, see [aws-sdk-go](https://github.com/aws/aws-sdk-go). * **v2:** Updated to v2.5.1. For more information, see [v2](https://github.com/onsi/ginkgo). ▼ v1.18.58 ## Changes * Bump postgres from 10.22 to 10.23 (#772) * Bump postgres from 11.17 to 11.18 (#773) * Bump postgres from 13.8 to 13.9 (#774) ## Dependencies * **google-cloud-go:** Updated to v1.28.0. For more information, see [google-cloud-go](https://github.com/googleapis/google-cloud-go). * **aws-sdk-go:** Updated to v1.44.137. For more information, see [aws-sdk-go](https://github.com/aws/aws-sdk-go). * **bosh-backup-and-restore:** Updated to v1.9.38. For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore). * **ginkgo:** Updated to v2.5.0. For more information, see [ginkgo](https://github.com/onsi/ginkgo). * **gomega:** Updated to v1.24.1. For more information, see [gomega](https://github.com/onsi/gomega). * **google-api-go-client:** Updated to v0.103.0. For more information, see [google-api-go-client](https://github.com/googleapis/google-api-go-client).

Component	Version	Release Notes

binary-offline-buildpack	1.1.0	▼ 1.1.0
		`* Add support for cflinuxfs4 stack`
		`* Bumps go.mod go version to 1.19`
		`* Uncached buildpack SHA256: ccfe7b77068236753986f6425a00b960fa4edcb013f6c9e5a740de8aca52761d`
		`* Uncached buildpack SHA256: 0a493631418ab46f9efe527f6d1defd01566e031e81c6be56f82939e19cd3656`
		`* Uncached buildpack SHA256: b6a4f88f1180b03d476924368667457e45bea08dce584ff4ce4e967320d72b36`
		`* Uncached buildpack SHA256: c370f58b71c01ac6082d308b1ac51abefedcde0a2caeec47552965f3ff492de2`

bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.28	▼ v0.0.28 `- bump-golang to v0.114.0 for golang 1.19.4`
bpm	1.1.21	
capi	1.117.10	

Component	Version	Release Notes
cf-autoscaling	249.0.25	▼ v249.0.25 ## What's Changed * Bump golang to 1.19.4 * Check for race conditions in unit and repository tests by @ctlong in https://github.com/pivotal-cf/cf-autoscaling-release/pull/664 * Bump kotlinVersion from 1.7.20 to 1.7.21 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/728 * add quota events to the events stream by @Benjamintf1 in https://github.com/pivotal-cf/cf-autoscaling-release/pull/733 * Bump jackson-module-kotlin from 2.13.4 to 2.14.1 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/732 * Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/730 * Bump kotlinVersion from 1.7.21 to 1.7.22 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/736 * Bump github.com/go-sql-driver/mysql from 1.6.0 to 1.7.0 in /src by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/737 ▼ v249.0.24 - Bump to go1.19.3 - Bump to [spring-boot v2.7.5](https://github.com/spring-projects/spring-boot/releases/tag/v2.7.5)
cf-cli	1.41.0	
cf-networking	3.16.0	
cflinuxfs3	0.345.0	

Component	Version	Release Notes

credhub	2.12.16	▼ 2.12.16
		`## Bug Fixes`
		`* Fixes #74 (a known issue in CredHub release `2.12.4` and `2.12.5` where CredHub fails to connect to [postgres-release](https://github.com/cloudfoundry/postgres-release), and possibly other postgres servers, due to TLS handshake errors) by reverting the Java version bump to `jre8u352` in `2.12.4`. We will bring back this bump once the postgres TLS issue is addressed.`
		`* Fixes #71 where CredHub fails when deployed with a FIPS-enabled Bionic Stemcell.`
		`## Security Fixes`
		`- Bump various dependencies`
		`## New Contributors`
		`* @xandroc made their first contribution in https://github.com/pivotal/credhub-release/pull/75`
		`**Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.15...2.12.16`
		▼ 2.12.15
		`## Security Fixes`
		`- [bump dependency-check-gradle from 7.3.0 to 7.3.2](https://github.com/cloudfoundry/credhub/commit/164f3803d661bbf3684f22f66ec696688d9e4322)`
		`- [bump grpcVersion from 1.50.2 to 1.51.0](https://github.com/cloudfoundry/credhub/commit/ee931e16e5be76b9c485cfde915962dc6bb644e5)`
		`- [bump: springRestDocsVersion from 2.0.6 to 2.0.7](https://github.com/cloudfoundry/credhub/commit/c8b47a94870d8b86a65dd8d39fc24ced0608b752)`
		`## Known Issues`
		`- This release breaks compatibility with [postgres-release](https://github.com/cloudfoundry/postgres-release) (and possibly more postgres server) with a TLS handshake error. Do not use this release if you are using postgres. The fix is underway.`
		`**Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.14...2.12.15`
		▼ 2.12.14
		`## Security Fixes`
		`- Bump various dependencies`
		`## What's Changed`
		`* Update Java 1.8 to jre8u352`
		`## Known Issues`

Component	Version	Release Notes
		- This release breaks compatibility with [postgres-release](https://github.com/cloudfoundry/postgres-release) (and possibly more postgres server) with a TLS handshake error. Do not use this release if you are using postgres. The fix is underway. **Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.13...2.12.14 ▼ 2.12.13 ### Security Fixes - Bump various dependencies ## What's Changed * Improve pre-start script error log in https://github.com/pivotal/credhub-release/pull/72 **Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.12...2.12.13
diego	2.70.0	

Component	Version	Release Notes
dotnet-core-offline-buildpack	2.4.4	▼ 2.4.4

```
  * Add dotnet-sdk 7.0.100
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183841988)
  * Add dotnet-aspnetcore 7.0.0
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183841985)
  * Add dotnet-runtime 7.0.0
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183841983)
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | bower | 1.8.14 | cflinuxfs3, cflinuxfs4 |
  | dotnet-aspnetcore | 3.1.31 | cflinuxfs3 |
  | dotnet-aspnetcore | 6.0.11 | cflinuxfs3, cflinuxfs4 |
  | dotnet-aspnetcore | 7.0.0 | cflinuxfs3, cflinuxfs4 |
  | dotnet-runtime | 3.1.31 | cflinuxfs3 |
  | dotnet-runtime | 6.0.11 | cflinuxfs3, cflinuxfs4 |
  | dotnet-runtime | 7.0.0 | cflinuxfs3, cflinuxfs4 |
  | dotnet-sdk | 3.1.425 | cflinuxfs3 |
  | dotnet-sdk | 6.0.403 | cflinuxfs3, cflinuxfs4 |
  | dotnet-sdk | 7.0.100 | cflinuxfs3, cflinuxfs4 |
  | libgdiplus | 6.1 | cflinuxfs3 |
  | libgdiplus | 6.1 | cflinuxfs4 |
  | libunwind | 1.6.2 | cflinuxfs3 |
  | libunwind | 1.6.2 | cflinuxfs4 |
  | node | 18.12.1 | cflinuxfs3 |
  | node | 18.12.1 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | dotnet-runtime | 6.0.x |
  | dotnet-aspnetcore | 6.0.x |
  | dotnet-sdk | 6.0.x |
  | bower | 1.8.x |
  * Uncached buildpack SHA256: 07448060b0127c98d9fc2cee2ad4f7da1c8a4d192a4d12276f38b287cc97bdc0
  * Uncached buildpack SHA256: 9f12d944e5f905ac90ce2185e8592b33c9231d99173416360b06c78bfbb5a462
```

▼ 2.4.3

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
  * Add dotnet-aspnetcore 3.1.31, remove dotnet-aspnetcore 3.1.30
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183751987)
  * Add dotnet-runtime 3.1.31, remove dotnet-runtime 3.1.30
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183751988)
  * Add dotnet-sdk 6.0.403, remove dotnet-sdk 6.0.402
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183751455)
  * Add dotnet-aspnetcore 6.0.11, remove dotnet-aspnetcore 6.0.10
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183751561)
  * Add dotnet-runtime 6.0.11, remove dotnet-runtime 6.0.10
  for stack(s) cflinuxfs4, cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183751459)
  * Set node to version 18 (current LTS)
  * Add node 18.12.1, remove node 16.18.0
  for stack(s) cflinuxfs3, cflinuxfs4
  Packaged binaries:
name	version	cf_stacks
bower	1.8.14	cflinuxfs3, cflinuxfs4
dotnet-aspnetcore	3.1.31	cflinuxfs3
dotnet-aspnetcore	6.0.11	cflinuxfs3, cflinuxfs4
dotnet-runtime	3.1.31	cflinuxfs3
dotnet-runtime	6.0.11	cflinuxfs3, cflinuxfs4
dotnet-sdk	3.1.425	cflinuxfs3
dotnet-sdk	6.0.403	cflinuxfs3, cflinuxfs4
libgdiplus	6.1	cflinuxfs3
libgdiplus	6.1	cflinuxfs4
libunwind	1.6.2	cflinuxfs3
libunwind	1.6.2	cflinuxfs4
node	18.12.1	cflinuxfs3
node	18.12.1	cflinuxfs4
Default binary versions:
name	version
-	-
dotnet-runtime	6.0.x
dotnet-aspnetcore	6.0.x
dotnet-sdk	6.0.x
bower	1.8.x
  * Uncached buildpack SHA256: a4822cfc6111a2242b3d121db576724888e1b8de9d211cbf487e1aa332732d7b
</pre></td></tr>
</table>

Component	Version	Release Notes
		`* Uncached buildpack SHA256: f3a7ec75e95a02f4a97100555a730481dd65b1f9887a3ca65d014be3552eebaf`
garden-runc	1.22.5	

Component	Version	Release Notes
go-offline-buildpack	1.10.2	▼ 1.10.2

```
  * Add go 1.19.4, remove go 1.19.3 for stack
(s) cflinuxfs4, cflinuxfs3
  * Add go 1.18.9, remove go 1.18.8 for stack
(s) cflinuxfs4, cflinuxfs3
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | dep | 0.5.4 | cflinuxfs3 |
  | dep | 0.5.4 | cflinuxfs4 |
  | glide | 0.13.3 | cflinuxfs3 |
  | glide | 0.13.3 | cflinuxfs4 |
  | go | 1.18.9 | cflinuxfs3 |
  | go | 1.18.9 | cflinuxfs4 |
  | go | 1.19.4 | cflinuxfs3 |
  | go | 1.19.4 | cflinuxfs4 |
  | godep | 80 | cflinuxfs3 |
  | godep | 80 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
  | go | 1.18.x |
  * Uncached buildpack SHA256: 2058855e3a1a3b
a2d68bb0b44c6d82084b8b00b6354b4350ff1c15ff282
3dcde
  * Uncached buildpack SHA256: d3ac53055023dc
81ff128d134ad0010827f2f4311772dc981957cebd1fe
72a05
```

▼ 1.10.1

```
  * Add go 1.19.3, remove go 1.19.2
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/
183690473)
  * Add go 1.18.8, remove go 1.18.7
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/
183690496)
  Packaged binaries:
  | name | version | cf_stacks |
  |-|-|-|
  | dep | 0.5.4 | cflinuxfs3 |
  | dep | 0.5.4 | cflinuxfs4 |
  | glide | 0.13.3 | cflinuxfs3 |
  | glide | 0.13.3 | cflinuxfs4 |
  | go | 1.18.8 | cflinuxfs3 |
  | go | 1.18.8 | cflinuxfs4 |
  | go | 1.19.3 | cflinuxfs3 |
  | go | 1.19.3 | cflinuxfs4 |
  | godep | 80 | cflinuxfs3 |
  | godep | 80 | cflinuxfs4 |
  Default binary versions:
  | name | version |
  |-|-|
```

Component	Version	Release Notes
		| go | 1.18.x | * Uncached buildpack SHA256: 511f34140c55d72e37e49bf27deab0f92ffc53c163cb063c5d9ee76b8e5ca2f8 * Uncached buildpack SHA256: b343fc1ff8f5f0af887b9fcbdb4851fcd100f50facde4d33337459db0c376eb8
haproxy	11.10.2	
java-offline-buildpack	4.53	
log-cache	2.12.4	▼ v2.12.4 ## What's Changed * update golang release to 1.19.4 * dependency bumps **Full Changelog**: https://github.com/cloudfoundry/log-cache-release/compare/v2.12.3...v2.12.4
loggregator	106.7.5	▼ v106.7.5 ## What's Changed * Dependency bumps * Update golang to 1.19.4 **Full Changelog**: https://github.com/cloudfoundry/loggregator-release/compare/v106.7.4...v106.7.5 ▼ v106.7.4 ## What's Changed * Traffic controller no longer waits for log-cache to be available by @mkocher in https://github.com/cloudfoundry/loggregator-release/pull/482 **Full Changelog**: https://github.com/cloudfoundry/loggregator-release/compare/v106.7.3...v106.7.4

Component	Version	Release Notes
loggregator-agent	6.5.5	▼ v6.5.5 - bump-golang to v0.114.0 for golang 1.19.4 - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/valyala/fasthttp from 1.41.0 to 1.43.0 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src - Bump github.com/prometheus/client_model from 0.2.0 to 0.3.0 in /src - Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1
mapfs	1.2.12	
metric-registrar	1.2.14	▼ v1.2.14 ## What's Changed - Bump to go1.19.3 - Bump dependencies **Full Changelog**: https://github.com/pivotal-cf/metric-registrar-release/compare/v1.2.13...v1.2.14
metrics-discovery	3.2.4	▼ v3.2.4 - bump-golang to v0.114.0 for golang 1.19.4 - Bump github.com/nats-io/nats.go from 1.19.0 to 1.21.0 in /src - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump github.com/prometheus/client_golang from 1.13.1 to 1.14.0 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src - Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1
mysql-monitoring	9.23.0	▼ 9.23.0 Final release 9.23.0 - Build with Go 1.19.4

Component	Version	Release Notes
nats	42	
nfs-volume	7.1.8	
nginx-offline-buildpack	1.1.45	
nodejs-offline-buildpack	1.8.3	
notifications	62	
notifications-ui	40	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>php-offline-buildpack</td><td>4.4.68</td><td>▼ 4.4.68
<pre>
  * Add php 7.4.33, remove php 7.4.30
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183708118)
  * Add php 8.1.12, remove php 8.1.10 (#728)
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183661829)
  * Add php 8.0.25, remove php 8.0.23 (#725)
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183659835)
  * Add composer 2.4.4, remove composer 2.4.2
  for stack(s) cflinuxfs3
  * Add appdynamics 22.10.0-627, remove appdy
namics 22.8.0-588 (#723)
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183626767)
  * Update default Nginx version (#721)
  * Add nginx 1.23.2, remove nginx 1.23.1 (#7
20)
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/
183579482)
  Packaged binaries:
name	version	cf_stacks	modules
CAAPM	20.11.0	cflinuxfs3	
appdynamics	22.10.0-627	cflinuxfs3
composer	2.4.4	cflinuxfs3	
httpd	2.4.54	cflinuxfs3	
newrelic	9.20.0.310	cflinuxfs3	
nginx	1.22.0	cflinuxfs3	
nginx	1.23.2	cflinuxfs3	
php	7.4.32	cflinuxfs3	amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, io
ncube, ldap, lua, lzf, mailparse, maxminddb,
mbstring, memcached, mongodb, msgpack, mysql
i, oauth, opcache, openssl, pcntl, pdo, pdo_f
irebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_
sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredi
s, protobuf, pspell, psr, rdkafka, readline,
redis, shmop, snmp, soap, sockets, sodium, so
lr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sy
svshm, tideways_xhprof, tidy, xdebug, xmlrpc,
xsl, yaf, yaml, zip, zlib
php	7.4.33	cflinuxfs3	amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, io
ncube, ldap, lua, lzf, mailparse, maxminddb,
mbstring, memcached, mongodb, msgpack, mysql
i, oauth, opcache, openssl, pcntl, pdo, pdo_f
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
irebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_
sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredi
s, protobuf, pspell, psr, rdkafka, readline,
redis, shmop, snmp, soap, sockets, sodium, so
lr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sy
svshm, tideways_xhprof, tidy, xdebug, xmlrpc,
xsl, yaf, yaml, zip, zlib |
  | php | 8.0.24 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sql
srv, pgsql, phpiredis, pspell, psr, rdkafka,
readline, redis, shmop, snmp, soap, sockets,
sodium, solr, sqlsrv, ssh2, sysvmsg, sysvsem,
sysvshm, tideways_xhprof, tidy, xdebug, xsl,
yaf, yaml, zip, zlib |
  | php | 8.0.25 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sql
srv, pgsql, phpiredis, pspell, psr, rdkafka,
readline, redis, shmop, snmp, soap, sockets,
sodium, solr, sqlsrv, ssh2, sysvmsg, sysvsem,
sysvshm, tideways_xhprof, tidy, xdebug, xsl,
yaf, yaml, zip, zlib |
  | php | 8.1.11 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pgsql,
phpiredis, pspell, psr, rdkafka, readline, re
dis, shmop, snmp, soap, sockets, sodium, sol
r, ssh2, sysvmsg, sysvsem, sysvshm, tideways_
xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zl
ib |
  | php | 8.1.12 | cflinuxfs3 | amqp, apcu, b
z2, curl, dba, enchant, exif, fileinfo, ftp,
gd, gettext, gmp, igbinary, imagick, imap, ld
ap, lzf, mailparse, maxminddb, mbstring, memc
ached, mongodb, msgpack, mysqli, oauth, opcac
he, openssl, pcntl, pdo, pdo_firebird, pdo_my
sql, pdo_odbc, pdo_pgsql, pdo_sqlite, pgsql,
phpiredis, pspell, psr, rdkafka, readline, re
dis, shmop, snmp, soap, sockets, sodium, sol
r, ssh2, sysvmsg, sysvsem, sysvshm, tideways_
xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zl
ib |
  Default binary versions:
  | name | version |
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>|-|-|
php	7.4.32
httpd	2.4.54
newrelic	9.20.0.310
nginx	1.23.2
composer	2.4.4
CAAPM	20.11.0 | * Uncached buildpack SHA256: 5a4c0e4333e015a0a33464eedecc15a666efb56e958e257d90c5fafe690b07bf</td></tr>
<tr><td>push-apps-manager-release</td><td>675.0.5</td><td></td></tr>
<tr><td>push-usage-service-release</td><td>674.0.29</td><td>▼ v674.0.29
Dependencies
* **app-usage-service:** Updated to `v9035e15`.
For more information, see [app-usage-service](https://github.com/pivotal-cf/app-usage-service).
* **metric-registrar-cli:** Updated to `v1.3.7`.
For more information, see [metric-registrar-cli](https://github.com/pivotal-cf/metric-registrar-cli/releases/tag/1.3.7).</td></tr>
<tr><td>pxc</td><td>0.49.0</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>python-offline-buildpack</td><td>1.8.3</td><td>▼ 1.8.3 * Add python 3.11.0 for stack(s) cflinuxfs3, cflinuxfs4 (https://www.pivotaltracker.com/story/show/183753096) Packaged binaries: | name	version	cf_stacks
libffi	3.2.1	cflinuxfs3, cflinuxfs4
libmemcache	1.0.18	cflinuxfs3, cflinuxfs4
miniconda3-py39	4.12.0	cflinuxfs3, cflinuxfs4
pip	22.3.1	cflinuxfs3, cflinuxfs4
pipenv	2022.11.5	cflinuxfs3
pipenv	2022.11.5	cflinuxfs4
python	3.7.15	cflinuxfs3
python	3.7.15	cflinuxfs4
python	3.8.15	cflinuxfs3
python	3.8.15	cflinuxfs4
python	3.9.15	cflinuxfs3
python	3.9.15	cflinuxfs4
python	3.10.8	cflinuxfs3
python	3.10.8	cflinuxfs4
python	3.11.0	cflinuxfs3
python	3.11.0	cflinuxfs4
setuptools	65.5.1	cflinuxfs3, cflinuxfs4 | Default binary versions: | name	version
-	-
python	3.10.x	 * Uncached buildpack SHA256: a0e3d5872626d07a988b7648d56028ddb67f31cc14b9a937bcab6bb8cbed4249 * Uncached buildpack SHA256: d203d81f9e5f7f672c48e6c65aca697e25b692d96f4da7f3b85ded1c8768e298</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>r-offline-buildpack</td><td>1.1.35</td><td>▼ 1.1.35 * Add r 4.2.2, remove r 4.2.0 for stack(s) cflinuxfs3 with dependencies for stack cflinuxfs3: forecast 8.18, plumber 1.2.1, rserve 1.8.10, shiny 1.7.3 (https://www.pivotaltracker.com/story/show/183673027) * Rebuild r 4.2.1 for stack(s) cflinuxfs3 with dependencies for stack cflinuxfs3: forecast 8.18, plumber 1.2.1, rserve 1.8.10, shiny 1.7.3 (https://www.pivotaltracker.com/story/show/183633457) * Bumps go.mod go version to 1.19 Packaged binaries: | name	version	cf_stacks	modules
r	3.6.2	cflinuxfs3	forecast, plumber, rserve, shiny
r	3.6.3	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.1	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.2	cflinuxfs3	forecast, plumber, rserve, shiny | * Uncached buildpack SHA256: 9f8baeeefe3c1d2dc00f08203670810795541eba1b1904a2b2d59b58c0b44489</td></tr>
</table>

Component	Version	Release Notes
routing	0.251.0	▼ v0.251.0 ## What's Changed - When the `router.ca_certs` property switched from a multi-line string of certs, to an array of certs, gorouter started failing to start up if any of the certs provided were invalid. Previously they were ignored. This has been reverted, so that any invalid CA certs are ignored during startup. Thanks @ameowlia! **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.250.0...v0.251.0 ▼ v0.250.0 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.249.0...v0.250.0 ## Built with go 1.19.4 ▼ v0.249.0 ## What's Changed * Switch to healthecker package in cf-networking-helpers by @mariash in https://github.com/cloudfoundry/routing-release/pull/302 * Add healthchecker package to sync-package-specs file by @mariash in https://github.com/cloudfoundry/routing-release/pull/303 * **Potential Breaking Change:** In preperation for mtls between gorouter and routing api, add gorouter backends ca to routing-api. Rendering these certs depends on routing-api consuming a link from gorouter. If you have multiple gorouter instance groups (for example in the case of isolation segments), you will need to rename bosh links to prevent the error "Multiple link providers found. For an example of link renaming, see [this ops file](https://github.com/cloudfoundry/cf-deployment/blob/main/operations/test/add-persistent-isolation-segment-router.yml#L74) by @reneighbor in https://github.com/cloudfoundry/routing-release/pull/300 * Ensure gorouter-healthchecker doesn't restart gorouter forever on failure by @geofffranks in https://github.com/cloudfoundry/routing-release/pull/305 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.248.0...v0.249.0

Component	Version	Release Notes
		▼ v0.248.0 ## What's Changed * Handle nil ca cert in ca_certs property list ▼ v0.247.0 ## What's Changed * gorouter template cleans `router.ca_certs` property to remove empty certificates ▼ v0.246.0 ## What's Changed * Update `router.ca_certs` property to accept and array of certificates instead of a string block. Thanks @peanball! ▼ v0.245.0 ## What's Changed * Gorouter's pre-start script now reserves ports used by other CF components when it increases the number of ephemeral ports available via `/proc/sys/net/ipv4/ip_local_reserved_ports`. This resolves issues when components fail to start up during deploys/monit restarts due to accidental port collisions with outbound traffic from the VM. Thanks @ameowlia ! * Routing-release no longer makes use of the deprecated uaa-go-client, and uses go-uaa instead * The `routing_utils/nats_client` helper utility now supports saving + loading gorouter's routing tables! Thanks @domdom82 ! * Fixed a memory leak with `gorouter` that resulted in HTTP request objects being held open if a client canceled the connection before the App responded. Thanks @geofffranks ! * **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.244.0...v0.245.0 ## Built with go 1.19.3

<table>
<thead>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
</thead>
<tbody>
<tr><td>ruby-offline-buildpack</td><td>1.8.60</td><td>▼ 1.8.60
<pre>
  * Add bundler 2.3.25, remove bundler 2.3.24
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183700877)
  * Add rubygems 3.3.25, remove rubygems 3.3.24
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183700127)
  * Add node 18.12.1, remove node 16.18.0
  for stack(s) cflinuxfs3
  * Add jruby 9.3.9.0, remove jruby 9.3.8.0
  for stack(s) cflinuxfs3
  (https://www.pivotaltracker.com/story/show/183621295)
  Packaged binaries:
name	version	cf_stacks
bundler	1.17.3	cflinuxfs3
bundler	2.3.25	cflinuxfs3
jruby	9.2.21.0	cflinuxfs3
jruby	9.3.9.0	cflinuxfs3
node	18.12.1	cflinuxfs3
openjdk1.8-latest	1.8.0	cflinuxfs3
ruby	2.7.5	cflinuxfs3
ruby	2.7.6	cflinuxfs3
ruby	3.0.3	cflinuxfs3
ruby	3.0.4	cflinuxfs3
ruby	3.1.1	cflinuxfs3
ruby	3.1.2	cflinuxfs3
rubygems	3.3.25	cflinuxfs3
yarn	1.22.19	cflinuxfs3
Default binary versions:
name	version
-	-
ruby	2.7.x
  * Uncached buildpack SHA256: c5fd8da054202aa26d01516cf2ccb4c04b0f13fcc665abe7b5db3557e2956225
</pre></td></tr>
<tr><td>silk</td><td>3.16.0</td><td></td></tr>
<tr><td>smb-volume</td><td>3.1.6</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.8.1</td><td></td></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.35</td><td></td></tr>
</tbody>
</table>

Component	Version	Release Notes
statsd-injector	1.11.26	▼ v1.11.26 - bump-golang to v0.114.0 for golang 1.19.4 - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src
syslog	11.8.6	▼ v11.8.6 Update golang to 1.19.4 **Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.5...v11.8.6 ▼ v11.8.5 * update dependencies * update golang to 1.19.3 **Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.4...v11.8.5

Component	Version	Release Notes
system-metrics-scraper	3.3.4	▼ v3.3.4 - bump-golang to v0.114.0 - Bump github.com/nats-io/nats.go from 1.20.0 to 1.21.0 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump github.com/nats-io/nats.go from 1.19.1 to 1.20.0 in /src - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src ▼ v3.3.3 ## What's Changed - Bump to go1.19.3 - Bump dependencies **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.3.2...v3.3.3 ▼ v3.3.2 ## What's Changed * limits amount of scraping at a time by @Benjamintf1 @mkocher in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/55 * bump golang to 1.19.2 **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.3.1...v3.3.2 ▼ v3.3.1 ## What's Changed * Bump github.com/onsi/ginkgo/v2 from 2.1.6 to 2.2.0 in /src by @dependabot in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/43 * Bump github.com/hashicorp/go-hclog from 1.3.0 to 1.3.1 in /src by @dependabot in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/44 * Bump github.com/nats-io/nats.go from 1.16.0 to 1.17.0 in /src by @dependabot in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/42 * Bump golang to 1.19.1 by @Benjamintf1 in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/45

Component	Version	Release Notes
		## New Contributors * @rroberts2222 made their first contribution in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/46 **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.3.0...v3.3.1 ▼ v3.3.0 ## What's Changed * Add ReadHeaderTimeouts to all servers by @ctlong in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/35 * Bump dependencies by @dependabot in https://github.com/cloudfoundry/system-metrics-scraper-release/pull/36 * Bump to golang 1.18.6 **Full Changelog**: https://github.com/cloudfoundry/system-metrics-scraper-release/compare/v3.2.8...v3.3.0

Component	Version	Release Notes
uaa	74.5.60	▼ v74.5.60 ## What's Changed * Bump github.com/cloudfoundry/bosh-utils from 0.0.342 to 0.0.343 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/451 * Various dependency bumps in server source code. **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.59...v74.5.60 ▼ v74.5.59 ## What's Changed * Bump versions.tomcatVersion from 9.0.68 to 9.0.69 by @dependabot in https://github.com/cloudfoundry/uaa/pull/2098 * Bump zeitwerk from 2.6.5 to 2.6.6 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/449 * Bump github.com/cloudfoundry/bosh-utils from 0.0.341 to 0.0.342 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/448 * Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/447 * Upgrade Newrelic to [version 7.11.1](https://github.com/cloudfoundry/uaa-release/commit/39d50dff8e0618f1a78d23d7973bf3d81bbca39d) **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.58...v74.5.59 ▼ v74.5.58 ## What's Changed * Bump zeitwerk from 2.6.1 to 2.6.5 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/445 * Bump github.com/onsi/gomega from 1.23.0 to 1.24.0 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/444 **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.57...v74.5.58

v2.12.19

Release Date: 11/10/2022

- **[Security Fix]** Bump of nodejs-offline-buildpack to version 4.8.3 fully addresses CVE-2022-3602 and CVE-2022-3786 in OpenSSL 3.x; impacted versions of OpenSSL are otherwise not used in this version line of TAS.
- **[Feature]** Add “Max request header size in kb” property to Networking tab to allow operators to specify a limit on the aggregate size of request headers. Requests over this limit receive a 431 status code.
- **[Bug Fix]** Bump `java_buildpack_offline` to version `4.53` to fix the known issue relating to Spring Auto Reconfiguration. This is once again enabled by default. For more information, see java-offline-buildpack v4.52 Disallows Spring Auto Reconfiguration by Default below.
- Bump backup-and-restore-sdk to version `1.18.57`
- Bump binary-offline-buildpack to version `1.0.47`
- Bump bosh-system-metrics-forwarder to version `0.0.27`
- Bump cf-autoscaling to version `249.0.23`
- Bump cf-networking to version `3.14.0`
- Bump cflinuxfs3 to version `0.332.0`
- Bump credhub to version `2.12.12`
- Bump dotnet-core-offline-buildpack to version `2.4.2`
- Bump garden-runc to version `1.22.5`
- Bump go-offline-buildpack to version `1.10.0`
- Bump java-offline-buildpack to version `4.53`
- Bump log-cache to version `2.12.3`
- Bump loggregator to version `106.7.3`
- Bump loggregator-agent to version `6.5.4`
- Bump mapfs to version `1.2.12`
- Bump metric-registrar to version `1.2.13`
- Bump metrics-discovery to version `3.2.3`
- Bump nginx-offline-buildpack to version `1.1.45`
- Bump nodejs-offline-buildpack to version `1.8.3`
- Bump php-offline-buildpack to version `4.4.67`
- Bump pxc to version `0.47.0`
- Bump r-offline-buildpack to version `1.1.34`
- Bump routing to version `0.244.0`
- Bump ruby-offline-buildpack to version `1.8.59`

- Bump smb-volume to version `3.1.6`
- Bump smoke-tests to version `4.8.1`
- Bump staticfile-offline-buildpack to version `1.5.35`
- Bump statsd-injector to version `1.11.25`
- Bump syslog to version `11.8.4`
- Bump uaa to version `74.5.57`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.305	

Component	Version	Release Notes

backup-and-restore-sdk	1.18.57	▼ v1.18.57
		`## Changes`
		`* Add a script that the autobumper task can use for ncurses (#739)`
		`* Prep for a gomega upgrade (#743)`
		`* Support mysql-8.0 on ubuntu-xenial stemcells (#738)`
		`* Update vendored package golang-1-linux (#745)`
		`* Wire in the ncurses autobumper from PR 739 (#740)`
		`* chore(dockerize-release): compile BOSH packages concurrently (#741)`
		`## Dependencies`
		`* **gomega:** Updated to v1.23.0.`
		`For more information, see [gomega](https://github.com/onsi/gomega).`
		`* **api:** Updated to v0.101.0.`
		`For more information, see [api](https://github.com/googleapis/google-api-go-client).`
		`* **rspec:** Updated to v3.12.0.`
		`For more information, see [rspec](https://github.com/rspec/rspec-metagem).`
		▼ v1.18.55
		`## Changes`
		`* Bump mysql from 5.7.38 to 5.7.39 (#722)`
		`## Dependencies`
		`* **gomega:** Updated to v1.22.1.`
		`For more information, see [gomega](https://github.com/onsi/gomega).`
		▼ v1.18.54
		`## Changes`
		`* Add a unit tests workflow for sdk-release-tests (#714)`
		`* Add airgap tests (#704)`
		`* Bump mariadb from 10.6.9 to 10.6.10 (#711)`
		`* Update vendored package golang-1-linux (#718)`
		`## Dependencies`
		`* **storage:** Updated to v1.27.0.`
		`For more information, see [storage](https://github.com/googleapis/google-cloud-go).`
		`* **api:** Updated to v0.98.0.`
		`For more information, see [api](https://github.com/googleapis/google-api-go-client).`
		▼ v1.18.53

Component	Version	Release Notes
		## Changes * Update vendored package golang-1-linux (#677) ## Dependencies * **bosh-backup-and-restore:** Updated to v1.9.38. For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore). * **pq:** Updated to v1.10.7. For more information, see [pq](https://github.com/lib/pq). * **api:** Updated to v0.96.0. For more information, see [api](https://github.com/googleapis/google-api-go-client). ▼ v1.18.52 ## Changes * Add final release 1.18.51 [ci skip] * Make the mysql8 packaging script work in airgapped environments (#703) ▼ v1.18.51 ## Changes * Add final release 1.18.50 [ci skip] * Feature/mysql 8 support (#702) ## Dependencies * **storage:** Updated to v1.26.0. For more information, see [storage](https://github.com/googleapis/google-cloud-go). * **gomega:** Updated to v1.20.2. For more information, see [gomega](https://github.com/onsi/gomega).
binary-offline-buildpack	1.0.47	▼ 1.0.47 * Update libbuildpack * Uncached buildpack SHA256: 4dd3ba6d0802797fd8a27f2624660aa877d7d6990c977e37f6391e6592376ca8 * Uncached buildpack SHA256: f7cec524cf452026823432b755e3acdffc39a0f498272d0be8bddec00ec16d67 * Uncached buildpack SHA256: c65dae7aaa1e4744fcac88d48f6aac3c85012cb6f42b671798a7a50300e6a333
bosh-dns-aliases	0.0.4	

Component	Version	Release Notes
bosh-system-metrics-forwarder	0.0.27	
bpm	1.1.19	
capi	1.117.9	
cf-autoscaling	249.0.23	
cf-cli	1.41.0	
cf-networking	3.14.0	
cflinuxfs3	0.332.0	

credhub	2.12.12	▼ 2.12.12
		`### Fixes`
		`- Handles Postgres queries when there are more than 65535 certificates that the user has permission to access.`
		`### Security Fixes`
		`- Bump various dependencies`
		`**Full Changelog**: https://github.com/pivotal/credhub-release/compare/2.12.11...2.12.12`
		▼ 2.12.11
		`### Security Fixes`
		`- Bump various dependencies`

diego	2.69.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>dotnet-core-offline-buildpack</td><td>2.4.2</td><td>▼ 2.4.2 * Add node 16.18.0, remove node 16.17.0 for stack(s) cflinuxfs3, cflinuxfs4 * Add dotnet-sdk 3.1.424, remove dotnet-sdk 3.1.423 for stack(s) cflinuxfs3 * Add dotnet-runtime 3.1.30, remove dotnet-runtime 3.1.29 for stack(s) cflinuxfs3 * Add dotnet-aspnetcore 3.1.30, remove dotnet-aspnetcore 3.1.29 for stack(s) cflinuxfs3 * Add dotnet-sdk 6.0.402, remove dotnet-sdk 6.0.401 for stack(s) cflinuxfs4, cflinuxfs3 * Add dotnet-runtime 6.0.10, remove dotnet-runtime 6.0.9 for stack(s) cflinuxfs4, cflinuxfs3 * Add dotnet-aspnetcore 6.0.10, remove dotnet-aspnetcore 6.0.9 for stack(s) cflinuxfs4, cflinuxfs3 Packaged binaries: | name	version	cf_stacks
bower	1.8.14	cflinuxfs3, cflinuxfs4
dotnet-aspnetcore	3.1.30	cflinuxfs3
dotnet-aspnetcore	6.0.10	cflinuxfs3, cflinuxfs4
dotnet-runtime	3.1.30	cflinuxfs3
dotnet-runtime	6.0.10	cflinuxfs3, cflinuxfs4
dotnet-sdk	3.1.424	cflinuxfs3
dotnet-sdk	6.0.402	cflinuxfs3, cflinuxfs4
libgdiplus	6.1	cflinuxfs3
libgdiplus	6.1	cflinuxfs4
libunwind	1.6.2	cflinuxfs3
libunwind	1.6.2	cflinuxfs4
node	16.18.0	cflinuxfs3
node	16.18.0	cflinuxfs4 | Default binary versions: | name	version
-	-
dotnet-runtime	6.0.x
dotnet-aspnetcore	6.0.x
dotnet-sdk	6.0.x
bower	1.8.x	 * Uncached buildpack SHA256: 6d39ad86c292780ffb933b17b9ea2b3dbf8dfc8a4051e6b8eee4a287a48fad89 * Uncached buildpack SHA256: 19e73da01d2aa75447f8e77486222601fe0d88c92f3661e9646151c9ba49e814</td></tr>
<tr><td>garden-runc</td><td>1.22.5</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>go-offline-buildpack</td><td>1.10.0</td><td>▼ 1.10.0
<pre>
  * Add support for cflinuxfs4 stack
  * Add go 1.19.2, remove go 1.19 for stack(s) cflinuxfs4, cflinuxfs3
  * Update Go Buildpack dependencies to only keep 1 of each patch version (latest)
  Packaged binaries:
name	version	cf_stacks
dep	0.5.4	cflinuxfs3
dep	0.5.4	cflinuxfs4
glide	0.13.3	cflinuxfs3
glide	0.13.3	cflinuxfs4
go	1.18.7	cflinuxfs3
go	1.18.7	cflinuxfs4
go	1.19.2	cflinuxfs3
go	1.19.2	cflinuxfs4
godep	80	cflinuxfs3
godep	80	cflinuxfs4
Default binary versions:
name	version
-	-
go	1.18.x
  * Uncached buildpack SHA256: 35f3b6b61c19104db056610fea83cc19b3c04788aec3772be4c0c9d4a46fc50f
  * Uncached buildpack SHA256: 802aa0da6b655ad5f05ccbba452d609a77a61a7daf4d7a05c1ce493a90de8256
</pre></td></tr>
<tr><td>haproxy</td><td>11.10.2</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>java-offline-buildpack</td><td>4.53</td><td>▼ 4.53 I'm pleased to announce the release of the java-buildpack, version 4.53. It primarily includes new OpenJDK versions, which are based on the Oracle Java Quarterly Updates for Oct 2022. - This release also includes an enhancement to the Sealights framework, more field are supported for a User-Provided service (Thanks to @alonweiss-sl via #964) ## Deprecation of Spring Cloud Connectors & Spring Auto Reconfiguration This release reverts the change made in v4.51 which disabled the Spring Auto Reconfiguration framework by default. From this release, it will be enabled by default as per all versions < 4.51. This is to provide users with more time to migrate to the recommended alternative library, `java-cfenv`. The default of disabled will now happen in a release after March 2023, and the library will be completely removed in a release after March 2024. As before, you may [post feedback/comments to this issue](https://github.com/cloudfoundry/java-buildpack/issues/951). For a more detailed look at the changes in 4.53, please take a look at the [commit log](https://github.com/cloudfoundry/java-buildpack/compare/v4.52...v4.53). The packaged version of the buildpack, suitable for use with create-buildpack and update-buildpack, can be found attached to this release. | Dependency	Version	CVEs	Release Notes
AppDynamics Agent	`22.9.1_34265`		[Release Notes](https://docs.appdynamics.com/4.5.x/en/product-and-release-announcements/release-notes/language-agent-notes/java-agent-notes)
Azure Application Insights Agent	`2.6.2`		[Release Notes](https://github.com/Microsoft/ApplicationInsights-Java/releases)
CA Introscope APM Framework	`22.8.0_48`		
Client Certificate Mapper	`1.11.0_RELEASE`	Included inline above	Included inline above
Container Security Provider	`1.19.0_RELEASE`	Included inline above	Included inline above
Contrast Security Agent	`3.18.1`		[Release Notes](https://docs.contrastsecurity.com/en/java-agent-release-notes.html)
Datadog APM Javaagent	`0.114.0`		[Re</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>lease Notes](https://github.com/DataDog/dd-tr
ace-java/releases) |
  | Elastic APM Agent | `1.34.1` | | [Release
Notes](https://www.elastic.co/guide/en/apm/ag
ent/java/current/release-notes.html) |
  | Gemalto Luna Security Provider | `7.4.0`
| | [Release Notes](https://www.thalesdocs.co
m/gphsm/luna/7/docs/network/Content/CRN/Luna/
CRN_Luna.htm) |
  | Geode Tomcat Session Store | `1.12.5` | |
|
  | Google Stackdriver Debugger | `2.32.0` |
| [Release Notes](https://cloud.google.com/de
bugger/docs/release-notes) |
  | Google Stackdriver Profiler | `0.1.0` | |
[Release Notes](https://cloud.google.com/prof
iler/docs/release-notes) |
  | Groovy | `2.5.19` | | [Release Notes](htt
p://www.groovy-lang.org/releases.html) |
  | JaCoCo Agent | `0.8.8` | | [Release Note
s](https://github.com/jacoco/jacoco/releases)
|
  | Java Memory Assistant Agent | `0.5.0` | |
|
  | Java Memory Assistant Clean Up | `0.1.0`
| | |
  | JProfiler Profiler | `12.0.4` | | [Change
Log](https://www.ej-technologies.com/downloa
d/jprofiler/changelog.html) |
  | JRebel Agent | `2022.4.1` | | [ChangeLog]
(https://www.jrebel.com/products/jrebel/chang
elog) |
  | jvmkill Agent | `1.17.0_RELEASE` |Include
d inline above | Included inline above |
  | MariaDB JDBC Driver | `2.7.2` | | [Releas
e Notes](https://mariadb.com/kb/en/mariadb-co
nnector-j-274-release-notes/) |
  | Memory Calculator | `3.13.0_RELEASE` |Inc
luded inline above | Included inline above |
  | Metric Writer | `3.5.0_RELEASE` |Included
inline above | Included inline above |
  | New Relic Agent | `7.11.0` | | [Release N
otes](https://docs.newrelic.com/docs/release-
notes/agent-release-notes/java-release-note
s/) |
  | OpenJDK JRE 11 | `11.0.17_7` |[Risk Matri
x](https://www.oracle.com/security-alerts/cpu
oct2022.html#AppendixJAVA) | [Release Notes]
(https://bell-sw.com/pages/liberica-release-n
otes-11.0.17/) |
  | OpenJDK JRE 17 | `17.0.5_8` |[Risk Matri
x](https://www.oracle.com/security-alerts/cpu
oct2022.html#AppendixJAVA) | [Release Notes]
(https://bell-sw.com/pages/liberica-release-n
otes-17.0.5/) |
  | OpenJDK JRE 8 | `1.8.0_352` |[Risk Matri
x](https://www.oracle.com/security-alerts/cpu</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>

```
oct2022.html#AppendixJAVA) | [Release Notes](https://bell-sw.com/pages/liberica-release-notes-8u352/) |
  | PostgreSQL JDBC Driver | `42.5.0` | | [ChangeLog](https://jdbc.postgresql.org/documentation/changelog.html) |
  | Redis Session Store | `1.3.6_RELEASE` |Included inline above | Included inline above |
  | Riverbed Appinternals Agent | `11.8.5_BL527` | |  |
  | SeaLights Agent | `4.0.2145` | |  |
  | SkyWalking | `8.9.0` | | [ChangeLog](https://github.com/apache/skywalking/tree/master/changes) |
  | Spring Auto-reconfiguration | `2.12.0_RELEASE` |Included inline above | Included inline above |
  | Spring Boot CLI | `2.7.5` | |  |
  | Spring Boot Container Customizer | `2.6.0_RELEASE` |Included inline above | Included inline above |
  | Tomcat | `9.0.68` |[Security](https://tomcat.apache.org/security-9.html) | [ChangeLog](https://tomcat.apache.org/tomcat-9.0-doc/changelog.html) |
  | Tomcat Access Logging Support | `3.3.0_RELEASE` |Included inline above | Included inline above |
  | Tomcat Lifecycle Support | `3.3.0_RELEASE` |Included inline above | Included inline above |
  | Tomcat Logging Support | `3.3.0_RELEASE` |Included inline above | Included inline above |
  | YourKit Profiler | `2022.9.171` | | [Release Notes](https://www.yourkit.com/download/yjp_2022_3_builds.jsp) |
```

</td></tr>
<tr><td>log-cache</td><td>2.12.3</td><td></td></tr>
<tr><td>loggregator</td><td>106.7.3</td><td></td></tr>
<tr><td>loggregator-agent</td><td>6.5.4</td><td></td></tr>
</table>

Component	Version	Release Notes
mapfs	1.2.12	▼ v1.2.12 ## Changes * Replace `go get` with `go install` (#23) * Update vendored package golang-1-linux (#26) * Update vendored package golang-1-linux (#27) ## Dependencies * **mapfs:** Updated to v`27f8711`. For more information, see [mapfs](https://github.com/cloudfoundry/mapfs).
metric-registrar	1.2.13	
metrics-discovery	3.2.3	
mysql-monitoring	9.22.0	
nats	42	
nfs-volume	7.1.3	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nginx-offline-buildpack</td><td>1.1.45</td><td>▼ 1.1.45 * Add nginx 1.23.2, remove nginx 1.23.1 for stack(s) cflinuxfs3 * Add nginx 1.22.1, remove nginx 1.22.0 for stack(s) cflinuxfs3 Packaged binaries: | name	version	cf_stacks
nginx	1.22.1	cflinuxfs3
nginx	1.23.2	cflinuxfs3
openresty	1.13.6.2	cflinuxfs3
openresty	1.15.8.3	cflinuxfs3
openresty	1.17.8.2	cflinuxfs3
openresty	1.19.9.1	cflinuxfs3
openresty	1.21.4.1	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x | * Uncached buildpack SHA256: 59553ae241975530230e065f996a0434b7a85df061b3d79ec8f3073f3701c720 ▼ 1.1.44 * Update libbuildpack Packaged binaries: | name	version	cf_stacks
nginx	1.22.0	cflinuxfs3
nginx	1.23.1	cflinuxfs3
openresty	1.13.6.2	cflinuxfs3
openresty	1.15.8.3	cflinuxfs3
openresty	1.17.8.2	cflinuxfs3
openresty	1.19.9.1	cflinuxfs3
openresty	1.21.4.1	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x	 * Uncached buildpack SHA256: 29cc1a2aea42cfb654f31ee6e0c7bdb14521679fab127b26fa396b055b9c8033</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>nodejs-offline-buildpack</td><td>1.8.3</td><td>▼ 1.8.3
<pre>
  * Add node 18.12.1, remove node 18.9.0
  for stack(s) cflinuxfs3, cflinuxfs4
  * Add node 14.21.1, remove node 14.20.0
  for stack(s) cflinuxfs3, cflinuxfs4
  * Add node 16.18.1, remove node 16.17.1
  for stack(s) cflinuxfs3, cflinuxfs4
  (https://www.pivotaltracker.com/story/show/183724805)
  * Bumps go.mod go version to 1.19
  Packaged binaries:
name	version	cf_stacks
node	14.20.1	cflinuxfs3
node	14.20.1	cflinuxfs4
node	14.21.1	cflinuxfs3
node	14.21.1	cflinuxfs4
node	16.18.0	cflinuxfs3
node	16.18.0	cflinuxfs4
node	16.18.1	cflinuxfs3
node	16.18.1	cflinuxfs4
node	18.10.0	cflinuxfs3
node	18.10.0	cflinuxfs4
node	18.12.1	cflinuxfs3
node	18.12.1	cflinuxfs4
yarn	1.22.19	cflinuxfs3, cflinuxfs4
Default binary versions:
name	version
-	-
node	16.x
  * Uncached buildpack SHA256: 2891efd8d931f6e7c10253bc116cb90544a086a47966532eee13af60414f1ede
  * Uncached buildpack SHA256: cf64d739fafe8a94abff4573cccabf1873e96b923c6f744de14c2e5e106f5903
</pre>
▼ 1.8.2
<pre>
  * Add node 16.18.0, remove node 16.17.0 for stack(s) cflinuxfs4, cflinuxfs3
  * Adding error message when npm install fails (#522)
  Packaged binaries:
name	version	cf_stacks
node	14.20.0	cflinuxfs3
node	14.20.0	cflinuxfs4
node	14.20.1	cflinuxfs3
node	14.20.1	cflinuxfs4
node	16.17.1	cflinuxfs3
node	16.17.1	cflinuxfs4
node	16.18.0	cflinuxfs4
node	16.18.0	cflinuxfs3
node	18.9.0	cflinuxfs3
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>| node | 18.9.0 | cflinuxfs4 |
| node | 18.10.0 | cflinuxfs3 |
| node | 18.10.0 | cflinuxfs4 |
yarn	1.22.19	cflinuxfs3, cflinuxfs4 | Default binary versions: | name	version
node	16.x | * Uncached buildpack SHA256: fa3ba16a26ff03da9312e4717bbcd4640a8bf1d1d275193980075209a2c4cd62 * Uncached buildpack SHA256: 65afb62e378e5d9ba02d1ca6184aca3fcc901d772464e4e254594b6ee7342136 ▼ 1.8.0 * Add support for cflinuxfs4 stack Packaged binaries: | name	version	cf_stacks
-	-	-
node	14.19.3	cflinuxfs3
node	14.20.0	cflinuxfs3
node	14.20.0	cflinuxfs4
node	16.16.0	cflinuxfs3
node	16.17.0	cflinuxfs3
node	16.17.0	cflinuxfs4
node	18.7.0	cflinuxfs3
node	18.9.0	cflinuxfs3
node	18.9.0	cflinuxfs4
yarn	1.22.19	cflinuxfs3, cflinuxfs4 | Default binary versions: | name	version
node	16.x | * Uncached buildpack SHA256: 9c916f1475c3ebfa8cb1531a31a6b70829b29e36380d3496dd105c2d8109c5de * Uncached buildpack SHA256: 7b59d44895ad526dd3f87bf176a5fd7a08f55f0cf4f91dcb8b2c1fa474292385</td></tr>
<tr><td>notifications</td><td>62</td><td></td></tr>
<tr><td>notifications-ui</td><td>40</td><td></td></tr>
</table>

Component	Version	Release Notes
php-offline-buildpack	4.4.67	▼ 4.4.67

```
  * Add php 7.4.32, remove php 7.4.29 for stack(s) cflinuxfs3
  * Add php 8.1.11, remove php 8.1.9 for stack(s) cflinuxfs3
  * Add php 8.0.24, remove php 8.0.22 for stack(s) cflinuxfs3
  * Add composer 2.4.2, remove composer 2.4.1 for stack(s) cflinuxfs3
  Packaged binaries:
  | name | version | cf_stacks | modules |
  |-|-|-|-|
  | CAAPM | 20.11.0 | cflinuxfs3 |  |
  | appdynamics | 22.8.0-588 | cflinuxfs3 |  |
  | composer | 2.4.2 | cflinuxfs3 |  |
  | httpd | 2.4.54 | cflinuxfs3 |  |
  | newrelic | 9.20.0.310 | cflinuxfs3 |  |
  | nginx | 1.22.0 | cflinuxfs3 |  |
  | nginx | 1.23.1 | cflinuxfs3 |  |
  | php | 7.4.30 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lua, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, protobuf, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xmlrpc, xsl, yaf, yaml, zip, zlib |
  | php | 7.4.32 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ioncube, ldap, lua, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phalcon, phpiredis, protobuf, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, stomp, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xmlrpc, xsl, yaf, yaml, zip, zlib |
  | php | 8.0.23 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, sysvmsg, sysvsem,
```

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
  | php | 8.0.24 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pdo_sqlsrv, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, sqlsrv, ssh2, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
  | php | 8.1.10 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, ssh2, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
  | php | 8.1.11 | cflinuxfs3 | amqp, apcu, bz2, curl, dba, enchant, exif, fileinfo, ftp, gd, gettext, gmp, igbinary, imagick, imap, ldap, lzf, mailparse, maxminddb, mbstring, memcached, mongodb, msgpack, mysqli, oauth, opcache, openssl, pcntl, pdo, pdo_firebird, pdo_mysql, pdo_odbc, pdo_pgsql, pdo_sqlite, pgsql, phpiredis, pspell, psr, rdkafka, readline, redis, shmop, snmp, soap, sockets, sodium, solr, ssh2, sysvmsg, sysvsem, sysvshm, tideways_xhprof, tidy, xdebug, xsl, yaf, yaml, zip, zlib |
  Default binary versions:
  | name | version |
  |-|-|
  | php | 7.4.32 |
  | httpd | 2.4.54 |
  | newrelic | 9.20.0.310 |
  | nginx | 1.23.1 |
  | composer | 2.4.2 |
  | CAAPM | 20.11.0 |
  * Uncached buildpack SHA256: 637cbc9daa642ce3d3f42377ea2d0555d3db0959e9fedd81fa1a2fc0e2097000</td></tr>
<tr><td>push-apps-manager-release</td><td>675.0.5</td><td></td></tr>
<tr><td>push-usage-service-release</td><td>674.0.25</td><td></td></tr>
<tr><td>pxc</td><td>0.47.0</td><td></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>python-offline-buildpack</td><td>1.8.0</td><td></td></tr>
<tr><td>r-offline-buildpack</td><td>1.1.34</td><td>▼ 1.1.34
<pre>
  * Update r 4.2.1 dependencies for stack(s) cflinuxfs3:
  * forecast from 8.17.0 to 8.18.0
  Packaged binaries:
name	version	cf_stacks	modules
r	3.6.2	cflinuxfs3	forecast, plumber, rserve, shiny
r	3.6.3	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.0	cflinuxfs3	forecast, plumber, rserve, shiny
r	4.2.1	cflinuxfs3	forecast, plumber, rserve, shiny
  * Uncached buildpack SHA256: f93a3cbfb21e159a96c84c61dec8911ea7c4337fe664da2dcc70ca5efe7ed13d
</pre></td></tr>
<tr><td>routing</td><td>0.244.0</td><td>▼ v0.244.0
<pre>
  ## What's Changed
  * Emit access logs for 431 responses to Loggegator [gorouter PR #331](https://github.com/cloudfoundry/gorouter/pull/331). Thanks @dsabeti !
  * Always suspend pruning when nats is down https://github.com/cloudfoundry/routing-release/pull/287. Thanks @ameowlia !
  * **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.243.0...v0.244.0
  ##   Built with go 1.19.2
</pre>
▼ v0.243.0
<pre>
    Bumped to go1.19.2
  Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.242.0...v0.243.0
</pre></td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>ruby-offline-buildpack</td><td>1.8.59</td><td>▼ 1.8.59
<pre>
  * Add bundler 2.3.24, remove bundler 2.3.22 for stack(s) cflinuxfs3
  * Add jruby 9.3.8.0, remove jruby 9.3.7.0 for stack(s) cflinuxfs3
  * Add node 16.18.0, remove node 16.16.0 for stack(s) cflinuxfs3
  * Add rubygems 3.3.24, remove rubygems 3.3.22 for stack(s) cflinuxfs3
  Packaged binaries:
name	version	cf_stacks
bundler	1.17.3	cflinuxfs3
bundler	2.3.24	cflinuxfs3
jruby	9.2.21.0	cflinuxfs3
jruby	9.3.8.0	cflinuxfs3
node	16.18.0	cflinuxfs3
openjdk1.8-latest	1.8.0	cflinuxfs3
ruby	2.7.5	cflinuxfs3
ruby	2.7.6	cflinuxfs3
ruby	3.0.3	cflinuxfs3
ruby	3.0.4	cflinuxfs3
ruby	3.1.1	cflinuxfs3
ruby	3.1.2	cflinuxfs3
rubygems	3.3.24	cflinuxfs3
yarn	1.22.19	cflinuxfs3
Default binary versions:
name	version
-	-
ruby	2.7.x
  * Uncached buildpack SHA256: 30c6e52afd35ea30bc2fdf6a105dda864928bb7e8a68b7b2341e89bc9cb83e3c
</pre></td></tr>
<tr><td>silk</td><td>3.14.0</td><td></td></tr>
</table>

Component	Version	Release Notes

smb-volume	3.1.6	▼ v3.1.6
		`## Changes`
		`* Update vendored package golang-1-linux (#67)`
		`* Update vendored package golang-1-linux (#70)`
		`## Dependencies`
		`* **bosh-template:** Updated to v2.3.0. For more information, see [bosh-template](https://github.com/cloudfoundry/bosh).`
		`* **smbbroker:** Updated to v`89a0251`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker).`
		`* **smbdriver:** Updated to v`68ff9d8`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).`

smoke-tests	4.8.1	▼ 4.8.1
		`Create bosh final release 4.8.1`
		▼ 4.8.0
		`Create bosh final release 4.8.0`

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.35</td><td>▼ 1.5.35 * Add nginx 1.23.2, remove nginx 1.23.1 for stack(s) cflinuxfs3 * Add nginx 1.22.1, remove nginx 1.22.0 for stack(s) cflinuxfs3 Packaged binaries: | name	version	cf_stacks
nginx	1.22.1	cflinuxfs3
nginx	1.23.2	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x | * Uncached buildpack SHA256: db23d06c3be3ac43c9dca5bd988ee1b6097d5896afa5309d684b446430ae65ab ▼ 1.5.34 * Update libbuildpack Packaged binaries: | name	version	cf_stacks
nginx	1.22.0	cflinuxfs3
nginx	1.23.1	cflinuxfs3 | Default binary versions: | name	version
-	-
nginx	1.23.x	 * Uncached buildpack SHA256: 89420cdafed63dc8dae495c6fce445fb0c5262ad6d149a90cd0e32aa58d3c408</td></tr>
<tr><td>statsd-injector</td><td>1.11.25</td><td></td></tr>
<tr><td>syslog</td><td>11.8.4</td><td></td></tr>
<tr><td>system-metrics-scraper</td><td>3.2.8</td><td></td></tr>
</table>

Component	Version	Release Notes
uaa	74.5.57	▼ v74.5.57 ### Dependency bumps * [Upgrade Newrelic to version 7.11.0](https://github.com/cloudfoundry/uaa-release/commit/50df349712142ab93b937dd52a5041c58a772f65) * Bump rspec-mocks from 3.11.1 to 3.11.2 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/436 * Bump rspec from 3.11.0 to 3.12.0 by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/437 * Bump github.com/onsi/gomega from 1.22.1 to 1.23.0 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/438 * Bump github.com/cloudfoundry/bosh-utils from 0.0.340 to 0.0.341 in /src/acceptance_tests by @dependabot in https://github.com/cloudfoundry/uaa-release/pull/439 **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.56...v74.5.57 ▼ v74.5.56 ### Dependency bumps - Various dependency bumps - Replace jaxb implementation library from Sun's to Glassfish's ▼ v74.5.55 ### Fixes - Fixes an issue where UAA server might crash (error: `java.io.IOException: Too many open files`) because the hung connections to an external LDAP server are not cleaned up. ### Dependency bumps - Various dependency bumps **Full Changelog**: https://github.com/cloudfoundry/uaa-release/compare/v74.5.54...v74.5.55 ▼ v74.5.54 ### Dependency bumps - Bump Tomcat to version 9.0.68

v2.12.18

Release Date: 10/19/2022

- **[Breaking Change]** A change in behavior to line-based log rate limits has been made. Previously, logs were buffered and released at the allowed rate, now logs exceeding the limit will be dropped.
- **[Feature]** Allow operators to specify isolation segments that should be routable through TAS routers
- **[Known Issue]** `java-offline-buildpack` v4.52 makes TAS for VMs incompatible with some service tiles and prevents apps that use `java-offline-buildpack` v4.52 from binding to service instances. For more information, see java-offline-buildpack v4.52 Disallows Spring Auto Reconfiguration by Default below.
- **[Bug Fix]** Fixes UAA's compatibility with Amazon Aurora MySQL
- **[Bug Fix]** Fix Capi compatibility with MySQL 8
- Bump binary-offline-buildpack to version `1.0.46`
- Bump capi to version `1.117.9`
- Bump cf-cli to version `1.41.0`
- Bump cf-networking to version `3.13.0`
- Bump cflinuxfs3 to version `0.328.0`
- Bump credhub to version `2.12.10`
- Bump diego to version `2.69.0`
- Bump dotnet-core-offline-buildpack to version `2.4.1`
- Bump garden-runc to version `1.22.4`
- Bump go-offline-buildpack to version `1.9.50`
- Bump java-offline-buildpack to version `4.52`
- Bump log-cache to version `2.12.1`
- Bump loggregator-agent to version `6.5.1`
- Bump mapfs to version `1.2.11`
- Bump metrics-discovery to version `3.2.1`
- Bump mysql-monitoring to version `9.22.0`
- Bump nfs-volume to version `7.1.3`
- Bump nginx-offline-buildpack to version `1.1.43`
- Bump php-offline-buildpack to version `4.4.66`
- Bump push-usage-service-release to version `674.0.25`
- Bump pxc to version `0.46.0`
- Bump python-offline-buildpack to version `1.8.0`
- Bump r-offline-buildpack to version `1.1.33`
- Bump routing to version `0.242.0`

- Bump ruby-offline-buildpack to version `1.8.58`
- Bump silk to version `3.14.0`
- Bump smb-volume to version `3.1.5`
- Bump smoke-tests to version `4.7.0`
- Bump statsd-injector to version `1.11.23`
- Bump syslog to version `11.8.3`
- Bump uaa to version `74.5.53`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.296	
backup-and-restore-sdk	1.18.50	
binary-offline-buildpack	1.0.46	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.24	
bpm	1.1.19	
capi	1.117.9	
cf-autoscaling	249.0.17	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>cf-cli</td><td>1.41.0</td><td>▼ v1.41.0 ### This release contains the following versions of the CF CLI | Major version	Prior version	Current version
v8	**8.4.0**	[8.5.0](https://github.com/cloudfoundry/cli/releases/tag/v8.5.0)
v7	**7.5.0**	[7.5.0](https://github.com/cloudfoundry/cli/releases/tag/v7.5.0)
v6	**6.53.0**	**[6.53.0](https://github.com/cloudfoundry/cli/releases/tag/v6.53.0)** | ▼ v1.39.0 ### This release contains the following versions of the CF CLI | Major version	Prior version	Current version
v8	**8.3.0**	[8.4.0](https://github.com/cloudfoundry/cli/releases/tag/v8.4.0)
v7	**7.4.0**	[7.5.0](https://github.com/cloudfoundry/cli/releases/tag/v7.5.0)
v6	**6.53.0**	**[6.53.0](https://github.com/cloudfoundry/cli/releases/tag/v6.53.0)** |</td></tr>
<tr><td>cf-networking</td><td>3.13.0</td><td></td></tr>
<tr><td>cflinuxfs3</td><td>0.328.0</td><td></td></tr>
</table>

Component	Version	Release Notes
credhub	2.12.10	▼ 2.12.10 ### Security Fixes - Bump various dependencies - Bump Spring Boot from 2.7.3 to 2.7.4 ▼ 2.12.9 ### Security Fixes - Bump various dependencies ### Enhancements - Improve Find a Credential by Name-Like query performance
diego	2.69.0	
dotnet-core-offline-buildpack	2.4.1	
garden-runc	1.22.4	
go-offline-buildpack	1.9.50	
haproxy	11.10.2	
java-offline-buildpack	4.52	
log-cache	2.12.1	
loggregator	106.6.9	
loggregator-agent	6.5.1	

Component	Version	Release Notes
mapfs	1.2.11	▼ v1.2.11 ## Changes * Update vendored package golang-1-linux (#21) ▼ v1.2.8 ## What's Changed * Bump src/mapfs to `0ee84aa` #18 ▼ v1.2.7 - [Bumps mapfs submodule to master@1600494](https://github.com/cloudfoundry/mapfs/commit/160049400a47577b0f3a8b2948974bc38ce76f18) - [Bump golang from 1.13 to 1.17](https://github.com/cloudfoundry/mapfs-release/commit/c287adda5cbdf345ff1b4985ae93cb72f1618f95)
metric-registrar	1.2.10	
metrics-discovery	3.2.1	
mysql-monitoring	9.22.0	▼ 9.22.0 Final release 9.22.0 - Build w/ go 1.19.2 ▼ v9.21.0 - Build with Go 1.19.1 ▼ v9.19.0 **New Features** The `mysql-metrics` job now supports a `mysql-metrics.port` option to specify the MySQL port when scraping database metrics. This port defaults to 3306 but may be configured to a non-standard port.
nats	42	
nfs-volume	7.1.3	
nginx-offline-buildpack	1.1.43	

Component	Version	Release Notes
nodejs-offline-buildpack	1.7.73	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.66	
push-apps-manager-release	675.0.5	
push-usage-service-release	674.0.25	▼ v674.0.25 ## Changes * Bump src/app-usage-service from [`99b5b7a`](https://github.com/pivotal-cf/app-usage-service/commit/99b5b7aa4e32a14293418f5b5237e54ec74ee07d) to [`aad81af`](https://github.com/pivotal-cf/app-usage-service/commit/aad81af5ada02ff2795e5ac69bba94e7591752ab) * Remove dependabot configs for legacy support branches (https://github.com/pivotal-cf/usage-service-release/pull/252)
pxc	0.46.0	
python-offline-buildpack	1.8.0	
r-offline-buildpack	1.1.33	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>routing</td><td>0.242.0</td><td>▼ v0.242.0
What's Changed
- `tcp_router` is now more verbose when running `haproxy_reloader` to assist in diagnosting failed reloads. Thanks @geofffranks! ([PR 9](https://github.com/cloudfoundry/cf-tcp-router/pull/9))
- `gorouter` will now truncate access logs that exceed loggregator + UDP packet limits, so that we no longer drop access log messages sent to the firehose. Thanks @ameowlia @ebroberson! 😸 ([PR 328](https://github.com/cloudfoundry/gorouter/pull/328) and [PR 329](https://github.com/cloudfoundry/gorouter/pull/329))
Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.241.0...v0.242.0
Built with go 1.19.1

▼ v0.241.0
Bumped to go1.19.1
* @plowin submitted [gorouter PR 327](https://github.com/cloudfoundry/gorouter/pull/327) to adjust endpoint-not-unregistered log-level to 'info'
Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.240.0...v0.241.0

▼ v0.240.0
What's Changed
* @geofffranks and @ameowlia added property `router.max_header_bytes` to the gorouter job.
* This value controls the maximum number of bytes the gorouter will read parsing the request header's keys and values, including the request line.
* It does not limit the size of the request body.
* An additional padding of 4096 bytes is added to this value by go.
* Requests with larger headers will result in a 431 status code.
Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.239.0...v0.240.0
Manifest Property Changes
| Job | Property | 0.237.0 | 0.238.0 |
| --- | --- | --- | --- |</td></tr>
</table>

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>| `gorouter` | `router.max_header_bytes` | didn't exist | 1048576 (1MB) |
Built with go 1.18.6

▼ v0.239.0

What's Changed
- Bumped Golang to 1.18.6 to mitigate [CVE-2022-27664](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27664)
Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.238.0...v0.239.0
Built with go 1.18.6

▼ v0.238.0

What's Changed
- Gorouter once again supports hairpinning for route-service requests, for more information, see [the proposed update.](https://github.com/cloudfoundry/routing-release/issues/281) `router.route_services_internal_lookup_allowlist` can be used to control which domains of route services can be hairpinned. Thanks @peanball!!
- Gorouter has a new websocket-specific dial timeout (`websocket_dial_timeout`), configurable separately from the default endpoint dial timeout. Thanks @peanball for this one too!!
Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.237.0...v0.238.0
Manifest Property Changes
| Job | Property | 0.237.0 | 0.238.0 |
| --- | --- | --- | --- |
| `gorouter` | `websocket_dial_timeout_in_seconds` | didn't exist | Defaults to `endpoint_dial_timeout_in_seconds`'s value |
| `gorouter` | `router.route_services_internal_lookup_allowlist` | didn't exist | No internal lookups allowed for route services. |
Built with go 1.18.5

▼ v0.237.0

What's Changed
- ⚠ Bump to golang 1.18
Breaking Changes: The routing components are now more strict about the protocols used in TLS communications, causing integrations with systems using older, insecure protocols to fail. These components have been updated t</td></tr>
</table>

Component	Version	Release Notes
		o Go 1.18, and will no longer support TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases. Please see this golang 1.18 release notes [section](https://tip.golang.org/doc/go1.18#tls10) for more information about the golang 1.18 change. ### * Update uaa-go-client; by @joergdw in https://github.com/cloudfoundry/routing-release/pull/277 * updated spec files to match packages by @ebroberson in https://github.com/cloudfoundry/routing-release/pull/282 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.236.0...v0.237.0 ## New Contributors * @joergdw made their first contribution in https://github.com/cloudfoundry/routing-release/pull/277 * @ebroberson made their first contribution in https://github.com/cloudfoundry/routing-release/pull/282 ## Built with go 1.18.4
ruby-offline-buildpack	1.8.58	
silk	3.14.0	

Component	Version	Release Notes
smb-volume	3.1.5	▼ v3.1.5 ## Changes * Update vendored package golang-1-linux (#58) ▼ v3.1.4 ## Release Notes - Fix issue when multiple cf versions are included (#55) ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on the errand VM. ▼ v3.1.3 ## Release Notes - Added support for CF CLI v8 to errands (#45) - Fixed Jammy compilation issues (#53) ## Dependencies - Bump [src/code.cloudfoundry.org/smbbroker](https://github.com/cloudfoundry/smbbroker) (#41, #50) - Bump [src/code.cloudfoundry.org/smbdriver](https://github.com/cloudfoundry/smbdriver) (#47, #48, #51) ▼ v3.1.2 ## Release Notes - Support Bionic Stemcell #16 - Add blobs for the `keyutils` package for both `bionic` and `jammy`. - We now install this package on any VM that runs the `smbdriver` bosh job iff that VM uses a `bionic` or `jammy` stemcell - This should allow the `smbdriver` to reliably mount SMB volumes on those stemcells, as discussed in #16 ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on t

Component	Version	Release Notes
		he errand VM. ▼ v3.1.1 ## Release Notes * Bumps [bosh-template](https://github.com/cloudfoundry/bosh) from 2.2.0 to 2.2.1 (#22) * Bumps [rspec-its](https://github.com/rspec/rspec-its) from 1.2.0 to 1.3.0 (#23) * Bumps [rspec](https://github.com/rspec/rspec-metagem) to 3.11.0. (#37) * Bumps [src/code.cloudfoundry.org/smbdriver](https://github.com/cloudfoundry/smbdriver) to `1e97c5d` (#34) * Bumps [src/code.cloudfoundry.org/smbbroker](https://github.com/cloudfoundry/smbbroker) to `64ba567` (#36) * Bumps automake from 1.15 to 1.15.1 (#43 - fixes Bionic compilation) ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on the errand VM.
smoke-tests	4.7.0	▼ 4.7.0 Create bosh final release 4.7.0
staticfile-offline-buildpack	1.5.33	
statsd-injector	1.11.23	
syslog	11.8.3	
system-metrics-scraper	3.2.8	

Component	Version	Release Notes

uaa	74.5.53	▼ v74.5.53
		### Dependency bumps
		- Various dependency bumps
		▼ v74.5.52
		### Dependency bumps
		- Various dependency bumps, including some dependencies with security fixes.
		▼ v74.5.49
		### Features
		- Add `uaa.database.additionalParameters` to set additional parameters in the database connection url. Operators can enable compatibility with Amazon Aurora for MySQL by adding this config:
		```
		uaa:
		database:
		additionalParameters:
		usePipelineAuth: "false"
		useBatchMultiSend: "false"
		```
		### Dependency bumps
		- Various dependency bumps.

v2.12.17

Release Date: 09/21/2022

- **[Security Fix]** Bump Cloud Controller Ruby version to 2.7.6 and Go to 1.18.5
- **[Security Fix]** Update Content-Security-Policy
- **[Feature]** Enables TLS for all internal MySQL galera and monitoring components
- **[Feature Improvement]** Bump golang to 1.18 for diego, routing, cf-networking, and silk
- **[Feature Improvement]** Use the latest version of nats-release
- Bump backup-and-restore-sdk to version `1.18.50`
- Bump bosh-system-metrics-forwarder to version `0.0.24`
- Bump bpm to version `1.1.19`
- Bump capi to version `1.117.8`
- Bump cflinuxfs3 to version `0.319.0`
- Bump credhub to version `2.12.8`

- Bump dotnet-core-offline-buildpack to version `2.4.0`
- Bump garden-runc to version `1.22.0`
- Bump go-offline-buildpack to version `1.9.49`
- Bump log-cache to version `2.11.13`
- Bump loggregator to version `106.6.9`
- Bump loggregator-agent to version `6.4.4`
- Bump metric-registrar to version `1.2.10`
- Bump metrics-discovery to version `3.1.2`
- Bump mysql-monitoring to version `9.18.0`
- Bump nginx-offline-buildpack to version `1.1.42`
- Bump nodejs-offline-buildpack to version `1.7.73`
- Bump php-offline-buildpack to version `4.4.65`
- Bump push-apps-manager-release to version `675.0.5`
- Bump python-offline-buildpack to version `1.7.57`
- Bump r-offline-buildpack to version `1.1.32`
- Bump ruby-offline-buildpack to version `1.8.57`
- Bump silk to version `3.12.0`
- Bump staticfile-offline-buildpack to version `1.5.33`
- Bump statsd-injector to version `1.11.21`
- Bump syslog to version `11.8.2`
- Bump system-metrics-scraper to version `3.2.8`
- Bump uaa to version `74.5.48`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.265	

Component	Version	Release Notes

backup-and-restore-sdk	1.18.50	▼ v1.18.50
		`## Changes`
		`* Add final release 1.18.49 [ci skip]`
		`* Bump mariadb from 10.6.8 to 10.6.9 (#688)`
		`* Bump mysql from 5.7.37 to 5.7.38 (#674)`
		`* Bump postgres from 10.21 to 10.22 (#682)`
		`* Bump postgres from 11.16 to 11.17 (#683)`
		`* Bump postgres from 13.7 to 13.8 (#684)`
		`* Fix deploy postres ci job (#687)`
		`* [ci] Replace Xenial by Jammy (#689)`
		`## Dependencies`
		`* **storage:** Updated to v1.25.0.`
		`For more information, see [storage](https://github.com/googleapis/google-cloud-go).`
		`* **bosh-backup-and-restore:** Updated to v1.9.37.`
		`For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore).`
		`* **api:** Updated to v0.94.0.`
		`For more information, see [api](https://github.com/googleapis/google-api-go-client).`
		▼ v1.18.49
		`## Changes`
		`* Add final release 1.18.47 [ci skip]`
		`## Dependencies`
		`* **storage:** Updated to v1.24.0.`
		`For more information, see [storage](https://github.com/googleapis/google-cloud-go).`
		`* **bosh-backup-and-restore:** Updated to v1.9.35.`
		`For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore).`
		`* **gomega:** Updated to v1.20.0.`
		`For more information, see [gomega](https://github.com/onsi/gomega).`
		▼ v1.18.48
		`## Changes`
		`* Add final release 1.18.47 [ci skip]`
		`* Fix bpm-release download url`
		`* Parametrise minis-host and minio-port`
		`* Remove explicit port in BOSH_GW_HOST`
		`* Remove hardcoded port`
		`* Replace secrets in task definition`
		`## Dependencies`
		`* **storage:** Updated to v1.24.0.`
		`For more information, see [storage](https://github.com/googleapis/google-cloud-go).`

Component	Version	Release Notes
		* **bosh-backup-and-restore:** Updated to v1.9.35. For more information, see [bosh-backup-and-restore](https://github.com/cloudfoundry-incubator/bosh-backup-and-restore). * **gomega:** Updated to v1.20.0. For more information, see [gomega](https://github.com/onsi/gomega).
binary-offline-buildpack	1.0.45	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.24	
bpm	1.1.19	
capi	1.117.8	
cf-autoscaling	249.0.17	
cf-cli	1.38.0	
cf-networking	3.11.0	
cflinuxfs3	0.319.0	
credhub	2.12.8	▼ 2.12.8 ### Security Fixes - Bump various dependencies, including bumping postgresql from 42.4.0 to 42.4.1, which addresses [CVE-2022-31197](https://nvd.nist.gov/vuln/detail/CVE-2022-31197) ▼ 2.12.7 ### Security Fixes - Bump various dependencies ### Bug Fixes - Improved test robustness on platforms with slow random number generation - Improved test robustness for several tests that handle database setup
diego	2.62.0	
dotnet-core-offline-buildpack	2.4.0	
garden-runc	1.22.0	
go-offline-buildpack	1.9.49	
haproxy	11.10.2	

Component	Version	Release Notes
java-offline-buildpack	4.50	
log-cache	2.11.13	
loggregator	106.6.9	
loggregator-agent	6.4.4	
mapfs	1.2.6	
metric-registrar	1.2.10	
metrics-discovery	3.1.2	
mysql-monitoring	9.18.0	▼ v9.18.0 **Bugs Fixed** Fixed a bug where mysql-metrics would fail to restart under ubuntu-jammy stemcells As part of this fix, mysql-metrics and the mysql-diag-agent jobs now use bpm for process management. ▼ v9.17.0 - `mysql-diag-agent` and `replication-canary` support TLS ▼ v9.16.0 - Bump golang to version 1.18.2
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.42	
nodejs-offline-buildpack	1.7.73	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.65	
push-apps-manager-release	675.0.5	▼ 675.0.5 - Update Content-Security-Policy
push-usage-service-release	674.0.24	
pxc	0.44.0	

Component	Version	Release Notes
python-offline-buildpack	1.7.57	
r-offline-buildpack	1.1.32	
routing	0.236.0	
ruby-offline-buildpack	1.8.57	
silk	3.12.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	
staticfile-offline-buildpack	1.5.33	
statsd-injector	1.11.21	
syslog	11.8.2	
system-metrics-scraper	3.2.8	
uaa	74.5.48	▼ v74.5.48 ### Dependency bumps - Various dependency bumps. ▼ v74.5.47 ### Fixes - Fixes a sporadic pre-start script failure due to a race condition of the `update-ca-certificates` commands [#391] ### Dependency bumps - Various dependency bumps.

v2.12.16

Release Date: 08/10/2022

- Bump backup-and-restore-sdk to version `1.18.47`
- Bump bosh-system-metrics-forwarder to version `0.0.23`
- Bump cf-autoscaling to version `249.0.17`
- Bump cf-networking to version `3.11.0`
- Bump cflinuxfs3 to version `0.312.0`
- Bump credhub to version `2.12.6`
- Bump dotnet-core-offline-buildpack to version `2.3.44`
- Bump go-offline-buildpack to version `1.9.48`
- Bump haproxy to version `11.10.2`

- Bump java-offline-buildpack to version `4.50`
- Bump log-cache to version `2.11.12`
- Bump loggregator to version `106.6.8`
- Bump loggregator-agent to version `6.4.3`
- Bump metric-registrar to version `1.2.9`
- Bump metrics-discovery to version `3.1.1`
- Bump nginx-offline-buildpack to version `1.1.41`
- Bump nodejs-offline-buildpack to version `1.7.72`
- Bump php-offline-buildpack to version `4.4.64`
- Bump pxc to version `0.44.0`
- Bump python-offline-buildpack to version `1.7.56`
- Bump r-offline-buildpack to version `1.1.31`
- Bump routing to version `0.236.0`
- Bump ruby-offline-buildpack to version `1.8.56`
- Bump silk to version `3.11.0`
- Bump staticfile-offline-buildpack to version `1.5.32`
- Bump statsd-injector to version `1.11.20`
- Bump syslog to version `11.8.1`
- Bump system-metrics-scraper to version `3.2.7`
- Bump uaa to version `74.5.46`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.261	
backup-and-restore-sdk	1.18.47	
binary-offline-buildpack	1.0.45	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.23	
bpm	1.1.18	
capi	1.117.7	
cf-autoscaling	249.0.17	
cf-cli	1.38.0	
cf-networking	3.11.0	
cflinuxfs3	0.312.0	

Component	Version	Release Notes

credhub	2.12.6	▼ 2.12.6
		`### Security Fixes`
		`- Bump various dependencies`

diego	2.62.0	
dotnet-core-offline-buildpack	2.3.44	
garden-runc	1.20.8	
go-offline-buildpack	1.9.48	
haproxy	11.10.2	
java-offline-buildpack	4.50	
log-cache	2.11.12	
loggregator	106.6.8	
loggregator-agent	6.4.3	
mapfs	1.2.6	
metric-registrar	1.2.9	
metrics-discovery	3.1.1	
mysql-monitoring	9.15.0	
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.41	
nodejs-offline-buildpack	1.7.72	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.64	
push-apps-manager-release	675.0.4	
push-usage-service-release	674.0.24	
pxc	0.44.0	
python-offline-buildpack	1.7.56	
r-offline-buildpack	1.1.31	

Component	Version	Release Notes
routing	0.236.0	▼ v0.236.0 ## What's Changed * Gorouter restart script waits for the gorouter to be running before reloading monit ## Built with go 1.17.12 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.235.0...0.236.0 ▼ 0.235.0 ## What's Changed * Gorouter healthchecker retries connection instead of monit (https://github.com/cloudfoundry/routing-release/pull/275) ## Built with go 1.17.11 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.234.0...0.235.0 ▼ 0.234.0 ## What's Changed * Gorouter: the metrics package now uses `lsof` to monitor file descriptors on MacOS @domdom82 https://github.com/cloudfoundry/gorouter/pull/312 * Bumped the `lager` dependency to resolve issues where the timeFormat flag was not honored, resulting in epoch timestamps vs human readable. Thanks @ameowlia! * Now tested with the bionic stemcell in CI ## Built with go 1.17.11 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.233.0...0.234.0 ▼ 0.233.0 ## What's Changed * TCP Router: Add locking to the haproxy_reloader script to avoid haproxy reload/restart race conditions by @geofffranks in https://github.com/cloudfoundry/routing-release/pull/269 * TCP Router: Bump HAProxy from 1.8.13 to 2.5.4 by @cunnie in https://github.com/cloudfoundry/routing-release/pull/266 * Gorouter: fix proxy round tripper race condition by @ameowlia and @geofffranks in https://github.com/cloudfoundry/gorouter/pull/318

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>* Routing API: fix timestamp precision issue that caused routes to be pruned unexpectedly by @geofffranks in https://github.com/cloudfoundry/routing-api/pull/24 * Routing API: remove `golang.x509ignoreCN` bosh property by @geofffranks and @mariash * Routing API: fix bug that caused TCP Router's HAProxy to reload every minute by @jrussett in https://github.com/cloudfoundry/routing-api/pull/26. ## Manifest Property Changes | Job	Property	Notes
`routing-api`	`golang.x509ignoreCN`	This property exposed a go debug flag for go version 1.15. Since go 1.16 this go debug flag has had no affect. Removing this bosh property is part of our effort to keep our code base free of cruft. | ## Built with go 1.17.10 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.232.0...0.233.0</td></tr>
<tr><td>ruby-offline-buildpack</td><td>1.8.56</td><td></td></tr>
<tr><td>silk</td><td>3.11.0</td><td></td></tr>
<tr><td>smb-volume</td><td>3.1.0</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.5.0</td><td></td></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.32</td><td></td></tr>
<tr><td>statsd-injector</td><td>1.11.20</td><td></td></tr>
<tr><td>syslog</td><td>11.8.1</td><td></td></tr>
<tr><td>system-metrics-scraper</td><td>3.2.7</td><td></td></tr>
<tr><td>uaa</td><td>74.5.46</td><td></td></tr>
</table>

v2.12.15

Release Date: 07/19/2022

- **[Security Fix]** Update Content-Security-Policy
- **[Feature]** Enable telemetry for iptables rules on Diego cells
- **[Feature]** User has the ability to manage step up scaling app instances using Apps Manager
- **[Feature Improvement]** Deprecate Spring Cloud Connectors & Spring Auto Configuration support in Java Buildpack.
- **[Bug Fix]** Add health check script for Bosh DNS for Cloud Controller
- **[Bug Fix]** Fix dummy routes showing in the User Interface

- **[Bug Fix]** Fix role assignment when users are created through the CLI
- **[Bug Fix]** Fix share domain with organization screen from erroring out
- **[Bug Fix]** Resolves an issue with HAProxy log rotation creating null bytes and not freeing disk space after rotation
- **[Bug Fix]** Use Content-Disposition header as heapdump filename
- **[Bug Fix]** When Autoscaler is configured to use the **RabbitMQ Queue Depth** scaling metric in an autoscaling rule, you can specify a RabbitMQ service instance. If you specify a service instance, Autoscaler only requests metrics from that service instance.
- **[Bug Fix]** Autoscaler migration correctly handles manually-created service bindings index.
- **[Bug Fix]** Fixes Autoscaler edge case when using `http_throughput` rules with scaling factor larger than `1`.
- Bump backup-and-restore-sdk to version `1.18.43`
- Bump cf-autoscaling to version `249.0.13`
- Bump cf-cli to version `1.38.0`
- Bump cf-networking to version `3.9.0`
- Bump cflinuxfs3 to version `0.306.0`
- Bump credhub to version `2.12.5`
- Bump diego to version `2.62.0`
- Bump garden-runc to version `1.20.8`
- Bump go-offline-buildpack to version `1.9.47`
- Bump java-offline-buildpack to version `4.49`
- Bump metrics-discovery to version `3.1.0`
- Bump nginx-offline-buildpack to version `1.1.39`
- Bump nodejs-offline-buildpack to version `1.7.71`
- Bump push-apps-manager-release to version `675.0.4`
- Bump pxc to version `0.43.0`
- Bump silk to version `3.9.0`
- Bump uaa to version `74.5.44`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.252	
backup-and-restore-sdk	1.18.43	
binary-offline-buildpack	1.0.45	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.22	
bpm	1.1.18	

Component	Version	Release Notes
capi	1.117.7	
cf-autoscaling	249.0.13	
cf-cli	1.38.0	
cf-networking	3.9.0	
cflinuxfs3	0.306.0	

credhub	2.12.5	▼ 2.12.5
		`### Security Fixes`
		`- Bump various dependencies`
		`### Bug Fixes`
		`- Fix for URL path handling on Windows ([cloudfoundry/credhub issue 266](https://github.com/cloudfoundry/credhub/issues/266))`
		`### Features`
		`- CredHub now logs as info instead of error when a credential isn't found`
		`- Added support for jammy-based stemcells that have openssl 3 ([pivotal/credhub-release issue 65](https://github.com/pivotal/credhub-release/issues/65))`

diego	2.62.0	
dotnet-core-offline-buildpack	2.3.42	
garden-runc	1.20.8	
go-offline-buildpack	1.9.47	
haproxy	11.6.0	
java-offline-buildpack	4.49	
log-cache	2.11.11	
loggregator	106.6.7	
loggregator-agent	6.4.1	
mapfs	1.2.6	
metric-registrar	1.2.6	
metrics-discovery	3.1.0	
mysql-monitoring	9.15.0	
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.39	

Component	Version	Release Notes
nodejs-offline-buildpack	1.7.71	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.61	
push-apps-manager-release	675.0.4	
push-usage-service-release	674.0.24	
pxc	0.43.0	
python-offline-buildpack	1.7.54	
r-offline-buildpack	1.1.28	
routing	0.232.0	
ruby-offline-buildpack	1.8.54	
silk	3.9.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	
staticfile-offline-buildpack	1.5.30	
statsd-injector	1.11.19	
syslog	11.7.10	
system-metrics-scraper	3.2.5	
uaa	74.5.44	

v2.12.14

Release Date: 06/23/2022

⚠ **Warning: Upcoming reduction in maintenance and security release coverage**
In future patches, no sooner than July 1st 2022, some TAS components will become more strict about the protocols used in TLS communications, causing integrations with systems using older, insecure protocols to fail. Specifically, components that use Go will no longer support TLS 1.0 or 1.1, or certificates using SHA-1. Use supported TLS protocols to avoid breaking changes and continue receiving maintenance and security releases.

- Bump diego to version `2.62.0`

Component	Version
ubuntu-xenial stemcell	621.244

Component	Version
backup-and-restore-sdk	1.18.42
binary-offline-buildpack	1.0.45
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.22
bpm	1.1.18
capi	1.117.7
cf-autoscaling	249.0.7
cf-cli	1.33.0
cf-networking	3.6.0
cflinuxfs3	0.299.0
credhub	2.12.4
diego	2.62.0
dotnet-core-offline-buildpack	2.3.42
garden-runc	1.20.6
go-offline-buildpack	1.9.46
haproxy	11.6.0
java-offline-buildpack	4.48.3
log-cache	2.11.11
loggregator	106.6.7
loggregator-agent	6.4.1
mapfs	1.2.6
metric-registrar	1.2.6
metrics-discovery	3.0.13
mysql-monitoring	9.15.0
nats	42
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.38
nodejs-offline-buildpack	1.7.70
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.61

Component	Version
push-apps-manager-release	675.0.3
push-usage-service-release	674.0.24
pxc	0.42.0
python-offline-buildpack	1.7.54
r-offline-buildpack	1.1.28
routing	0.232.0
ruby-offline-buildpack	1.8.54
silk	3.6.0
smb-volume	3.1.0
smoke-tests	4.5.0
staticfile-offline-buildpack	1.5.30
statsd-injector	1.11.19
syslog	11.7.10
system-metrics-scraper	3.2.5
uaa	74.5.41

v2.12.13

Release Date: 06/09/2022

- **[Security Fix]** Added Content-Security-Policy headers in UAA responses
- **[Bug Fix]** Fix metric registrar secure scraping with isolation segments
- **[Bug Fix]** Sticky sessions no longer break when used with route-services that return HTTP 4xx/5xx responses
- **[Bug Fix/Improvement]** Stop emitting debug metrics for agents and log-cache by default. Reduces load on logging system by >=720 metrics per vm per minute
- **[Breaking Change]** If you followed the procedure in Autoscale application fails with MySQL Deadlock errors to manually add an index to an Autoscale database, and the index is not dropped before you upgrade to TAS for VMs v2.12.13, upgrading causes an error.
- Bump backup-and-restore-sdk to version `1.18.42`
- Bump binary-offline-buildpack to version `1.0.45`
- Bump bosh-system-metrics-forwarder to version `0.0.22`
- Bump bpm to version `1.1.18`
- Bump capi to version `1.117.7`
- Bump cf-autoscaling to version `249.0.7`
- Bump cf-networking to version `3.6.0`

- Bump cflinuxfs3 to version `0.299.0`
- Bump diego to version `2.64.0`
- Bump dotnet-core-offline-buildpack to version `2.3.42`
- Bump garden-runc to version `1.20.6`
- Bump go-offline-buildpack to version `1.9.46`
- Bump java-offline-buildpack to version `4.48.3`
- Bump log-cache to version `2.11.11`
- Bump loggregator to version `106.6.7`
- Bump loggregator-agent to version `6.4.1`
- Bump metric-registrar to version `1.2.6`
- Bump metrics-discovery to version `3.0.13`
- Bump nginx-offline-buildpack to version `1.1.38`
- Bump nodejs-offline-buildpack to version `1.7.70`
- Bump php-offline-buildpack to version `4.4.61`
- Bump push-usage-service-release to version `674.0.24`
- Bump python-offline-buildpack to version `1.7.54`
- Bump routing to version `0.232.0`
- Bump ruby-offline-buildpack to version `1.8.54`
- Bump silk to version `3.6.0`
- Bump staticfile-offline-buildpack to version `1.5.30`
- Bump statsd-injector to version `1.11.19`
- Bump syslog to version `11.7.10`
- Bump system-metrics-scraper to version `3.2.5`
- Bump uaa to version `74.5.41`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.244	
backup-and-restore-sdk	1.18.42	
binary-offline-buildpack	1.0.45	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.22	
bpm	1.1.18	
capi	1.117.7	
cf-autoscaling	249.0.7	

Component	Version	Release Notes
cf-cli	1.33.0	
cf-networking	3.6.0	
cflinuxfs3	0.299.0	
credhub	2.12.4	
diego	2.64.0	
dotnet-core-offline-buildpack	2.3.42	
garden-runc	1.20.6	
go-offline-buildpack	1.9.46	
haproxy	11.6.0	
java-offline-buildpack	4.48.3	
log-cache	2.11.11	
loggregator	106.6.7	
loggregator-agent	6.4.1	
mapfs	1.2.6	
metric-registrar	1.2.6	
metrics-discovery	3.0.13	
mysql-monitoring	9.15.0	
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.38	
nodejs-offline-buildpack	1.7.70	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.61	
push-apps-manager-release	675.0.3	
push-usage-service-release	674.0.24	
pxc	0.42.0	
python-offline-buildpack	1.7.54	
r-offline-buildpack	1.1.28	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>routing</td><td>0.232.0</td><td>▼ 0.232.0 ## What's Changed * Fixing issue #250: Return a 503 not a 404 when all instances down by @kecirlotfi in https://github.com/cloudfoundry/routing-release/pull/268 and https://github.com/cloudfoundry/gorouter/pull/314 * Fixing issue https://github.com/cloudfoundry/gorouter/pull/315: Fix route service pruning by @geofffranks ## Manifest Property Changes | Job	Property	default	notes
`gorouter`	`for_backwards_compatibility_only.empty_pool_response_code_503`	`0s`	This property was added to enable https://github.com/cloudfoundry/routing-release/pull/268 | ## New Contributors * @kecirlotfi made their first contribution! Thanks so much! ## Built with go 1.17.9 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.231.0...0.232.0</td></tr>
<tr><td>ruby-offline-buildpack</td><td>1.8.54</td><td></td></tr>
<tr><td>silk</td><td>3.6.0</td><td></td></tr>
<tr><td>smb-volume</td><td>3.1.0</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.5.0</td><td></td></tr>
<tr><td>staticfile-offline-buildpack</td><td>1.5.30</td><td></td></tr>
<tr><td>statsd-injector</td><td>1.11.19</td><td></td></tr>
<tr><td>syslog</td><td>11.7.10</td><td></td></tr>
<tr><td>system-metrics-scraper</td><td>3.2.5</td><td></td></tr>
<tr><td>uaa</td><td>74.5.41</td><td></td></tr>
</table>

v2.12.12

Release Date: 04/20/2022

- **[Feature Improvement]** Add option to configure CC BBR health check timeout
- **[Feature Improvement]** Enforce service name uniqueness in shared services in spaces
- **[Breaking Change]** Syslog drains configured to use TLS now reject certificates signed with the SHA-1 hash function.

- Bump backup-and-restore-sdk to version `1.18.39`
- Bump binary-offline-buildpack to version `1.0.43`
- Bump capi to version `1.117.6`
- Bump cf-autoscaling to version `249.0.2`
- Bump cflinuxfs3 to version `0.285.0`
- Bump credhub to version `2.12.4`
- Bump diego to version `2.62.0`
- Bump dotnet-core-offline-buildpack to version `2.3.41`
- Bump go-offline-buildpack to version `1.9.42`
- Bump java-offline-buildpack to version `4.48.2`
- Bump log-cache to version `2.11.8`
- Bump loggregator to version `106.6.4`
- Bump loggregator-agent to version `6.3.11`
- Bump metrics-discovery to version `3.0.10`
- Bump nginx-offline-buildpack to version `1.1.37`
- Bump nodejs-offline-buildpack to version `1.7.69`
- Bump php-offline-buildpack to version `4.4.59`
- Bump pxc to version `0.42.0`
- Bump python-offline-buildpack to version `1.7.53`
- Bump r-offline-buildpack to version `1.1.28`
- Bump ruby-offline-buildpack to version `1.8.53`
- Bump uaa to version `74.5.37`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.224	
backup-and-restore-sdk	1.18.39	
binary-offline-buildpack	1.0.43	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.21	
bpm	1.1.16	
capi	1.117.6	

Component	Version	Release Notes
cf-autoscaling	249.0.2	▼ v249.0.2 ## What's Changed * Bump github.com/onsi/gomega from 1.18.1 to 1.19.0 in /src by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/640 * bump spring boot for cve CVE-2022-22965 by @Benjamintf1 in https://github.com/pivotal-cf/cf-autoscaling-release/pull/646 * Bump log4j-to-slf4j from 2.17.1 to 2.17.2 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/619 * Bump log4j-api from 2.17.1 to 2.17.2 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/618 * Bump gson from 2.8.6 to 2.9.0 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/611 * Bump spock-core from 2.0-groovy-3.0 to 2.1-groovy-3.0 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/613 * Bump objenesis from 3.1 to 3.2 in /src/cf-autoscaling/api by @dependabot in https://github.com/pivotal-cf/cf-autoscaling-release/pull/479 **Full Changelog**: https://github.com/pivotal-cf/cf-autoscaling-release/compare/v249.0.1...v249.0.2 ▼ v249.0.1 ## What's Changed * Pin jackson-databind to 2.13.2.2 to address [CVE-2020-36518](https://nvd.nist.gov/vuln/detail/CVE-2020-36518) * Unpin tomcat dependencies in autoscale API in https://github.com/pivotal-cf/cf-autoscaling-release/pull/636 * Bump autoscale API dependencies in https://github.com/pivotal-cf/cf-autoscaling-release/pull/612, https://github.com/pivotal-cf/cf-autoscaling-release/pull/625, https://github.com/pivotal-cf/cf-autoscaling-release/pull/525, https://github.com/pivotal-cf/cf-autoscaling-release/pull/634 **Full Changelog**: https://github.com/pivotal-cf/cf-autoscaling-release/compare/v249...v249.0.1

Component	Version	Release Notes
cf-cli	1.33.0	
cf-networking	3.3.0	
cflinuxfs3	0.285.0	
credhub	2.12.4	▼ 2.12.4 ### Security Fixes - Bump various dependencies. ▼ 2.12.3 ### Security Fixes - Bump various dependencies.
diego	2.62.0	
dotnet-core-offline-buildpack	2.3.41	
garden-runc	1.20.3	
go-offline-buildpack	1.9.42	
haproxy	11.6.0	
java-offline-buildpack	4.48.2	
log-cache	2.11.8	▼ v2.11.8 ## Release Highlights Pin Go back to go1.17. [Go 1.18 includes changes to memory management](https://tip.golang.org/doc/go1.18#runtime) and we'd like to get more familiarity with these changes and their impact before bumping. ### Built with golang 1.17.8 ▼ v2.11.7 - fix bug with large messages (#58) - bump-golang to v0.100.0(now 1.18) ▼ v2.11.6 * fix prom scraper config (#55) * bump-golang to v0.99.0 * Remove useless GODEBUG flag `x509ignoreCN`

Component	Version	Release Notes

loggregator	106.6.4	▼ v106.6.4
		`- fix bug with large messages (#430)`
		`- bump-golang to v0.100.0(now 1.18)`

loggregator-agent	6.3.11	▼ v6.3.11
		`- fix bug with large messages (#89)`
		`- bump-golang to v0.100.0(now 1.18)`

mapfs	1.2.6	
metric-registrar	1.2.5	

metrics-discovery	3.0.10	▼ v3.0.10
		`- fix bug with large messages (#22)`
		`- bump-golang to v0.100.0(now 1.18)`

mysql-monitoring	9.15.0	
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.37	
nodejs-offline-buildpack	1.7.69	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.59	
push-apps-manager-release	675.0.3	
push-usage-service-release	674.0.23	
pxc	0.42.0	
python-offline-buildpack	1.7.53	
r-offline-buildpack	1.1.28	
routing	0.231.0	
ruby-offline-buildpack	1.8.53	
silk	3.3.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	

Component	Version	Release Notes
staticfile-offline-buildpack	1.5.29	
statsd-injector	1.11.18	
syslog	11.7.7	
system-metrics-scraper	3.2.4	
uaa	74.5.37	

v2.12.11

Release Date: 04/06/2022

- **[Security Fix]** This release fixes CVE-2022-22965; note that the “fix” in the immediately prior version did not actually address the vulnerability, as Spring framework dependencies in UAA that should have been updated, were not. We have confirmed this version actually contains the dependency bumps, and that it is no longer vulnerable to our confirmed exploit. We consider this patch necessary for secure operation; see the VMware Security Advisory here for more details. This release also includes a new version of the Java Buildpack.
- Bump java-offline-buildpack to version `4.48.2`
- Bump uaa to version `74.5.37`

Component	Version
ubuntu-xenial stemcell	621.224
backup-and-restore-sdk	1.18.34
binary-offline-buildpack	1.0.42
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.21
bpm	1.1.16
capi	1.117.4
cf-autoscaling	249
cf-cli	1.33.0
cf-networking	3.3.0
cflinuxfs3	0.279.0
credhub	2.12.1
diego	2.61.0
dotnet-core-offline-buildpack	2.3.40
garden-runc	1.20.3
go-offline-buildpack	1.9.41

Component	Version
haproxy	11.6.0
java-offline-buildpack	4.48.2
log-cache	2.11.5
loggregator	106.6.3
loggregator-agent	6.3.10
mapfs	1.2.6
metric-registrar	1.2.5
metrics-discovery	3.0.9
mysql-monitoring	9.15.0
nats	42
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.36
nodejs-offline-buildpack	1.7.67
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.57
push-apps-manager-release	675.0.3
push-usage-service-release	674.0.23
pxc	0.41.0
python-offline-buildpack	1.7.51
r-offline-buildpack	1.1.27
routing	0.231.0
ruby-offline-buildpack	1.8.52
silk	3.3.0
smb-volume	3.1.0
smoke-tests	4.5.0
staticfile-offline-buildpack	1.5.29
statsd-injector	1.11.18
syslog	11.7.7
system-metrics-scraper	3.2.4
uaa	74.5.37

v2.12.10

Release Date: 03/31/2022

- **[Security Fix]** This release was intended to address CVE-2022-22965, but did not actually update the vulnerable dependencies. Upgrade to a more recent patch version instead. See the VMware Security Advisory here for more details.
- Bump uaa to version `74.5.36`

Component	Version
ubuntu-xenial stemcell	621.224
backup-and-restore-sdk	1.18.34
binary-offline-buildpack	1.0.42
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.21
bpm	1.1.16
capi	1.117.4
cf-autoscaling	249
cf-cli	1.33.0
cf-networking	3.3.0
cflinuxfs3	0.279.0
credhub	2.12.1
diego	2.61.0
dotnet-core-offline-buildpack	2.3.40
garden-runc	1.20.3
go-offline-buildpack	1.9.41
haproxy	11.6.0
java-offline-buildpack	4.48
log-cache	2.11.5
loggregator	106.6.3
loggregator-agent	6.3.10
mapfs	1.2.6
metric-registrar	1.2.5
metrics-discovery	3.0.9
mysql-monitoring	9.15.0
nats	42

Component	Version
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.36
nodejs-offline-buildpack	1.7.67
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.57
push-apps-manager-release	675.0.3
push-usage-service-release	674.0.23
pxc	0.41.0
python-offline-buildpack	1.7.51
r-offline-buildpack	1.1.27
routing	0.231.0
ruby-offline-buildpack	1.8.52
silk	3.3.0
smb-volume	3.1.0
smoke-tests	4.5.0
staticfile-offline-buildpack	1.5.29
statsd-injector	1.11.18
syslog	11.7.7
system-metrics-scraper	3.2.4
uaa	74.5.36

v2.12.9

Release Date: 03/31/2022

- **[Security Fix]** This release fixes CVE-2022-23806 and CVE-2022-23772.
- **[Bug Fix]** Assign cloud_controller.read and cloud_controller.write scopes to service brokers created using CF CLI v8
- **[Bug Fix]** CAPI: Quota metrics are no longer filtered when syslog ingress is turned on
- **[Bug Fix]** Propagate updated user provided service environment variables to bound applications for CF CLI v8
- **[Bug Fix]** Resolve an issue resulting in tcp-router repeatedly respawning haproxy until it hits a forked process limit

- **[Bug Fix]** Resolves an issue where invalid seeded router group values should caused breaking changes
- **[Bug fix]** Remove x509ignoreCN option in Gorouter
- Bump capi to version `1.117.4`
- Bump cf-autoscaling to version `249`
- Bump cf-networking to version `3.3.0`
- Bump cflinuxfs3 to version `0.279.0`
- Bump credhub to version `2.12.1`
- Bump diego to version `2.61.0`
- Bump dotnet-core-offline-buildpack to version `2.3.40`
- Bump garden-runc to version `1.20.3`
- Bump go-offline-buildpack to version `1.9.41`
- Bump loggregator to version `106.6.3`
- Bump loggregator-agent to version `6.3.10`
- Bump metric-registrar to version `1.2.5`
- Bump metrics-discovery to version `3.0.9`
- Bump nginx-offline-buildpack to version `1.1.36`
- Bump nodejs-offline-buildpack to version `1.7.67`
- Bump php-offline-buildpack to version `4.4.57`
- Bump python-offline-buildpack to version `1.7.51`
- Bump r-offline-buildpack to version `1.1.27`
- Bump routing to version `0.231.0`
- Bump ruby-offline-buildpack to version `1.8.52`
- Bump silk to version `3.3.0`
- Bump staticfile-offline-buildpack to version `1.5.29`
- Bump uaa to version `74.5.35`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.224	
backup-and-restore-sdk	1.18.34	
binary-offline-buildpack	1.0.42	
bosh-dns-aliases	0.0.4	
bosh-system-metrics-forwarder	0.0.21	
bpm	1.1.16	
capi	1.117.4	

Component	Version	Release Notes
cf-autoscaling	249	
cf-cli (v7/v8)*	1.33.0	
cf-networking	3.3.0	
cflinuxfs3	0.279.0	

Component	Version	Release Notes
credhub	2.12.1	▼ 2.12.1 ### Security Fixes - Bump various dependencies. ▼ 2.12.0 ### Security Fixes - Bump various dependencies. ### Bug Fixes - Fixes an issue where CredHub experiences downtime during certificate rotation process by making CredHub properly load concatenated mTLS CA certificates. ### Features - CredHub is now compatible with Postgres 13, 14. ▼ 2.11.1 ### Dependency Bumps - Bumps log4j2 to 2.17.1 ▼ 2.11.0 ### Security Fixes - Further addresses [CVE with Log4j library](https://github.com/advisories/GHSA-jfh8-c2jp-5v3q) and [its prior incomplete fix](https://github.com/advisories/GHSA-7rjr-3q55-vv33) by bumping to log4j2 2.16.0. ▼ 2.10.0 ### Security Fixes - Addresses [CVE with Log4j library](https://github.com/advisories/GHSA-jfh8-c2jp-5v3q) ### Features - Adds a minimum duration server-level configuration fields for leaf and CA certificates: `certificates.leaf_minimum_duration_in_days` and `certificates.ca_minimum_duration_in_days`. When these fields are configured, if a request to generate or regenerate a certificate has a duration lower than the minimum, then the minimum duration is used instead. (https://github.com/cloudfoundry/credhub/pull/201)

Component	Version	Release Notes
diego	2.61.0	
dotnet-core-offline-buildpack	2.3.40	
garden-runc	1.20.3	
go-offline-buildpack	1.9.41	
haproxy	11.6.0	
java-offline-buildpack	4.48	
log-cache	2.11.5	
loggregator	106.6.3	
loggregator-agent	6.3.10	
mapfs	1.2.6	
metric-registrar	1.2.5	
metrics-discovery	3.0.9	
mysql-monitoring	9.15.0	
nats	42	
nfs-volume	7.1.1	
nginx-offline-buildpack	1.1.36	
nodejs-offline-buildpack	1.7.67	
notifications	62	
notifications-ui	40	
php-offline-buildpack	4.4.57	
push-apps-manager-release	675.0.3	
push-usage-service-release	674.0.23	
pxc	0.41.0	
python-offline-buildpack	1.7.51	
r-offline-buildpack	1.1.27	

Component	Version	Release Notes
routing	0.231.0	▼ 0.231.0 ## Bug Fixes - Removed the x509ignoreCN property. Now that `gorouter` is built on golang 1.17, itno longer has any effect on gorouter behavior, and was only adding to confusion inthe properties - Resolve an issue with route-registrar using the same TTL as it's RegistrationIntervalfor tcp routes, leading to unnecessary churn of pruned + re-registered routes. - Resolve an issue with Routing API where upserts to tcp routes were causing changeevents to be emitted when the only change was a bump in TTL. This led to an issuewhere tcp-router was constantly reloading haproxy with every route's heartbeatregistration call. ## Manifest Property Changes | Job | Property | 0.230.0 | 0.231.0 | | --- | --- | --- | --- | | `gorouter` | `golang.x509ignoreCN` | false | No longer exists | | `route_registrar` | `golang.x509ignoreCN` | false | No longer exists | | `tcp_router` | `golang.x509ignoreCN` | false | No longer exists | ### Built with golang 1.17.8 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.230.0...0.231.0 ▼ 0.230.0 ## Feature * update gorouter for prometheus scraping by @Benjamintf1 in https://github.com/cloudfoundry/routing-release/pull/258 ## Bug Fix * Invalid seeded router group manifest values should no longer cause breaking changes by default by @ameowlia in https://github.com/cloudfoundry/routing-release/pull/261 ### Built with golang 1.17.7 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.229.0...0.230.0
ruby-offline-buildpack	1.8.52	
silk	3.3.0	
smb-volume	3.1.0	

Component	Version	Release Notes
smoke-tests	4.5.0	
staticfile-offline-buildpack	1.5.29	
statsd-injector	1.11.18	
syslog	11.7.7	
system-metrics-scraper	3.2.4	
uaa	74.5.35	

v2.12.8

Release Date: 02/28/2022

- **[Feature Improvement]** Due to routing-release now being built with Golang 1.17, all certificates provided must contain SAN entries on them. The previous workaround of setting “Enable temporary workaround for certs without SANs” no longer functions.
- **[Feature Improvement]** Per Golang 1.17’s new and stricter IP parsing standards, any IP addrs with leading zeros in any octets will result in a BOSH template failure to allow operators to remove the leading zeros and try again (affects properties fed into diego-release, garden-runc-release, winc-release, nats-release, and routing-release),.
- **[Feature Improvement]** UAA is compatible with MySQL 8
- **[Feature Improvement]** You can configure container-to-container traffic to use TLS. For more information, see Securing Container-to-Container Traffic in *Container-to-Container Networking*.

  > ⚠ **Warning:** This feature introduces a migration to the BBS database. Rolling your TAS for VMs deployment back to v2.12.7 or earlier causes database issues.

- **[Bug Fix]** Fix default metric registrar blocked tags to include ‘ip’ and remove ‘id’
- **[Bug Fix]** Fix metric-registrar blocked tags configuration
- **[Bug Fix]** Fixes an issue related to the parsing of the X-B3-TraceId and X-B3-SpanId HTTP headers
- **[Bug Fix]** Restore missing networking and garden metrics
- **[Bug Fix]** Smoke tests support for TLSv1.3 only option
- Bump backup-and-restore-sdk to version `1.18.34`
- Bump cf-autoscaling to version `248`
- Bump cflinuxfs3 to version `0.274.0`
- Bump credhub to version `2.9.9`
- Bump diego to version `2.58.1`

- Bump garden-runc to version `1.20.0`
- Bump loggregator-agent to version `6.3.8`
- Bump metric-registrar to version `1.2.4`
- Bump metrics-discovery to version `3.0.8`
- Bump nats to version `42`
- Bump routing to version `0.229.0`
- Bump smoke-tests to version `4.5.0`
- Bump uaa to version `74.5.34`

Component	Version
ubuntu-xenial stemcell	621.211
backup-and-restore-sdk	1.18.34
binary-offline-buildpack	1.0.42
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.21
bpm	1.1.16
capi	1.117.2
cf-autoscaling	248
cf-cli (v7/v8)*	1.33.0
cf-networking	2.43.0
cflinuxfs3	0.274.0
credhub	2.9.9
diego	2.58.1
dotnet-core-offline-buildpack	2.3.38
garden-runc	1.20.0
go-offline-buildpack	1.9.38
haproxy	11.6.0
java-offline-buildpack	4.48
log-cache	2.11.5
loggregator	106.6.2
loggregator-agent	6.3.8
mapfs	1.2.6
metric-registrar	1.2.4
metrics-discovery	3.0.8

Component	Version
mysql-monitoring	9.15.0
nats	42
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.34
nodejs-offline-buildpack	1.7.66
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.55
push-apps-manager-release	675.0.3
push-usage-service-release	674.0.23
pxc	0.41.0
python-offline-buildpack	1.7.49
r-offline-buildpack	1.1.25
routing	0.229.0
ruby-offline-buildpack	1.8.50
silk	2.43.0
smb-volume	3.1.0
smoke-tests	4.5.0
staticfile-offline-buildpack	1.5.28
statsd-injector	1.11.18
syslog	11.7.7
system-metrics-scraper	3.2.4
uaa	74.5.34

* The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.7

Release Date: 02/07/2022

> **Note:** This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

- **[Security Fix]** Diego - Bump containerd to v1.5.9 to address (CVE-2021-43816)
- **[Security Fix]** Bump routing release to v0.228.0 to address (CVE-2021-44716)

- **[Feature]** Monit thresholds for the Cloud Controller worker are configurable
- **[Feature]** Operators can sort apps by status in Apps Manager
- **[Feature]** Apps can be step-scaled up or down in Autoscaler. See About App Autoscaler.
- **[Feature Improvement]** Apps Manager supports HTTP/2
- **[Feature Improvement]** Operators can assign the Space Supporter role in Apps Manager
- **[Feature Improvement]** Golang v1.17 contains stricter IP parsing standards, so IP addresses with leading zeros in any octets cause a BOSH template failure. Operators can remove the leading zeros and try deploying again. This affects properties that feed into cf-networking-release, silk-release, loggregator-agent-release, and syslog-release. Syslog drains and metric registrar endpoints registered using user-provided services might also be affected.
- **[Bug Fix]** Adds the ability to parse the cost object in service plan
- **[Bug Fix]** Fix race conditions that could cause Autoscaler to crash
- Bump backup-and-restore-sdk to version `1.18.32`
- Bump binary-offline-buildpack to version `1.0.42`
- Bump bosh-system-metrics-forwarder to version `0.0.21`
- Bump bpm to version `1.1.16`
- Bump cf-autoscaling to version `247`
- Bump cf-networking to version `2.43.0`
- Bump cflinuxfs3 to version `0.272.0`
- Bump diego to version `2.57.0`
- Bump dotnet-core-offline-buildpack to version `2.3.38`
- Bump go-offline-buildpack to version `1.9.38`
- Bump java-offline-buildpack to version `4.48`
- Bump log-cache to version `2.11.5`
- Bump loggregator to version `106.6.2`
- Bump loggregator-agent to version `6.3.7`
- Bump metric-registrar to version `1.2.3`
- Bump metrics-discovery to version `3.0.7`
- Bump nats to version `41`
- Bump nginx-offline-buildpack to version `1.1.34`
- Bump nodejs-offline-buildpack to version `1.7.66`
- Bump php-offline-buildpack to version `4.4.55`
- Bump push-apps-manager-release to version `675.0.3`
- Bump pxc to version `0.41.0`
- Bump python-offline-buildpack to version `1.7.49`

- Bump r-offline-buildpack to version `1.1.25`
- Bump routing to version `0.228.0`
- Bump ruby-offline-buildpack to version `1.8.50`
- Bump silk to version `2.43.0`
- Bump smoke-tests to version `4.4.0`
- Bump staticfile-offline-buildpack to version `1.5.28`
- Bump statsd-injector to version `1.11.18`
- Bump syslog to version `11.7.7`
- Bump system-metrics-scraper to version `3.2.4`
- Bump uaa to version `74.5.31`

Component	Version
ubuntu-xenial stemcell	621.198
backup-and-restore-sdk	1.18.32
binary-offline-buildpack	1.0.42
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.21
bpm	1.1.16
capi	1.117.2
cf-autoscaling	247
cf-cli (v7/v8)*	1.33.0
cf-networking	2.43.0
cflinuxfs3	0.272.0
credhub	2.9.8
diego	2.57.0
dotnet-core-offline-buildpack	2.3.38
garden-runc	1.19.30
go-offline-buildpack	1.9.38
haproxy	11.6.0
java-offline-buildpack	4.48
log-cache	2.11.5
loggregator	106.6.2
loggregator-agent	6.3.7
mapfs	1.2.6

Component	Version
metric-registrar	1.2.3
metrics-discovery	3.0.7
mysql-monitoring	9.15.0
nats	41
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.34
nodejs-offline-buildpack	1.7.66
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.55
push-apps-manager-release	675.0.3
push-usage-service-release	674.0.23
pxc	0.41.0
python-offline-buildpack	1.7.49
r-offline-buildpack	1.1.25
routing	0.228.0
ruby-offline-buildpack	1.8.50
silk	2.43.0
smb-volume	3.1.0
smoke-tests	4.4.0
staticfile-offline-buildpack	1.5.28
statsd-injector	1.11.18
syslog	11.7.7
system-metrics-scraper	3.2.4
uaa	74.5.31

* The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.6

Release Date: 12/21/2021

⚠ **Warning:**

See the following warnings:

- VMware recommends upgrading TAS for VMs as soon as possible to address CVE-2021-44228. If you are unable to upgrade, you can mitigate this CVE manually. See Workaround instructions to address CVE-2021-44228 and CVE-2021-45046 in Tanzu Application Service v2.7 - v2.12.
- This vulnerability also affects Ops Manager. VMware recommends upgrading to Ops Manager v2.10.24 as soon as possible. For more information, see the Ops Manager v2.10.24 release notes.

Note: This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

- **[Security Fix]** Fix uncontrolled recursion related to Log4j (CVE-2021-45105)
- **[Bug Fix]** Cloud Controller Worker: PruneExcessAppRevisions job is more memory efficient
- **[Breaking Change]** Gorouter: zipkin `trace-id` size complies with w3 standard of 16 bytes opposed to the previous 8 bytes.
- Bump credhub to version `2.9.8` which has Log4j `2.17.0`
- Bump java-offline-buildpack to version `4.47`
- Bump routing to version `0.227.0`
- Bump uaa to version `74.5.30` which has Log4j `2.17.0`

Component	Version
ubuntu-xenial stemcell	~621
backup-and-restore-sdk	1.18.28
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.15
capi	1.117.2
cf-autoscaling	242
cf-cli (v7/v8)*	1.33.0
cf-networking	2.42.0
cflinuxfs3	0.268.0
credhub	2.9.8
diego	2.54.0
dotnet-core-offline-buildpack	2.3.36
garden-runc	1.19.30

Component	Version
go-offline-buildpack	1.9.37
haproxy	11.6.0
java-offline-buildpack	4.47
log-cache	2.11.4
loggregator	106.6.1
loggregator-agent	6.3.5
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32
nodejs-offline-buildpack	1.7.63
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.53
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.47
r-offline-buildpack	1.1.23
routing	0.227.0
ruby-offline-buildpack	1.8.48
silk	2.41.0
smb-volume	3.1.0
smoke-tests	4.3.1
staticfile-offline-buildpack	1.5.26
statsd-injector	1.11.16
syslog	11.7.6
system-metrics-scraper	3.2.3

Component	Version
uaa	74.5.30

[*] The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.5

Release Date: 12/16/2021

Warning:

See the following warnings:

- VMware recommends upgrading TAS for VMs as soon as possible to address CVE-2021-44228. If you are unable to upgrade, you can mitigate this CVE manually. See Workaround instructions to address CVE-2021-44228 and CVE-2021-45046 in Tanzu Application Service v2.7 - v2.12.
- This vulnerability also affects Ops Manager. VMware recommends upgrading to Ops Manager v2.10.24 as soon as possible. For more information, see the Ops Manager v2.10.24 release notes.

Note: This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

- **[Security Fix]** Fix remote code execution vulnerability related to Log4j (CVE-2021-45046)
- Bump credhub to version `2.9.7` which has Log4j `2.16.0`
- Bump java-offline-buildpack to version `4.45`
- Bump php-offline-buildpack to version `4.4.53`
- Bump uaa to version `74.5.29` which has Log4j `2.16.0`

Component	Version
ubuntu-xenial stemcell	~621
backup-and-restore-sdk	1.18.28
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.15
capi	1.117.2
cf-autoscaling	242
cf-cli (v7/v8)[*]	1.33.0
cf-networking	2.42.0

Component	Version
cflinuxfs3	0.268.0
credhub	2.9.7
diego	2.54.0
dotnet-core-offline-buildpack	2.3.36
garden-runc	1.19.30
go-offline-buildpack	1.9.37
haproxy	11.6.0
java-offline-buildpack	4.45
log-cache	2.11.4
loggregator	106.6.1
loggregator-agent	6.3.5
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32
nodejs-offline-buildpack	1.7.63
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.53
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.47
r-offline-buildpack	1.1.23
routing	0.227.0
ruby-offline-buildpack	1.8.48
silk	2.41.0
smb-volume	3.1.0

Component	Version
smoke-tests	4.3.1
staticfile-offline-buildpack	1.5.26
statsd-injector	1.11.16
syslog	11.7.6
system-metrics-scraper	3.2.3
uaa	74.5.29

* The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.4

Release Date: 12/15/2021

Note: This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

Warning:

See the following warnings:

- VMware recommends upgrading TAS for VMs as soon as possible to address CVE-2021-44228. If you are unable to upgrade, you can mitigate this CVE manually. See Workaround instructions to address CVE-2021-44228 and CVE-2021-45046 in Tanzu Application Service v2.7 - v2.12.
- This vulnerability also affects Ops Manager. VMware recommends upgrading to Ops Manager v2.10.24 as soon as possible. For more information, see the Ops Manager v2.10.24 release notes.

- **[Security Fix]** Java and PHP Buildpacks - Fix remote code execution vulnerability related to Log4j (CVE-2021-44228)
- **[Bug Fix]** Fix “pre-start scripts failed. Failed Jobs: policy-server” error Upgrading to CF Networking Release v2.40.0
- **[Bug Fix]** Enables audit logging file rotation to reduce I/O load during log rotation
- **[Bug Fix]** Smoke Tests uses specified domain for Isolation Segments
- **[Feature Improvement]** Cloud Controller - Allow operators to configure the Cloud Controller monit healthcheck timeout
- Bump capi to version `1.117.2`
- Bump cf-cli to version `1.33.0`
- Bump cf-networking to version `2.42.0`
- Bump garden-runc to version `1.19.30`

- Bump haproxy to version `11.6.0`
- Bump java-offline-buildpack to version `4.44`
- Bump php-offline-buildpack to version `4.4.52`
- Bump smoke-tests to version `4.3.1`

Component	Version
ubuntu-xenial stemcell	~621
backup-and-restore-sdk	1.18.28
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.15
capi	1.117.2
cf-autoscaling	242
cf-cli (v7/v8)*	1.33.0
cf-networking	2.42.0
cflinuxfs3	0.268.0
credhub	2.9.6
diego	2.54.0
dotnet-core-offline-buildpack	2.3.36
garden-runc	1.19.30
go-offline-buildpack	1.9.37
haproxy	11.6.0
java-offline-buildpack	4.44
log-cache	2.11.4
loggregator	106.6.1
loggregator-agent	6.3.5
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32

Component	Version
nodejs-offline-buildpack	1.7.63
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.52
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.47
r-offline-buildpack	1.1.23
routing	0.227.0
ruby-offline-buildpack	1.8.48
silk	2.41.0
smb-volume	3.1.0
smoke-tests	4.3.1
staticfile-offline-buildpack	1.5.26
statsd-injector	1.11.16
syslog	11.7.6
system-metrics-scraper	3.2.3
uaa	74.5.28

[*] The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.3

Release Date: 12/13/2021

Warning:

See the following warnings:

- VMware recommends upgrading TAS for VMs as soon as possible to address CVE-2021-44228. If you are unable to upgrade, you can mitigate this CVE manually. See Instructions to address CVE-2021-44228 in Tanzu Application Service (2.7 through 2.12).
- This release does not contain the Java buildpack bump for this vulnerability. You must manually bump the Java buildpack and lock the version. Download the updated buildpack from VMware Tanzu Network.
- This vulnerability also affects Ops Manager. VMware recommends upgrading to Ops Manager v2.10.24 as soon as possible. For more information, see the

Ops Manager v2.10.24 release notes.

- **[Security Fix]** UAA and CredHub - Fix remote code execution vulnerability related to Log4j (CVE-2021-44228)
- **[Bug Fix]** Diego - Envoy v1.19 uses the original TCP connection pool so that it can accept more than 1024 downstream connections
- Bump credhub to version `2.9.6` which has Log4j `2.15.0`
- Bump diego to version `2.54.0`
- Bump uaa to version `74.5.28` which has Log4j `2.15.0`

Component	Version
ubuntu-xenial stemcell	621.176
backup-and-restore-sdk	1.18.26
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.15
capi	1.117.1
cf-autoscaling	241
cf-cli (v7/v8)*	1.33.0
cf-networking	2.40.0
cflinuxfs3	0.264.0
credhub	2.9.6
diego	2.54.0
dotnet-core-offline-buildpack	2.3.36
garden-runc	1.19.30
go-offline-buildpack	1.9.37
haproxy	11.6.0
java-offline-buildpack	4.42
log-cache	2.11.4
loggregator	106.6.1
loggregator-agent	6.3.4
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6

Component	Version
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32
nodejs-offline-buildpack	1.7.63
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.48
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.47
r-offline-buildpack	1.1.23
routing	0.226.0
ruby-offline-buildpack	1.8.48
silk	2.40.0
smb-volume	3.1.0
smoke-tests	4.3.0
staticfile-offline-buildpack	1.5.26
statsd-injector	1.11.16
syslog	11.7.5
system-metrics-scraper	3.2.3
uaa	74.5.28

* The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.2

Release Date: 11/23/2021

- Bump backup-and-restore-sdk to version `1.18.26`
- Bump bpm to version `1.1.15`
- Bump cf-autoscaling to version `241`
- Bump cf-networking to version `2.40.0`
- Bump cflinuxfs3 to version `0.264.0`

- Bump diego to version `2.53.1`
- Bump dotnet-core-offline-buildpack to version `2.3.36`
- Bump go-offline-buildpack to version `1.9.37`
- Bump haproxy to version `11.6.0`
- Bump loggregator to version `106.6.1`
- Bump nodejs-offline-buildpack to version `1.7.63`
- Bump php-offline-buildpack to version `4.4.48`
- Bump python-offline-buildpack to version `1.7.47`
- Bump r-offline-buildpack to version `1.1.23`
- Bump routing to version `0.226.0`
- Bump ruby-offline-buildpack to version `1.8.48`
- Bump silk to version `2.40.0`
- Bump staticfile-offline-buildpack to version `1.5.26`

Component	Version
ubuntu-xenial stemcell	~621
backup-and-restore-sdk	1.18.26
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.15
capi	1.117.1
cf-autoscaling	241
cf-cli (v7/v8)*	1.33.0
cf-networking	2.40.0
cflinuxfs3	0.264.0
credhub	2.9.4
diego	2.53.1
dotnet-core-offline-buildpack	2.3.36
garden-runc	1.19.30
go-offline-buildpack	1.9.37
haproxy	11.6.0
java-offline-buildpack	4.42
log-cache	2.11.4

Component	Version
loggregator	106.6.1
loggregator-agent	6.3.4
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32
nodejs-offline-buildpack	1.7.63
notifications	62
notifications-ui	40
php-offline-buildpack	4.4.48
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.47
r-offline-buildpack	1.1.23
routing	0.226.0
ruby-offline-buildpack	1.8.48
silk	2.40.0
smb-volume	3.1.0
smoke-tests	4.3.0
staticfile-offline-buildpack	1.5.26
statsd-injector	1.11.16
syslog	11.7.5
system-metrics-scraper	3.2.3
uaa	74.5.26

[*] The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.1

Release Date: 10/20/2021

- **[Security Fix]** CAPI - Cap label selectors at 50 in queries and improve label selector performance to mitigate DOS vulnerability CVE-2021-22101
- **[Feature Improvement]** HTTP/2 toggle disables Diego container proxy ALPN
- **[Feature Improvement]** Set default for system metrics scrape interval to 15 seconds
- **[Bug Fix]** CAPI - Some metrics for CAPI were not being properly emitted
- **[Bug Fix]** Fix certificate rotation by fixing CredHub's import of concatenated certificates
- **[Bug Fix]** Fix "System metrics scrape interval" configuration in manifest
- Bump backup-and-restore-sdk to version `1.18.22`
- Bump bpm to version `1.1.14`
- Bump capi to version `1.117.1`
- Bump cflinuxfs3 to version `0.262.0`
- Bump credhub to version `2.9.4`
- Bump log-cache to version `2.11.4`
- Bump nginx-offline-buildpack to version `1.1.32`
- Bump nodejs-offline-buildpack to version `1.7.61`
- Bump push-usage-service-release to version `674.0.23`
- Bump pxc to version `0.39.0`
- Bump python-offline-buildpack to version `1.7.46`
- Bump r-offline-buildpack to version `1.1.22`
- Bump ruby-offline-buildpack to version `1.8.47`
- Bump uaa to version `74.5.26`

Component	Version
ubuntu-xenial stemcell	621.0
backup-and-restore-sdk	1.18.22
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.14
capi	1.117.1
cf-autoscaling	239
cf-cli (v7/v8)*	1.33.0
cf-networking	2.38.0
cflinuxfs3	0.262.0

Component	Version
credhub	2.9.4
diego	2.53.0
dotnet-core-offline-buildpack	2.3.34
garden-runc	1.19.30
go-offline-buildpack	1.9.34
haproxy	11.4.4
java-offline-buildpack	4.42
log-cache	2.11.4
loggregator-agent	6.3.4
loggregator	106.6.0
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.32
nodejs-offline-buildpack	1.7.61
notifications-ui	40
notifications	62
php-offline-buildpack	4.4.45
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.23
pxc	0.39.0
python-offline-buildpack	1.7.46
r-offline-buildpack	1.1.22
routing	0.224.0
ruby-offline-buildpack	1.8.47
silk	2.38.0
smb-volume	3.1.0
smoke-tests	4.3.0

Component	Version
staticfile-offline-buildpack	1.5.24
statsd-injector	1.11.16
syslog	11.7.5
system-metrics-scraper	3.2.3
uaa	74.5.26

[*] The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

v2.12.0

Release Date: October 4, 2021

- See New Features in TAS for VMs v2.12
- See Breaking Changes

Component	Version
ubuntu-xenial stemcell	621.0
backup-and-restore-sdk	1.18.18
binary-offline-buildpack	1.0.40
bosh-dns-aliases	0.0.4
bosh-system-metrics-forwarder	0.0.20
bpm	1.1.13
capi	1.117.0
cf-autoscaling	239
cf-cli (v7/v8)[*]	1.33.0
cf-networking	2.38.0
cflinuxfs3	0.259.0
credhub	2.9.1
diego	2.53.0
dotnet-core-offline-buildpack	2.3.34
garden-runc	1.19.30
go-offline-buildpack	1.9.34
haproxy	11.4.4
java-offline-buildpack	4.42
log-cache	2.11.2
loggregator-agent	6.3.4

Component	Version
loggregator	106.6.0
mapfs	1.2.6
metric-registrar	1.2.2
metrics-discovery	3.0.6
mysql-monitoring	9.15.0
nats	40
nfs-volume	7.1.1
nginx-offline-buildpack	1.1.31
nodejs-offline-buildpack	1.7.57
notifications-ui	40
notifications	62
php-offline-buildpack	4.4.45
push-apps-manager-release	675.0.1
push-usage-service-release	674.0.20
pxc	0.37.0
python-offline-buildpack	1.7.45
r-offline-buildpack	1.1.21
routing	0.224.0
ruby-offline-buildpack	1.8.46
silk	2.38.0
smb-volume	3.1.0
smoke-tests	4.3.0
staticfile-offline-buildpack	1.5.24
statsd-injector	1.11.16
syslog	11.7.5
system-metrics-scraper	3.2.3
uaa	74.5.25

* The `cf-cli` version corresponds to the commercial distribution on VMware Tanzu Network.

How to Upgrade

To upgrade to TAS for VMs v2.12, see Configuring TAS for VMs for Upgrades.

When upgrading to TAS for VMs v2.12, be aware of the following upgrade considerations:

- If you previously used an earlier version of TAS for VMs, you must first upgrade to TAS for VMs v2.11 to successfully upgrade to TAS for VMs v2.12.
- Upgrade the cf CLI to the latest cf CLI v7 release, the latest cf CLI v8 release, or the commercial cf CLI distribution available on VMware Tanzu Network.
- To minimize downtime for developers pushing apps, upgrade from TAS for VMs v2.11.9 or later. Upgrading from earlier patch versions can result in an `Unknown Error` when pushing apps.
- Some partner service tiles may be incompatible with TAS for VMs v2.12. VMware is working with partners to ensure their tiles are updated to work with the latest versions of TAS for VMs. For information about which partner service releases are currently compatible with TAS for VMs v2.12, review the appropriate partners services release documentation at https://docs.pivotal.io or contact the partner organization that produces the tile.

New Features in TAS for VMs v2.12

TAS for VMs v2.12 includes the following major features:

TAS for VMs Is Compatible with cf CLI v8

TAS for VMs v2.12 paired with cf CLI v8 allows you to do the following:

- Push apps with end-to-end HTTP/2 routing
- Assign the Space Supporter role to users
- Manage services asynchronously

For more information, see Upgrading to cf CLI v8.

Gorouter Supports HTTP/2

Breaking Change:: See Envoy Advertises HTTP/2 Support Over ALPN below.

In TAS for VMs v2.12 and later, HTTP/2 support is enabled by default. HTTP/2 is the second major version of the the HTTP protocol.

HTTP/2 features the following improvements over HTTP/1.1:

- Uses a binary data format instead of plain text
- Compresses headers
- Multiplexes multiple HTTP requests over a single TCP connection

Together, these improvements can improve response times for some apps.

For more information about the HTTP/2 protocol, see RFC 7540.

For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.

For information about routing HTTP/2 traffic to your TAS for VMs apps, see Routing HTTP/2 and gRPC Traffic to Apps.

Gorouter Supports TLS v1.3

In TAS for VMs v2.12, the Gorouter supports TLS v1.3. New installations of TAS for VMs use TLS v1.3 for the Gorouter by default. If you are upgrading to TAS for VMs v2.12, the Gorouter uses TLS v1.2 by default.

You can select which versions of TLS that the Gorouter uses when you configure TAS for VMs. Selecting support for TLS v1.3 only is a beta feature in TAS for VMs v2.12.

For more information, see (Beta) Gorouter Can Support TLS v1.3 Connections Only below.

New User Role: Space Supporter

TAS for VMs v2.12 introduces the Space Supporter role. Users with the Space Supporter role can do the following:

- View app logs and audit events
- Start, stop, and restart apps
- Scale apps
- Read, bind, and unbind existing service instances

Users with the Space Supporter role cannot do any of the following:

- View credentials or app data
- Edit app source code
- SSH into app instances
- View the app environment
- Create or access service keys
- Create or update services
- Delete apps or services

The Space Supporter role is only available for the Cloud Controller V3 API. If a user with this role tries to access a V2 endpoint, the API returns a `403` error.

For more information, see User Roles in *Orgs, Spaces, Roles, and Permissions*.

TAS for VMs Version is in Apps Manager UI

You can find the current version of TAS for VMs in the footer of the Apps Manager UI.

Secure Endpoint for the Metric Registrar

TAS for VMs v2.12 allows operators to register a secure endpoint for the Metric Registrar CLI plugin to ingest app metrics. You can use the `cf register-metrics-endpoint` command to specify an internal port in your app when you register the endpoint to the Metric Registrar.

For more information, see Register a Metrics Endpoint in *Using Metric Registrar*.

Reduce Traffic to Syslog Drains

In TAS for VMs v2.12, the **System Logging** pane includes the **Default Loggregator drain metadata** checkbox. This configuration setting controls how much deployment metadata TAS for VMs sends in app and aggregate syslog drains.

If you activate this checkbox, TAS for VMs sends all metadata from your deployment to syslog drains.

If you deactivate this checkbox, TAS for VMs sends a reduced amount of metadata. This can reduce your external database logs by up to 50%.

For more information, see (Optional) Configure System Logging in *Configuring TAS for VMs*.

Aggregate Drains are Moved to the Syslog Binding Cache

Aggregate drains are now stored and retrieved from the syslog binding cache. This means that TAS for VMs deployments that change aggregate drains only deploy on the binding cache/clock VM. TAS for VMs does not deploy all VMs on the system. As a result, TAS for VMs deploys more quickly.

This change also removes the Log Cache drain when not in use, reducing BOSH error logs when syslog ingestion for Log Cache is not in use.

Supported cf CLI Container Images

VMware supports the following container images that contain the cf CLI:

- cloudfoundry/cli:8.0.0
- cloudfoundry/cli:7.3.0

VMware maintains these container images and updates them with the latest security patches.

Breaking Changes

TAS for VMs v2.12 includes the following breaking changes:

(Beta) Gorouter Can Support TLS v1.3 Connections Only

TLS v1.3 is not compatible with some versions of Java. If you configure TAS for VMs to support TLS v1.3 only, you might encounter errors with Java apps. For more information, see JSSE Client does not accept status_request extension in CertificateRequest messages for TLS 1.3 in the JDK Bug System.

The tile property that controls the TLS version in TAS for VMs changes in TAS for VMs v2.12. You must update any stored configuration files to reflect the change.

Envoy Advertises HTTP/2 Support Over ALPN

Envoy, the Diego container proxy, advertises HTTP/2 support using Application-Layer Protocol Negotiation (ALPN) for all apps. Internal clients that access the Envoy TLS port directly must negotiate down to HTTP/1.1 for apps that do not support HTTP/2. The Envoy TLS port is typically `61001`. Clients that connect to apps using the Gorouter are not affected.

Gorouter No Longer De-chunks Short Chunked Responses

In previous versions of TAS for VMs, the Gorouter de-chunked short chunked responses, set a Content-Length header, and sent a traditional body. This capability was available when Gorouter used Golang v1.15, which is out of support.

For versions of TAS for VMs that contain routing-release v0.214.0 and later, the Gorouter uses Golang v1.16 which sends a chunked response. If your clients or proxies that access apps cannot handle a chunked response, or expect a Content-Length header, they break.

For more information, see Clients receive responses with no Content-Length header and a chunked encoded body after upgrading Tanzu Application Service for VMs in the Knowledge Base.

Known Issues

TAS for VMs v2.12 includes the following known issues:

HAProxy Does Not Support HTTP/2

HAProxy is not configured to support HTTP/2 ingress traffic. HAProxy also does not send HTTP/2 traffic to the Gorouter, even when HTTP/2 is enabled.

To work around this issue, you can use an external load balancer to support HTTP/2 traffic. For more information, see Configure Load Balancers in *Configuring HTTP/2 Support*.

This issue is resolved in TAS for VMs v2.12.2 and later.

Pre-Start Scripts Fail on the policy-server Job

When upgrading to TAS for VMs v2.12.2, the `policy-server` pre-start script runs a database migration that drops a stored procedure that is no longer needed. If your `networkpolicyserver` database does not have the stored procedure, you might see the following error in `diego_database policy-server` stdout logs:

```
PROCEDURE networkpolicyserver.drop_destination_index does not exist handling 66
```

To work around this error, add the migration to your `networkpolicyserver.gorp_migrations` table and skip the migration.

For more information, see “pre-start scripts failed. Failed Jobs: policy-server” error Upgrading to CF Networking Release 2.40.0 in Tanzu Application Service for VMs in the Knowledge Base.

Gorouter Sets an Invalid X-B3-SpanID Header

An issue with the Gorouter’s implementation of `X-B3-SpanId` and `X-B3-TraceId` headers can cause invalid span IDs to be set after updating the `X-B3-TraceId` header to the new 16-byte standard. As a result, some applications and libraries invalidate the `X-B3-SpanId` value, breaking traces of the application.

This issue affects versions of TAS for VMs that contain routing-release v0.227.0 and v0.228.0.

java-offline-buildpack v4.52 Disallows Spring Auto Reconfiguration by Default

TAS for VMs v2.12.18 includes `java-offline-buildpack` v4.52, which disallows Spring Auto Reconfiguration by default. This change creates the following issues:

- TAS for VMs v2.12.18 is incompatible with some service tiles, including Spring Cloud Data Flow (SCDF).
- TAS for VMs cannot bind apps that use `java-offline-buildpack` v4.52 to service instances.

This known issue does not affect apps until you re-stage them.

Because the effects of this change are so disruptive, a patch release to fix this known issue is currently in development.

To temporarily restore the previous functionality of Spring Auto Reconfiguration for your apps:

1. Set the `JBP_CONFIG_SPRING_AUTO_RECONFIGURATION` environment variable for your apps by running one of the following commands:
    - To restore the previous functionality of Spring Auto Reconfiguration on a per-app basis, run:

      ```
      cf set-env APP-NAME JBP_CONFIG_SPRING_AUTO_RECONFIGURATION '{enabled: true}'
      ```

      Where `APP-NAME` is the name of your app.
    - To restore the previous functionality of Spring Auto Reconfiguration for all apps in your TAS for VMs deployment, run:

      ```
      cf set-staging-environment-variable-group '{"JBP_CONFIG_SPRING_AUTO_RECONFIGURATION":"{enabled: true}"}'
      ```

For more information about the deprecation of Spring Cloud Connectors and Spring Auto Reconfiguration in `java-offline-buildpack` v4.52, see Java Buildpack - Deprecation of Spring Cloud Connectors & Spring Auto Reconfiguration in the VMware Tanzu Knowledge Base.

VMware Tanzu Application Service for Windows v2.12 Release Notes

This topic contains release notes for VMware Tanzu Application Service for VMs [Windows] v2.12.

Because VMware uses the Percona Distribution for MySQL, expect a time lag between Oracle releasing a MySQL patch and VMware releasing TAS for VMs [Windows] containing that patch.

> ⚠ **Warning:** Windows stemcells v2019.44 and later include a version of `tar` that is incompatible with winfs2019-release v2.33.1 and earlier. For more information, see Windows Stemcell v2019.44 is Incompatible with winfs2019-release v2.33.1 and Earlier below.

Before you install the tile, review the Windows Stemcell Compatibility Matrix.

Releases

2.12.20

Release Date: 03/21/2023

- Bump diego to version `2.72.0`
- Bump envoy-nginx to version `0.15.0`
- Bump garden-runc to version `1.25.0`
- Bump hwc-offline-buildpack to version `3.1.28`
- Bump windowsfs-release to version `2.46.0`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.72.0	
envoy-nginx	0.15.0	
event-log	0.9.0	
garden-runc	1.25.0	
windows-syslog	1.1.13	

hwc-offline-buildpack	3.1.28	▼ 3.1.28
		`* Updates github-config (#104)`
		`* Bump github.com/onsi/gomega from 1.24.2 to 1.26.0`
		`Packaged binaries:`
		`| name | version | cf_stacks |`
		`|-|-|-|`
		`| hwc | 21.0.0 | windows, windows2016 |`
		`Default binary versions:`
		`| name | version |`
		`|-|-|`
		`| hwc | 21.0.0 |`
		`* Uncached buildpack SHA256: 92a52f3346131abbe2472b1f12a2ff4e304aeb6c497b7e3f10a3537be8c1e721`
		`* Uncached buildpack SHA256: 37813ac6ded74a0e87924a3a9cec19afb6d4e6dfc6a8ef8da83f7d02fecfdc82`

metrics-discovery	3.2.7	
smoke-tests	4.8.2	
loggregator-agent	6.5.8	
winc	2.10.0	
windows-utilities	0.14.0	
windowsfs-release	2.46.0	

2.12.19

Release Date: 02/28/2023

- Bump garden-runc to version `1.23.0`
- Bump windows-syslog to version `1.1.13`
- Bump metrics-discovery to version `3.2.7`
- Bump loggregator-agent to version `6.5.8`
- Bump winc to version `2.10.0`
- Bump windowsfs-release to version `2.44.0`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.71.0	
envoy-nginx	0.14.0	
event-log	0.9.0	
garden-runc	1.23.0	
windows-syslog	1.1.13	
windowsfs-release	2.44.0	

▼ v1.1.13
`* update golang to 1.20.1`
`**Full Changelog**: https://github.com/cloudfoundry/windows-syslog-release/compare/v1.1.12...v1.1.13`
▼ v1.1.12
`## What's Changed`
`* use go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/windows-syslog-release/pull/18`
`**Full Changelog**: https://github.com/cloudfoundry/windows-syslog-release/compare/v1.1.11...v1.1.12`		

hwc-offline-buildpack	3.1.27	

Component	Version	Release Notes
metrics-discovery	3.2.7	▼ v3.2.7 * update golang to 1.20.1 **Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.6...v3.2.7### ▼ v3.2.6 ## What's Changed * Upgrade to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/104 **Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.5...v3.2.6 ▼ v3.2.5 ## What's Changed * Update dependencies * Expire individual metrics by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/103 **Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.4...v3.2.5
smoke-tests	4.8.2	
loggregator-agent	6.5.8	▼ v6.5.8 ## What's Changed * update dependencies * Upgrade to go 1.20.1 by @rroberts2222 in https://github.com/cloudfoundry/loggregator-agent-release/pull/224 **Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.7...v6.5.8
winc	2.10.0	
windows-utilities	0.14.0	

v2.12.18

Release Date: 02/09/2023

- **[Bug Fix]** Allows docker app workloads without a `sh` binary in the docker image to execute properly.
- Bump loggregator-agent to version `6.5.7`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.71.0	
envoy-nginx	0.14.0	
event-log	0.9.0	
garden-runc	1.22.9	
windows-syslog	1.1.11	
hwc-offline-buildpack	3.1.27	
metrics-discovery	3.2.4	
smoke-tests	4.8.2	

loggregator-agent	6.5.7	▼ v6.5.7
		`## What's Changed`
		`* Sanitize ProcID in syslog messages so messages with utf-8 in the source_type are not dropped by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/202`
		`* Update dependencies`
		`**Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.6...v6.5.7`

winc	2.9.0	
windows-utilities	0.14.0	
windowsfs-release	2.42.0	

v2.12.17

Release Date: 01/30/2023

- Bump garden-runc to version `1.22.9`
- Bump windowsfs-release to version `2.42.0`

Component	Version
windows2019 stemcell	2019.44
diego	2.71.0

Component	Version
envoy-nginx	0.14.0
event-log	0.9.0
garden-runc	1.22.9
windows-syslog	1.1.11
hwc-offline-buildpack	3.1.27
metrics-discovery	3.2.4
smoke-tests	4.8.2
loggregator-agent	6.5.6
winc	2.9.0
windows-utilities	0.14.0
windowsfs-release	2.42.0

v2.12.16

Release Date: 01/17/2023

- Bump diego to version `2.71.0`
- Bump garden-runc to version `1.22.7`
- Bump smoke-tests to version `4.8.2`
- Bump loggregator-agent to version `6.5.6`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.71.0	
envoy-nginx	0.14.0	
event-log	0.9.0	
garden-runc	1.22.7	
windows-syslog	1.1.11	
hwc-offline-buildpack	3.1.27	
metrics-discovery	3.2.4	
smoke-tests	4.8.2	▼ 4.8.2 `Port assets/ruby_simple to Ruby 3`

Component	Version	Release Notes
loggregator-agent	6.5.6	▼ v6.5.6 ## What's Changed * fix scraping with non-positive intervals to preserve non-scraping behavior by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/174 * updated some dependencies. **Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.5...v6.5.6
winc	2.9.0	
windows-utilities	0.14.0	
windowsfs-release	2.41.0	

v2.12.15

Release Date: 12/15/2022

- **[Security Fix]** Fix CVE-2022-31733: Unsecured Application Port
- Bump diego to version `2.70.0`
- Bump envoy-nginx to version `0.14.0`
- Bump hwc-offline-buildpack to version `3.1.27`
- Bump metrics-discovery to version `3.2.4`
- Bump loggregator-agent to version `6.5.5`
- Bump winc to version `2.9.0`
- Bump windowsfs-release to version `2.40.0`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.70.0	
envoy-nginx	0.14.0	
event-log	0.9.0	
garden-runc	1.22.5	
windows-syslog	1.1.11	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>hwc-offline-buildpack</td><td>3.1.27</td><td>▼ 3.1.27 * Add hwc 21.0.0, remove hwc 20.0.0 for stack(s) windows2016, windows (https://www.pivotaltracker.com/story/show/183726731) * Bumps default version to match new HWC version * Bumps go.mod go version to 1.19 Packaged binaries: | name	version	cf_stacks
hwc	21.0.0	windows, windows2016 | Default binary versions: | name	version
-	-
hwc	21.0.0	 * Uncached buildpack SHA256: ae83488a72f50d1725fb37fc35e819133ed07af82d872b9fe7fb34e9de18b92e * Uncached buildpack SHA256: 2e2e474d7677112021cc892627eddef0768e28835b6ad98117a260ea022e4463</td></tr>
<tr><td>metrics-discovery</td><td>3.2.4</td><td>▼ v3.2.4
- bump-golang to v0.114.0 for golang 1.19.4
- Bump github.com/nats-io/nats.go from 1.19.0 to 1.21.0 in /src
- Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src
- Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src
- Bump github.com/prometheus/client_golang from 1.13.1 to 1.14.0 in /src
- Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src
- Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1</td></tr>
<tr><td>smoke-tests</td><td>4.8.1</td><td></td></tr>
</table>

Component	Version	Release Notes

loggregator-agent	6.5.5	▼ v6.5.5
		`  - bump-golang to v0.114.0 for golang 1.19.4`
		`  - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src`
		`  - Bump github.com/valyala/fasthttp from 1.41.0 to 1.43.0 in /src`
		`  - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src`
		`  - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src`
		`  - Bump github.com/prometheus/client_model from 0.2.0 to 0.3.0 in /src`
		`  - Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1`

winc	2.9.0	
windows-utilities	0.14.0	

v2.12.14

Release Date: 11/10/2022

- Bump garden-runc to version `1.22.5`
- Bump windows-syslog to version `1.1.11`
- Bump hwc-offline-buildpack to version `3.1.26`
- Bump metrics-discovery to version `3.2.3`
- Bump smoke-tests to version `4.8.1`
- Bump loggregator-agent to version `6.5.4`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.69.0	
envoy-nginx	0.13.0	
event-log	0.9.0	
garden-runc	1.22.5	
windows-syslog	1.1.11	
windowsfs-release	2.39.0	

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td>hwc-offline-buildpack</td><td>3.1.26</td><td>▼ 3.1.26 * Update libbuildpack Packaged binaries: | name	version	cf_stacks
hwc	20.0.0	windows, windows2016 | Default binary versions: | name	version
-	-
hwc	20.0.0	 * Uncached buildpack SHA256: b9b2cec9ada73d9a2933a14e8e56f025c35b02d8bed7e74e20b093a23e13ec43 * Uncached buildpack SHA256: f633f0f686fc9539ec8f4ef205e778c820602e51434730fd69f7caad4cfb3d4f</td></tr>
<tr><td>metrics-discovery</td><td>3.2.3</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.8.1</td><td>▼ 4.8.1
Create bosh final release 4.8.1
▼ 4.8.0
Create bosh final release 4.8.0
▼ 4.7.0
Create bosh final release 4.7.0</td></tr>
<tr><td>loggregator-agent</td><td>6.5.4</td><td></td></tr>
<tr><td>winc</td><td>2.8.0</td><td></td></tr>
<tr><td>windows-utilities</td><td>0.14.0</td><td></td></tr>
<tr><td>windowsfs-release</td><td>2.40.0</td><td></td></tr>
</table>

v2.12.13

Release Date: 10/19/2022

- **[Feature Improvement]** Add option for file logging and improved event logging. For more information about syslog, see Optional TAS for VMs [Windows] 3.0 compatible syslog option.
- Bump diego to version `2.69.0`

- Bump envoy-nginx to version `0.13.0`
- Bump garden-runc to version `1.22.4`
- Bump windows-syslog to version `1.1.7`
- Bump metrics-discovery to version `3.2.1`
- Bump loggregator-agent to version `6.5.1`
- Bump winc to version `2.8.0`
- Bump windowsfs-release to version `2.37.0`

Component	Version
windows2019 stemcell	2019.44
diego	2.69.0
envoy-nginx	0.13.0
event-log	0.9.0
garden-runc	1.22.4
windows-syslog	1.1.7
hwc-offline-buildpack	3.1.25
metrics-discovery	3.2.1
smoke-tests	4.5.0
loggregator-agent	6.5.1
winc	2.8.0
windows-utilities	0.14.0
windowsfs-release	2.37.0

v2.12.12

Release Date: 09/21/2022

- **[Security Fix]** Bump Cloud Controller Ruby version to 2.7.6 and Go to 1.18.5
- **[Security Fix]** Update Content-Security-Policy
- **[Breaking Change]** If you have configured an app log rate limit that measures app log rates in lines per second, Diego immediately drops app logs that exceed the app log rate limit. For more information, see Diego Drops App Logs That Exceed the App Log Rate Limit below.
- Bump diego to version `2.66.3`
- **[Feature]** Enables TLS for all internal MySQL galera and monitoring components
- **[Feature Improvement]** Bump golang to 1.18 for diego, routing, cf-networking, and silk
- **[Known Issue]** If Git is not installed in the `PATH` environment variable for your Windows stemcell when you deploy TAS for VMs [Windows], you may encounter a version control

system (VCS) stamping failure. For more information, see Windows Stemcells Without Git Installed Cause VSC Stamping Failures below.

- Bump envoy-nginx to version `0.11.0`
- Bump garden-runc to version `1.22.2`
- Bump hwc-offline-buildpack to version `3.1.25`
- Bump metrics-discovery to version `3.2.0`
- Bump loggregator-agent to version `6.5.0`
- Bump winc to version `2.7.0`
- Bump windowsfs-release to version `2.35.0`

Component	Version
windows2019 stemcell	2019.44
diego	2.66.3
envoy-nginx	0.11.0
event-log	0.9.0
garden-runc	1.22.2
hwc-offline-buildpack	3.1.25
metrics-discovery	3.2.0
smoke-tests	4.5.0
loggregator-agent	6.5.0
winc	2.7.0
windows-utilities	0.14.0
windowsfs-release	2.36.0

v2.12.11

Release Date: 08/10/2022

- Bump metrics-discovery to version `3.1.1`
- Bump loggregator-agent to version `6.4.3`

Component	Version
windows2019 stemcell	2019.44
diego	2.62.0
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.8
hwc-offline-buildpack	3.1.24

Component	Version
metrics-discovery	3.1.1
smoke-tests	4.5.0
loggregator-agent	6.4.3
winc	2.5.0
windows-utilities	0.14.0

v2.12.10

Release Date: 07/19/2022

- Bump diego to version `2.62.0`
- Bump garden-runc to version `1.20.8`
- Bump metrics-discovery to version `3.1.0`
- Bump loggregator-agent to version `6.4.2`

Component	Version
windows2019 stemcell	2019.44
diego	2.62.0
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.8
hwc-offline-buildpack	3.1.24
metrics-discovery	3.1.0
smoke-tests	4.5.0
loggregator-agent	6.4.2
winc	2.5.0
windows-utilities	0.14.0
windowsfs-release	2.35.0

v2.12.9

Release Date: 06/23/2022

Warning: Upcoming breaking changes! In future patches, no sooner than July 1st 2022, some components will become more strict about the protocols used in TLS communications, causing integrations with systems using older, insecure protocols to fail. Specifically, components using the Go programming language will be updated to Go 1.18, and will no longer support TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases. However, the pre-existing configuration for "TLS versions supported by the Gorouter" will still work. This change may not arrive all at once, as Go is used in systems throughout TAS for VMs. There will be a VMware Knowledge Base article about this change published prior to the changes rolling out. These changes will be clearly designated in the release notes of the versions they ship in; a version of this warning will appear on all patch versions until we are confident no systems remain to be updated.

- Bump diego to version `2.62.0`

Component	Version
windows2019 stemcell	2019.44
diego	2.62.0
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.6
hwc-offline-buildpack	3.1.24
metrics-discovery	3.0.13
smoke-tests	4.5.0
loggregator-agent	6.4.1
winc	2.5.0
windows-utilities	0.14.0
windowsfs-release	2.35.0

v2.12.8

Release Date: 06/09/2022

⚠ **Warning:** Breaking change. This version contains Diego 2.64.0, which bumps to Go 1.18. Go 1.18 no longer supports TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases. We stated earlier that we wouldn't bump to Go 1.18 until July 1, 2022. This TAS for VMs release with Diego 2.64.0 breaks that promise. We apologize. We are rolling back to Diego 2.62.0. If you already successfully deployed to this TAS for VMs release with Diego 2.64.0, then you are safe to continue using it.

- Bump diego to version `2.64.0`
- Bump garden-runc to version `1.20.6`
- Bump metrics-discovery to version `3.0.13`
- Bump loggregator-agent to version `6.4.1`

Component	Version
windows2019 stemcell	2019.44
diego	2.64.0
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.6
hwc-offline-buildpack	3.1.24
metrics-discovery	3.0.13
smoke-tests	4.5.0
loggregator-agent	6.4.1
winc	2.5.0
windows-utilities	0.14.0
windowsfs-release	2.35.0

v2.12.7

Release Date: 04/20/2022

- **[Breaking Change]** Syslog drains configured to use TLS now reject certificates signed with the SHA-1 hash function.
- Bump diego to version `2.62.0`
- Bump hwc-offline-buildpack to version `3.1.24`
- Bump metrics-discovery to version `3.0.10`
- Bump loggregator-agent to version `6.3.11`
- Bump winc to version `2.5.0`

Component	Version	Release Notes
windows2019 stemcell	2019.44	
diego	2.62.0	
envoy-nginx	0.9.0	
event-log	0.9.0	
garden-runc	1.20.3	
hwc-offline-buildpack	3.1.24	

Component	Version	Release Notes

metrics-discovery	3.0.10	▼ v3.0.10
		`- fix bug with large messages (#22)`
		`- bump-golang to v0.100.0(now 1.18)`

smoke-tests	4.5.0	

loggregator-agent	6.3.11	▼ v6.3.11
		`- fix bug with large messages (#89)`
		`- bump-golang to v0.100.0(now 1.18)`

winc	2.5.0	
windows-utilities	0.14.0	
windowsfs-release	2.35.0	

v2.12.6

Release Date: 03/31/2022

- Bump diego to version `2.61.0`
- Bump garden-runc to version `1.20.3`
- Bump hwc-offline-buildpack to version `3.1.23`
- Bump metrics-discovery to version `3.0.9`
- Bump loggregator-agent to version `6.3.10`
- Bump windowsfs-release to version `2.35.0`

Component	Version
windows2019 stemcell	2019.44
diego	2.61.0
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.3
hwc-offline-buildpack	3.1.23
metrics-discovery	3.0.9
smoke-tests	4.5.0
loggregator-agent	6.3.10
winc	2.3.0

Component	Version
windows-utilities	0.14.0
windowsfs-release	2.35.0

v2.12.5

Release Date: 02/28/2022

- **[Feature Improvement]** Golang v1.17 contains stricter IP parsing standards, so syslog drains registered using user-provided services cannot contain IP addresses with leading zeros in any octets. This affects properties that are fed into diego-release, garden-runc-release, winc-release, nats-release, and routing-release.
- **[Bug Fix]** Smoke tests support for TLSv1.3 only option
- Bump diego to version `2.58.1`
- Bump envoy-nginx to version `0.9.0`
- Bump garden-runc to version `1.20.0`
- Bump hwc-offline-buildpack to version `3.1.22`
- Bump metrics-discovery to version `3.0.8`
- Bump smoke-tests to version `4.5.0`
- Bump loggregator-agent to version `6.3.8`

Component	Version
windows2019 stemcell	2019.44
diego	2.58.1
envoy-nginx	0.9.0
event-log	0.9.0
garden-runc	1.20.0
hwc-offline-buildpack	3.1.22
metrics-discovery	3.0.8
smoke-tests	4.5.0
loggregator-agent	6.3.8
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.33.2

v2.12.4

Release Date: 02/08/2022

- **[Security Fix]** Diego - Bump containerd to v1.5.9 to fix CVE-2021-43816
- **[Feature Improvement]** TAS for VMs [Windows] supports compiled releases
- **[Feature Improvement]** Golang v1.17 contains stricter IP parsing standards, so syslog drains registered using user-provided services cannot contain IP addresses with leading zeros in any octets.
- **[Bug Fix]** windowsfs-release compilation issue - Cannot extract through symlink
- Bump diego to version `2.57.0`
- Bump hwc-offline-buildpack to version `3.1.21`
- Bump smoke-tests to version `4.4.0`
- Bump loggregator-agent to version `6.3.7`
- Bump windowsfs-release to version `2.33.2`

Component	Version
windows2019 stemcell	2019.44
diego	2.57.0
envoy-nginx	0.7.0
event-log	0.9.0
garden-runc	1.19.30
hwc-offline-buildpack	3.1.21
metrics-discovery	3.0.7
smoke-tests	4.4.0
loggregator-agent	6.3.7
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.33.2

v2.12.3

Release Date: 12/15/2021

> **Note:** Windows stemcells v2019.44 and later are not compatible with this version of TAS for VMs [Windows]. You must use a Windows stemcell version between v2019.0 and v2019.43 to install this release.

- **[Bug Fix]** Diego - Envoy v1.19 uses the original TCP connection pool so that it can accept more than 1024 downstream connections
- **[Bug Fix]** Smoke Tests uses specified domain for Isolation Segments
- Bump diego to version `2.54.0`

- Bump metrics-discovery to version `3.0.7`
- Bump smoke-tests to version `4.3.1`
- Bump loggregator-agent to version `6.3.5`

Component	Version
windows2019 stemcell	~2019
diego	2.54.0
envoy-nginx	0.7.0
event-log	0.9.0
garden-runc	1.19.30
hwc-offline-buildpack	3.1.20
metrics-discovery	3.0.7
smoke-tests	4.3.1
loggregator-agent	6.3.5
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.31.0

v2.12.2

Release Date: 11/23/2021

Note: Windows stemcells v2019.44 and later are not compatible with this version of TAS for VMs [Windows]. You must use a Windows stemcell version between v2019.0 and v2019.43 to install this release.

- Bump diego to version `2.53.1`
- Bump hwc-offline-buildpack to version `3.1.20`
- Bump windowsfs-release to version `2.31.0`

Component	Version
windows2019 stemcell	2019.0
diego	2.53.1
envoy-nginx	0.7.0
event-log	0.9.0
garden-runc	1.19.30
hwc-offline-buildpack	3.1.20
loggregator-agent	6.3.4

Component	Version
metrics-discovery	3.0.6
smoke-tests	4.3.0
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.31.0

v2.12.1

Release Date: 10/20/2021

> **Note:** Windows stemcells v2019.44 and later are not compatible with this version of TAS for VMs [Windows]. You must use a Windows stemcell version between v2019.0 and v2019.43 to install this release.

- **[Feature Improvement]** HTTP/2 toggle disables Diego container proxy ALPN
- No BOSH release bumps

Component	Version
windows2019 stemcell	2019.0
diego	2.53.0
envoy-nginx	0.7.0
event-log	0.9.0
garden-runc	1.19.30
hwc-offline-buildpack	3.1.18
loggregator-agent	6.3.4
metrics-discovery	3.0.6
smoke-tests	4.3.0
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.29.0

v2.12.0

Release Date: October 4, 2021

> **Note:** Windows stemcells v2019.44 and later are not compatible with this version of TAS for VMs [Windows]. You must use a Windows stemcell version between v2019.0 and v2019.43 to install this release.

- See New Features in TAS for VMs [Windows] v2.12

Component	Version
windows2019 stemcell	2019.0
diego	2.53.0
envoy-nginx	0.7.0
event-log	0.9.0
garden-runc	1.19.30
hwc-offline-buildpack	3.1.18
loggregator-agent	6.3.4
metrics-discovery	3.0.6
smoke-tests	4.3.0
winc	2.3.0
windows-utilities	0.14.0
windowsfs-release	2.29.0

How to Upgrade

The TAS for VMs [Windows] v2.12 tile is available with the release of VMware Tanzu Application Service for VMs (TAS for VMs) v2.12. To use the TAS for VMs [Windows] v2.12 tile, you must install VMware Tanzu Operations Manager v2.10 or later and TAS for VMs v2.12 or later.

New Features in TAS for VMs [Windows] v2.12

TAS for VMs [Windows] v2.12 includes the following major feature:

Gorouter Supports TLS v1.3

In TAS for VMs [Windows] v2.12, the Gorouter supports TLS v1.3. New installations of TAS for VMs use TLS v1.3 for the Gorouter by default. If you are upgrading to TAS for VMs v2.12, the Gorouter uses TLS v1.2 by default.

For more information, see Gorouter Supports TLS v1.3 in *VMware Tanzu Application Service for VMs v2.12 Release Notes*.

Breaking Changes

TAS for VMs [Windows] v2.12 includes the following breaking changes:

(Beta) Gorouter Can Support TLS v1.3 Connections Only

TLS v1.3 is not compatible with some versions of Java. If you configure TAS for VMs [Windows] to support TLS v1.3 only, you might encounter errors with Java apps. For more information, see JSSE

Client does not accept status_request extension in CertificateRequest messages for TLS 1.3 in the JDK Bug System.

The tile property that controls the TLS version in TAS for VMs [Windows] changes in TAS for VMs v2.12. You must update any stored configuration files to reflect the change.

Diego Drops App Logs That Exceed the App Log Rate Limit

As of TAS for VMs [Windows] v2.12.12, if you have configured an app log rate limit that measures app log rates in lines per second, Diego immediately drops app logs that exceed the app log rate limit.

In TAS for VMs [Windows] v2.12.11 and earlier, Diego buffers and releases approximately 5 MB to 10 MB of app logs that exceed the app log rate limit. This behavior has changed in TAS for VMs [Windows] v2.12.12 because Diego has been upgraded to a newer version.

If this change in behavior causes parts of your deployment to fail, VMware recommends that you either modify any automated scripts that rely on app log output or increase the app log rate limit.

For more information about app log rate limits, see App Log Rate Limiting.

Optional TAS for VMs [Windows] 3.0 compatible syslog option

TAS for VMs [Windows] 3.0 includes changes to syslog forwarding. These changes include:

- Changes in syslog format to better match other tiles.
- Allows securely forwarding logs with tls enabled.
- Includes bosh logs to match other tiles and to aid in debugging and auditing the system.

These changes will only be enabled if you turn off `compatibility_mode` in the system logging window of the settings.

The formatting changes are detailed as follows:

- The priority is changed from `kernel/debug(7)` to `user/info(14)`.
- The app name is changed from `Microsoft-Windows-Security-Auditing` to `event_logger`.
- The process number is changed from a numerical process ID to `rs2`.
- Logs contain structured data for instance and deployment details.
- The event log JSON string includes additional fields.
- In the event log JSON string, field names are written in camel case.
- In the event log JSON string, fields may appear in a different order.

The following example shows the previous log format:

```
<7>1 2022-07-06T22:19:38.1413061Z 10.0.4.14 Microsoft-Windows-Security-Auditi
ng 160 - - {"message":"A new process has been created.\r\n\r\nCreator Subjec
t:\r\n\tSecurity ID:\t\tS-1-5-18\r\n\tAccount Name:\t\tVM-7F65ECCF-0D0$\r\n\t
Account Domain:\t\tWORKGROUP\r\n\tLogon ID:\t\t0x3e7\r\n\r\nTarget Subject:\r
\n\tSecurity ID:\t\tS-1-0-0\r\n\tAccount Name:\t\t-\r\n\tAccount Domain:\t\t-
\r\n\tLogon ID:\t\t0x0\r\n\r\nProcess Information:\r\n\tNew Process ID:\t\t0x
1f7c\r\n\tNew Process Name:\tC\r\n\tToken Elevation Type:\t%%1936\r\n\tMandat
ory Label:\t\tS-1-16-16384\r\n\tCreator Process ID:\t0x248\r\n\tCreator Proce
ss Name:\tC\r\n\tProcess Command Line:\t\r\n\r\nToken Elevation Type indicate
s the type of token that was assigned to the new process in accordance with U
ser Account Control policy.\r\n\r\nType 1 is a full token with no privileges
removed or groups disabled.  A full token is only used if User Account Contro
l is disabled or if the user is the built-in Administrator account or a servi
ce account.\r\n\r\nType 2 is an elevated token with no privileges removed or
groups disabled.  An elevated token is used when User Account Control is enab
led and the user chooses to start the program using Run as administrator.  An
elevated token is also used when an application is configured to always requi
re administrative privilege or to always require maximum privilege, and the u
ser is a member of the Administrators group.\r\n\r\nType 3 is a limited token
with administrative privileges removed and administrative groups disabled.  T
he limited token is used when User Account Control is enabled, the applicatio
n does not require administrative privilege, and the user does not choose to
start the program using Run as administrator.","source":"Microsoft-Windows-Se
curity-Auditing"}
```

The following example shows the new log format:

```
<14>1 2022-07-11T22:27:08.279742Z 10.0.4.12 event_logger rs2 - [instance@4745
0 az="us-central1-b" deployment="pas-windows-dfc8956c7081f9369571" director
="" group="windows_diego_cell" id="d0564a0e-684f-4b58-99ee-6a59d1e7caf8"] {"M
achineName":"vm-c5547227-c4fa-44ae-79d0-ee56f96e82a4","Data":[],"Index":16225
7,"Category":"(13312)","CategoryNumber":13312,"EventID":4688,"EntryType":8,"M
essage":"A new process has been created.\r\n\r\nCreator Subject:\r\n\tSecurit
y ID:\t\tS-1-5-18\r\n\tAccount Name:\t\tVM-C5547227-C4F$\r\n\tAccount Domai
n:\t\tWORKGROUP\r\n\tLogon ID:\t\t0x3e7\r\n\r\nTarget Subject:\r\n\tSecurity
ID:\t\tS-1-0-0\r\n\tAccount Name:\t\t-\r\n\tAccount Domain:\t\t-\r\n\tLogon I
D:\t\t0x0\r\n\r\nProcess Information:\r\n\tNew Process ID:\t\t0x1590\r\n\tNew
Process Name:\tC\r\n\tToken Elevation Type:\t%%1936\r\n\tMandatory Label:\t\t
S-1-16-16384\r\n\tCreator Process ID:\t0x1120\r\n\tCreator Process Name:\tC\r
\n\tProcess Command Line:\t\r\n\r\nToken Elevation Type indicates the type of
token that was assigned to the new process in accordance with User Account Co
ntrol policy.\r\n\r\nType 1 is a full token with no privileges removed or gro
ups disabled.  A full token is only used if User Account Control is disabled
or if the user is the built-in Administrator account or a service account.\r
\n\r\nType 2 is an elevated token with no privileges removed or groups disabl
ed.  An elevated token is used when User Account Control is enabled and the u
ser chooses to start the program using Run as administrator.  An elevated tok
en is also used when an application is configured to always require administr
ative privilege or to always require maximum privilege, and the user is a mem
ber of the Administrators group.\r\n\r\nType 3 is a limited token with admini
strative privileges removed and administrative groups disabled.  The limited
token is used when User Account Control is enabled, the application does not
require administrative privilege, and the user does not choose to start the p
rogram using Run as administrator.","Source":"Microsoft-Windows-Security-Audi
ting","ReplacementStrings":["S-1-5-18","VM-C5547227-C4F$","WORKGROUP","0x3e
7","0x1590","C:\Windows\System32\wbem\WMIC.exe","%%1936","0x1120","","S-1-0-
0","-","-","0x0","C:\bosh\bosh-agent.exe","S-1-16-16384"],"InstanceId":468
8,"TimeGenerated":"\/Date(1657578421000)\/","TimeWritten":"\/Date(16575784210
00)\/","UserName":null,"Site":null,"Container":null}
```

Known Issues

TAS for VMs [Windows] v2.12 includes the following known issues:

Upgrades Fail When the Stemcell Does Not Change

If you upgrade to a version of TAS for VMs [Windows] that uses the same stemcell, TAS for VMs [Windows] can fail to create containers, causing the deployment to fail. If there are stemcell changes or if the Microsoft base layer changes, this error is unlikely to occur.

For more information, see Failure to create containers when upgrading with shared Microsoft base image in the VMware Tanzu Knowledge Base.

Smoke Tests Fail When Isolation Segment is Deployed

The Smoke Test errand runs extra, failing tests when TAS for VMs [Windows] is deployed with Isolation Segment. They are “Compute isolation disabled” and “Application Workflow Linux Applications”.

To work around this issue, disable Smoke Tests. For more information, see Windows Tile smoke-tests fails for isolation segment segments in the VMware Tanzu Knowledge Base.

Windows Stemcell v2019.44 is Incompatible with winfs2019-release v2.33.1 and Earlier

Windows stemcells v2019.44 and later include a newer version of `tar` that is incompatible with `winfs2019-release` v2.33.1 and earlier. TAS for VMs [Windows] deployments that use Windows stemcells v2019.44 and later cannot untar `winfs2019-release` v2.33.1 and earlier. Compatible Windows stemcells for `winfs2019-release` v2.33.1 and earlier include v2019.0 through v2019.43.

Windows Stemcells Without Git Installed Cause VSC Stamping Failures

If Git is not installed either on your Windows stemcell or in the `PATH` environment variable for your Windows stemcell when you deploy TAS for VMs [Windows] v2.12.12, you may see the following error:

```
Stderr:         Use -buildvcs=false to disable VCS stamping.
```

This occurs because some TAS for VMs [Windows] v2.12.12 use Go v1.18, which embeds VSC information in binaries. As a result, releases that contain `.git` files require that Git is installed either on your Windows stemcell or in the `PATH` for your Windows stemcell. If you do not have Git installed in either location and have not set the `buildvcs` property to `false`, Go v1.18 fails to build the release.

TAS for VMs [Windows] v2.12.12 contains `windows2019fs-release`. Because `windows2019fs-release` contains `.git` files, deployments of TAS for VMs [Windows] v2.12.12 using Windows stemcells that do not have Git installed on them or in their `PATH` fail with the VSC stamping error above.

Isolation Segment v2.12 Release notes

Caution: This release has been removed from General Support.

Here you will find the release notes for Isolation Segment v2.12.

Because VMware uses the Percona Distribution for MySQL, expect a time lag between Oracle releasing a MySQL patch and VMware releasing VMware Tanzu Application Service for VMs (TAS for VMs) containing that patch.

Releases

2.12.20

Release Date: 03/21/2023

Note: This version of TAS for VMs contains a known issue with Gorouter error handling for backend app requests. Failures that previously returned HTTP Status Codes 496, 499, 503, 525, or 526 may instead return 502. Additionally, stale routes may fail to be pruned properly, which could result in apps unexpectedly returning HTTP Status Codes 502.

- Bump cf-networking to version `3.23.0`
- Bump cflinuxfs3 to version `0.356.0`
- Bump diego to version `2.72.0`
- Bump garden-runc to version `1.25.0`
- Bump mapfs to version `1.2.13`
- Bump nfs-volume to version `7.1.9`
- Bump routing to version `0.259.0`
- Bump silk to version `3.23.0`
- Bump smb-volume to version `3.1.10`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.448	
bpm	1.1.21	
cf-networking	3.23.0	
cflinuxfs3	0.356.0	
diego	2.72.0	
garden-runc	1.25.0	
haproxy	11.10.2	

Component	Version	Release Notes
loggregator-agent	6.5.8	
mapfs	1.2.13	▼ v1.2.13 ## Changes * Golang: Updated to v1.19.4 (#32) * Golang: Updated to v1.19.5 (#37) * Golang: Updated to v1.19.5 (#44) * Golang: Updated to v1.20.1 (#48) ## Dependencies * **mapfs:** Updated to v`98da9f0`. For more information, see [mapfs](https://github.com/cloudfoundry/mapfs).
metrics-discovery	3.2.7	
nfs-volume	7.1.9	

Component	Version	Release Notes

routing	0.259.0	▼ v0.259.0
		## Changes
		- No changes from last version.
		- Fixing CI so that artifacts are generated correctly for github release.
		## Built with go 1.20.1
		Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.258.0...v0.259.0
		## Resources
		- [Download release v0.259.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.259.0).
		▼ v0.258.0
		## Changes
		- Update healthchecker to [0.4.0](https://github.com/cloudfoundry/healthchecker-release/releases/tag/v0.4.0)
		- Increase startup delay default to 30 seconds [PR](https://github.com/cloudfoundry/healthchecker-release/pull/2)
		- Upgrade golang to 1.20.1
		## Bosh Job Spec changes:
		```diff
		diff --git a/jobs/acceptance_tests/spec b/jobs/acceptance_tests/spec
		index 65bf4c30..6a73b9ae 100644
		--- a/jobs/acceptance_tests/spec
		+++ b/jobs/acceptance_tests/spec
		@@ -7,7 +7,7 @@ templates:
		bpm.yml.erb: config/bpm.yml
		packages:
		- - golang-1.19-linux
		+ - golang-1.20-linux
		- acceptance_tests
		- rtr
		- cf-cli-6-linux
		diff --git a/jobs/smoke_tests/spec b/jobs/smoke_tests/spec
		index b16357ed..0426dc99 100644
		--- a/jobs/smoke_tests/spec
		+++ b/jobs/smoke_tests/spec
		@@ -7,7 +7,7 @@ templates:
		bpm.yml.erb: config/bpm.yml
		packages:
		- - golang-1.19-linux
		+ - golang-1.20-linux
		- acceptance_tests
		- cf-cli-6-linux
		```
		## Built with go 1.20.1
		Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.257.0...v

Component	Version	Release Notes

		`0.258.0`
		`## Resources`
		`- [Download release v0.258.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.258.0).`

silk	3.23.0	

smb-volume	3.1.10	▼ v3.1.10
		`## Changes`
		`* Backfill property tests for force_noserverino (#103)`
		`## Dependencies`
		`* **smbbroker:** Updated to v`17e471d`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker).`
		`* **smbdriver:** Updated to v`fcb9ca4`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).`

smoke-tests	4.8.2	
syslog	11.8.8	

2.12.19

Release Date: 02/28/2023

- Bump cf-networking to version `3.22.0`
- Bump cflinuxfs3 to version `0.352.0`
- Bump garden-runc to version `1.23.0`
- Bump loggregator-agent to version `6.5.8`
- Bump metrics-discovery to version `3.2.7`
- Bump routing to version `0.257.0`
- Bump silk to version `3.22.0`
- Bump smb-volume to version `3.1.9`
- Bump syslog to version `11.8.8`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.418	
bpm	1.1.21	
cf-networking	3.22.0	

Component	Version	Release Notes
cflinuxfs3	0.352.0	
diego	2.71.0	
garden-runc	1.23.0	
haproxy	11.10.2	

loggregator-agent	6.5.8	▼ v6.5.8
		`## What's Changed`
		`* update dependencies`
		`* Upgrade to go 1.20.1 by @rroberts2222 in https://github.com/cloudfoundry/loggregator-agent-release/pull/224`
		`**Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.7...v6.5.8`

mapfs	1.2.12	

metrics-discovery	3.2.7	▼ v3.2.7
		`* update golang to 1.20.1`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.6...v3.2.7###`
		▼ v3.2.6
		`## What's Changed`
		`* Upgrade to go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/104`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.5...v3.2.6`
		▼ v3.2.5
		`## What's Changed`
		`* Update dependencies`
		`* Expire individual metrics by @rroberts2222 in https://github.com/cloudfoundry/metrics-discovery-release/pull/103`
		`**Full Changelog**: https://github.com/cloudfoundry/metrics-discovery-release/compare/v3.2.4...v3.2.5`

nfs-volume	7.1.8	

Component	Version	Release Notes

routing	0.257.0	▼ v0.257.0
		`## Changes`
		`- Bumped to build with golang 1.19.6`
		`##    Built with go 1.19.6`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.256.0...v0.257.0`
		`## Resources`
		`- [Download release v0.257.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.257.0).`
		▼ v0.256.0
		`## Changes`
		`- Update healthchecker in release to stable version`
		`##    Built with go 1.19.5`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.255.0...v0.256.0`
		`## Resources`
		`- [Download release v0.256.0 from bosh.io](https://bosh.io/releases/github.com/cloudfoundry/routing-release?version=0.256.0).`

silk	3.22.0	

Component	Version	Release Notes
smb-volume	3.1.9	▼ v3.1.9 ## Changes * Add force_noserverino property in smbdriver job (#102) ## Dependencies * **bosh-template:** Updated to v2.4.0. For more information, see [bosh-template](https://github.com/cloudfoundry/bosh). * **smbdriver:** Updated to v`adc77c7`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver). ▼ v3.1.8 ## Dependencies * **smbdriver:** Updated to v`6cc617a`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver). ▼ v3.1.7 ## Changes * Golang: Updated to v1.19.4 (#76) ## Dependencies * **rspec:** Updated to v3.12.0. For more information, see [rspec](https://github.com/rspec/rspec-metagem). * **smbbroker:** Updated to v`114bb05`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker). * **smbdriver:** Updated to v`f0b92e3`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).
smoke-tests	4.8.2	

Component	Version	Release Notes

syslog	11.8.8	▼ v11.8.8
		`* update to golang 1.20.1`
		`**Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.7...v11.8.8`
		▼ v11.8.7
		`## What's Changed`
		`* use go 1.20 by @rroberts2222 in https://github.com/cloudfoundry/syslog-release/pull/117`
		`**Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.6...v11.8.7`

v2.12.18

Release Date: 02/09/2023

- **[Bug Fix]** Allows docker app workloads without a `sh` binary in the docker image to execute properly.
- Bump loggregator-agent to version `6.5.7`
- Bump routing to version `0.255.0`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.401	
bpm	1.1.21	
cf-networking	3.19.0	
cflinuxfs3	0.350.0	
diego	2.71.0	
garden-runc	1.22.9	
haproxy	11.10.2	

Component	Version	Release Notes

loggregator-agent	6.5.7	▼ v6.5.7
		`## What's Changed`
		`* Sanitize ProcID in syslog messages so messages with utf-8 in the source_type are not dropped by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/202`
		`* Update dependencies`
		`**Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.6...v6.5.7`

mapfs	1.2.12	
metrics-discovery	3.2.4	
nfs-volume	7.1.8	

routing	0.255.0	▼ v0.255.0
		`[Upgrade healthchecker in release](https://github.com/cloudfoundry/routing-release/commit/ddb43e9e746b009d0ea6e6cf8cf8e7eb059ffafc). In order to limit the scope of packages brought in with the introduction of http healthchecker, we migrated the healthchecker package out of cf-networking-helpers into its own release.`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.254.0...v0.255.0`
		`Built with go 1.19.5`

silk	3.19.0	
smb-volume	3.1.6	
smoke-tests	4.8.2	
syslog	11.8.6	

v2.12.17

Release Date: 01/30/2023

- Bump cf-networking to version `3.19.0`
- Bump cflinuxfs3 to version `0.350.0`
- Bump garden-runc to version `1.22.9`
- Bump routing to version `0.254.0`

- Bump silk to version `3.19.0`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.376	
bpm	1.1.21	
cf-networking	3.19.0	
cflinuxfs3	0.350.0	
diego	2.71.0	
garden-runc	1.22.9	
haproxy	11.10.2	
loggregator-agent	6.5.6	
mapfs	1.2.12	
metrics-discovery	3.2.4	
nfs-volume	7.1.8	

routing	0.254.0	▼ v0.254.0
		`Built with go 1.19.5`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.253.0...v0.254.0`
		▼ v0.253.0
		`## What's Changed`
		`* Specs to make maxRetries configurable for endpoints and route-services by @domdom82 in https://github.com/cloudfoundry/routing-release/pull/298`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.252.0...v0.253.0`

silk	3.19.0	
smb-volume	3.1.6	
smoke-tests	4.8.2	
syslog	11.8.6	

v2.12.16

Release Date: 01/17/2023

- Bump cf-networking to version `3.17.0`

- Bump cflinuxfs3 to version `0.347.0`
- Bump diego to version `2.71.0`
- Bump garden-runc to version `1.22.7`
- Bump loggregator-agent to version `6.5.6`
- Bump routing to version `0.252.0`
- Bump silk to version `3.17.0`
- Bump smoke-tests to version `4.8.2`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.364	
bpm	1.1.21	
cf-networking	3.17.0	
cflinuxfs3	0.347.0	
diego	2.71.0	
garden-runc	1.22.7	
haproxy	11.10.2	

loggregator-agent	6.5.6	▼ v6.5.6
		`## What's Changed`
		`* fix scraping with non-positive intervals to preserve non-scraping behavior by @Benjamintf1 in https://github.com/cloudfoundry/loggregator-agent-release/pull/174`
		`* updated some dependencies.`
		`**Full Changelog**: https://github.com/cloudfoundry/loggregator-agent-release/compare/v6.5.5...v6.5.6`

mapfs	1.2.12	
metrics-discovery	3.2.4	
nfs-volume	7.1.8	

routing	0.252.0	▼ v0.252.0
		`## What's Changed`
		`- Improve random source for least connection pool to be thread safe. Thanks Daniel Lynch!`
		`**Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.251.0...v0.252.0`

Component	Version	Release Notes
silk	3.17.0	
smb-volume	3.1.6	
smoke-tests	4.8.2	▼ 4.8.2 `Port assets/ruby_simple to Ruby 3`
syslog	11.8.6	

v2.12.15

Release Date: 12/15/2022

- **[Security Fix]** Fix CVE-2022-31733: Unsecured Application Port
- Bump bpm to version `1.1.21`
- Bump cf-networking to version `3.16.0`
- Bump cflinuxfs3 to version `0.345.0`
- Bump diego to version `2.70.0`
- Bump loggregator-agent to version `6.5.5`
- Bump metrics-discovery to version `3.2.4`
- Bump nfs-volume to version `7.1.8`
- Bump routing to version `0.251.0`
- Bump silk to version `3.16.0`
- Bump syslog to version `11.8.6`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.364	
bpm	1.1.21	
cf-networking	3.16.0	
cflinuxfs3	0.345.0	
diego	2.70.0	
garden-runc	1.22.5	
haproxy	11.10.2	

Component	Version	Release Notes
loggregator-agent	6.5.5	▼ v6.5.5 - bump-golang to v0.114.0 for golang 1.19.4 - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/valyala/fasthttp from 1.41.0 to 1.43.0 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src - Bump github.com/prometheus/client_model from 0.2.0 to 0.3.0 in /src - Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1
mapfs	1.2.12	
metrics-discovery	3.2.4	▼ v3.2.4 - bump-golang to v0.114.0 for golang 1.19.4 - Bump github.com/nats-io/nats.go from 1.19.0 to 1.21.0 in /src - Bump google.golang.org/grpc from 1.50.1 to 1.51.0 in /src - Bump github.com/onsi/ginkgo/v2 from 2.5.0 to 2.5.1 in /src - Bump github.com/prometheus/client_golang from 1.13.1 to 1.14.0 in /src - Bump github.com/onsi/gomega from 1.24.0 to 1.24.1 in /src - Bump golangci/golangci-lint-action from 3.3.0 to 3.3.1
nfs-volume	7.1.8	

Component	Version	Release Notes
routing	0.251.0	▼ v0.251.0 ## What's Changed - When the `router.ca_certs` property switched from a multi-line string of certs, to an array of certs, gorouter started failing to start up if any of the certs provided were invalid. Previously they were ignored. This has been reverted, so that any invalid CA certs are ignored during startup. Thanks @ameowlia! **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.250.0...v0.251.0 ▼ v0.250.0 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.249.0...v0.250.0 ## Built with go 1.19.4 ▼ v0.249.0 ## What's Changed * Switch to healthecker package in cf-networking-helpers by @mariash in https://github.com/cloudfoundry/routing-release/pull/302 * Add healthchecker package to sync-package-specs file by @mariash in https://github.com/cloudfoundry/routing-release/pull/303 * **Potential Breaking Change:** In preperation for mtls between gorouter and routing api, add gorouter backends ca to routing-api. Rendering these certs depends on routing-api consuming a link from gorouter. If you have multiple gorouter instance groups (for example in the case of isolation segments), you will need to rename bosh links to prevent the error "Multiple link providers found. For an example of link renaming, see [this ops file](https://github.com/cloudfoundry/cf-deployment/blob/main/operations/test/add-persistent-isolation-segment-router.yml#L74) by @reneighbor in https://github.com/cloudfoundry/routing-release/pull/300 * Ensure gorouter-healthchecker doesn't restart gorouter forever on failure by @geofffranks in https://github.com/cloudfoundry/routing-release/pull/305 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.248.0...v0.249.0

Component	Version	Release Notes
		▼ v0.248.0 ## What's Changed * Handle nil ca cert in ca_certs property list ▼ v0.247.0 ## What's Changed * gorouter template cleans `router.ca_certs` property to remove empty certificates ▼ v0.246.0 ## What's Changed * Update `router.ca_certs` property to accept and array of certificates instead of a string block. Thanks @peanball! ▼ v0.245.0 ## What's Changed * Gorouter's pre-start script now reserves ports used by other CF components when it increases the number of ephemeral ports available via `/proc/sys/net/ipv4/ip_local_reserved_ports`. This resolves issues when components fail to start up during deploys/monit restarts due to accidental port collisions with outbound traffic from the VM. Thanks @ameowlia ! * Routing-release no longer makes use of the deprecated uaa-go-client, and uses go-uaa instead * The `routing_utils/nats_client` helper utility now supports saving + loading gorouter's routing tables! Thanks @domdom82 ! * Fixed a memory leak with `gorouter` that resulted in HTTP request objects being held open if a client canceled the connection before the App responded. Thanks @geofffranks ! * **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.244.0...v0.245.0 ## Built with go 1.19.3
silk	3.16.0	
smb-volume	3.1.6	
smoke-tests	4.8.1	

Component	Version	Release Notes

syslog	11.8.6	▼ v11.8.6
		`Update golang to 1.19.4`
		`**Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.5...v11.8.6`
		▼ v11.8.5
		`* update dependencies`
		`* update golang to 1.19.3`
		`**Full Changelog**: https://github.com/cloudfoundry/syslog-release/compare/v11.8.4...v11.8.5`

v2.12.14

Release Date: 11/10/2022

- **[Feature]** Add “Max request header size in kb” property to Networking tab to allow operators to specify a limit on the aggregate size of request headers. Requests over this limit receive a 431 status code.
- Bump cf-networking to version `3.14.0`
- Bump cflinuxfs3 to version `0.332.0`
- Bump garden-runc to version `1.22.5`
- Bump loggregator-agent to version `6.5.4`
- Bump mapfs to version `1.2.12`
- Bump metrics-discovery to version `3.2.3`
- Bump routing to version `0.244.0`
- Bump smb-volume to version `3.1.6`
- Bump smoke-tests to version `4.8.1`
- Bump syslog to version `11.8.4`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.305	
bpm	1.1.19	
cf-networking	3.14.0	
cflinuxfs3	0.332.0	
diego	2.69.0	
garden-runc	1.22.5	

Component	Version	Release Notes
haproxy	11.10.2	
loggregator-agent	6.5.4	
mapfs	1.2.12	▼ v1.2.12 ## Changes * Replace `go get` with `go install` (#23) * Update vendored package golang-1-linux (#26) * Update vendored package golang-1-linux (#27) ## Dependencies * **mapfs:** Updated to v`27f8711`. For more information, see [mapfs](https://github.com/cloudfoundry/mapfs).
metrics-discovery	3.2.3	
nfs-volume	7.1.3	
routing	0.244.0	▼ v0.244.0 ## What's Changed * Emit access logs for 431 responses to Loggegator [gorouter PR #331](https://github.com/cloudfoundry/gorouter/pull/331). Thanks @dsabeti ! * Always suspend pruning when nats is down https://github.com/cloudfoundry/routing-release/pull/287. Thanks @ameowlia ! * **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.243.0...v0.244.0 ## Built with go 1.19.2 ▼ v0.243.0 Bumped to go1.19.2 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.242.0...v0.243.0
silk	3.14.0	

Component	Version	Release Notes
smb-volume	3.1.6	▼ v3.1.6 ## Changes * Update vendored package golang-1-linux (#67) * Update vendored package golang-1-linux (#70) ## Dependencies * **bosh-template:** Updated to v2.3.0. For more information, see [bosh-template](https://github.com/cloudfoundry/bosh). * **smbbroker:** Updated to v`89a0251`. For more information, see [smbbroker](https://github.com/cloudfoundry/smbbroker). * **smbdriver:** Updated to v`68ff9d8`. For more information, see [smbdriver](https://github.com/cloudfoundry/smbdriver).
smoke-tests	4.8.1	▼ 4.8.1 Create bosh final release 4.8.1 ▼ 4.8.0 Create bosh final release 4.8.0
syslog	11.8.4	

v2.12.13

Release Date: 10/19/2022

- **[Breaking Change]** A change in behavior to line-based log rate limits has been made. Previously, logs were buffered and released at the allowed rate, now logs exceeding the limit will be dropped.
- Bump cf-networking to version `3.13.0`
- Bump cflinuxfs3 to version `0.328.0`
- Bump diego to version `2.69.0`
- Bump garden-runc to version `1.22.4`
- Bump loggregator-agent to version `6.5.1`
- Bump mapfs to version `1.2.11`
- Bump metrics-discovery to version `3.2.1`
- Bump nfs-volume to version `7.1.3`
- Bump routing to version `0.242.0`

- Bump silk to version `3.14.0`
- Bump smb-volume to version `3.1.5`
- Bump smoke-tests to version `4.7.0`
- Bump syslog to version `11.8.3`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.296	
bpm	1.1.19	
cf-networking	3.13.0	
cflinuxfs3	0.328.0	
diego	2.69.0	
garden-runc	1.22.4	
haproxy	11.10.2	
loggregator-agent	6.5.1	
mapfs	1.2.11	▼ v1.2.11 ## Changes * Update vendored package golang-1-linux (#21) ▼ v1.2.8 ## What's Changed * Bump src/mapfs to `0ee84aa` #18 ▼ v1.2.7 - [Bumps mapfs submodule to master@1600494](https://github.com/cloudfoundry/mapfs/commit/160049400a47577b0f3a8b2948974bc38ce76f18) - [Bump golang from 1.13 to 1.17](https://github.com/cloudfoundry/mapfs-release/commit/c287adda5cbdf345ff1b4985ae93cb72f1618f95)
metrics-discovery	3.2.1	
nfs-volume	7.1.3	

Component	Version	Release Notes
routing	0.242.0	▼ v0.242.0 ## What's Changed - `tcp_router` is now more verbose when running `haproxy_reloader` to assist in diagnosting failed reloads. Thanks @geofffranks! ([PR 9](https://github.com/cloudfoundry/cf-tcp-router/pull/9)) - `gorouter` will now truncate access logs that exceed loggregator + UDP packet limits, so that we no longer drop access log messages sent to the firehose. Thanks @ameowlia @ebroberson! 😺 ([PR 328](https://github.com/cloudfoundry/gorouter/pull/328) and [PR 329](https://github.com/cloudfoundry/gorouter/pull/329)) **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.241.0...v0.242.0 ## Built with go 1.19.1 ▼ v0.241.0 Bumped to go1.19.1 * @plowin submitted [gorouter PR 327](https://github.com/cloudfoundry/gorouter/pull/327) to adjust endpoint-not-unregistered log-level to 'info' **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.240.0...v0.241.0 ▼ v0.240.0 ## What's Changed * @geofffranks and @ameowlia added property `router.max_header_bytes` to the gorouter job. * This value controls the maximum number of bytes the gorouter will read parsing the request header's keys and values, including the request line. * It does not limit the size of the request body. * An additional padding of 4096 bytes is added to this value by go. * Requests with larger headers will result in a 431 status code. **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/v0.239.0...v0.240.0 ## Manifest Property Changes | Job | Property | 0.237.0 | 0.238.0 | | --- | --- | --- | --- |

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td>
<pre>
  | `gorouter` | `router.max_header_bytes` | didn't exist | 1048576 (1MB) |
  ##   Built with go 1.18.6
</pre>
▼ v0.239.0
<pre>
  ## What's Changed
  - Bumped Golang to 1.18.6 to mitigate [CVE-2022-27664](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-27664)
  Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.238.0...v0.239.0
  ##   Built with go 1.18.6
</pre>
▼ v0.238.0
<pre>
  ## What's Changed
  - Gorouter once again supports hairpinning for route-service requests, for more information, see [the proposed update.](https://github.com/cloudfoundry/routing-release/issues/281) `router.route_services_internal_lookup_allowlist` can be used to control which domains of route services can be hairpinned. Thanks @peanball!!
  - Gorouter has a new websocket-specific dial timeout (`websocket_dial_timeout`), configurable separately from the default endpoint dial timeout. Thanks @peanball for this one too!!
  Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.237.0...v0.238.0
  ## Manifest Property Changes
Job	Property	0.237.0	0.238.0
`gorouter`	`websocket_dial_timeout_in_seconds`	didn't exist	Defaults to `endpoint_dial_timeout_in_seconds`'s value
`gorouter`	`router.route_services_internal_lookup_allowlist`	didn't exist	No internal lookups allowed for route services.
  ##   Built with go 1.18.5
</pre>
▼ v0.237.0
<pre>
  ## What's Changed
  - ⚠ Bump to golang 1.18
  Breaking Changes: The routing components are now more strict about the protocols used in TLS communications, causing integrations with systems using older, insecure protocols to fail. These components have been updated t
</pre>
</td></tr>
</table>

Component	Version	Release Notes

		o Go 1.18, and will no longer support TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases.
		  Please see this golang 1.18 release notes [section](https://tip.golang.org/doc/go1.18#tls10) for more information about the golang 1.18 change.
		  ###
		  * Update uaa-go-client; by @joergdw in https://github.com/cloudfoundry/routing-release/pull/277
		  * updated spec files to match packages by @ebroberson in https://github.com/cloudfoundry/routing-release/pull/282
		  Full Changelog: https://github.com/cloudfoundry/routing-release/compare/v0.236.0...v0.237.0
		  ## New Contributors
		  * @joergdw made their first contribution in https://github.com/cloudfoundry/routing-release/pull/277
		  * @ebroberson made their first contribution in https://github.com/cloudfoundry/routing-release/pull/282
		  ## Built with go 1.18.4

silk	3.14.0	

Component	Version	Release Notes
smb-volume	3.1.5	▼ v3.1.5 ## Changes * Update vendored package golang-1-linux (#58) ▼ v3.1.4 ## Release Notes - Fix issue when multiple cf versions are included (#55) ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on the errand VM. ▼ v3.1.3 ## Release Notes - Added support for CF CLI v8 to errands (#45) - Fixed Jammy compilation issues (#53) ## Dependencies - Bump [src/code.cloudfoundry.org/smbbroker](https://github.com/cloudfoundry/smbbroker) (#41, #50) - Bump [src/code.cloudfoundry.org/smbdriver](https://github.com/cloudfoundry/smbdriver) (#47, #48, #51) ▼ v3.1.2 ## Release Notes - Support Bionic Stemcell #16 - Add blobs for the `keyutils` package for both `bionic` and `jammy`. - We now install this package on any VM that runs the `smbdriver` bosh job iff that VM uses a `bionic` or `jammy` stemcell - This should allow the `smbdriver` to reliably mount SMB volumes on those stemcells, as discussed in #16 ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on t

Component	Version	Release Notes
		he errand VM. ▼ v3.1.1 ## Release Notes * Bumps [bosh-template](https://github.com/cloudfoundry/bosh) from 2.2.0 to 2.2.1 (#22) * Bumps [rspec-its](https://github.com/rspec/rspec-its) from 1.2.0 to 1.3.0 (#23) * Bumps [rspec](https://github.com/rspec/rspec-metagem) to 3.11.0. (#37) * Bumps [src/code.cloudfoundry.org/smbdriver](https://github.com/cloudfoundry/smbdriver) to `1e97c5d` (#34) * Bumps [src/code.cloudfoundry.org/smbbroker](https://github.com/cloudfoundry/smbbroker) to `64ba567` (#36) * Bumps automake from 1.15 to 1.15.1 (#43 - fixes Bionic compilation) ## Dependencies - The `smbbrokerpush` and `bbr-smbbroker` errands require either the `cf-cli-7-linux` or `cf-cli-6-linux` job from [cf-cli-release](https://bosh.io/releases/github.com/bosh-packages/cf-cli-release?all=1) to be colocated on the errand VM.
smoke-tests	4.7.0	▼ 4.7.0 Create bosh final release 4.7.0
syslog	11.8.3	

v2.12.12

Release Date: 09/21/2022

- **[Feature Improvement]** Bump golang to 1.18 for diego, routing, cf-networking, and silk
- Bump bpm to version `1.1.19`
- Bump cflinuxfs3 to version `0.319.0`
- Bump garden-runc to version `1.22.0`
- Bump loggregator-agent to version `6.4.4`
- Bump metrics-discovery to version `3.1.2`
- Bump silk to version `3.12.0`
- Bump syslog to version `11.8.2`

Component	Version
ubuntu-xenial stemcell	621.265
bpm	1.1.19
cf-networking	3.11.0
cflinuxfs3	0.319.0
diego	2.62.0
garden-runc	1.22.0
haproxy	11.10.2
loggregator-agent	6.4.4
mapfs	1.2.6
metrics-discovery	3.1.2
nfs-volume	7.1.1
routing	0.236.0
silk	3.12.0
smb-volume	3.1.0
smoke-tests	4.5.0
syslog	11.8.2

v2.12.11

Release Date: 08/10/2022

- Bump cf-networking to version `3.11.0`
- Bump cflinuxfs3 to version `0.312.0`
- Bump haproxy to version `11.10.2`
- Bump loggregator-agent to version `6.4.3`
- Bump metrics-discovery to version `3.1.1`
- Bump routing to version `0.236.0`
- Bump silk to version `3.11.0`
- Bump syslog to version `11.8.1`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.261	
bpm	1.1.18	
cf-networking	3.11.0	
cflinuxfs3	0.312.0	

Component	Version	Release Notes
diego	2.62.0	
garden-runc	1.20.8	
haproxy	11.10.2	
loggregator-agent	6.4.3	
mapfs	1.2.6	
metrics-discovery	3.1.1	
nfs-volume	7.1.1	

Component	Version	Release Notes
routing	0.236.0	▼ v0.236.0 ## What's Changed * Gorouter restart script waits for the gorouter to be running before reloading monit ## Built with go 1.17.12 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.235.0...0.236.0 ▼ 0.235.0 ## What's Changed * Gorouter healthchecker retries connection instead of monit (https://github.com/cloudfoundry/routing-release/pull/275) ## Built with go 1.17.11 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.234.0...0.235.0 ▼ 0.234.0 ## What's Changed * Gorouter: the metrics package now uses `lsof` to monitor file descriptors on MacOS @domdom82 https://github.com/cloudfoundry/gorouter/pull/312 * Bumped the `lager` dependency to resolve issues where the timeFormat flag was not honored, resulting in epoch timestamps vs human readable. Thanks @ameowlia! * Now tested with the bionic stemcell in CI ## Built with go 1.17.11 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.233.0...0.234.0 ▼ 0.233.0 ## What's Changed * TCP Router: Add locking to the haproxy_reloader script to avoid haproxy reload/restart race conditions by @geofffranks in https://github.com/cloudfoundry/routing-release/pull/269 * TCP Router: Bump HAProxy from 1.8.13 to 2.5.4 by @cunnie in https://github.com/cloudfoundry/routing-release/pull/266 * Gorouter: fix proxy round tripper race condition by @ameowlia and @geofffranks in https://github.com/cloudfoundry/gorouter/pull/318

<table>
<tr><th>Component</th><th>Version</th><th>Release Notes</th></tr>
<tr><td></td><td></td><td><pre>
  * Routing API: fix timestamp precision issue that caused routes to be pruned unexpectedly by @geofffranks in https://github.com/cloudfoundry/routing-api/pull/24
  * Routing API: remove `golang.x509ignoreCN` bosh property by @geofffranks and @mariash
  * Routing API: fix bug that caused TCP Router's HAProxy to reload every minute by @jrussett in https://github.com/cloudfoundry/routing-api/pull/26.
  ## Manifest Property Changes
Job	Property	Notes
`routing-api`	`golang.x509ignoreCN`	This property exposed a go debug flag for go version 1.15. Since go 1.16 this go debug flag has had no affect. Removing this bosh property is part of our effort to keep our code base free of cruft.
  ##   Built with go 1.17.10
  Full Changelog: https://github.com/cloudfoundry/routing-release/compare/0.232.0...0.233.0
</pre></td></tr>
<tr><td>silk</td><td>3.11.0</td><td></td></tr>
<tr><td>smb-volume</td><td>3.1.0</td><td></td></tr>
<tr><td>smoke-tests</td><td>4.5.0</td><td></td></tr>
<tr><td>syslog</td><td>11.8.1</td><td></td></tr>
</table>

v2.12.10

Release Date: 07/19/2022

- **[Feature]** Enable telemetry for iptables rules on Diego cells
- **[Bug Fix]** Resolves an issue with HAProxy log rotation creating null bytes and not freeing disk space after rotation
- Bump cf-networking to version `3.9.0`
- Bump cflinuxfs3 to version `0.306.0`
- Bump diego to version `2.62.0`
- Bump garden-runc to version `1.20.8`
- Bump metrics-discovery to version `3.1.0`
- Bump silk to version `3.9.0`

Component	Version
ubuntu-xenial stemcell	621.252

Component	Version
bpm	1.1.18
cf-networking	3.9.0
cflinuxfs3	0.306.0
diego	2.62.0
garden-runc	1.20.8
haproxy	11.6.0
loggregator-agent	6.4.1
mapfs	1.2.6
metrics-discovery	3.1.0
nfs-volume	7.1.1
routing	0.232.0
silk	3.9.0
smb-volume	3.1.0
smoke-tests	4.5.0
syslog	11.7.10

v2.12.9

Release Date: 06/23/2022

⚠ **Warning:** Upcoming breaking changes! In future patches, no sooner than July 1st 2022, some components will become more strict about the protocols used in TLS communications, causing integrations with systems using older, insecure protocols to fail. Specifically, components using the Go programming language will be updated to Go 1.18, and will no longer support TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases. However, the pre-existing configuration for "TLS versions supported by the Gorouter" will still work. This change may not arrive all at once, as Go is used in systems throughout TAS. There will be a VMware Knowledge Base article about this change published prior to the changes rolling out. These changes will be clearly designated in the release notes of the versions they ship in; a version of this warning will appear on all patch versions until we are confident no systems remain to be updated.

- Bump diego to version `2.62.0`

Component	Version
ubuntu-xenial stemcell	621.244

Component	Version
bpm	1.1.18
cf-networking	3.6.0
cflinuxfs3	0.299.0
diego	2.62.0
garden-runc	1.20.6
haproxy	11.6.0
loggregator-agent	6.4.1
mapfs	1.2.6
metrics-discovery	3.0.13
nfs-volume	7.1.1
routing	0.232.0
silk	3.6.0
smb-volume	3.1.0
smoke-tests	4.5.0
syslog	11.7.10

v2.12.8

Release Date: 06/09/2022

⚠ **Warning:** Breaking change. This version contains Diego 2.64.0, which bumps to Go 1.18. Go 1.18 no longer supports TLS 1.0 and 1.1 connections or certificates with a SHA-1 checksum. This is most likely to affect connections with external databases. We stated earlier that we wouldn't bump to Go 1.18 until July 1, 2022. This TAS release with Diego 2.64.0 breaks that promise. We apologize. We are rolling back to Diego 2.62.0. If you already successfully deployed to this TAS release with Diego 2.64.0, then you are safe to continue using it.

- **[Bug Fix]** Fix metric registrar secure scraping with isolation segments
- **[Bug Fix]** Sticky sessions no longer break when used with route-services that return HTTP 4xx/5xx responses
- Bump bpm to version `1.1.18`
- Bump cf-networking to version `3.6.0`
- Bump cflinuxfs3 to version `0.299.0`
- Bump diego to version `2.64.0`
- Bump garden-runc to version `1.20.6`

- Bump loggregator-agent to version `6.4.1`
- Bump metrics-discovery to version `3.0.13`
- Bump routing to version `0.232.0`
- Bump silk to version `3.6.0`
- Bump syslog to version `11.7.10`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.244	
bpm	1.1.18	
cf-networking	3.6.0	
cflinuxfs3	0.299.0	
diego	2.64.0	
garden-runc	1.20.6	
haproxy	11.6.0	
loggregator-agent	6.4.1	
mapfs	1.2.6	
metrics-discovery	3.0.13	
nfs-volume	7.1.1	
routing	0.232.0	▼ 0.232.0 ## What's Changed * Fixing issue #250: Return a 503 not a 404 when all instances down by @kecirlotfi in https://github.com/cloudfoundry/routing-release/pull/268 and https://github.com/cloudfoundry/gorouter/pull/314 * Fixing issue https://github.com/cloudfoundry/gorouter/pull/315: Fix route service pruning by @geofffranks ## Manifest Property Changes | Job | Property | default | notes | | --- | --- | --- | --- | | `gorouter` | `for_backwards_compatibility_only.empty_pool_response_code_503` | `0s` | This property was added to enable https://github.com/cloudfoundry/routing-release/pull/268 | ## New Contributors * @kecirlotfi made their first contribution! Thanks so much! ## Built with go 1.17.9 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.231.0...0.232.0

Component	Version	Release Notes
silk	3.6.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	
syslog	11.7.10	

v2.12.7

Release Date: 04/20/2022

- **[Breaking Change]** Syslog drains configured to use TLS now reject certificates signed with the SHA-1 hash function.
- Bump cflinuxfs3 to version `0.285.0`
- Bump diego to version `2.62.0`
- Bump loggregator-agent to version `6.3.11`
- Bump metrics-discovery to version `3.0.10`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.224	
bpm	1.1.16	
cf-networking	3.3.0	
cflinuxfs3	0.285.0	
diego	2.62.0	
garden-runc	1.20.3	
haproxy	11.6.0	

loggregator-agent	6.3.11	▼ v6.3.11
		`- fix bug with large messages (#89)`
		`- bump-golang to v0.100.0(now 1.18)`

mapfs	1.2.6	

metrics-discovery	3.0.10	▼ v3.0.10
		`- fix bug with large messages (#22)`
		`- bump-golang to v0.100.0(now 1.18)`

nfs-volume	7.1.1	
routing	0.231.0	

Component	Version	Release Notes
silk	3.3.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	
syslog	11.7.7	

v2.12.6

Release Date: 03/31/2022

- **[Security Fix]** This release fixes CVE-2022-23806 and CVE-2022-23772.
- **[Bug Fix]** Resolve an issue resulting in tcp-router repeatedly respawning haproxy until it hits a forked process limit
- **[Bug Fix]** Resolves an issue where invalid seeded router group values should caused breaking changes
- **[Bug fix]** Remove x509ignoreCN option in Gorouter
- Bump cf-networking to version `3.3.0`
- Bump cflinuxfs3 to version `0.279.0`
- Bump diego to version `2.61.0`
- Bump garden-runc to version `1.20.3`
- Bump loggregator-agent to version `6.3.10`
- Bump metrics-discovery to version `3.0.9`
- Bump routing to version `0.231.0`
- Bump silk to version `3.3.0`

Component	Version	Release Notes
ubuntu-xenial stemcell	621.224	
bpm	1.1.16	
cf-networking	3.3.0	
cflinuxfs3	0.279.0	
diego	2.61.0	
garden-runc	1.20.3	
haproxy	11.6.0	
loggregator-agent	6.3.10	
mapfs	1.2.6	
metrics-discovery	3.0.9	
nfs-volume	7.1.1	

Component	Version	Release Notes
routing	0.231.0	▼ 0.231.0 ## Bug Fixes - Removed the x509ignoreCN property. Now that `gorouter` is built on golang 1.17, itno longer has any effect on gorouter behavior, and was only adding to confusion inthe properties - Resolve an issue with route-registrar using the same TTL as it's RegistrationIntervalfor tcp routes, leading to unnecessary churn of pruned + re-registered routes. - Resolve an issue with Routing API where upserts to tcp routes were causing changeevents to be emitted when the only change was a bump in TTL. This led to an issuewhere tcp-router was constantly reloading haproxy with every route's heartbeatregistration call. ## Manifest Property Changes | Job | Property | 0.230.0 | 0.231.0 | | --- | --- | --- | --- | | `gorouter` | `golang.x509ignoreCN` | false | No longer exists | | `route_registrar` | `golang.x509ignoreCN` | false | No longer exists | | `tcp_router` | `golang.x509ignoreCN` | false | No longer exists | ### Built with golang 1.17.8 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.230.0...0.231.0 ▼ 0.230.0 ## Feature * update gorouter for prometheus scraping by @Benjamintf1 in https://github.com/cloudfoundry/routing-release/pull/258 ## Bug Fix * Invalid seeded router group manifest values should no longer cause breaking changes by default by @ameowlia in https://github.com/cloudfoundry/routing-release/pull/261 ### Built with golang 1.17.7 **Full Changelog**: https://github.com/cloudfoundry/routing-release/compare/0.229.0...0.230.0
silk	3.3.0	
smb-volume	3.1.0	
smoke-tests	4.5.0	

Component	Version	Release Notes
syslog	11.7.7	

v2.12.5

Release Date: 02/28/2022

- **[Feature Improvement]** Due to routing-release now being built with Golang 1.17, all certificates provided MUST contain SAN entries on them. The previous workaround of setting "Enable temporary workaround for certs without SANs" will no longer function.
- **[Feature Improvement]** Per Golang 1.17's new and stricter IP parsing standards, any IP addrs with leading zeros in any octets will result in a BOSH template failure to allow operators to remove the leading zeros and try again (affects properties fed into diego-release, garden-runc-release, winc-release, nats-release, and routing-release),.
- **[Bug Fix]** Fixes an issue related to the parsing of the X-B3-TraceId and X-B3-SpanId HTTP headers
- **[Bug Fix]** Smoke tests support for TLSv1.3 only option
- Bump cflinuxfs3 to version `0.274.0`
- Bump diego to version `2.58.1`
- Bump garden-runc to version `1.20.0`
- Bump loggregator-agent to version `6.3.8`
- Bump metrics-discovery to version `3.0.8`
- Bump routing to version `0.229.0`
- Bump smoke-tests to version `4.5.0`

Component	Version
ubuntu-xenial stemcell	621.211
bpm	1.1.16
cf-networking	2.43.0
cflinuxfs3	0.274.0
diego	2.58.1
garden-runc	1.20.0
haproxy	11.6.0
loggregator-agent	6.3.8
mapfs	1.2.6
metrics-discovery	3.0.8
nfs-volume	7.1.1
routing	0.229.0

Component	Version
silk	2.43.0
smb-volume	3.1.0
smoke-tests	4.5.0
syslog	11.7.7

v2.12.4

Release Date: 02/07/2022

> **Note:** This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

- **[Security Fix]** Bump routing release to v0.228.0 to address (CVE-2021-44716)
- **[Feature Improvement]** Golang v1.17 contains stricter IP parsing standards, so IP addresses with leading zeros in any octets cause a BOSH template failure. Operators can remove the leading zeros and try deploying again. This affects properties that feed into cf-networking-release, silk-release, loggregator-agent-release, and syslog-release. Syslog drains and metric registrar endpoints registered using user-provided services might also be affected.
- Bump bpm to version `1.1.16`
- Bump cf-networking to version `2.43.0`
- Bump cflinuxfs3 to version `0.272.0`
- Bump diego to version `2.57.0`
- Bump loggregator-agent to version `6.3.7`
- Bump metrics-discovery to version `3.0.7`
- Bump routing to version `0.228.0`
- Bump silk to version `2.43.0`
- Bump smoke-tests to version `4.4.0`
- Bump syslog to version `11.7.7`

Component	Version
ubuntu-xenial stemcell	621.198
bpm	1.1.16
cf-networking	2.43.0
cflinuxfs3	0.272.0
diego	2.57.0
garden-runc	1.19.30
haproxy	11.6.0

Component	Version
loggregator-agent	6.3.7
mapfs	1.2.6
metrics-discovery	3.0.7
nfs-volume	7.1.1
routing	0.228.0
silk	2.43.0
smb-volume	3.1.0
smoke-tests	4.4.0
syslog	11.7.7

v2.12.3

Release Date: 12/15/2021

Note: This version of TAS for VMs contains a known issue that can cause application traces to break. See Gorouter Sets an Invalid X-B3-SpanID Header in *Known Issues*.

- **[Bug Fix]** Fix “pre-start scripts failed. Failed Jobs: policy-server” error Upgrading to CF Networking Release 2.40.0
- **[Bug Fix]** Diego - Envoy v1.19 uses the original TCP connection pool so that it can accept more than 1024 downstream connections.
- **[Bug Fix]** Smoke Tests uses specified domain for Isolation Segments
- Bump cf-networking to version `2.42.0`
- Bump cflinuxfs3 to version `0.268.0`
- Bump diego to version `2.54.0`
- Bump loggregator-agent to version `6.3.5`
- Bump routing to version `0.227.0`
- Bump silk to version `2.41.0`
- Bump smoke-tests to version `4.3.1`
- Bump syslog to version `11.7.6`

Component	Version
ubuntu-xenial stemcell	~621
bpm	1.1.15
cf-networking	2.42.0
cflinuxfs3	0.268.0

Component	Version
diego	2.54.0
garden-runc	1.19.30
haproxy	11.6.0
loggregator-agent	6.3.5
mapfs	1.2.6
metrics-discovery	3.0.6
nfs-volume	7.1.1
routing	0.227.0
silk	2.41.0
smb-volume	3.1.0
smoke-tests	4.3.1
syslog	11.7.6

v2.12.2

Release Date: 11/23/2021

- **[Feature Improvement]** Enable HTTP/2 for HAProxy
- **[Bug Fix]** Breaking Change: Any customers with gorouter certificates lacking a SubjectAltName extension will experience failures upon deployment. As a workaround to complete deployment while new certificates are procured, enable the "Enable temporary workaround for certs without SANs" property in the Networking section of the TAS tile. For more information on updating certs, see the support articlxe Routing and Golang 1.15 X.509 CommonName deprecation
- Bump bpm to version `1.1.15`
- Bump cf-networking to version `2.40.0`
- Bump cflinuxfs3 to version `0.264.0`
- Bump diego to version `2.53.1`
- Bump haproxy to version `11.6.0`
- Bump routing to version `0.226.0`
- Bump silk to version `2.40.0`

Component	Version
ubuntu-xenial stemcell	621.0
bpm	1.1.15
cf-networking	2.40.0
cflinuxfs3	0.264.0

Component	Version
diego	2.53.1
garden-runc	1.19.30
haproxy	11.6.0
loggregator-agent	6.3.4
mapfs	1.2.6
metrics-discovery	3.0.6
nfs-volume	7.1.1
routing	0.226.0
silk	2.40.0
smb-volume	3.1.0
smoke-tests	4.3.0
syslog	11.7.5

v2.12.1

Release Date: 10/20/2021

- **[Feature Improvement]** HTTP/2 toggle disables Diego container proxy ALPN
- Bump bpm to version `1.1.14`
- Bump cflinuxfs3 to version `0.262.0`

Component	Version
ubuntu-xenial stemcell	621.0
bpm	1.1.14
cf-networking	2.38.0
cflinuxfs3	0.262.0
diego	2.53.0
garden-runc	1.19.30
haproxy	11.4.4
loggregator-agent	6.3.4
mapfs	1.2.6
metrics-discovery	3.0.6
nfs-volume	7.1.1
routing	0.224.0
silk	2.38.0

Component	Version
smb-volume	3.1.0
smoke-tests	4.3.0
syslog	11.7.5

v2.12.0

Release Date: October 4, 2021

- See New Features in Isolation Segment v2.12
- See Breaking Changes

Component	Version
ubuntu-xenial stemcell	621.0
bpm	1.1.13
cf-networking	2.38.0
cflinuxfs3	0.259.0
diego	2.53.0
garden-runc	1.19.30
haproxy	11.4.4
loggregator-agent	6.3.4
mapfs	1.2.6
metrics-discovery	3.0.6
nfs-volume	7.1.1
routing	0.224.0
silk	2.38.0
smb-volume	3.1.0
smoke-tests	4.3.0
syslog	11.7.5

About Isolation Segment

The Isolation Segment v2.12 tile is available for installation with Ops Manager v2.10.

Isolation segments provide dedicated pools of resources where you can deploy apps and isolate workloads. Using isolation segments separates app resources as completely as if they were in different deployments but avoids redundant management and network complexity. For more information about isolation segments, see Isolation Segments in *TAS for VMs Security*.

For more information about using isolation segments in your deployment, see Managing Isolation Segments.

How to Install

To install Isolation Segment v2.12, see Installing Isolation Segment.

To install Isolation Segment v2.12, you must first install Ops Manager v2.10. For more information, see the Ops Manager documentation.

New Features in Isolation Segment v2.12

Isolation Segment v2.12 includes the following major features:

Gorouter Supports TLS v1.3

In Isolation Segment v2.12, the Gorouter supports TLS v1.3. New installations of Isolation Segment use TLS v1.3 for the Gorouter by default. If you are upgrading to Isolation Segment v2.12, the Gorouter uses TLS v1.2 by default.

You can select which versions of TLS that the Gorouter uses when you configure Isolation Segment. Selecting support for TLS v1.3 only is a beta feature in Isolation Segment v2.12.

For more information, see (Beta) Gorouter Can Support TLS v1.3 Connections Only below.

Gorouter Supports HTTP/2

Breaking Change: See Envoy Advertises HTTP/2 Support Over ALPN below.

In TAS for VMs v2.12 and later, HTTP/2 support is enabled by default. HTTP/2 is the second major version of the the HTTP protocol.

HTTP/2 features the following improvements over HTTP/1.1:

- Uses a binary data format instead of plain text
- Compresses headers
- Multiplexes multiple HTTP requests over a single TCP connection

Together, these improvements can improve response times for some apps.

For more information about the HTTP/2 protocol, see RFC 7540.

For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.

For information about routing HTTP/2 traffic to your TAS for VMs apps, see Routing HTTP/2 and gRPC Traffic to Apps.

Breaking Changes

Isolation Segment v2.12 includes the following breaking changes:

(Beta) Gorouter Can Support TLS v1.3 Connections Only

TLS v1.3 is not compatible with some versions of Java. If you configure Isolation Segment to support TLS v1.3 only, you might encounter errors with Java apps. For more information, see JSSE Client does not accept status_request extension in CertificateRequest messages for TLS 1.3 in the JDK Bug System.

The tile property that controls the TLS version in Isolation Segment changes in TAS for VMs v2.12. You must update any stored configuration files to reflect the change.

Envoy Advertises HTTP/2 Support Over ALPN

Envoy, the Diego container proxy, advertises HTTP/2 support using Application-Layer Protocol Negotiation (ALPN) for all apps. Internal clients that access the Envoy TLS port directly must negotiate down to HTTP/1.1 for apps that do not support HTTP/2. The Envoy TLS port is typically `61001`. Clients that connect to apps using the Gorouter are not affected.

Gorouter No Longer De-chunks Short Chunked Responses

In previous versions of TAS for VMs, the Gorouter de-chunked short chunked responses, set a Content-Length header, and sent a traditional body. This capability was available when Gorouter used Golang v1.15, which is out of support.

For versions of TAS for VMs that contain routing-release v0.214.0 and later, the Gorouter uses Golang v1.16 which sends a chunked response. If your clients or proxies that access apps cannot handle a chunked response, or expect a Content-Length header, they break.

For more information, see Clients receive responses with no Content-Length header and a chunked encoded body after upgrading Tanzu Application Service for VMs in the Knowledge Base.

Known Issues

Isolation Segment v2.12 includes the following known issue:

HAProxy Does Not Support HTTP/2

HAProxy is not configured to support HTTP/2 ingress traffic. HAProxy also does not send HTTP/2 traffic to the Gorouter, even when HTTP/2 is enabled.

To work around this issue, you can use an external load balancer to support HTTP/2 traffic. For more information, see Configure Load Balancers in *Configuring HTTP/2 Support*.

This issue is resolved in TAS for VMs v2.12.2 and later.

Gorouter Sets an Invalid X-B3-SpanID Header

An issue with the Gorouter's implementation of `X-B3-SpanId` and `X-B3-TraceId` headers can cause invalid span IDs to be set after updating the `X-B3-TraceId` header to the new 16-byte standard. As a result, some applications and libraries invalidate the `X-B3-SpanId` value, breaking traces of the application.

This issue affects versions of TAS for VMs that contain routing-release v0.227.0 and v0.228.0.

Architecture

In this section:

- **Components**
  - TAS for VMs Components
  - Diego Components and Architecture
  - Buildpacks
  - TAS for VMs Routing Architecture
  - Cloud Controller
  - Cloud Controller Blobstore
  - Garden
  - GrootFS Disk Usage
  - HTTP Routing
  - UAA Overview
  - Services
  - CredHub
- **App Management**
  - How Apps Are Staged
  - App Container Lifecycle
  - How Diego Balances App Processes
  - High Availability in TAS for VMs
  - How TAS for VMs Maintains High Availability
  - TAS for VMs Security
  - General Data Protection Regulation and Tanzu Application Service for VMs

Components

In this section:

- TAS for VMs Components
- Diego Components and Architecture
- Buildpacks

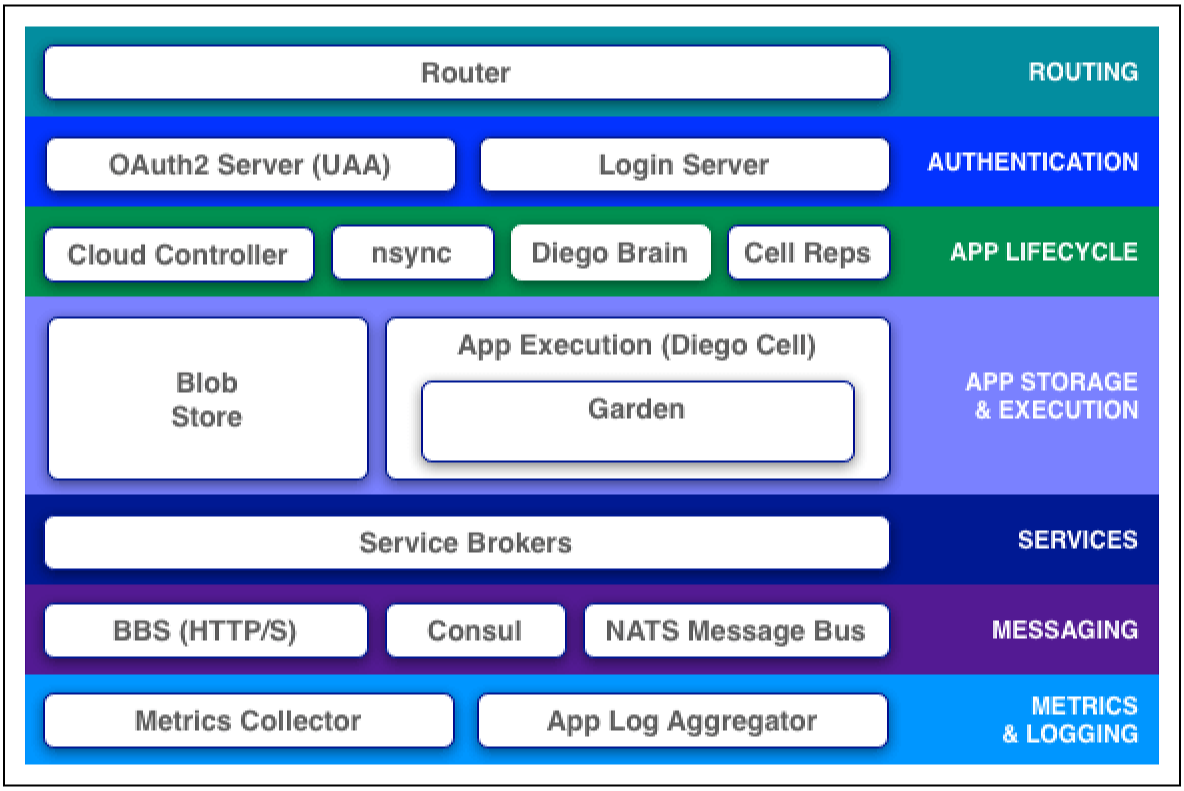
- TAS for VMs Routing Architecture
- Cloud Controller
- Cloud Controller Blobstore
- Garden
- GrootFS Disk Usage
- HTTP Routing
- UAA Overview
- Services
- CredHub

TAS for VMs Components

This topic tells you about the VMware Tanzu Application Service for VMs (TAS for VMs) runtime components.

TAS for VMs components include a self-service application execution engine, an automation engine for application deployment and lifecycle management, and a scriptable command line interface (CLI), as well as integration with development tools to ease deployment processes. TAS for VMs has an open architecture that includes a buildpack mechanism for adding frameworks, an application services interface, and a cloud provider interface.

See the descriptions below for more information about TAS for VMs components. Some descriptions include links to more detailed documentation.



Routing

Router

The router routes incoming traffic to the appropriate component, either a Cloud Controller component or a hosted application running on a Diego Cell.

The router periodically queries the Diego Bulletin Board System (BBS) to determine which cells and containers each application currently runs on. Using this information, the router recomputes new routing tables based on the IP addresses of each cell virtual machine (VM) and the host-side port numbers for the cell's containers.

For more information on the routing tier, including the router, see TAS for VMs Routing Architecture.

Authentication

OAuth2 Server (UAA) and Login Server

The OAuth2 server (the UAA) and Login Server work together to provide identity management.

App Lifecycle

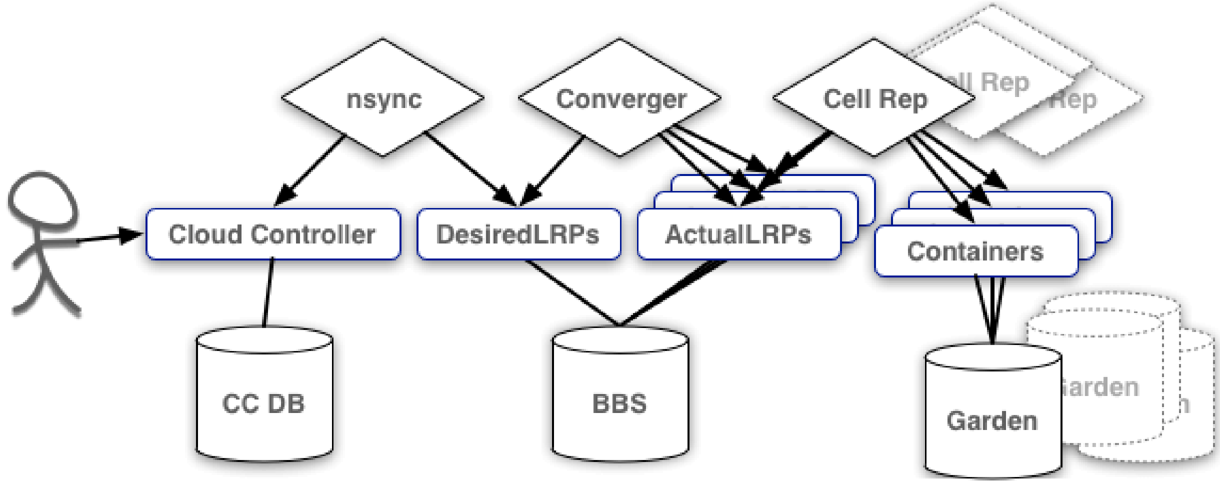
Cloud Controller and Diego Brain

The Cloud Controller (CC) directs the deployment of applications. To push an app to TAS for VMs, you target the Cloud Controller. The Cloud Controller then directs the Diego Brain through the CC-Bridge components to coordinate individual Diego cells to stage and run applications.

The Cloud Controller also maintain records of orgs, spaces, user roles, services, and more.

nsync, BBS, and Cell Reps

To keep applications available, cloud deployments must constantly monitor their states and reconcile them with their expected states, starting and stopping processes as required.



App Monitoring and Syncing with Diego

The nsync, BBS, and Cell Rep components work together along a chain to keep apps running. At one end is the user. At the other end are the instances of applications running on widelydistributed VMs, which may crash or become unavailable.

Here is how the components work together:

- **nsync** receives a message from the Cloud Controller when the user scales an app. It writes the number of instances into a `DesiredLRP` structure in the Diego BBS database.
- **BBS** uses its convergence process to monitor the `DesiredLRP` and `ActualLRP` values. It launches or stops application instances as appropriate to ensure the `ActualLRP` count matches the `DesiredLRP` count.
- **Cell Rep** monitors the containers and provides the `ActualLRP` value.

App Storage and Execution

Blobstore

The blobstore is a repository for large binary files, which Github cannot easily manage because GitHub is designed for code. The blobstore contains the following:

- Application code packages
- Buildpacks
- Droplets

You can configure the blobstore as either an internal server or an external S3 or S3-compatible endpoint. For more information about the blobstore, see How to use Elastic Runtime BLOB Storage Data in the Knowledge Base.

Diego Cell

Application instances, application tasks, and staging tasks all run as Garden containers on the Diego Cell VMs. The Diego cell rep component manages the lifecycle of those containers and the processes running in them, reports their status to the Diego BBS, and emits their logs and metrics to Loggregator.

Services

Service Brokers

Applications typically depend on services such as databases or third-party SaaS providers. When a developer provisions and binds a service to an application, the service broker for that service is responsible for providing the service instance.

Messaging

Internal HTTPS and BBS

TAS for VMs component VMs communicate with each other internally through HTTP and HTTPS protocols, sharing temporary messages and data stored in Diego’s Bulletin Board System (BBS).

- BOSH Director colocates a BOSH DNS server on every deployed VM. All VMs keep up-to-date DNS records for all the other VMs in the same foundation, enabling service discovery between VMs. BOSH DNS also provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available.

- Diego's Bulletin Board System (BBS) stores more frequently updated and disposable data such as cell and app status, unallocated work, and heartbeat messages, as well as longer-lived distributed locks. The BBS stores data in MySQL, using the Go MySQL Driver.

The route-emitter component uses the NATS protocol to broadcast the latest routing tables to the routers.

Metrics and Logging

Loggregator

The Loggregator (log aggregator) system streams application logs to developers.

For more information, see Loggregator Architecture.

Diego components and architecture

This topic tells you about the components that form and interact with the Diego system in VMware Tanzu Application Service for VMs (TAS for VMs).

TAS for VMs uses the Diego system to manage app containers. Diego components assume app scheduling and management responsibility from the Cloud Controller.

Diego is a self-healing container management system that attempts to keep the correct number of instances running in Diego cells to avoid network failures and crashes. Diego schedules and runs Tasks and Long-Running Processes (LRP). For more information about Tasks and LRPs, see How the Diego Auction Allocates Jobs.

You can submit, update, and retrieve the desired number of Tasks and LRPs using the Bulletin Board System (BBS) API. For more information, see the BBS Server repository on GitHub.

Learning how Diego runs an app

The following sections describe how Diego handles a request to run an app. This is only one of the processes that happen in Diego. For example, running an app assumes the app has already been staged.

For more information about the staging process, see How Apps are Staged.

The following illustrations and descriptions do not include all of the components of Diego. For information about each Diego component, see Diego Components.

The architecture discussed in the following steps includes the following high level blocks:

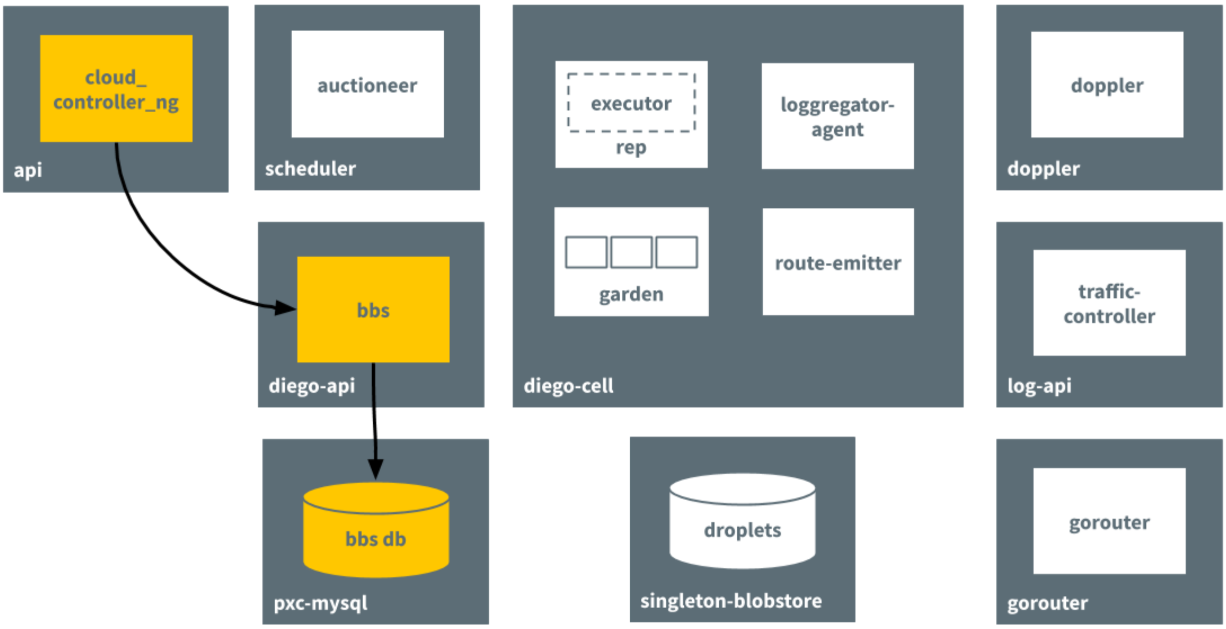
- api - cloud_controller_ng
- scheduler - auctioneer
- diego-api - bbs
- pxc-mysql - bbs db
- diego-cell - rep/executor, garden, loggregator-agent, route-emitter

- singleton-blobstore - droplets
- doppler - doppler
- log-api - traffic-controller
- gorouter - gorouter

Note: The images below are based on the VM names in an open-source deployment of Cloud Foundry Application Runtime. In TAS for VMs, the processes interact in the same way, but are on different VMs. Correct VM names for each process are in the components sections of this topic.

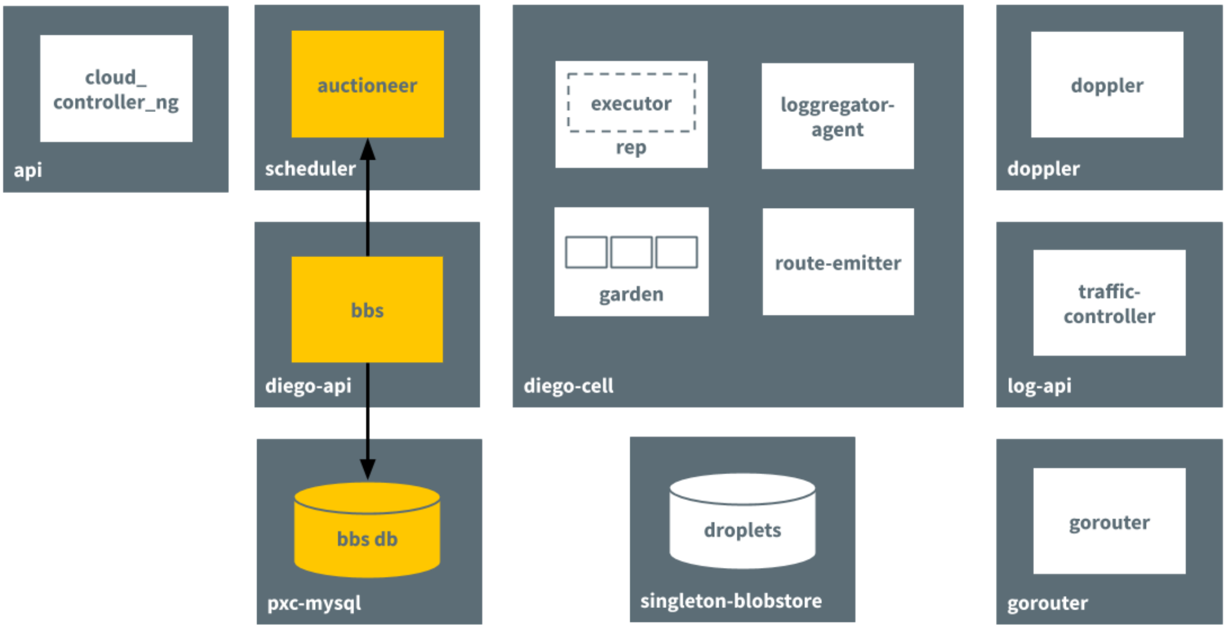
Step 1: Receiving the request to run an app

Cloud Controller passes requests to run apps to the Diego BBS, which stores information about the request in its database.



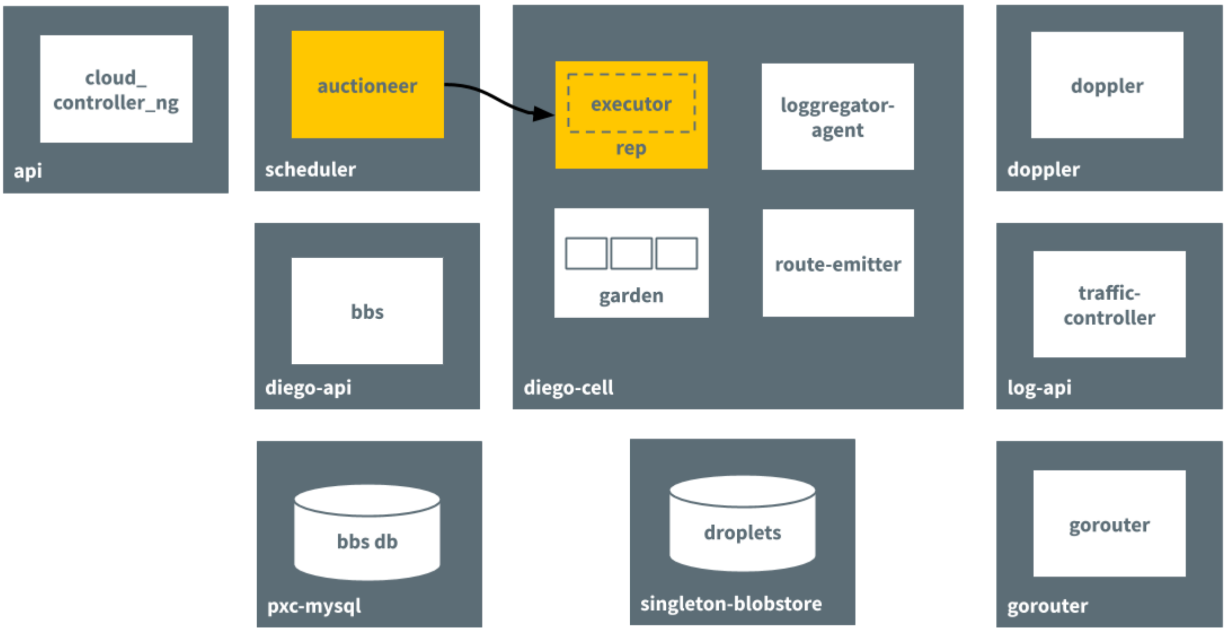
Step 2: Passing the request to the auctioneer process

The BBS contacts the Auctioneer to create an auction based on the desired resources for the app. It references the information stored in its database.



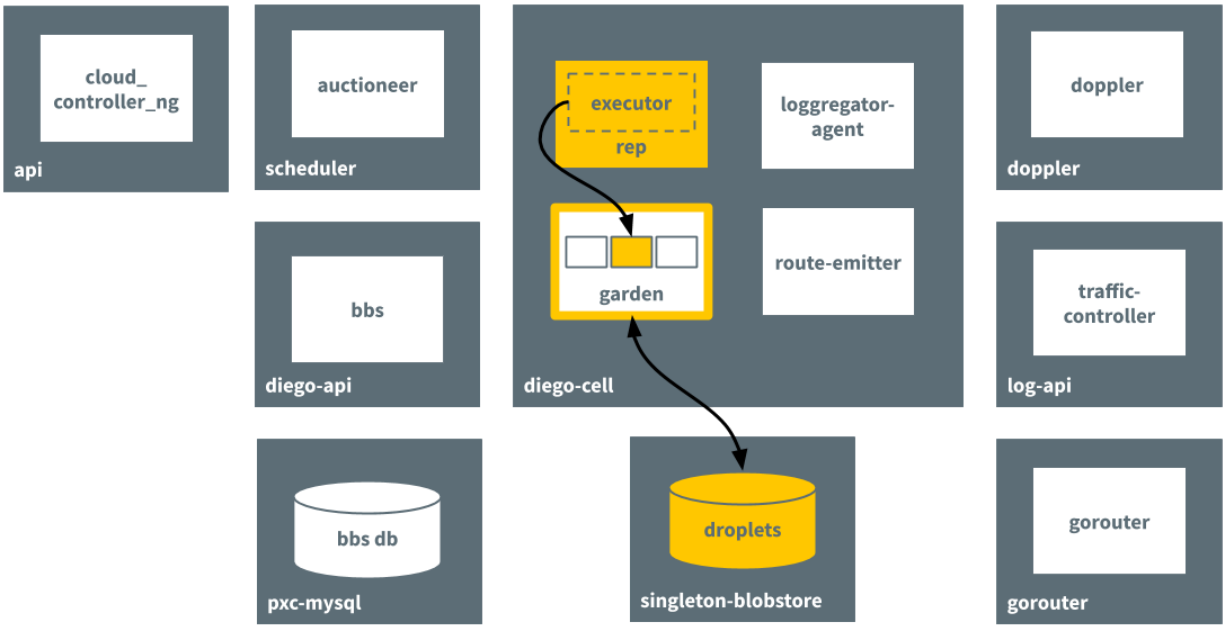
Step 3: Performing the auction

Through an auction, the Auctioneer finds a Diego Cell to run the app on. The Rep job on the Diego Cell accepts the auction request.



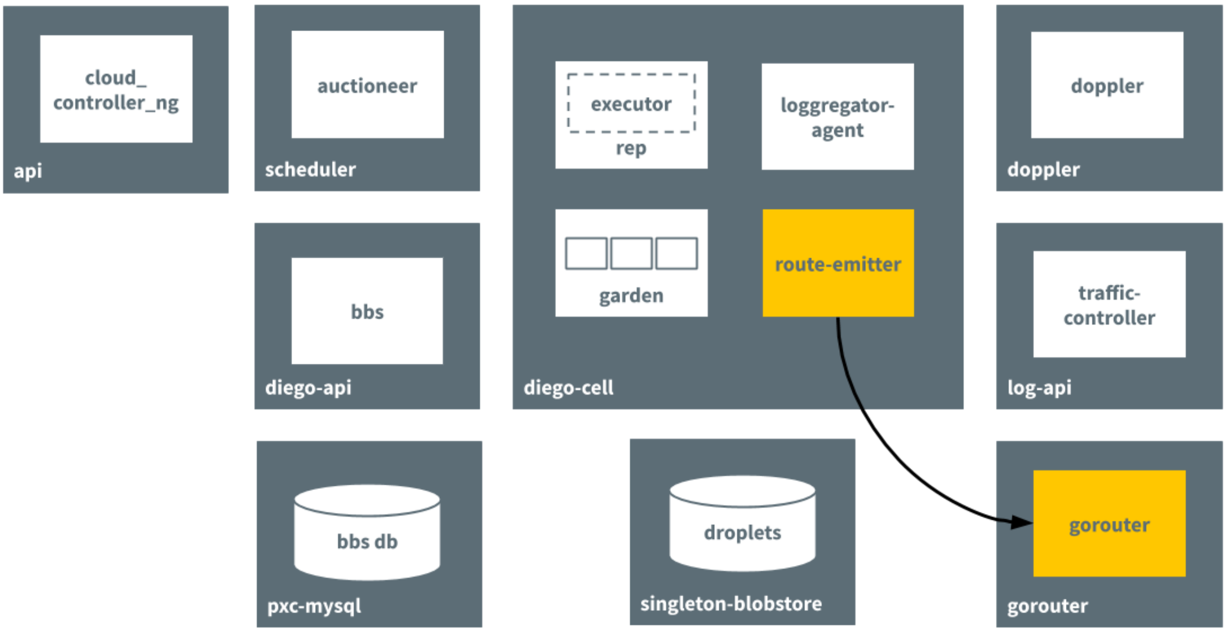
Step 4: Creating the container and running the app

The in process Executor creates a Garden container in the Diego Cell. Garden downloads the droplet that resulted from the staging process and runs the app in the container.



Step 5: Emitting a route for the app

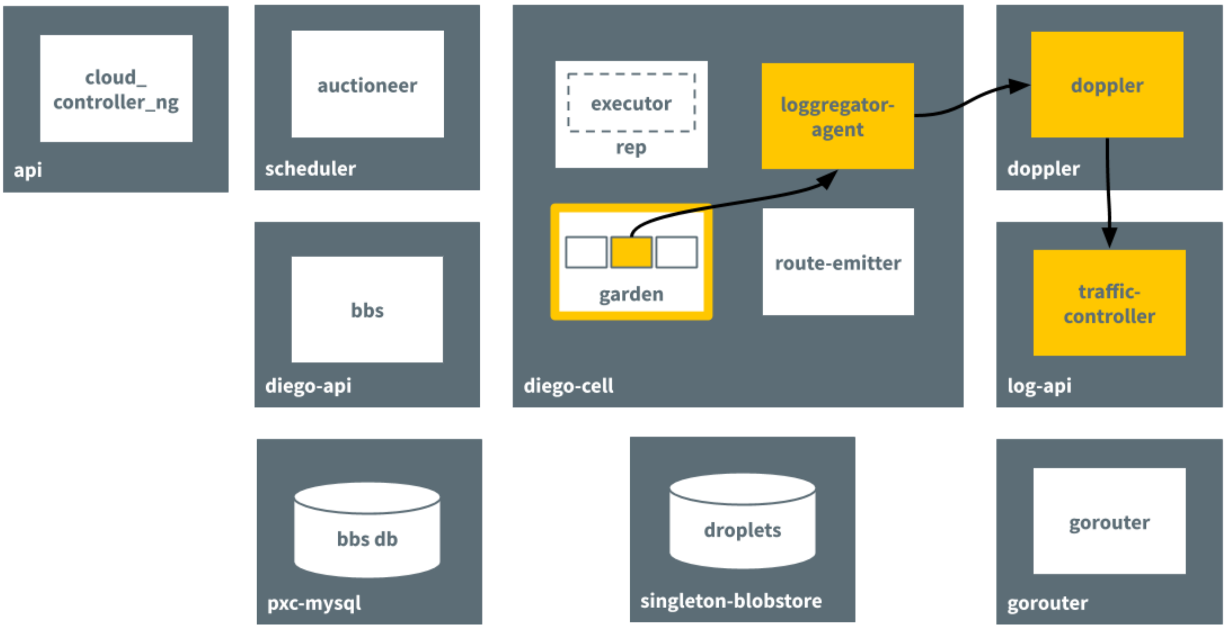
The `route-emitter` process emits a route registration message to Gorouter for the new app running on the Diego Cell.



Step 6: Sending logs to the Loggregator

Loggregator agent forwards app logs, errors, and metrics to the TAS for VMs Loggregator.

For more information, see App Logging in TAS for VMs.



Diego components

The following table describes the jobs that are part of the TAS for VMs Diego BOSH release.

Component	Function
Job: auctioneer **VM:** diego_brain	- Distributes work through auction to Cell Reps over SSL/TLS. For more information, see How the Diego Auction Allocates Jobs. - Maintains a lock in Locket to ensure only one auctioneer handles auctions at a time.
Job: bbs **VM:** diego_database	- Maintains a real-time representation of the state of the Diego cluster, including desired LRPs, running LRPs, and in-flight Tasks. - Provides an RPC-style API over HTTP to Diego Core components for external clients as well as internal clients, including the SSH Proxy and Route Emitter. - Ensures consistency and fault tolerance for Tasks and LRPs by comparing desired state with actual state. - Keeps `DesiredLRP` and `ActualLRP` counts synchronized. If the `DesiredLRP` count exceeds the `ActualLRP` count, requests a start auction from the Auctioneer. If the `ActualLRP` count exceeds the `DesiredLRP` count, sends a stop message to the Rep on the Diego Cell hosting an instance
Job: file_server **VM:** diego_brain	- Serves static assets that can include general-purpose App Lifecycle binaries

Job: locket **VM:** diego_database	• Provides a consistent key-value store for maintenance of distributed locks and component presence
Job: rep **VM:** diego_cell	• Represents a Diego Cell in Diego Auctions for Tasks and LRPs • Runs Tasks and LRPs by creating a container and then running actions in it • Periodically ensures its set of Tasks and ActualLRPs in the BBS is in sync with the containers actually present on the Diego Cell • Manages container allocations against resource constraints on the Diego Cell, such as memory and disk space • Streams stdout and stderr from container processes to the metron-agent running on the Diego Cell, which in turn forwards to the Loggregator system • Periodically collects container metrics and emits them to Loggregator • Mediates all communication between the Diego Cell and the BBS • Maintains a presence record for the Diego Cell in Locket
Job: route_emitter **VM:** diego_cell	• Monitors `DesiredLRP` and `ActualLRP` states, emitting route registration and unregistration messages to Gorouter when it detects changes. • Periodically emits the entire routing table to the TAS for VMs Gorouter.
Job: ssh_proxy **VM:** diego_brain	• Brokers connections between SSH clients and SSH servers • Runs inside instance containers and authorizes access to app instances based on Cloud Controller roles

Additional information

The following resources provide more information about Diego components:

- The Diego Release repository on GitHub.
- The Auctioneer repository on GitHub.
- The Bulletin Board System repository on GitHub.
- The File Server repository on GitHub.
- The Rep repository on GitHub.
- The Executor repository on GitHub.
- The Route-Emitter repository on GitHub.
- App SSH, App SSH Overview, and the Diego SSH repository on GitHub.

Maximum recommended Diego Cells

The maximum recommended Diego Cells is 250 Cells for each TAS for VMs deployment. By default, there is a hard limit of 256 addresses for vSphere deployments that use Silk for networking. This hard limit is described in the Silk Release documentation on GitHub.

The default CIDR address block for the overlay network is 10.255.0.0/16. Each Diego Cell requires a subnet, and subnets (0-255) for each Diego Cell are allocated out of this network.

TAS for VMs deployments that do not use Silk for networking do not have a hard limit. However, operating a foundation with more than 250 Diego Cells is not recommended for the following reasons:

- Changes to the foundation can take a long time, potentially days or weeks depending on the `max-in-flight` value. For example, if there is a certificate expiring in a week, there might not be enough time to rotate the certificates before expiry. For more information, see Basic Advice in *Configuring TAS for VMs for Upgrades*.
- A single foundation still has single points of failure, such as the certificates on the platform. The RAM that 250 Diego Cells provides is enough to host many business-critical apps.

Components from other releases

The following table describes jobs that interact closely with Diego but are not part of the Diego TAS for VMs BOSH release.

Component	Function
Job: bosh-dns-aliases **VM:** all	• Provides service discovery through colocated DNS servers on all BOSH-deployed VMs • Provides client-side load-balancing by randomly selecting a healthy VM when multiple VMs are available
Job: cc_uploader **VM:** diego_brain	• Mediates uploads from the Executor to the Cloud Controller • Translates simple HTTP POST requests from the Executor into complex multipart-form uploads for the Cloud Controller
Job: database **VM:** mysql	• Provides a consistent key-value data store to Diego
Job: loggregator-agent **VM:** all	• Forwards app logs, errors, and app and Diego metrics to the Loggregator Doppler component
Job: cloud_controller_clock **VM:** clock_global	• Runs a Diego sync process to ensure desired app data in Diego is in sync with the Cloud Controller.

App lifecycle binaries

The following platform-specific binaries deploy apps and govern their lifecycle:

- The **Builder**, which stages a TAS for VMs app. The Builder runs as a Task on every staging request. It performs static analysis on the app code and does any necessary pre-processing before the app is first run.
- The **Launcher**, which runs a TAS for VMs app. The Launcher is set as the Action on the `DesiredLRP` for the app. It executes the start command with the correct system context, including working directory and environment variables.
- The **Healthcheck**, which performs a status check on running TAS for VMs app from inside the container. The Healthcheck is set as the Monitor action on the `DesiredLRP` for the app.

Current implementations

- The buildpack app lifecycle implements the TAS for VMs buildpack-based deployment strategy. For more information, see the buildpackapplifecycle repository on GitHub.
- The Docker app lifecycle implements a Docker deployment strategy. For more information, see the dockerapplifecycle repository on GitHub.

Additional information

The following resources provide more information about components from other releases that interact closely with Diego:

- The CC-Uploader repository on GitHub.
- Garden or the Garden repository on GitHub.
- The Loggregator Release repository on GitHub.
- The BOSH DNS documentation.
- The TPS repository on GitHub.

Cloud Foundry buildpacks

Buildpacks provide framework and runtime support for your apps. Buildpacks typically examine your apps to determine what dependencies to download and how to configure your apps to communicate with bound services.

When you push an app, Cloud Foundry automatically detects an appropriate buildpack for it. This buildpack is used to compile or prepare your app for launch.

Cloud Foundry deployments often have limited access to dependencies. This limitation occurs when the deployment is behind a firewall, or when administrators want to use local mirrors and proxies. In these circumstances, Cloud Foundry provides a Buildpack Packager app. For more information, see the Buildpack Packager repository on GitHub.

Using buildpacks

The following topics discuss different usage scenarios for buildpacks in Cloud Foundry:

- How Buildpacks Work.
- Stack Association.

- Pushing an App with Multiple Buildpacks.
- Using a Proxy.
- Supported Binary Dependencies.
- Production Server Configuration.

System buildpacks

For information about the existing buildpacks that Cloud Foundry supports, see System buildpacks.

Sidecar buildpacks

For information about deploying a sidecar buildpack, see Sidecar buildpack.

Community buildpacks

For a list of unsupported, community created buildpacks, see Community created section of the *Buildpack information for Cloud Foundry v2* page in the cf-docs-contrib repository on GitHub.

Customizing and developing buildpacks

For information about customizing existing buildpacks and developing new buildpacks, see Customizing and developing buildpacks.

Using continuous integration for buildpacks

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for buildpacks. You can use this topic as a model when working with Concourse to build, and release new versions of your own buildpacks.

TAS for VMs Routing Architecture

This topic tells you about the routing architecture and flow in VMware Tanzu Application Service for VMs (TAS for VMs).

Routing architecture components

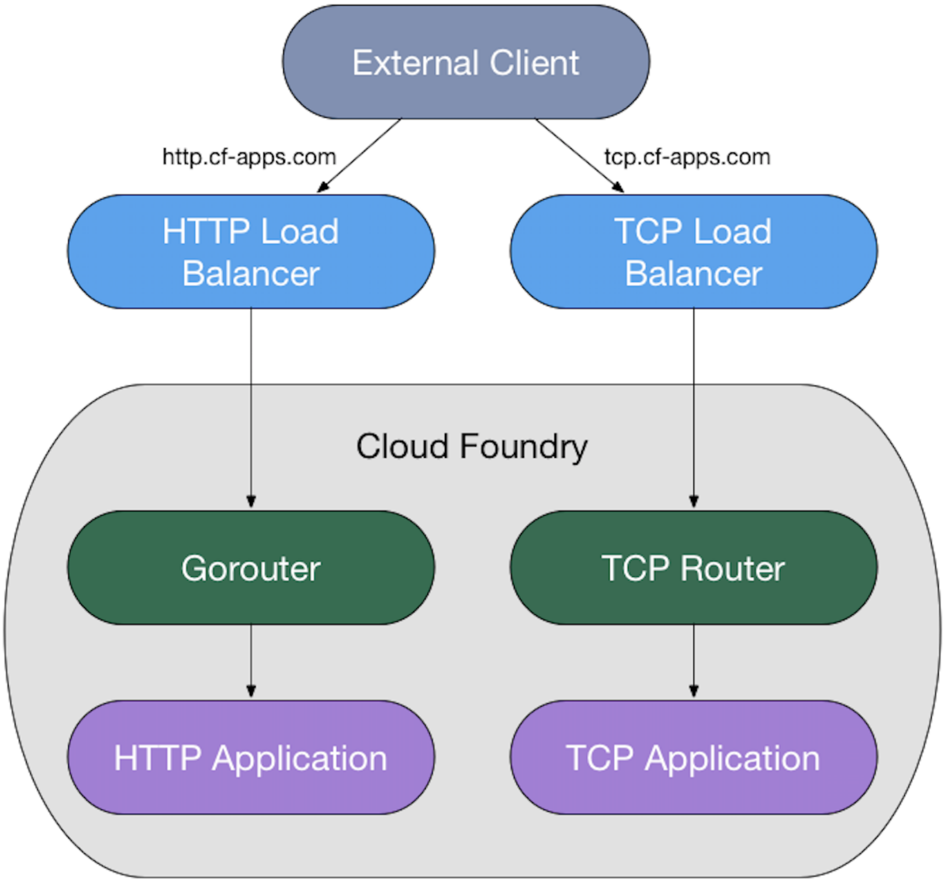
The following summarizes the roles and responsibilities of various components depicted in the TAS for VMs routing architecture diagrams. These summaries are limited to the roles and responsibilities these components have pertaining to routing. For more complete descriptions of these components, see TAS for VMs Concepts, TAS for VMs Components, and Diego Components and Architecture.

Component Name	Summary
BOSH manifest	Used to configure route registrar with route(s) for system components such as UAA and Loggregator.
Cloud Controller	Contains route metadata, including whether they are HTTP or TCP.
Diego BBS	Contains IP and port metadata as well as route metadata from Cloud Controller, which route emitter discovers.
Diego cell	Manages app instances and tasks and long-running processes related to them. A route emitter runs on each cell.
Gorouter	Routes HTTP traffic coming into TAS for VMs to the appropriate component. Receives route updates through NATS. Routes that have not been updated in two minutes are pruned from the Gorouter's database.
NATS	Receives routing configuration from route emitter and provides same to Gorouter.
Route registrar	Sends routing metadata described in BOSH manifest for system components such as UAA and Loggregator to NATS. This is because the Diego cell does not have information about system components, only about user spaces.
Route emitter	Periodically emits route, IP, and port metadata to NATS or Routing API as registration and unregistration messages. Does not know about app instances on Diego cell, but knows what cell it belongs to and learns about what app instances are running on its cell by asking Diego BBS for information about app instances on the same cell.
Routing API	Receives routing configuration from route emitter and other internal clients, and provides routing configuration for TCP router.
Routing database	Saves some routing data from Routing API. If the Gorouter misses a message about an unmapped route from NATS, it will not get it again, so TCP router and Routing API can consult routing database for current state of routes.
TCP router	Routes TCP traffic coming into TAS for VMs to the appropriate component. Receives route updates through the routing API.

Making a request to external client

The following process tells you about how an external client makes a request to an app running on TAS for VMs:

1. The external client sends its request.
2. Your DNS service sends the request to the HTTP or TCP load balancer based on the prefix of the DNS name in the client request, such as `http` in `http.example.com`.
3. The load balancer sends the request to the load balancer's corresponding router.
4. The router sends the request to the app.



Maintaing updated routing tables

Because each app can have many instances, one app route can go to multiple containers. As Diego moves and scales app instances, the route emitter on each cell sends messages to NATS or the Routing API to register and deregister routes to the cell's app instances.

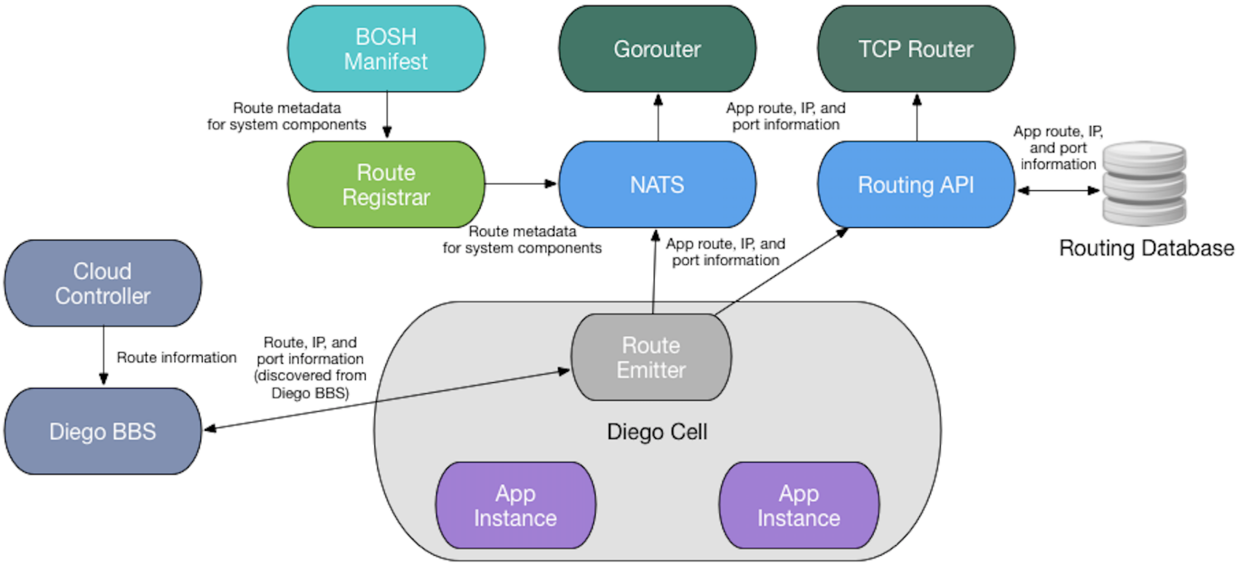
The route emitter periodically emits the routes it discovers from Diego BBS to NATS and the Routing API as registration and unregistration messages every twenty seconds. The Gorouter uses TLS to verify app identity and confirm that its routes are up-to-date. For more information about how TAS for VMs maintains route consistency, see Preventing Misrouting in *HTTP Routing*.

The following process describes how a router obtains information about routes for an app running on TAS for VMs:

1. Cloud Controller component sends app route information to Diego BBS. For HTTP routing, route information includes the host and path of an external URL, as shown in the format of the `router.register` message in the Gorouter documentation on GitHub. For TCP routing, route information includes the route port on which the TCP connection was received; for more information, see the Routing API documentation on GitHub.

2. Diego BBS coordinates the back end IP address and port where each instance of the app runs. When queried by the route emitter, the BBS sends this information along with Cloud Controller's app route information to the route emitter on the Diego cell where instances of the app are located.

3. If a route is HTTP, the route emitter on the Diego cells sends app route, IP, and port information to NATS, which then sends it to the Gorouter. If a route is TCP, the route emitter sends that information to the Routing API, which then sends it to the TCP router.
4. Gorouter and TCP router use the route, IP, and port information from the route emitter to map incoming app requests to back end app instance locations.


BOSH Manifest
Gorouter
TCP Router
Route metadata for system components
App route, IP, and port information
App route, IP, and port information
Route Registrar
NATS
Routing API
Routing Database
Route metadata for system components
App route, IP, and port information
Cloud Controller
Route information
Route, IP, and port information (discovered from Diego BBS)
Route Emitter
Diego BBS
Diego Cell
App Instance
App Instance

Routing table recovery after component failure

Cloud Controller and Diego BBS have their own databases, while NATS and the Gorouter only store their data in memory. If NATS or the Gorouter are restarted, they lose all of their data and must wait for the route emitter to send data to them again. If Diego BBS is restarted, it can retrieve its data from Cloud Controller.

If Cloud Controller is restarted, you must retrieve its data from a backup.

Cloud Controller

Cloud Controller in VMware Tanzu Application Service for VMs (TAS for VMs) provides you with REST API endpoints to access the system. Cloud Controller maintains a database with tables for orgs, spaces, services, user roles, and more.

Diego Auction

The Cloud Controller uses the Diego Auction to balance application processes over the Diego Cells in a TAS for VMs installation.

Database (CC_DB)

The Cloud Controller database has been tested with MySQL.

Blobstore

To stage and run apps, TAS for VMs manages and stores the following types of binary large object (blob) files:

Blob Type	Description	Location in Blobstore
App Packages	Full contents of app directories, including source code and resource files, zipped into single blob files.	`/cc-packages`
Buildpacks	Buildpack directories, which Diego cells download to compile and stage apps with.	`/cc-buildpacks`
Resource Cache	Large files from app packages that the Cloud Controller stores with a SHA for later re-use. To save bandwidth, the TAS for VMs Command Line Interface (cf CLI) only uploads large application files that the Cloud Controller has not already stored in the resource cache.	`/cc-resources`
Buildpack Cache	Large files that buildpacks generate during staging, stored for later re-use. This cache lets buildpacks run more quickly when staging apps that have been staged previously.	`cc-droplets/buildpack_cache`
Droplets	Staged apps packaged with everything needed to run in a container.	`/cc-droplets`

TAS for VMs blobstores use the Fog Ruby gem to store blobs in services like Amazon S3, WebDAV, or the NFS filesystem. The file system location of an internal blobstore is `/var/vcap/store/shared`.

A single blobstore typically stores all five types of blobs, but you can configure the Cloud Controller to use separate blobstores for each type.

Automatic Blob Cleanup

After a blob deletion fails silently or something else goes wrong, the blobstore may contain blobs that the Cloud Controller no longer needs or lists in its database. These are called orphan blobs, and they waste blobstore capacity.

Cloud Controller detects and removes orphan blobs by scanning part of the blobstore daily and checking for any blobs that its database does not account for. The process scans through the entire blobstore every week, and only removes blobs that show as orphans for three consecutive days.

Cloud Controller performs this automatic cleanup when the `cloud_controller_worker` job property `cc.perform_blob_cleanup` is set to `true`.

Manual Blob Cleanup

Cloud Controller does not track resource cache and buildpack cache blob types in its database, so it does not clean them up automatically as it does with app package, buildpack, and droplet type blobs.

To clean up the buildpack cache, admin users can run `cf curl -X DELETE /v2/blobstores/buildpack_cache`. This empties the buildpack cache completely, which is a safe operation.

To clean up the resource cache, delete it as follows:

- **Internal blobstore**: Run `bosh ssh` to connect to the blobstore VM (NFS or WebDav) and `rm *` the contents of the `/var/vcap/store/shared/cc-resources` directory.
- **External blobstore**: Use the file store's API to delete the contents of the `resources` bucket.

Do not manually delete app package, buildpack, or droplet blobs from the blobstore. To free up resources from those locations, run `cf delete-buildpack` for buildpacks or `cf delete` for app packages and droplets.

Testing

By default `rspec` runs a test suite with the SQLite in-memory database. Specify a connection string using the `DB_CONNECTION` environment variable to test against MySQL. For example:

```
DB_CONNECTION="mysql2://root:password@localhost:3306/ccng" rspec
```

Cloud Controller blobstore

TAS for VMs uses a blobstore to store the source code that enables you to push, stage, and run.

This topic references staging and treats all blobstores as generic object stores.

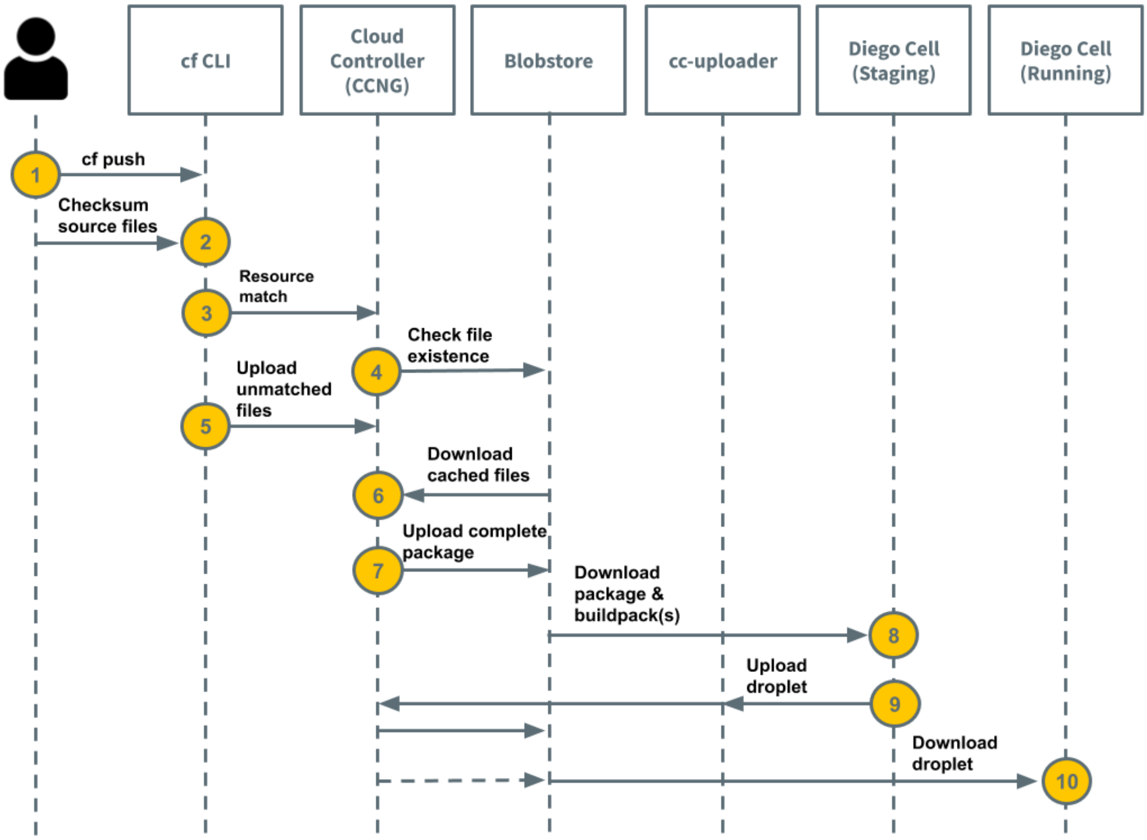
For more information about staging, see How Apps Are Staged.

For more information about how specific third-party blobstores can be configured, see Configuring File Storage for TAS for VMs.

Staging using the blobstore

This section describes how staging buildpack apps uses the blobstore.

The following diagram illustrates how the staging process uses the blobstore. To walk through the same diagram in an app staging context, see How Diego Stages Buildpack Apps.



The process in which the staging process uses the blobstore is as follows:

1. **cf push:** A developer runs `cf push`.
2. **Create app:** The Cloud Foundry Command Line Interface (cf CLI) gathers local source code files and computes a checksum of each.
3. **Store app metadata:** The cf CLI makes a `resource_matches` request, which matches resources to Cloud Controller. The request lists file names and their checksums. For more information and an example API request, see Resource Matches in the TAS for VMs API documentation.
4. **Check file existence** includes the following:
    a. The Cloud Controller makes a series of `HEAD` requests to the blobstore to find out which files it has cached.
    b. Cloud Controller content-addresses its cached files so that changes to a file result in it being stored as a different object.
    c. Cloud Controller computes which files it has and which it needs the cf CLI to upload. This process can take a long time.
    d. In response to the resource match request, Cloud Controller lists the files the cf CLI needs to upload.
5. **Upload unmatched files:** The cf CLI compresses and uploads the unmatched files to Cloud Controller.
6. **Download cached files:** Cloud Controller downloads, to its local disk, the matched files that are cached in the blobstore.

7. **Upload complete package** includes the following:
   a. Cloud Controller compresses the newly uploaded files with the downloaded cached files in a ZIP file.
   b. Cloud Controller uploads the complete package to the blobstore.
8. **Download package & buildpack(s):** A Diego Cell downloads the package and its buildpacks into a container and stages the app.
9. **Upload droplet** includes the following:
   a. After the app has been staged, the Diego Cell uploads the complete droplet to `cc-uploader`.
   b. `cc-uploader` makes a multi-part upload request to upload the droplet to Cloud Controller.
   c. Cloud Controller enqueues an asynchronous job to upload to the blobstore.
10. **Download droplet** includes the following:
    a. A Diego Cell attempts to download the droplet from Cloud Controller into the app container.
    b. Cloud Controller asks the blobstore for a signed URL.
    c. Cloud Controller redirects the Diego Cell droplet download request to the blobstore.
    d. A Diego Cell downloads the app droplet from the blobstore and runs it.

Blobstore load

The load that Cloud Controller generates on its blobstore is unique due to resource matching. Many blobstores that perform well under normal read, write, and delete loada are overwhelmed by Cloud Controller's heavy use of HEAD requests during resource matching.

Pushing an app with large number of files causes Cloud Controller to check the blobstore for the existence of each file.

Parallel BOSH deployments of Diego Cells can also generate significant read load on the Cloud Controller blobstore as the cells perform evacuation. For more information, see the Evacuation section of the *App Container Lifecycle* topic.

How Cloud Controller reaps expired packages, droplets, and buildpacks

As new droplets and packages are created, the oldest ones associated with an app are marked as `EXPIRED` if they exceed the configured limits for packages and droplets stored per app.

Each night, starting at midnight, Cloud Controller runs a series of jobs to delete the data associated with expired packages, droplets, and buildpacks.

Enabling the native versioning feature on your blobstore increases the number of resources stored and costs. For more information, see Using Versioning in the AWS documentation.

Blobstore interaction timeouts

Cloud Controller inherits default blobstore operation timeouts from Excon. Excon defaults to 60 second read, write, and connect timeouts.

For more information, see the excon repository on GitHub.

Garden component

You can use Garden, the component that VMware Tanzu Application Service for VMs (TAS for VMs) uses to create and manage isolated environments called containers. Each instance of an app deployed to TAS for VMs runs within a container.

For more information about how containers work, see Container Mechanics in *Container Security*.

Plug-in back ends

Garden has plug-in back ends for different platforms and runtimes. It specifies a set of interfaces that each platform specific back end must implement.

These interfaces contain methods to perform the following actions:

- Create and delete containers.
- Apply resource limits to containers.
- Open and attach network ports to containers.
- Copy files into and out of containers.
- Run processes within containers.
- Stream `STDOUT` and `STDERR` data out of containers.
- Annotate containers with arbitrary metadata.
- Snapshot containers for redeploys without downtime.

For more information, see the Garden repository on GitHub.

Garden-runC back end

TAS for VMs currently uses the Garden-runC back end, a Linux-specific implementation of the Garden interface using the Open Container Interface (OCI) standard. Previous versions of TAS for VMs used the Garden-Linux back end. For more information, see the Garden-Linux repository on GitHub.

Garden-runC has the following features:

- Uses the same OCI low-level container execution code as Docker and Kubernetes, so container images run identically across all three platforms
- AppArmor is configured and enforced by default for all unprivileged containers
- Seccomp allowlisting restricts the set of system calls a container can access, reducing the risk of container breakout
- Allows pluggable networking and rootfs management

For more information, see the Garden-runC repository on GitHub.

Garden RootFS plug-in

Garden manages container file systems through a plug-in interface. TAS for VMs uses the Garden RootFS (GrootFS) plug-in for this task. GrootFS is a Linux-specific implementation of the Garden volume plug-in interface.

GrootFS performs the following actions:

- Creates container file systems based on buildpacks and droplets.
- Creates container file systems based on remote docker images.
- Authenticates with remote registries when using remote images.
- Properly maps UID/GID for all files inside an image.
- Runs garbage collection to remove unused volumes.
- Applies per container disk quotas.
- Provides per container disk usage stats.

For more information, see GrootFS Disk Usage and the GrootFS repository on GitHub.

GrootFS disk usage

This topic tells you about the concepts related to GrootFS disk space management in VMware Tanzu Application Service for VMs (TAS for VMs).

GrootFS stores

GrootFS is the container root filesystem management component for Garden. A container root filesystem or *rootfs* is often referred to as an **image**.

A **GrootFS store** is the directory in which rootfs layers and container images are cached. This directory is configured by GrootFS and mounted on an XFS-formatted volume by the Garden job during BOSH VM creation.

Individual container root filesystems are provided via OverlayFS mounts.

Supplying GrootFS with an already formatted XFS volume for its store is not yet supported for BOSH-controlled deployments.

Garbage collection behavior in GrootFS stores

GrootFS stores are initialized to use the entirety of `/var/vcap/data`. If the `reserved_space_for_other_jobs_in_mb` is not set high enough, or if there are many images with few shared volumes, the store can use up all available space.

The thresholder component calculates and sets a value so that GrootFS's garbage collector can attempt to ensure that a small reserved space is kept free for other jobs. GrootFS only tries to garbage collect when that threshold is reached. However, if all the rootfs layers are actively in use by images, then garbage collection cannot occur and that space is used up.

Volumes

Underlying layers in rootfs images are known as `volumes` in GrootFS. They are read-only and their changesets are layered together through an **OverlayFS** mount to create the root filesystems for containers.

When GrootFS writes each filesystem volume to disk, it also stores the number of bytes written to a file in a `meta` directory. The size of an individual volume is available in its corresponding metadata file. GrootFS also stores the SHA of each underlying volume used by an image in the `meta` folder.

For each container, GrootFS mounts the underlying `volumes` using overlay to a point in the `images` directory. This mount point is the rootfs for the container and is read write.

On disk, the read-write layer for each container can be found at `/var/vcap/data/grootfs/store/unprivileged/images/CONTAINER-ID/diff` (or `/var/vcap/data/grootfs/store/privileged/images/CONTAINER-ID/diff` for privileged containers.)

When GrootFS calls on the built-in XFS quota tooling to get disk usage for a container, it takes into account data written to those `diff` directories and not the data in the read-only volumes.

Volume Cleanup Example

When `clean` is called in GrootFS, any layers that are not being used by an existing rootfs are deleted from the store. The cleanup only takes into account the `volumes` folders in the store.

For example, imagine that there are two rootfs images from different base images, Image A and Image B:

```
- Image A
  Layers:
    - layer-1
    - layer-2
    - layer-3

- Image B
  Layers:
    - layer-1
    - layer-4
    - layer-5
```

They have a layer in common, layer-1. And after deleting Image B, layer-4 and layer-5 can be collected by clean, but not layer-1 because Image A still uses that layer.

For more information on how to calculate GrootFS disk usage in your deployment, see Examining GrootFS Disk Usage.

Additional information

For more information, see the following sections of `garden-runc-release`:

- overlay-xfs-setup
- grootfs-utils
- thresholder

HTTP routing

This topic tells you how the Gorouter, the main component in the VMware Tanzu Application Service for VMs (TAS for VMs) routing tier, routes HTTP traffic within TAS for VMs.

For more information about routing, see Routing in *TAS for VMs Components*.

HTTP headers

Header size limit

The Gorouter has a limit of 1 MB for HTTP Headers.

The specific language, framework, and configuration of the back end app container determine the effective header size limit. For example, the default header size for the Tomcat container is 8 kB.

X-Forwarded Proto

The `X-Forwarded-Proto` header gives the scheme of the HTTP request from the client.

If an incoming request includes the `X-Forwarded-Proto` header, the Gorouter:

- Appends it to the existing header
- Sets the scheme to HTTP if the client made an insecure request, meaning a request on port 80
- Sets the scheme to HTTPS if the client made a secure request, meaning a request on port 443

Developers can configure their apps to reject insecure requests by inspecting the `X-Forwarded-Proto` HTTP header on incoming traffic. The header may have multiple values represented as a comma-separated list, so developers must ensure the app rejects traffic that includes any `X-Forwarded-Proto` values that are not HTTPS.

X-Forwarded-For

If `X-Forwarded-For` is present, the Gorouter appends the load balancer’s IP address to it and forwards the list. If `X-Forwarded-For` is not present, then the Gorouter sets it to the IP address of the load balancer in the forwarded request (some load balancers masquerade the client IP). If a load balancer sends the client IP using the PROXY protocol, then the Gorouter uses the client IP address to set `X-Forwarded-For`.

If your load balancer terminates TLS on the client side of the Gorouter, it must append these headers to requests forwarded to the Gorouter. For more information, see Securing Traffic into TAS for VMs.

HTTP Headers for Zipkin Tracing

If your load balancer terminates TLS on the client side of the Gorouter, it must append these headers to requests forwarded to the Gorouter. For more information, see Securing Traffic into TAS for VMs.

HTTP headers for Zipkin tracing

Zipkin is a tracing system that allows app developers to troubleshoot failures or latency issues. Zipkin provides the ability to trace requests and responses across distributed systems. For more information about Zipkin tracing, see Zipkin.io.

When Zipkin tracing is enabled in TAS for VMs, the Gorouter examines the HTTP request headers and performs the following:

- If the `X-B3-TraceId` and `X-B3-SpanId` HTTP headers are not present in the request, the Gorouter generates values for these and inserts the headers into the request forwarded to an app. These values are also found in the Gorouter access log message for the request: `x_b3_traceid` and `x_b3_spanid`.
- If the `X-B3-TraceId` and `X-B3-SpanId` HTTP headers are present in the request, the Gorouter forwards them unmodified. In addition to these trace and span IDs, the Gorouter access log message for the request includes `x_b3_parentspanid`.

Developers can then add Zipkin trace IDs to their app logging in order to trace app requests and responses in TAS for VMs.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs myapp` to correlate the trace and span IDs logged by the Gorouter with the trace ids logged by their app. To correlate trace IDs for a request through multiple apps, each app must forward appropriate values for the headers with requests to other apps.

For more information about Zipkin tracing, see Enabling Zipkin Tracing.

HTTP headers for app instance routing

Developers who want to obtain debug data for a specific instance of an app can use the HTTP header `X-Cf-App-Instance` to make a request to an app instance.

To make an HTTP request to a specific app instance:

1. Obtain the GUID of your app:

   ```
   $ cf app YOUR-APP --guid
   ```

2. List your app instances and retrieve the index number of the instance you want to debug:

   ```
   $ cf app YOUR-APP
   ```

3. Make a request to the app route using the HTTP header `X-Cf-App-Instance` set to the concatenated values of the app GUID and the instance index:

   ```
   $ curl app.example.com -H "X-Cf-App-Instance":"YOUR-APP-GUID:YOUR-INSTANCE-INDEX"
   ```

> Use of the `X-Cf-App-Instance` header is only available for users on the Diego architecture.

If this header is set to an invalid value, Gorouter resturns a `400` status code and the response from Gorouter contains a `X-Cf-Routererror` header with more information about the error. The following table describes the possible error responses:

X-Cf-Routererror Value	Reason for Error	Response Body
`invalid_cf_app_instance_header`	The value provided for `X-Cf-App-Instance` was not a properly formatted GUID.	None
`unknown_route`	The value provided for `X-Cf-App-Instance` is a correctly formatted GUID, but there is no instance found with that guid for the route requested.	`400 Bad Request: Requested instance ('1') with guid ('aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa') does not exist for route ('myroute.cf.com')`

Forward Client Certificate to Apps

Apps that require mutual TLS (mTLS) need metadata from client certificates to authorize requests. TAS for VMs supports this use case without bypassing layer-7 load balancers and the Gorouter.

The HTTP header `X-Forwarded-Client-Cert` (XFCC) may be used to pass the originating client certificate along the data path to the application. Each component in the data path must trust that the back end component has not allowed the header to be tampered with.

If you configure the load balancer to terminate TLS and set the XFCC header from the received client certificate, you must also configure the load balancer to strip this header if it is present in client requests. This configuration is required to prevent spoofing of the client certificate.

1. From the terminal output, record the the index number of the instance you want to debug.
2. Make a request to the app route by running:

   ```
   curl APP-FQDN -H "X-Cf-App-Instance":"APP-GUID:INSTANCE-INDEX-NUMBER"
   ```

   Where:

   - `APP-FQDN` is the fully-qualified domain name (FQDN) of your app. For example, `app.example.com`.
   - `APP-GUID` is the app GUID that you recorded in a previous step.
   - `INSTANCE-INDEX-NUMBER` is the instance index number that you recorded in the previous step.

   Note: You can only use `X-Cf-App-Instance` header on the Diego architecture.

   If either of the values you provide in the above command are invalid, Gorouter returns a `400` error, and the response from Gorouter contains a `X-Cf-Routererror` header with more information about the error. The following table describes the possible error responses:

X-Cf-Routererror Value	Reason for Error	Response Body

`invalid_cf_app_instance_header`	The value provided for `X-Cf-App-Instance` includes an incorrectly formatted app GUID.	None
`unknown_route`	The value provided for `X-Cf-App-Instance` includes a correctly formatted app GUID, but the app instance index number was not found for the requested route.	`400 Bad Request: Requested instance ('1') with guid ('aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa') does not exist for route ('example-route.cf.com')`

Forwarding client certificate to apps

Apps that require mutual TLS (mTLS) need metadata from client certificates to authorize requests. TAS for VMs supports this use case without bypassing layer-7 load balancers and the Gorouter.

The HTTP header `X-Forwarded-Client-Cert` (XFCC) may be used to pass the originating client certificate along the data path to the app. Each component in the data path must trust that the back-end component has not allowed the header to be tampered with.

If you configure the load balancer to terminate TLS and set the XFCC header from the received client certificate, you must also configure the load balancer to strip this header if it is present in client requests. This configuration is required to prevent spoofing of the client certificate.

The sections below describe supported deployment configurations.

Terminate TLS at Load Balancer

By default, the Gorouter forwards arbitrary headers that are not otherwise mentioned in the docs. This includes the XFCC header.

For apps to receive the XFCC header, configure your load balancer to set the XFCC header with the contents of the client certificate received in the TLS handshake.

To enable this mode:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **TLS termination point**, select **Infrastructure load balancer**.

Terminate TLS at HAProxy

This option allows you to configure support for the XFCC header while leveraging HAProxy. When selected, HAProxy sets the XFCC header to the contents of the client certificate received in the TLS handshake.

This option requires you to configure the load balancer in front of HAProxy to pass through the TLS handshake to HAProxy through TCP.

To enable this mode:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **TLS termination point**, select **HAProxy**.

HAProxy trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between apps running on Ops Manager.

Terminate TLS at the Gorouter

If the Gorouter is the first component to terminate TLS, such that it receives the certificate of the originating client in the mutual TLS handshake, you should select this option. When selected, the Gorouter sets the XFCC header to the contents of the client certificate received in the TLS handshake and strips the XFCC header when present in a request.

This option requires you to configure the load balancer in front of the Gorouter to pass through TLS handshake to the Gorouter through TCP.

To enable this mode:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **TLS termination point**, select **Gorouter**.

The Gorouter trusts the Diego intermediate certificate authority. This trust is enabled automatically and permits mutual authentication between apps running on Ops Manager.

Client-Side TLS

Depending on your needs, you can configure your deployment to terminate TLS at the Gorouter, at the Gorouter and the load balancer, or at the load balancer only. For more information, see Securing Traffic into TAS for VMs.

TLS to apps and other back end services

The Gorouter supports TLS and mutual authentication to back end destinations, including app instances, platform services, and any other routable endpoints.

This has the following benefits:

- Improved availability for apps by keeping routes in the Gorouter's routing table when TTL expires
- Increased guarantees against misrouting by validating the identity of back ends before forwarding requests
- Increased security by encrypting data in flight from the Gorouter to back ends

Preventing Misrouting

Preventing misroutes

Network partitions or NATS failures can cause the Gorouter's routing table to fall out of sync, as TAS for VMs continues to re-create containers across hosts to keep apps running. This can lead to routing of requests to incorrect destinations.

Before forwarding traffic to an app instance, the Gorouter initiates a TLS handshake with an Envoy proxy running in each app container. In the TLS handshake, the Envoy proxy presents a certificate generated by Diego for each container which uniquely identifies the container using the same app instance identifier sent by the Route-Emitter, configured in the certificate as a domain Subject Alternative Name (SAN). For more information, see Envoyproxy.io.

If the Gorouter confirms that the app instance identifier in the certificate matches the one received in the route registration message, the Gorouter forwards the HTTP request over the TLS session, and the Envoy proxy then forwards it to the app process. If the instance identifiers do not match, the Gorouter removes the app instance from its routing table and transparently retries another instance of the app.

Currently, only Linux cells support the Gorouter validating app instance identities using TLS by default. With Windows cells, the Gorouter connects to back ends without TLS, forwarding requests to Windows apps over plain text and pruning based on route TTL.

Configuring validation of app instance identity with TLS

Configure Validation of App Instance Identity with TLS

Verifying app identity using TLS improves resiliency and consistency for app routes.

The **App Containers** pane of the TAS for VMs tile includes these options under **Gorouter app identity verification**:

- **The Gorouter uses TLS to verify app identity:** Enables the Gorouter to verify app identity using TLS. This is the default option.
- **The Gorouter and apps use mutual TLS to verify each other's identity:** Enables the Gorouter and your apps to verify each other's identity using TLS.

> **Breaking Change:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing.

To enable TLS to backends running on Windows cells, the same options can be configured under **(Beta) Enable TLS Connections From Router To Applications** on the **Advanced Features** tab of the Windows tile.

Router Balancing Algorithm

Router balancing algorithm

Gorouter can be configured to use different load balancing algorithms for routing incoming requests to app instances. The Gorouter maintains a dynamically updated list of app instances for each route. Depending on which algorithm is selected, it forwards to one of the app instances.

- `round-robin` for round-robin algorithm,
- `least-connection` for least connection algorithm,

By default, the Gorouter uses the round-robin algorithm.

Round robin load balancing

Incoming requests for a given route are forwarded to all app instances one after another, looping back to the first one after they have each received a request. This algorithm is suitable for most use cases and evenly distributes the load between app instances.

Least connection load balancing

Each request for a given route is forwarded to the app instance with the least number of open connections. This algorithm can be more suitable for some cases. For example, if app instances have long-lived connections and are scaled up, then new instances receive fewer connections, causing a disproportionate load. In this case, choosing a least-connection algorithm sends new connections to new instances to equalize the load.

WebSockets

WebSockets is a protocol providing bi-directional communication over a single, long-lived TCP connection, commonly implemented by web clients and servers. WebSockets are initiated through HTTP as an upgrade request. The Gorouter supports this upgrade handshake, and holds the TCP connection open with the selected app instance. To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see Supporting WebSockets.

Session affinity

Gorouter supports session affinity, or *sticky sessions*, for incoming HTTP requests to compatible apps.

With sticky sessions, when multiple instances of an app are running on TAS for VMs, requests from a particular client always reach the same app instance. This allows apps to store session data specific to a user session.

To support sticky sessions, configure your app to return a sticky session cookie in responses. The default value for this field is `JSESSIONID`. You can configure the cookie names that the routing tier uses for sticky sessions. To configure the names of the cookies, see Configure Networking in *Configuring TAS for VMs.*

If an app returns a sticky session cookie to a client request, the CF routing tier generates a unique `VCAP_ID` for the app instance based on its GUID with the same expiry, sameSite, and secure attributes as `JSESSIONID`. The `VCAP_ID` is in the following format:

```
323f211e-fea3-4161-9bd1-615392327913
```

On subsequent requests, the client must provide both the sticky session and `VCAP_ID` cookies.

CF routing tier uses the `VCAP_ID` cookie to forward client requests to the same app instance every time. The sticky session cookie is forwarded to the app instance to enable session continuity. If the app instance identified by the `VCAP_ID` crashes, the Gorouter attempts to route the request to a different instance of the app. If the Gorouter finds a healthy instance of the app, it initiates a new sticky session.

For more information, see Session Affinity in GitHub.

> **Note:** TAS for VMs does not persist or replicate HTTP session data across app instances. If an app instance crashes or is stopped, session data for that instance is lost. If you require session data to persist across crashed or stopped instances, or to be shared by all instances of an app, store session data in a CF marketplace service that offers data persistence.

Keep alive connections

Front end clients

Gorouter supports keep alive connections from clients and does not close the TCP connection with clients immediately after returning an HTTP response. Clients are responsible for closing these connections.

Back end servers

If keep-alive connections are disabled, the Gorouter closes the TCP connection with an app instance or system component after receiving an HTTP response.

If keep-alive connections are enabled, the Gorouter maintains established TCP connections to back ends. The Gorouter supports up to 100 idle connections to each back end:

- If an idle connection exists for a given back end, the Gorouter reuses it to route subsequent requests.
- If no idle connection exists for this back end, the Gorouter creates a new connection.

Transparent retries

If the Gorouter cannot establish a TCP connection with a selected app instance, the Gorouter considers the instance ineligible for requests for 30 seconds and transparently attempts to connect to another app instance. Once the Gorouter has established a TCP connection with an app instance, the Gorouter forwards the HTTP request.

When you deploy an app that requires Diego Cells to restart or recreate, the app may not respond to a Gorouter request before the keep-alive connection breaks. The following table describes how the Gorouter behaves if it cannot establish a TCP connection to an app:

If the Gorouter...	and the back end...	then the Gorouter...
cannot establish a TCP connection to a back end	N/A	retries another back end, no more than 3 times
establishes a TCP connection to a back end and forwards the request	does not respond	waits 15 minutes for a response, and if it errors, does not retry another back end
establishes a TCP connection to a back end and forwards the request	returns a TCP connection error	returns an error to the client, marks backend ineligible, and does not retry another back end

In all cases, if the app still does not respond to the request, the Gorouter returns a `502` error. For more information, see Troubleshooting Router Error Responses.

User Accounts and Authentication overview

Cloud Foundry User Account and Authentication (UAA) is an open-source identity server project under the Cloud Foundry Foundation.

UAA provides enterprise-scale identity management features. It is used by these commercial services, for example:

- Single sign-on for VMware Tanzu: Identity services for apps and APIs running on VMware Tanzu Application Service (TAS for VMs)
- GE Predix Identity Services
- SAP HANA Identity Services

What is UAA?

UAA provides identity-based security for apps and APIs. It supports open standards for authentication and authorization, including:

- OAuth
- OpenID Connect
- SAML
- LDAP
- SCIM

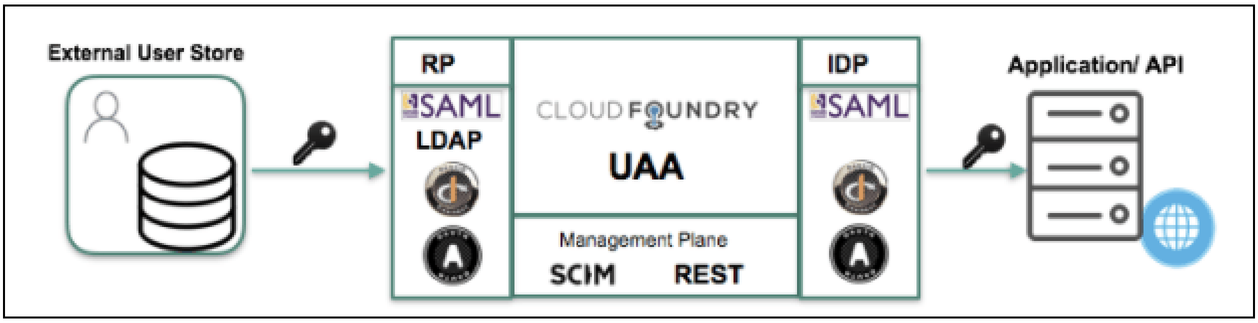
The major features of UAA include:

- User Single Sign-On (SSO) using federated identity protocols
- API security with OAuth
- User and group management
- Multi-tenancy support
- Support for **JWT** and **opaque** as a token format
- Token revocation
- Operational flexibility

  - Operate and run as a BOSH release, which uses multi-cloud deployment capabilities
  - Push as an app to TAS for VMs
- Database flexibility, including support for MySQL and Postgres
- Auditing, logging, and monitoring
- Token exchange for SAML and JWT bearers
- Rest APIs for authentication, authorization, and configuration management

UAA architecture

The diagram below illustrates the architecture of UAA:



The table below describes the protocols UAA can use:

Protocol	Purpose	Profiles
OAuth 2.0	Authorizes apps and APIs	Authorization Server, Relying Party
OpenID Connect 1.0	Federates to external identity providers (IDPs) and acts as an IDP for SSO	Identity Provider, Relying Party
SAML 2.0	Federates to external IDPs and acts as an IDP for SSO	Identity Provider, Service Provider
LDAP	Authenticates users in external user store	LDAP Client
SCIM 1.0	Manages users and groups	Identity Provisioning

Client-side tools and libraries

The table below describes the client-side tools and libraries UAA uses:

Name	Language
UAAC CF-UAA-LIB	Ruby
Spring Security OAuth	Java
CF Java Client	Java
UAA Javascript SDK (Singular)	JS

The role of UAA in securing TAS for VMs

TAS for VMs relies on UAA for identity and access management requirements. UAA secures user and system access to TAS for VMs installations.

Since TAS for VMs is primarily used in the enterprise context, UAA supports enterprise SSO workflows. If a user has already authenticated against the enterprise IDP, they can access TAS for VMs without re-entering credentials.

Some of the major components of TAS for VMs that use UAA include:

- Cloud Controller
- Gorouter
- Loggregator
- Container networking

Each of these components expose APIs for user and system interaction. UAA uses OAuth to secure the APIs exposed by core TAS for VMs components.

UAA secures many different TAS for VMs components, including:

- Cloud Foundry Command Line Interface (cf CLI)
- Cloud Controller
- Loggregator
- Notifications
- Gorouter
- Container Networking
- Diego
- BOSH Director
- Autoscaler

Token management

After authenticating, client apps access resources and services using tokens rather than account authentication credentials. UAA generates, manages, distributes, and validates these tokens. UAA provides new tokens only to client apps with valid authentication credentials.

Once granted, token validity is unchallenged until the token expires due to timeout.

UAA distributes two types of tokens: access tokens and refresh tokens.

Access tokens

Client apps use UAA-provided access tokens to request access to API endpoints, resources, and services. The access token presented by a client app must be valid in order to access a resource.

Access tokens are valid for a short period of time.

After a UAA-provided access token has expired, a client app can request UAA provide a replacement. To request a replacement access token from UAA, a client app must present a valid refresh token.

Refresh tokens

Client apps use UAA-provided refresh tokens to request replacements for expired access tokens. The refresh token presented by a client app must be valid in order to replace an expired access token.

Refresh tokens are valid for a long period of time.

UAA does not provide replacement refresh tokens. To obtain a new refresh token, a client app must re-authenticate.

Token timeouts

The token interactions listed above produce certain end results.

Authenticated client apps are running and interacting with services using tokens, not the service account's authentication. An admin can revoke a service account's access and privileges, but a client app running under that service account can continue to interact with the environment, unchallenged, as long as its token is valid. Therefore:

- The access token timeout is the maximum period during which an already running app can continue to access services after an admin has revoked privileges from the service account the app is running under.
- The refresh token timeout is the maximum period during which a client app can access services without re-authentication.

The default access and refresh token timeout values are the UAA token timeout values VMware recommends. Before altering token timeout settings, consider:

- Short default token timeout periods create overhead within your system.
    - A short access token timeout requires a client app to frequently request a replacement access token from the UAA server.
    - A short refresh token timeout requires frequent re-authentication, which might be impossible at the designated timeout frequency.

  Setting a short token timeout period, such as `0`, can render all associated services and resources inaccessible.
- Long timeout periods represent a security risk.
    - A long access token timeout allows an authenticated process to continue accessing services and resources for a long period after an admin has revoked privileges from the service account used by the process.
    - A long refresh token timeout allows an already authenticated app to continue running for a long period of time on stale authentication credentials without being challenged to re-authenticate.

  Setting a long timeout period, such as one lasting days, can result in a client app running unchallenged for days.

Services

This topic provides an overview of services in VMware Tanzu Application Service for VMs (TAS for VMs).

Services and Service Instances

TAS for VMs offers a marketplace of services that operators can use to provision reserved resources on demand. Marketplace services include resources such as databases on a shared or dedicated server, or accounts on a SaaS app. These resources are known as service instances and the systems that deliver and operate these resources are known as services.

For a service to be available in the Marketplace, an operator must integrate the service with TAS for VMs using APIs.

For more information about provisioning service instances and other lifecycle operations, see Managing Service Instances.

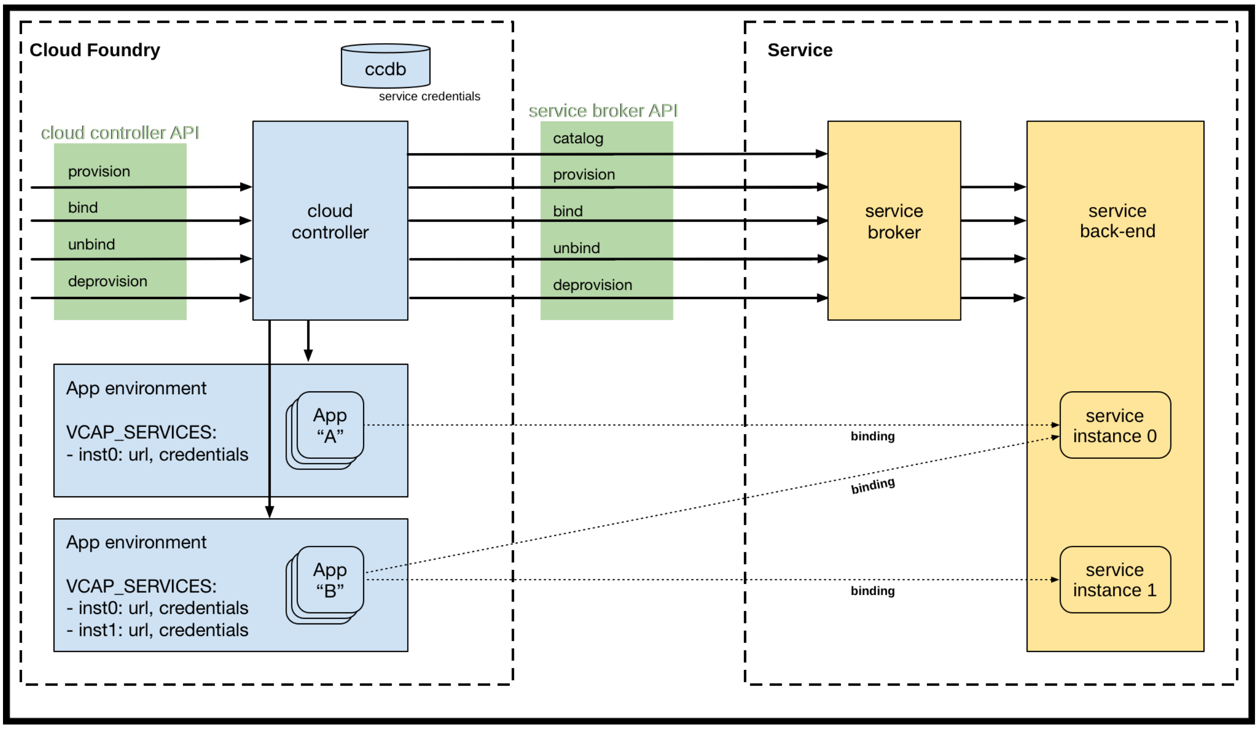
TAS for VMs enables users to use services that are not available in the Marketplace using user-provided service instances (UPSI). For more information, see User-Provided Service Instances.

Architecture and Terminology

Services are integrated with TAS for VMs by using a documented API for which the Cloud Controller is the client, called the Service Broker API. This should not be confused with the Cloud Controller API (CAPI), often used to refer to the version of Cloud Foundry. The Service Broker API is versioned independently of CAPI.

The component of the service that uses the Service Broker API is called the *service broker*. This component was formerly referred to as a service gateway. However, as traffic between apps and services does not flow through the service broker, the term caused confusion. Although “gateway” still appears in old code, the term “broker” is used in conversation, in new code, and in documentation.

Service brokers advertise a catalog of service offerings and service plans, as well as interpreting calls for provision, bind, unbind, and deprovision. What a broker does with each call can vary between services. In general, ‘provision’ reserves resources on a service and ‘bind’ delivers information to an app necessary for accessing the resource. The reserved resource is called a *service instance*. What a service instance represents can vary by service. It could be a single database on a multi-tenant server, a dedicated cluster, or even just an account on a web app.



View a larger version of this image.

Service Instance Credentials

TAS for VMs enables users to provision credentials needed to interface with a service instance. You can use app binding to automatically deliver these credentials to your TAS for VMs app. For external and local clients, you can use service keys to generate credentials to communicate directly with a service instance.

App Binding

Service instance credentials can be delivered automatically to apps running on TAS for VMs in an environment variable. For more information, see Delivering Service Credentials to an App.

For information about binding to a specific app development framework, see Buildpacks.

Service Keys

Credentials managed manually are known as service keys. Use service keys when you want a set of credentials for use by clients other than the app in the same space. For instance, you can use service keys to connect to a service instance from a local client, or from an app in another space, or even from outside of TAS for VMs.

For more information about creating a user-provided service instance with service keys, see the User-Provided Service Instances topic. For more information about service keys, see the Managing Service Keys topic.

Note: Not all services support service keys. Some services support credentials through app binding only.

Implementation and Deployment

How a service is implemented is up to the service provider or developer. TAS for VMs only requires that the service provider implement the Service Broker API. A broker can be implemented as a separate app, or by adding the required HTTP endpoints to an existing service.

Because TAS for VMs only requires that a service implements the Service Broker API to be available to TAS for VMs end users, many deployment models are possible. The following are examples of valid deployment models:

- Entire service packaged and deployed by BOSH alongside TAS for VMs
- Service broker packaged and deployed by BOSH alongside TAS for VMs, rest of the service deployed and maintained by other means
- Broker (and optionally service) pushed as an app to TAS for VMs user space
- Entire service, including broker, deployed and maintained outside of TAS for VMs by other means

Communication Between Apps and Service Instances

To allow an app to communicate with a service external to TAS for VMs, you might need to configure the service to accept connections from your app based on its outbound IP address.

In your external service configuration, you must do one of the following:

- Add the entire IP address range for the Diego Cell where the app is deployed to your allow list.
- Derive the app IP address from its DNS name using a command-line tool such as `dig`, `host`, or `nslookup`. In your external service configuration, add the IP address or range of the app instance to your allow list.

Stream App Logs to Log Management Services

To learn how your app logs can be streamed to third-party log management services, see Streaming App Logs to Log Management Services.

User-provided service instances can be used to drain app logs to a service not available in the Marketplace. This is also known as setting up a syslog drain. For guidance on configuring some third-party log management services, see Service-Specific Instructions for Streaming App Logs.

Manage App Requests with Route Services

To learn how Marketplace services (and user-provided service instances) can be used to perform preprocessing on app requests, see Managing App Requests with Route Services.

Migrate a Database Schema

If your app relies on a relational database, you must apply schema changes periodically. To perform database schema migrations on TAS for VMs-managed services, run a database migration task with the TAS for VMs Command Line Interface (cf CLI) tool.

For more information about running cf CLI tasks, see Running Tasks.

> **Note:** To run tasks with the cf CLI, you must install cf CLI v6.23.0 or later. For information about downloading, installing, and uninstalling the cf CLI, see Installing the cf CLI.

To perform a database schema migration with the cf CLI:

1. Push the app:

   ```
   $ cf push APP-NAME
   ```

   Where `APP-NAME` is the name of the app.

   > **Note:** To run a task without starting the app, push the app with `cf push -i 0` and then run the task. You can run the app later by scaling up its instance count.

2. Perform a database schema migration as a task on the app:

   ```
   $ cf run-task APP-NAME --command "bin/rails db:migrate" --name TASK-NAM
   E
   Creating task for app APP-NAME in org jdoe-org / space development as j
   doe@pivotal.io...
   OK
   Task 1 has been submitted successfully for execution.
   ```

   Where:

   - `APP-NAME` is the name of the app.
   - `TASK-NAME` is the name of the task.

CredHub

For an overview of the purpose and functionality of the CredHub component, which is designed for centralized credential management, see the CredHub documentation.

App Management

In this section:

- How Apps Are Staged
- App Container Lifecycle
- How Diego Balances App Processes
- High Availability in TAS for VMs
- How TAS for VMs Maintains High Availability
- TAS for VMs Security

- General Data Protection Regulation and Tanzu Application Service for VMs

How apps are staged

This topic tells you how Diego stages buildpack apps and Docker images in VMware Tanzu Application Service for VMs (TAS for VMs).

TAS for VMs uses Diego to manage app containers. It is a self-healing system that attempts to keep the correct number of instances running in Diego Cells to avoid network failures and crashes. For more information about Diego, see Diego components and architecture.

Learn about tasks and long-running processes (LRPs).

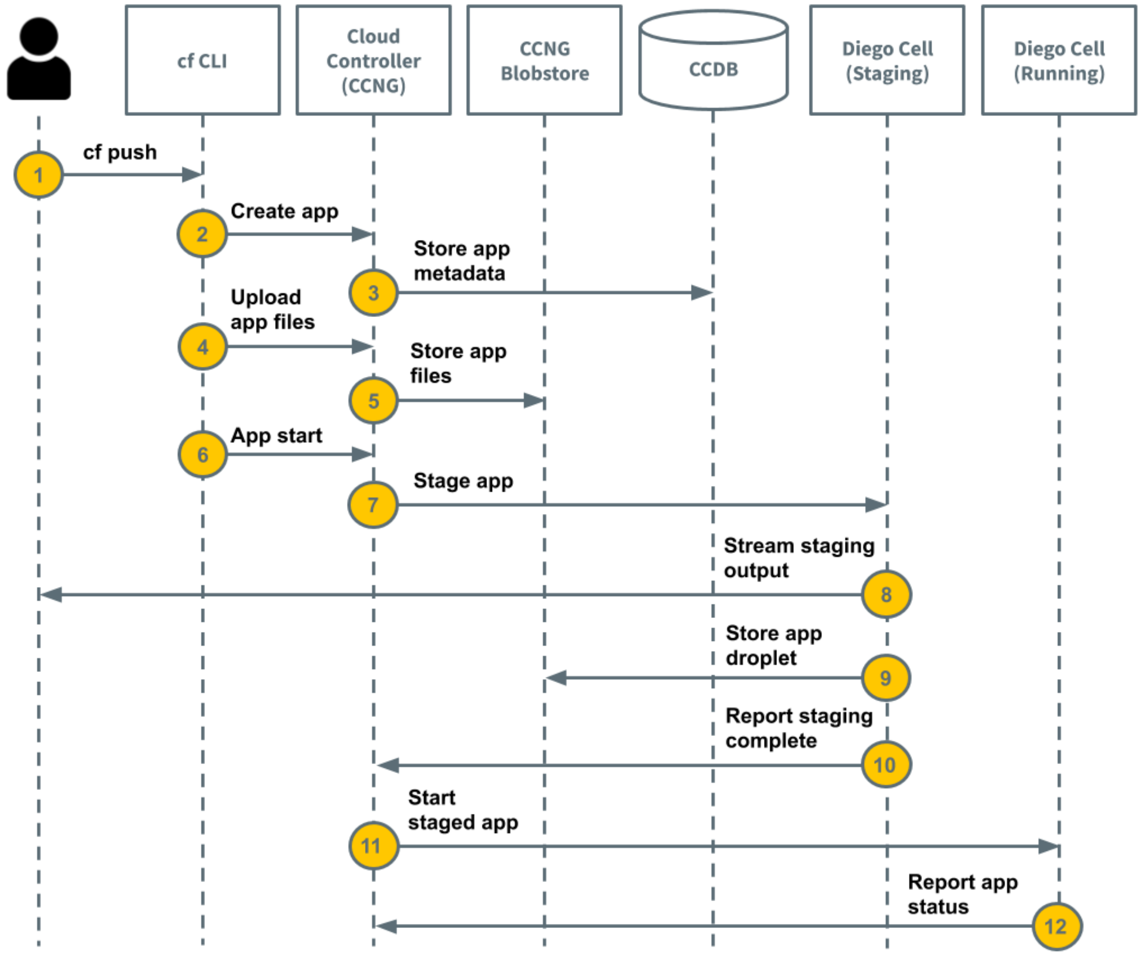
For more information about these, see the Tasks and Long-Running Processes section of the *How Diego Balances App Processes* topic.

To better understand the flow and caching of source bits during staging, see the How Staging Uses the Blobstore section of the *Cloud Controller blobstore* topic.

How Diego stages buildpack apps

Learn how Diego stages buildpack apps.

The following diagram illustrates the steps and components involved in the process of staging a buildpack app. The staging process for buildpack apps includes a developer and the following components: CF Command Line, Cloud Controller (CCNG), CCNG Blobstore, CCDB, Diego Cell (Staging), and Diego Cell (Running). Step 1: cf push from Developer to CF Command Line. Step 2: Create App from CF Command Line to Cloud Controller (CCNG). Step 3: Stores App Metadata from CCNG to the CCDB. Step 4: Upload App Files from the CF Command Line to CCNG. Step 5: Store App Files from CCNG to CCNG Blobstore. Step 6: App Start from CF Command Line to CCNG. Step 7: Stage App from CCNG to Diego Cell (Staging). Step 8: Stream Staging Output from Diego Cell (Staging) to the Developer. Step 9: Store App Droplet from Diego Cell (Staging) to CCNG Blobstore. Step 10: Report Staging Complete from Diego Cell (Staging) to CCNG. Step 11: Start Staged App from CCNG to Diego Cell (Running). Step 12: Report App Status from Diego Cell (Running) to CCNG.



The following steps describe the process of staging a buildpack app:

1. A developer runs `cf push`.
2. Cloud Foundry Command Line Interface (cf CLI) tells the Cloud Controller to create a record for the app. For more information about the Cloud Controller, see Cloud Controller.
3. Cloud Controller stores the app metadata. App metadata can include the app name, number of instances, buildpack, and other information about the app.
4. This step includes:
    a. The cf CLI requests a resource match from the Cloud Controller.
    b. The cf CLI uploads the app source files, omitting any app files that already exist in the resource cache.
    c. The Cloud Controller combines the uploaded app files with files from the resource cache to create the app package.
5. Cloud Controller stores the app package in the blobstore. For more information, see the Blobstore section of the *TAS for VMs Components* topic.
6. The cf CLI issues a request to start the app.
7. This step includes:
    a. The Cloud Controller issues a staging request to Diego.

   b. Diego schedules a Diego Cell to run the staging task.

   c. The task downloads buildpacks and the app buildpack cache, if present.

   d. The task uses the buildpack to compile and stage the app.

8. Diego Cell streams the output of the staging process. You might need to view the output to troubleshoot staging problems.

9. This step includes:

   a. The task creates a tarball, or droplet, with the compiled and staged app.

   b. The Diego Cell stores the droplet in the blobstore.

   c. The task uploads the buildpack cache to the blobstore for use the next time the app is staged.

10. Diego Bulletin Board System (BBS) reports to the Cloud Controller that staging is complete. If staging does not complete within 15 minutes, it fails.

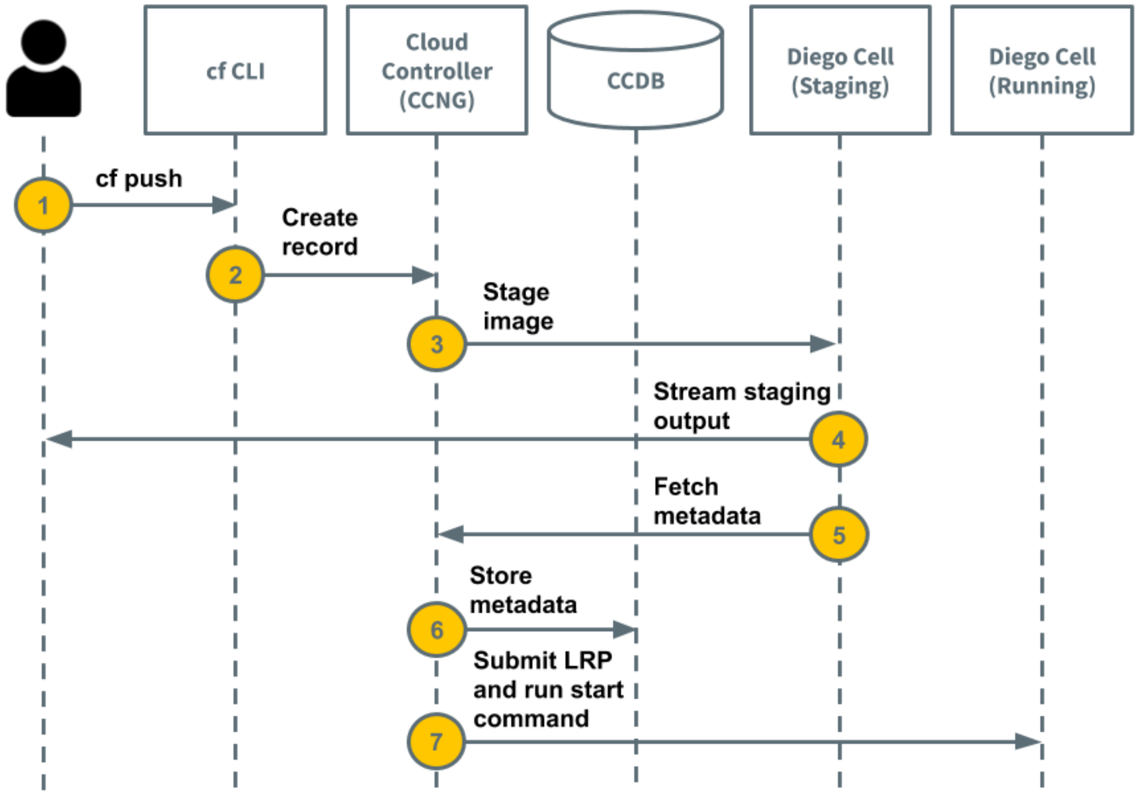
11. Diego schedules the app as a LRP on one or more Diego Cells.

12. Diego Cells report the status of the app to the Cloud Controller.

How Diego stages Docker images

This section describes how Diego stages Docker images.

The following diagram illustrates the steps and components involved in the process of staging a Docker image. The following diagram illustrates the steps and components involved in the process of staging a Docker image. The staging process for Docker images includes a developer and the following components: CF Command Line, Cloud Controller (CCNG), CCDB, Diego Cell (Staging), and Diego Cell (Running). Step 1: CF Push from Developer to CF Command Line. Step 2: Create Record from CF Command Line to CCNG. Step 3: Stage Image from CCNG to CCDB. Step 4: Stream Staging Output from Diego Cell (Staging) to Developer. Step 5: Fetch Metadata from Diego Cell (Staging) to CCNG. Step 6: Store Metadata from CCNG to CCDB. Step 7: Submit LRP an Run Start Command from CCNG to Diego Cell (Running).



The following describe each step in the process of staging a Docker image:

1. A developer runs `cf push` and includes the name of a Docker image in an accessible Docker Registry.
2. The cf CLI tells the Cloud Controller to create a record for the Docker image.
3. This step includes:
   a. The Cloud Controller issues a staging request to Diego.
   b. Diego schedules a Diego Cell to run the task.
4. Diego Cell streams the output of the staging process. You might need to view the output to troubleshoot staging problems.
5. The task fetches the metadata associated with the Docker image and returns a portion of it to the Cloud Controller.
6. Cloud Controller stores the metadata in the Cloud Controller database.
7. This step includes:
   a. Cloud Controller uses the Docker image metadata to construct a LRP that runs the start command specified in the Dockerfile.
   b. Cloud Controller submits the LRP to Diego.
   c. Diego schedules the LRP on one or more Diego Cells.
   d. Cloud Controller instructs Diego and the Gorouter to route traffic to the Docker image.

> **Note:** Cloud Controller takes into account any user-specified overrides specified in the Dockerfile, such as environment variables.

The app container lifecycle on the Diego architecture

The lifecycle stages of an app container for your VMware Tanzu Application Service for VMs (TAS for VMs) deployments running on the Diego architecture include deployment, crash events, evacuation, and shutdown.

Deployment

The app deployment process involves uploading, staging, and starting the app in a container. Your app must successfully complete each of these phases within certain time limits. The default time limits for the phases are as follows:

- Upload: 15 minutes
- Stage: 15 minutes
- Start: 60 seconds

> **Note:** Your administrator can change these defaults. Check with your administrator for the actual time limits set for app deployment.

Developers can change the time limit for starting apps through an app manifest or on the command line. For more information, see Deploying with App Manifests and Using App Health Checks.

Crash Events

If an app instance crashes, TAS for VMs automatically restarts it by rescheduling the instance on another container three times. After three failed restarts, TAS for VMs waits thirty seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, TAS for VMs stops trying to restart the app instance.

Evacuation

Certain operator actions require restarting VMs with containers hosting app instances. For example, an operator who updates stemcells or installs a new version of TAS for VMs must restart all the VMs in a deployment.

TAS for VMs automatically relocates the instances on VMs that are shutting down through a process called evacuation. TAS for VMs recreates the app instances on another VM, waits until they are healthy, and then shuts down the old instances. During an evacuation, developers may see their app instances in a duplicated state for a brief period.

During this app duplication process, singleton app instances may become temporarily unavailable if the replacement instance does not become healthy within the Diego Cell's evacuation timeout, which defaults to 10 minutes. Because of this, app developers with a low tolerance for brief downtime may prefer to run several instances of their app. See Run Multiple Instances to Increase Availability.

Shutdown

TAS for VMs requests a shutdown of your app instance in the following scenarios:

- When a user runs `cf scale`, `cf stop`, `cf push`, `cf delete`, or `cf restart-app-instance`
- As a result of a system event, such as the replacement procedure during Diego Cell evacuation or when an app instance stops because of a failed health check probe

To shut down the app, TAS for VMs sends the app process in the container a SIGTERM. By default, the process has ten seconds to shut down gracefully. If the process has not exited after ten seconds, TAS for VMs sends a SIGKILL.

By default, apps must finish their in-flight jobs within ten seconds of receiving the SIGTERM before TAS for VMs terminates the app with a SIGKILL. For instance, a web app must finish processing existing requests and stop accepting new requests. To modify the timeout period on the TAS for VMs tile or IST tile, go to the Advanced Settings tab and edit the “app graceful shutdown period” property. NOTE: This may increase the time it takes to drain diego cells causing increased deployment time.

If your apps require a longer period of time to finish in-flight jobs and gracefully shut down, you can increase the graceful shutdown period. For more information, see Configure Advanced Features in *Configuring TAS for VMs*.

Note: One exception to the cases mentioned above is when monit restarts a crashed Diego Cell rep or Garden server. In this case, TAS for VMs immediately stops the apps that are still running using SIGKILL.

How Diego balances app processes

Diego balances app processes over the virtual machines (VMs) in a TAS for VMs installation using the Diego Auction. When new processes need to be allocated to VMs, the Diego Auction determines which ones must run on which physical machines. The auction algorithm balances the load on VMs and optimizes app availability and resilience. This topic explains how the Diego Auction works.

For more information, see Diego Components and Architecture and the Auction repository on GitHub.

Tasks and long running processes

Diego Auction distinguishes between two types of jobs: **Tasks** and **Long-Running Processes** (LRPs).

- **Tasks** run once, for a finite amount of time. A common example is a staging task that compiles an app’s dependencies, to form a self-contained droplet that makes the app portable and runnable on multiple VMs. Other examples of tasks include making a database schema change, bulk importing data to initialize a database, and setting up a connected service.

- **Long-Running Processes** run continuously, for an indefinite amount of time. LRPs terminate only if they crash or are stopped. Examples include web servers, asynchronous background workers, and other applications and services that continuously accept and process input. To make high-demand LRPs more available, Diego may allocate multiple instances of the same application to run simultaneously on different VMs, often spread across Availability Zones that serve users in different geographic regions.

Diego Auction process is repeated whenever new jobs need to be allocated to VMs. Each auction distributes a current **batch** of work, Tasks and LRPs, that can include newly created jobs, jobs left unallocated in the previous auction, and jobs left orphaned by failed VMs. Diego does not redistribute jobs that are already running on VMs. Only one auction can take place at a time, which prevents placement collisions.

Ordering the auction batch

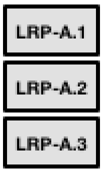
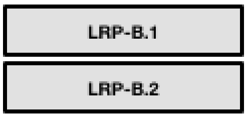
Diego Auction algorithm allocates jobs to VMs to fulfill the following outcomes, in decreasing **priority** order:

1. Keep at least one instance of each LRP running.
2. Run all of the Tasks in the current batch.
3. Distribute as much of the total desired LRP load as possible over the remaining available VMs, by spreading multiple LRP instances broadly across VMs and their Availability Zones.

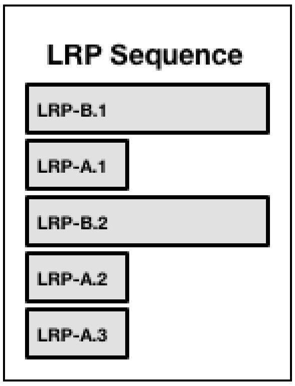
To achieve these outcomes, each auction begins with the Diego Auctioneer component arranging the batch jobs into a priority order. Some of these jobs might be duplicate instances of the same process that Diego needs to allocate for high-traffic LRPs to meet demand. The Auctioneer creates a list of multiple LRP instances based on the desired instance count configured for each process.

For more information, see the Step 2: Passing a request to the auctioneer process section of the *Diego Components and Architecture* topic.

For example, if the process LRP-A has a desired instance count of 3 and a memory load of 2, and process LRP-B has 2 desired instances and a load of 5, the Auctioneer creates a list of jobs for each process as follows:

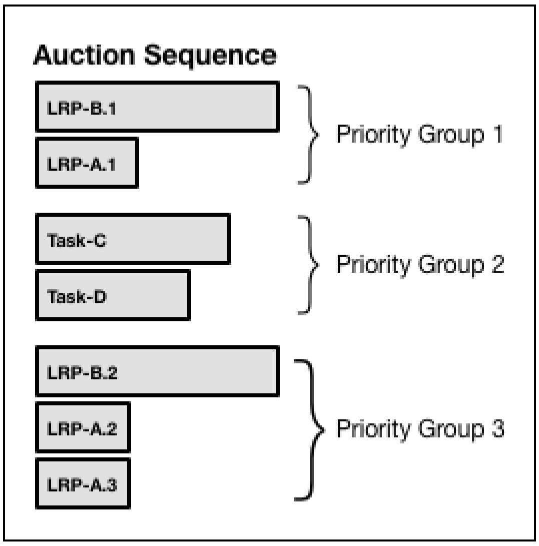
Process	Desired Instances	Load	Jobs
LRP-A	3		
LRP-B	2		

The Auctioneer then builds an ordered sequence of LRP instances by cycling through the list of LRPs in decreasing order of load. With each cycle, it adds another instance of each LRP to the sequence, until all desired instances of the LRP have been added. With the example above, the Auctioneer would order the LRPs like this:



The Auctioneer then builds an ordered sequence for all jobs, both LRPs and Tasks. Reflecting the auction batch priority order, the first instances of LRPs are first priority. Tasks are next, in decreasing order of load. Duplicate LRP jobs come last.

Adding one-time Task-C (load = 4) and Task-D (load = 3) to the above example, the priority order becomes:



The diagram above shows the following content:

- Title: Auction Sequence
  - Priority Group 1
    - LRP-B.1 (wide box)
    - LRP-A.1 (narrow box)
  - Priority Group 2
    - Task-C (medium-wide box)
    - Task-D (narrower box)
  - Priority Group 3
    - LRP-B.2 (wide box)

- LRP-A.2 (narrow box)
- LRP-A.3 (narrow box)

Auctioning the batch to the Diego Cells

With all jobs sorted in priority order, the Auctioneer allocates each in turn to one of the VMs. The process resembles an auction, where VMs “bid” with their suitability to run each job. Facilitating this process, each app VM has a resident Diego Cell that monitors and allocates the machine’s operation. The Diego Cell participates in the auction on behalf of the virtual machine that it runs on. For more information, see the Diego Components section of the *Diego Components and Architecture* topic.

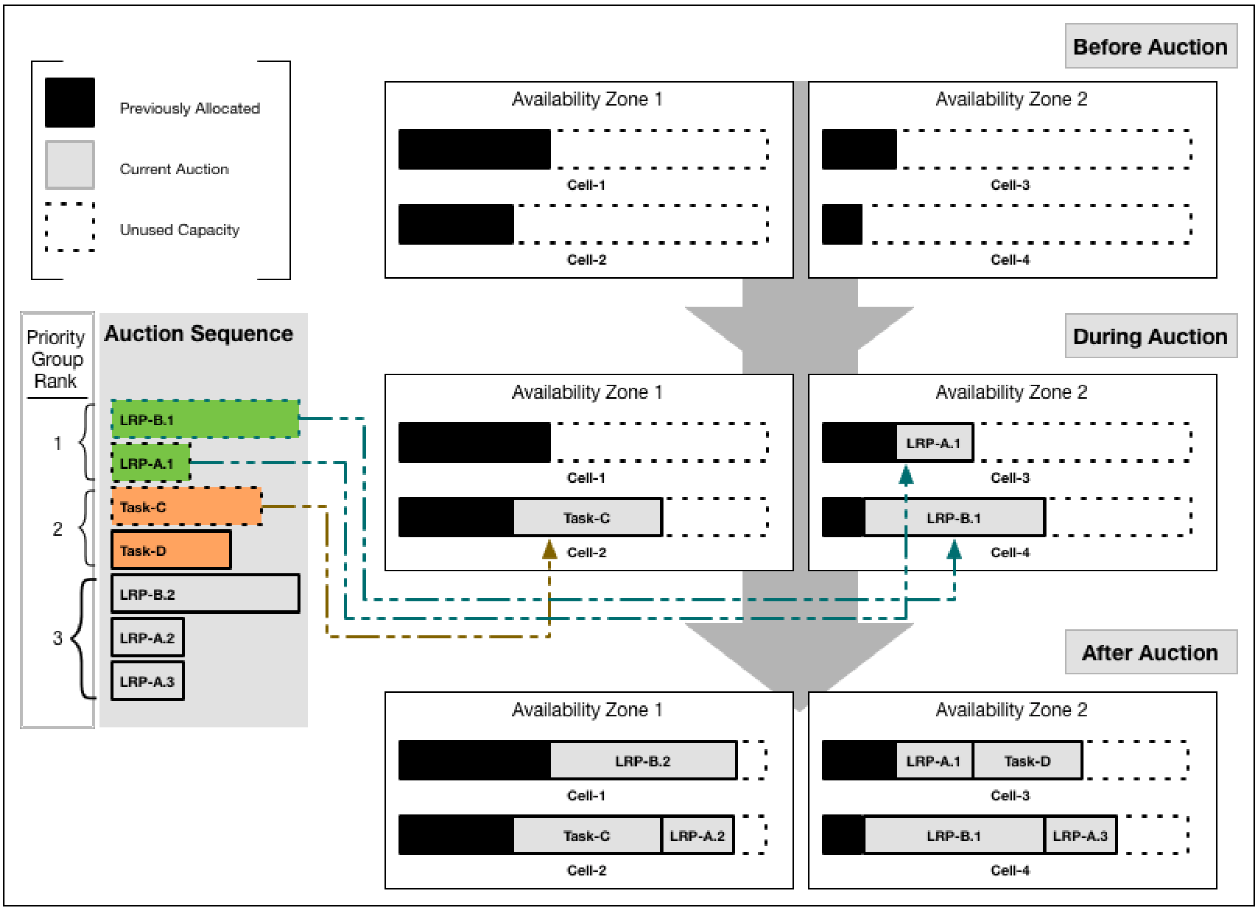
Starting with the highest priority job in the ordered sequence, the Auctioneer polls all the Diego Cells on their fitness to run the currently-auctioned job.

Diego Cells “bid” to host each job according to the following priorities, in decreasing order:

1. Allocate all jobs only to Diego Cells that have the correct software stack to host them, and sufficient resources given their allocation so far during this auction.
2. Allocate LRP instances into Availability Zones that are not already hosting other instances of the same LRP.
3. Within each Availability Zone, allocate LRP instances to run on Diego Cells that are not already hosting other instances of the same LRP.
4. Allocate any job to the Diego Cell that has lightest load, from both the current auction and jobs it has been running already. In other words, distribute the total load evenly across all Diego Cells.

This example auction sequence has seven jobs: five LRP instances and two Tasks.

The following diagram shows how the Auctioneer might distribute this work across four Diego Cells running in two Availability Zones:



If the Auctioneer reaches the end of its sequence of jobs, having distributed all jobs to the Diego Cells, it submits requests to the Diego Cells to execute their allotted work. If the Diego Cells run out of capacity to handle all jobs in the sequence, the Auctioneer carries the unallocated jobs over and merges them into the next auction batch, to be allocated in the next auction.

Triggering another auction

The Diego Auction process repeats to adapt a TAS for VMs deployment to its changing workload. For example, the BBS initiates a new auction when it detects that the actual number of running instances of LRPs does not match the number desired. Diego's BBS component monitors the number of instances of each LRP that are currently running. The BBS component periodically compares this number with the desired number of LRP instances, as configured by the user. If the actual number falls short of what is desired, the BBS triggers a new auction. In the case of a surplus of app instances, the BBS stops the extra instances and initiates another auction.

Cloud Controller also starts an auction whenever a Diego Cell fails. After any auction, if a Diego Cell responds to its work request with a message that it cannot perform the work after all, the Auctioneer carries the unallocated work over into the next batch. But if the Diego Cell fails to respond entirely, for example if its connection times out, the unresponsive Diego Cell might still be running its' work. In this case, the Auctioneer does not automatically carry the Diego Cell's work over to the next batch. Instead, the Auctioneer defers to the BBS to continue monitoring the states of the Diego Cells, and to reassign unassigned work later if needed.

High Availability in TAS for VMs

This topic tells you about the components used to ensure high availability in VMware Tanzu Application Service for VMs (TAS for VMs), vertical and horizontal scaling, and the infrastructure required to support scaling component VMs for high availability.

A system with high availability provides higher than typical uptime through redundancy of apps and component VMs. You can create the redundancy required for high availability in several ways, such as running VMs in multiple availability zones and using external blob storage solutions.

This topic provides guidance on configuring your TAS for VMs deployment for high availability.

Components of high availability deployments

You can use availability zones, external load balancers, and external blob storage to ensure high availability for your deployment.

Availability zones

Availability Zones (AZs) are locations where public cloud services offer data centers.

You can assign and scale components in multiple AZs to help maintain high availability through redundancy. To configure sufficient redundancy, deploy TAS for VMs across three or more AZs and assign multiple component instances to different AZs.

Always use an odd number of AZs. This ensures that your deployment remains available as long as greater than half of the AZs are available.

For example, a deployment with three AZs stays available when one AZ is unavailable. A deployment with five AZs stays available when two AZs are unavailable.

External load balancers

External load balancers distribute traffic coming from the internet to your internal network.

To ensure high availability for production environments, use a highly-available customer-provided external load balancing solution that does the following:

- Provides load balancing to each of the TAS for VMs Router IP addresses
- Supports SSL termination with wildcard DNS location
- Adds appropriate x-forwarded-for and x-forwarded-proto HTTP headers to incoming requests
- (Optional) Supports WebSockets

For lab and test environments, the `use-haproxy.yml` ops file enables HAProxy for your foundation.

For more information, see Using Your Own Load Balancer.

External blob storage

Blobs are large binary files, such as PDFs or images. To store blobs for high availability, use external storage such as Amazon S3 or an S3-compatible service.

You can also store blobs internally using WebDAV or NFS. These components run as single instances and you cannot scale them. For these deployments, use the high availability features of your IaaS to immediately recover your WebDAV or NFS server VM if it fails. Contact Support if you need assistance.

The singleton compilation components do not affect platform availability.

Scaling platform capacity

You can scale platform capacity in the following ways:

- **Vertical scaling**: Add memory and disk to each VM.
- **Horizontal scaling**: Add more VMs that run instances of TAS for VMs components.

The type of apps you host on TAS for VMs determines whether you scale vertically or horizontally.

For more information about scaling applications and maintaining app uptime, see the following topics:

- Scaling an app using cf scale
- Using blue-green deployment to reduce downtime

Scaling vertically

Scaling vertically means adding memory and disk to your component VMs.

To scale vertically, allocate and maintain the following:

- Free space on host Diego cell VMs so that apps expected to deploy can successfully be staged and run.
- Disk space and memory in your deployment so that if one host VM is down, all instances of apps can be placed on the remaining host VMs.
- Free space to handle one AZ going down if deploying in multiple AZs.

Scaling horizontally

Scaling horizontally means increasing the number of instances of VMs that run a functional component of a system.

You can horizontally scale most TAS for VMs component VMs to multiple instances for high availability.

You should also distribute the instances of components across different AZs to minimize downtime during ongoing operation, product updates, and platform upgrades. For more information about using AZs, see Availability Zones.

Recommended instance counts for high availability

For more information regarding rolling app deployments, see Scaling TAS for VMs.

The table below provides the instance counts VMware recommends for a high-availability deployment and the minimum instances for a functional deployment:

VMware Tanzu Application Service for VMs (TAS for VMs) Job	Recommended Instance Number for HA	Minimum Instance Number	Notes
Diego Cell	≥ 3	1	The optimal balance between CPU and memory sizing and instance count depends on the performance characteristics of the apps that run on Diego Cells. Scaling vertically with larger Diego Cells makes for larger points of failure, and more apps go down when a Diego Cell fails. On the other hand, scaling horizontally decreases the speed at which the system re-balances apps. Re-balancing 100 Diego Cells takes longer and demands more processing overhead than re-balancing 20 Diego Cells.
Diego Brain	≥ 2	1	For high availability, use at least one per AZ, or at least two if only one AZ.
Diego BBS	≥ 2	1	For high availability in a multi-AZ deployment, use at least one instance per AZ. Scale Diego BBS to at least two instances for high availability in a single-AZ deployment.
MySQL Server	3	1	If you use an external database in your deployment, then you can set the MySQL Server instance count to `0`. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.
MySQL Proxy	2	1	If you use an external database in your deployment, then you can set the MySQL Proxy instance count to `0`.
NATS Server	≥ 2	1	In a high-availability deployment, you might run a single NATS instance if your deployment lacks the resources to deploy two stable NATS servers. Components using NATS are resilient to message failures and the BOSH Resurrector recovers the NATS VM quickly if it becomes non-responsive.
Cloud Controller	≥ 2	1	Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.
Clock Global	≥ 2	1	For a high-availability deployment, scale the Clock Global job to a value greater than 1 or to the number of AZs you have.
Router	≥ 2	1	Scale the Gorouter to accommodate the number of incoming requests. Additional instances increase available bandwidth. In general, this load is much less than the load on Diego Cells.
UAA	≥ 2	1	
Doppler Server	≥ 2	1	Deploying additional Doppler servers splits traffic across them. For a high-availability deployment, VMware recommends at least two per AZ.
Loggregator Traffic Controller	≥ 2	1	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. For a high-availability deployment, VMware recommends at least two per AZ.
Syslog Scheduler	≥ 2	1	The Syslog Scheduler is a scalable component. For high availability, use at least one instance per AZ, or at least two instances if only one AZ is present.
CredHub	≥ 3	2	CredHub is a scalable component. For high availability, use at least one instance per AZ, or at least three instances if only one AZ is present.

Infrastructure for component scaling

The ability to scale component VMs is important for high availability. To scale component VMs, you must ensure that the surrounding infrastructure of your deployment supports VM scaling.

Learn about the infrastructure required to support scaling component VMs for high availability.

BOSH Resurrector

The BOSH Resurrector increases VMware Tanzu Application Service for VMs (TAS for VMs) availability in the following ways:

- Reacts to hardware failure and network disruptions by recreating VMs on active, stable hosts
- Detects operating system failures by continuously monitoring VMs and recreating them as required
- Continuously monitors the BOSH Agent running on each VM and recreates the VMs as required

The BOSH Resurrector continuously monitors the status of all VMs in a TAS for VMs deployment. The Resurrector also monitors the BOSH Agent on each VM. If either the VM or the BOSH Agent fail, the Resurrector recreates the VM on another active host. To enable the BOSH Resurrector, see the Enable BOSH Resurrector section of the *Using the BOSH Resurrector* topic.

Resource pools

Each IaaS has different ways of limiting resource consumption for scaling VMs. Consult with your IaaS administrator to ensure additional VMs and related resources, like IPs and storage, are available to scale.

For more information about configuring your resource pools according to the requirements of your deployment, see the Ops Manager configuration topic for your IaaS.

For information about configuring resource pools for Amazon Web Services, see Amazon EC2 FAQs in the Amazon documentation.

For information about configuring resource pools for OpenStack, see Manage projects and users in the OpenStack documentation.

For information about configuring resource pools for vSphere, see the Resource Config Pane section of the *Configuring BOSH Director on vSphere* topic.

Databases

For database services deployed outside TAS for VMs, use the high availability features included with your infrastructure. Also, configure backup and restore where possible. For more information about scaling internal database components, see Scaling TAS for VMs.

> **Note:** Data services may have single points of failure depending on their configuration.

If you need assistance, contact Support.

How TAS for VMs Maintains High Availability

VMware Tanzu Application Service for VMs (TAS for VMs) deployments make up several layers of high availability to keep your apps running during system failure. These layers include AZs, app health management, process monitoring, and VM resurrection.

Availability zones

Ops Manager supports deploying apps instances across multiple AZs. This level of high availability requires that you define AZs in your IaaS. Ops Manager balances the apps you deploy across the AZs you defined. If an AZ goes down, you still have app instances running in another.

You can configure your deployment so that Diego Cells are created across these AZs in the **Assign AZs and Networks** pane of the TAS for VMs tile.

Health management for app instances

If you lose app instances for any reason, such as a bug in the app or an AZ going down, Ops Manager restarts new instances to maintain capacity. Under Diego architecture, the nsync, BBS, and Cell Rep components track the number of instances of each app that are running across all of the Diego cells. When these components detect a discrepancy between the actual state of the app instances in the cloud and the desired state as known by the Cloud Controller, they advise the Cloud Controller of the difference and the Cloud Controller initiates the deployment of new app instances.

For more information about the nsync, BBS, and Cell Rep components, see the nsync, BBS, and Cell Rep section of the *TAS for VMs Components* topic.

Process monitoring

Ops Manager uses a BOSH agent, monit, to monitor the processes on the component VMs that work together to keep your apps running, such as nsync, BBS, and Cell Rep. If monit detects a failure, it restarts the process and notifies the BOSH agent on the VM. The BOSH agent notifies the BOSH Health Monitor, which triggers responders through plugins such as email notifications or paging.

Resurrector VMs

BOSH detects if a VM is present by listening for heartbeat messages that are sent from the BOSH agent every 60 seconds. The BOSH Health Monitor listens for those heartbeats. When the Health Monitor finds that a VM is not responding, it passes an alert to the Resurrector component. If the Resurrector is enabled, it sends the IaaS a request to create a new VM instance to replace the one that failed.

To enable the Resurrector, see the following pages for your particular IaaS: AWS, Azure, GCP, OpenStack, or vSphere.

TAS for VMs Security

This topic provides you with an overview of VMware Tanzu Application Service for VMs (TAS for VMs) security. For an overview of container security, see Container Security.

TAS for VMs implements the following measures to mitigate against security threats:

- Minimizes network surface area
- Isolates customer apps and data in containers
- Encrypts connections
- Uses role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access
- Ensures security of app bits in a multi-tenant environment
- Prevents possible denial of service attacks through resource starvation

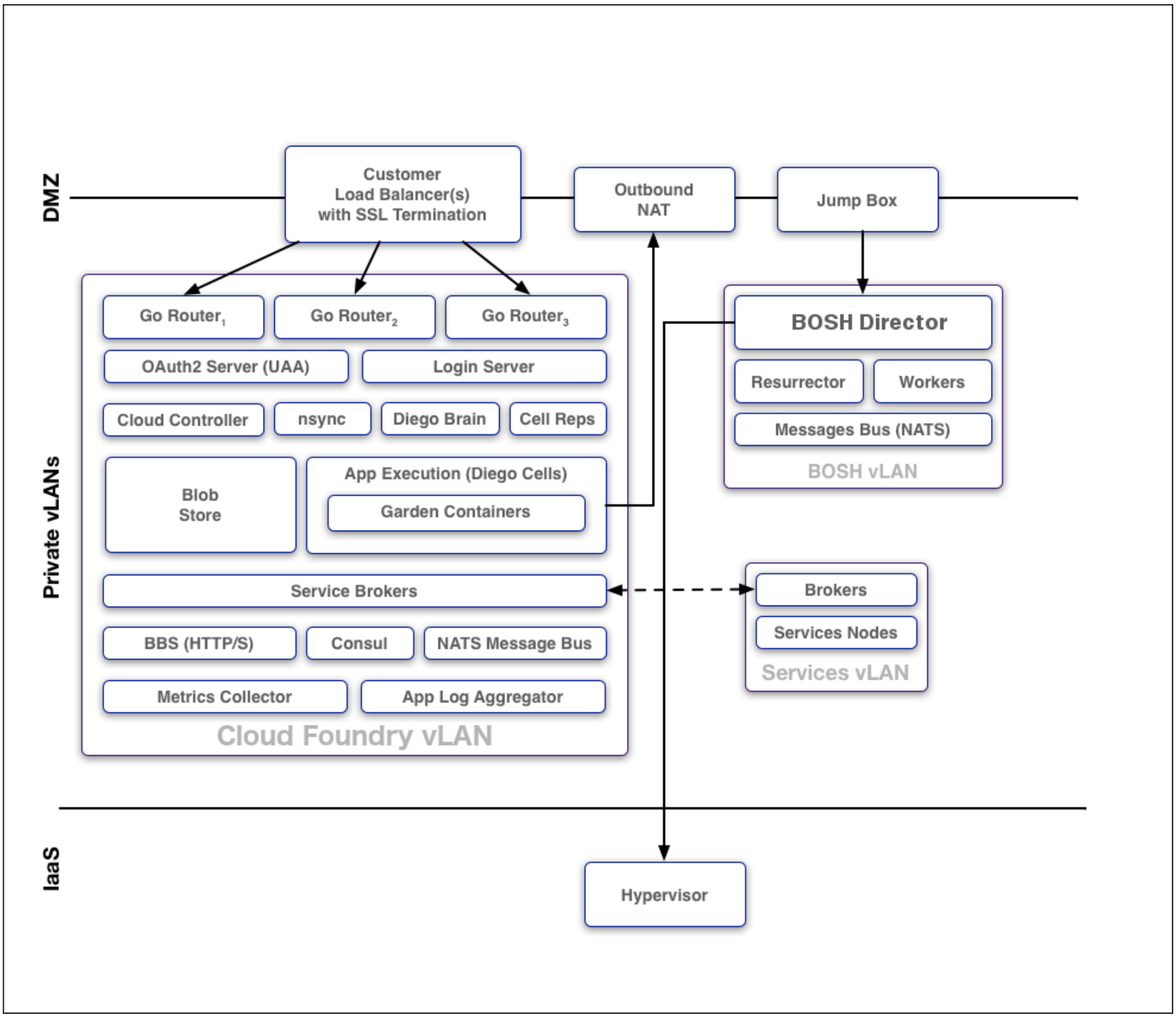
System boundaries and access

In a typical deployment of TAS for VMs, the components run on virtual machines (VMs) that exist within a VLAN.

In this configuration, the only access points visible on a public network are a load balancer that maps to one or more TAS for VMs routers and, optionally, a NAT VM and a jumpbox. Because of the limited number of contact points with the public internet, the surface area for possible security vulnerabilities is minimized.

As the diagram below shows the architecture of a typical deployment of TAS for VMs:

> **Note:** VMware recommends that you also install a NAT VM for outbound requests and a jumpbox to access the BOSH Director, though these access points are optional depending on your network configuration.



The above diagram shows the following components:

- DMZ
  - Customer Load Balancers with SSL Termination
  - Outbound NAT
  - Jumpbox
- Private vLANs
  - Cloud Foundry vLAN
    - Three Go Routers
    - OAuth2 Server (UAA)
    - Login Server
    - Cloud Controller
    - nsync
    - Diego Brain
    - Cell Reps
    - Blob Store

        - App Execution (Diego Cells) with Garden Containers
        - Service Brokers
        - BBS (HTTP/S)
        - Consul
        - NATS Message Bus
        - Metrics Collector
        - App Log Aggregator
    - Services vLAN
        - BOSH Director
        - Resurrector
        - Workers -Message Bus (NATS)
    - BOSH vLAN
        - Brokers
        - Services Nodes
- IaaS
    - Hypervisor

The diagram above shows the following communications:

- Customer Load Balancers send communication to the three Go Routers
- Outbound NAT communicates with Hypervisor
- BOSH Director communicates with the App Execution (Diego Cells)
- Service Brokers in Cloud Foundry vLAN communicates with Brokers in Services vLAN

Protocols

All traffic from the public internet to the Cloud Controller and UAA happens over HTTPS. Inside the boundary of the system, components communicate over a publish-subscribe (pub-sub) message bus NATS, HTTP, and SSL/TLS.

BOSH

Operators deploy TAS for VMs with BOSH. The BOSH Director is the core orchestrating component in BOSH: it controls VM creation and deployment, as well as other software and service lifecycle events. You use HTTPS to ensure secure communication to the BOSH Director.

> **Note:** VMware recommends that you deploy the BOSH Director on a subnet that is not publicly accessible, and access the BOSH Director from a jumpbox on the subnet or through VPN.

BOSH includes the following functionality for security:

- Communicates with the VMs it launches over NATS. Because NATS cannot be accessed from outside TAS for VMs, this ensures that published messages can only originate from a component within your deployment.
- Provides an audit trail through the `bosh tasks --all` and `bosh tasks --recent=VALUE` commands. `bosh tasks --all` returns a table that shows all BOSH actions taken by an operator or other running processes. `bosh tasks --recent=VALUE` returns a table of recent tasks, with `VALUE` being the number of recent tasks you want to view.
- Allows you to set up individual login accounts for each operator. BOSH operators have root access.

Note: BOSH does not encrypt data stored on BOSH VMs. Your IaaS might encrypt this data.

Isolation segments

Isolation segments provide dedicated pools of resources to which apps can be deployed to isolate workloads. Using isolation segments separates app resources as completely as if they were in different TAS for VMs deployments but avoids redundant management components and unneeded network complexity.

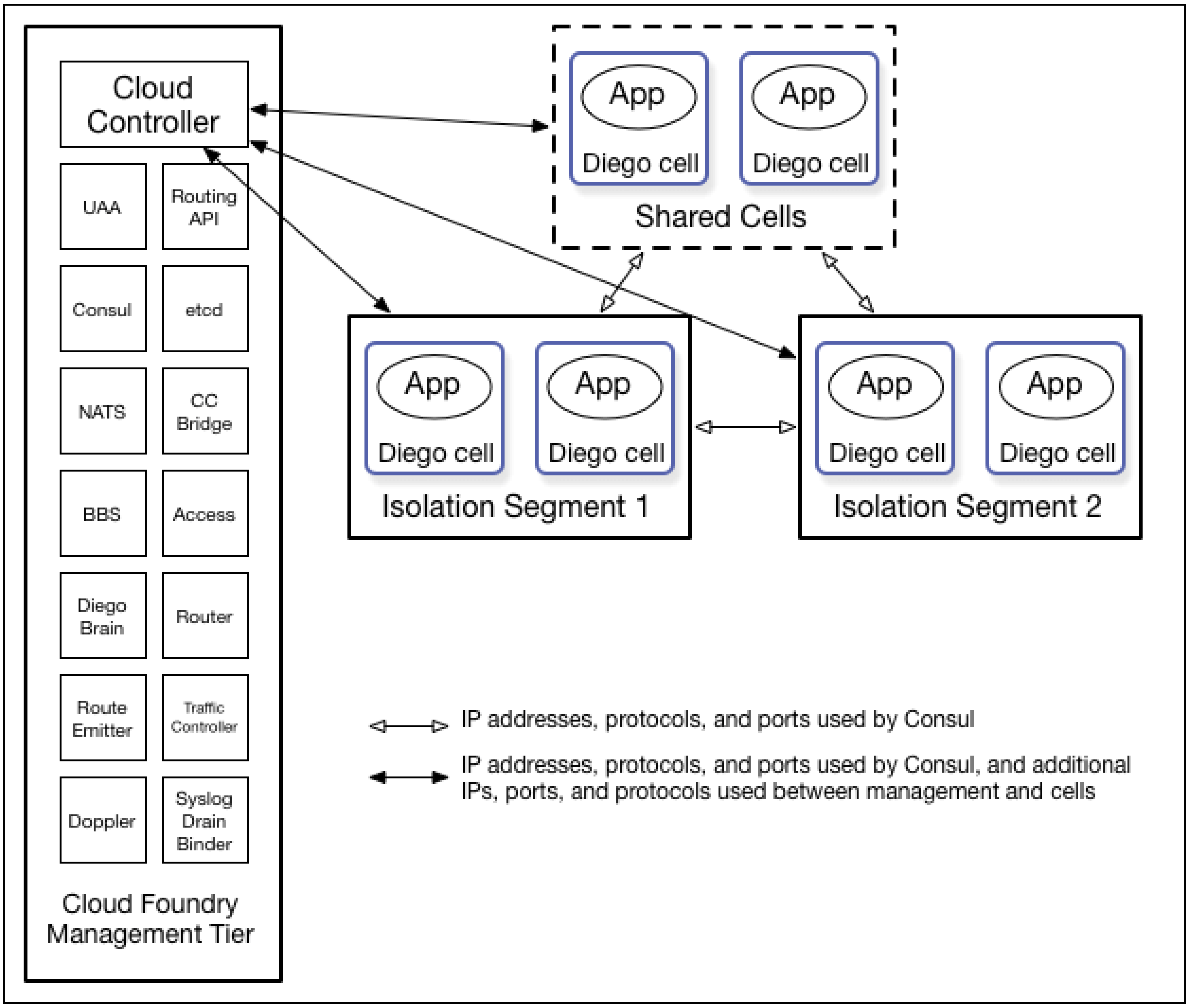
You can designate isolation segments for exclusive use by orgs and spaces within TAS for VMs. This guarantees that apps within the org or space use resources that are not also used by other orgs or spaces. For more information, see Orgs, Spaces, Roles, and Permissions.

Customers can use isolation segments for different reasons, including:

- To follow regulatory restrictions that require separation between different types of apps. For example, a health care company might not be able to host medical records and billing systems on the same machines.
- To dedicate specific hardware to different isolation segments. For example, to ensure that high-priority apps run on a cluster of high-performance hosts.
- To separate data on multiple clients, to strengthen a security story, or offer different hosting tiers.

In TAS for VMs, the Cloud Controller database (CCDB) identifies isolation segments by name and GUID, for example `30dd879c-ee2f-11db-8314-0800200c9a66`. The isolation segment object has no internal structure beyond these two properties at the TAS for VMs level, but BOSH associates the name of the isolation segment with Diego Cells, through their `placement_tag` property.

This diagram shows how isolation segments keep apps running on different pools of Diego Cells, and how the Diego Cells communicate with each other and with the management components:



For more information about how to create and manage isolation segments in an Ops Manager deployment, see Installing Isolation Segment and Managing Isolation Segments.

For API commands related to isolation segments, see Isolation Segments in the Cloud Controller API (CAPI) Reference.

Authentication and authorization

User Account and Authentication (UAA) is the central identity management service for TAS for VMs and its various components.

UAA acts as an OAuth2 Authorization Server and issues access tokens for apps that request platform resources. The tokens are based on the JSON Web Token and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the TAS for VMs platform, UAA acts as the user store and stores user passwords in the UAA database using bcrypt. UAA also supports connecting to external user stores through LDAP and SAML. Once an operator has configured the external user store, such as a corporate Microsoft Active Directory, users can use their LDAP credentials to gain access to the TAS for VMs platform instead of registering a separate account. Alternatively, operators can use SAML to connect to an external user store and enable single sign-on for users into the TAS for VMs platform.

Standard TAS for VMs deployments based on cf-deployment provide a UAA client `cf` that can be used to create OAuth 2 tokens using the Password Grant Flow for TAS for VMs users that are needed to access the CF API. This UAA client is also used by the CF CLI. The UAA client `cf` doesn't require a `client_secret`.

Managing user access with Role-Based Access Control

Apps that users deploy to TAS for VMs exist within a space. Spaces exist within orgs. To view and access an org or a space, a user must be a member of it. TAS for VMs uses role-based access control (RBAC), with each role granted permissions to either an org or a specified space. For more information about roles and permissions, see Orgs, Spaces, Roles, and Permissions.

For more information, see Getting Started with Apps Manager and Managing User Roles using Apps Manager.

Security for service broker integration

The Cloud Controller authenticates every request with the Service Broker API using HTTP or HTTPS, depending on which protocol that you specify during broker registration. The Cloud Controller rejects any broker registration that does not contain a username and password.

Service instances bound to an app contain credential data. Users specify the binding credentials for user-provided service instances, while third-party brokers specify the binding credentials for managed service instances. The VCAP_SERVICES environment variable contains credential information for any service bound to an app. TAS for VMs constructs this value from encrypted data that it stores in the CCDB. For more information about user-provided service instances, see User-Provided Service Instances.

> **Note:** The selected third-party broker controls how securely to communicate managed service credentials.

A third-party broker might offer a dashboard client in its catalog. Dashboard clients require a text string defined as a `client_secret`. TAS for VMs does not store this secret in the CCDB. Instead, TAS for VMs passes the secret to the UAA component for verification using HTTP or HTTPS.

Managing software vulnerabilities

TAS for VMs manages software vulnerability using releases and BOSH stemcells. New TAS for VMs releases are created with updates to address code issues, while new stemcells are created with patches for the latest security fixes to address any underlying operating system issues.

Ensuring security for app artifacts

TAS for VMs secures both the code and the configuration of an app using the following functionality:

- App developers push their code using the TAS for VMs API. TAS for VMs secures each call to the TAS for VMs API using the UAA and SSL.

- The Cloud Controller uses RBAC to ensure that only authorized users can access a particular app.
- The Cloud Controller stores the configuration for an app in an encrypted database table. This configuration data includes user-specified environment variables and service credentials for any services bound to the app.
- TAS for VMs runs the app inside a secure container. For more information, see Container Security.
- TAS for VMs operators can configure network traffic rules to control inbound communication to and outbound communication from an app. For more information, see the Network Traffic Rules section of the *Container Security* topic.

Security event logging and auditing

For operators, TAS for VMs provides an audit trail through the `bosh tasks` command. This command shows all actions that an operator has taken with the platform. Additionally, operators can redirect TAS for VMs component logs to a standard syslog server using the `syslog_daemon_config` property in the `metron_agent` job of `cf-release`.

For users, TAS for VMs records an audit trail of all relevant API invocations of an app. The Cloud Foundry Command Line Interface (cf CLI) command `cf events` returns this information.

Recommendations for running a secure deployment

To help run a secure deployment, VMware recommends:

- Configure UAA clients and users using a BOSH manifest. Limit and manage these clients and users as you would any other kind of privileged account.
- Deploy within a VLAN that limits network traffic to individual VMs. This reduces the possibility of unauthorized access to the VMs within your BOSH-managed cloud.
- Enable HTTPS for apps and SSL database connections to protect sensitive data transmitted to and from apps.
- Ensure that the jumpbox is secure, along with the load balancer and NAT VM.
- Encrypt stored files and data within databases to meet your data security requirements. Deploy using industry standard encryption and the best practices for your language or framework.
- Prohibit promiscuous network interfaces on the trusted network.
- Review and monitor data sharing and security practices with third-party services that you use to provide additional functionality to your app.
- Store SSH keys securely to prevent disclosure, and promptly replace lost or compromised keys.
- Use TAS for VMs’s RBAC model to restrict your users’ access to only what is necessary to complete their tasks.
- Use a strong passphrase for both your TAS for VMs user account and SSH keys.

- Use the IPsec for VMware Tanzu to encrypt IP data traffic within your deployment.

Compliance

For information about compliance in TAS for VMs, see the following topics:

- NIST Controls and VMware Tanzu Application Service for VMs: Provides a dedicated site that assesses TAS for VMs against NIST SP 800-53(r4) Controls.
- General Data Protection Regulation: Provides an overview of the General Data Protection Regulation (GDPR) and where TAS for VMs might store personal data.

Security for apps and services

This section links to topics that describe how TAS for VMs and TAS for VMs users manage security for apps and service instances.

- App Security Groups: Describes how App Security Groups (ASGs) work and how to manage them in TAS for VMs.
- Configuring SSH Access for TAS for VMs: Explains how to configure TAS for VMs to allow SSH access to app instances for debugging.
- Restricting App Access to Internal TAS for VMs Components: Details how to create ASGs, rules that allow internal outgoing communications from all apps in TAS for VMs, or the apps running in the same space.
- Configuring App Security Groups for Email Notifications: Describes how to define an ASG to enable app-generated notifications.
- Trusted System Certificates: Explains where apps can find trusted system certificates.
- Managing Access to Service Plans: Describes how to enable or disable access to service plans for a subset of users.
- Delivering Service Credentials to an App: Describes how to bind apps to service instances, which generate the credentials that enable the apps to use the service.
- Managing Service Keys: Explains how to create and manage service keys that enable apps to use service instances.

General Data Protection Regulation and TAS for VMs

This topic provides an overview of the General Data Protection Regulation (GDPR) and where VMware Tanzu Application Service for VMs (TAS for VMs) may store personal data.

Overview

GDPR came into effect on May 25, 2018 and impacts any company processing the data of EU citizens or residents, even if the company is not EU-based. The GDPR sets forth how companies can handle privacy issues, securely store data, and respond to security breaches.

Understand Personal Data

The GDPR grants data subjects certain rights, such as the right to obtain a copy of their personal data, object to the processing of personal data, and the right to have their personal data erased. Organizations subject to GDPR need to ensure that they can address and respond to requests by data subjects if they are processing their personal data.

Article 4, Section 1 of the GDPR defines personal data as follows:

'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

For more information, see the GDPR text.

Personal data can be collected, stored, and processed in a TAS for VMs deployment. VMware has performed a review of TAS for VMs components and determined that personal data may reside in the following areas:

- User Account and Authentication (UAA)
- Cloud Foundry API
- Routing
- Diego
- Notifications Service

Where Personal Data May Reside

The following sections explain how different TAS for VMs components collect personal data.

User Account and Authentication (UAA)

UAA is an open-source Cloud Foundry component that provides identity management features and identity-based security for apps and APIs. For more information, see User Account and Authentication.

GDPR	Workflow	What personal data is collected?	When is it collected?	Where is it stored?	How is it processed?	Who has access to it?

Business Initiation	User registers	• Username • Email address • First name (optional) • Last name (optional) • User ID (UAA GUID, generated)	User registration submission	UAA DB	Stored in UAA DB	• End user • UAA administrators
	Just-in-time provisioning: create user on user login	• Username • Email address • First name (optional) • Last name (optional) • User ID (UAA GUID, generated) • Additional attributes as defined by the organization	User login	UAA DB	Stored in UAA DB	UAA administrators
	Admin user makes a creation API call	• Username • Email address • First name (optional) • Last name (optional) • User ID (UAA GUID, generated) • Additional attributes as defined by the organization	Admin API call	UAA DB	Stored in UAA DB	UAA administrators

Business Execution	User self-updates profile	- Email address - First name (optional) - Last name (optional)	User registration submission	UAA DB	Stored in UAA DB	- End user - UAA administrators
	Just-in-time provisioning: user update	- Email address - First name (optional) - Last Name (optional) - Additional attributes as defined by the organization	User login	UAA DB	Stored in UAA DB	UAA administrators
	User logs in	- Current account cookie (generated) - Saved account cookie (generated)	User login	User browser	By UAA	- End user - UAA login page
	Admin user makes an update API call	- Email address - First name (optional) - Last name (optional) - Additional attributes as defined by the organization	Admin API call	UAA DB	Stored in UAA DB	UAA administrators
Delete User Flow	Admin user makes a hard delete API call	n/a	n/a	n/a	Deleted from UAA DB	UAA administrators
	Admin user makes a deactivation API call	n/a	n/a	n/a	Soft delete (records still held in database but user cannot login)	UAA administrators
Reports/Logs	Event or debug logs	Any information	When event happens	UAA logs	Depends on setup of Loggregator and log forwarding	BOSH administrators

Cloud Foundry API

The Cloud Foundry API release contains several components, including the Cloud Controller. For more information, see the Cloud Foundry API release README.

GDPR	Workflow	What personal data is collected?	When is it collected?	Where is it stored?	How is it processed?	Who has access to it?	How long is it kept?
Business Initiation	User makes a request for the first time	User ID	The first time a user makes a request to the API	Cloud Controller DB	It is used to identify permissions for the user	Ops Manager operator	As long as the user is part of the system
Business Execution	Troubleshooting API requests	• User ID • User agent • IP address	On each request	• Local VM: component and logs • Log aggregator used by Ops Manager operator	n/a	Ops Manager operator	• Local VM: 4 week maximum by default • Log aggregator as configured by Ops Manager operator

Audit Trails	Audit what changes a user makes	- Name - User ID - Email address	On specific API requests that mutate the state of resources	Audit Event Table in the Cloud Controller DB	n/a	- Ops Manager operator - Users that can view the resource that had an audited change	31 days
	Audit what changes a user makes	- IP Address - Email address - User ID - User name	On each request	- Local VM: CEF logs - Log aggregator used by Ops Manager operator	n/a	Ops Manager operator	- Local VM: 4 week maximum by default - Log aggregator as configured by Ops Manager operator
	Audit what user created a resource	- Name - User ID - Email address	When API resources are created	As part of the resource row in Cloud Controller DB	n/a	- Ops Manager operator - Users that can view the resource	As long as the resource exists

Routing

By default, the Gorouter logs include the `X-Forwarded-For` header, which may include the originating client IP. Under GDPR, client IP addresses should be considered personal data.

Deactivate Client IP Logging

In TAS for VMs v2.0 and later and Elastic Runtime v1.12, operators can deactivate logging of client IP addresses in the Gorouter.

To deactivate logging of client IP addresses:

1. Go to the Ops Manager Installation Dashboard and click the **TAS for VMs** or **Elastic Runtime** tile.
2. Click **Networking**.
3. Under **Logging of client IPs in the Gorouter**, select one of the two options:

- If the source IP address exposed by your load balancer is its own IP address, select **Disable logging of X-Forwarded-For header only**.
- If the source IP address exposed by your load balancer belongs to the downstream client, select **Disable logging of both source IP and X-Forwarded-For header**.

4. Click **Save**.
5. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes** to redeploy.

Diego

Diego is the container management system for TAS for VMs. For more information, see Diego Components and Architecture.

GDPR	Workflow	What personal data is collected?	When is it collected?	Where is it stored?	How is it processed?	Who has access to it?	How can I delete it?
Business Execution	Executing apps and tasks	No personal data is collected explicitly, but personal data may be encoded in app file contents or runtime metadata such as environment variables or start commands.	Runtime metadata is collected when Cloud Controller submits work specification to the Diego BBS API. File contents are collected when Diego schedules an app or a task on a Diego Cell.	Runtime metadata is stored in the Diego BBS DB. App file contents are cached on Diego Cells.	Runtime metadata is used to start processes inside app instance containers and to configure their environment. App file contents are presented as part of the app instance container filesystem.	Platform operators and other developers with access to the Cloud Controller space containing that app can view the data.	• To delete the runtime metadata stored in the Diego BBS DB, stop the app or cancel the task that includes that data. • To delete the app file contents stored in the running app and task containers, stop the app or cancel the task to destroy the containers. To destroy the app file contents stored in the download cache on the Diego Cells, recreate the Diego Cell VMs.

Reports/Logs	SSH proxy logs TAS for VMs user access.	UAA user name and ID	When the user authenticates for SSH access to an app.	The data is stored in a log file collocated with the SSH proxy instance handling the authentication request. This log file may also have its contents forwarded to a remote syslog destination.	No processing of the local log file is done automatically. If the log file contents are forwarded to a log aggregation service, they may be parsed and processed arbitrarily.	Only platform operators have access to the local log file. Platform operators or auditors may have access to these log contents in a downstream log aggregation service.	To delete the log lines containing the user ID: 1. Run `bosh recreate` on the VMs hosting the SSH proxy processes to remove all the logs on ephemeral disk. 2. `bosh ssh` into the VMs hosting the SSH proxy processes and remove specific log lines containing user IDs. 3. Scrub corresponding log lines from any log aggregation service.

Notifications Service

The Notifications Service enables operators to configure components of TAS for VMs to send emails to end users. For more information, see Getting Started with the Notifications Service.

GDPR	Workflow	What personal data is collected?	When is it collected?	Where is it stored?	How is it processed?	Who has access to it?
Business Execution	Sending email to UAA users	User ID	First email sent	The `receipts` table in the Notifications database	Stored in the Notifications database	Notifications operator making a database query
	UAA user unsubscribes globally	User ID	When the UAA user unsubscribes	The `global_unsubscribes` table in the Notifications database	Stored in the Notifications database	Notifications operator making a database query
	UAA user unsubscribes from a specific kind of email	User ID	When the UAA user unsubscribes	The `unsubscribes` table in the Notifications database	Stored in the Notifications database	Notifications operator making a database query
	UAA user unsubscribes from a campaign in the v2 API	User ID	When the UAA user unsubscribes	The `unsubscribes` table in the Notifications database	Stored in the Notifications database	Notifications operator making a database query

Reports/Logs	UAA user unsubscribes	User email address	When the UAA user unsubscribes	Log output	Loggregator	Loggregator Firehose users

Installing

In this section:

Installing TAS for VMs

In this section:

TAS for VMs Resource Requirements

This topic describes the resource requirements for installing VMware Tanzu Application Service for VMs (TAS for VMs).

General Requirements

The general requirements for deploying and managing an installation of Ops Manager and TAS for VMs are:

- A wildcard DNS record that points to your Gorouter or load balancer. Alternatively, you can use a service such as xip.io. For example, `203.0.113.0.xip.io`.
  - TAS for VMs gives each app its own hostname in your app domain.
  - With a wildcard DNS record, every hostname in your domain resolves to the IP address of your Gorouter or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your load balancer or Gorouter, every app deployed to the `example.com` domain resolves to the IP address of your Gorouter.
- At least one wildcard TLS certificate that matches the DNS record you set up above, `*.example.com`.
- Sufficient IP allocation:
  - One static IP address for either HAProxy or one of your Gorouters.
  - One static IP address for each job in the TAS for VMs tile. For a full list, see the **Resource Config** pane for each tile.
  - One static IP address for each job listed below:
    - Consul
    - NATS
    - File Storage

        - MySQL Proxy
        - MySQL Server
        - Backup Restore Node
        - HAProxy
        - Router
        - MySQL Monitor
        - Diego Brain
        - TCP Router
    - One IP for each VM instance created by the service.
    - An additional IP address for each compilation worker. The formula for total IPs needed is `IPs needed = static IPs + VM instances + compilation workers`.

> **Note:** VMware recommends that you allocate at least 36 dynamic IP addresses when installing TAS for VMs. BOSH requires additional dynamic IP addresses during installation to compile and deploy VMs, install TAS for VMs, and connect to services.

- One or more NTP servers if not already provided by your IaaS.
- (**Recommended**) A network without DHCP available for deploying the TAS for VMs VMs.

> **Note:** If you have DHCP, see Troubleshooting Deployment Problems in the Ops Manager documentation for information about avoiding issues with your installation.

- (**Optional**) External storage. When you deploy TAS for VMs, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the TAS for VMs tile. VMware recommends using external storage whenever possible. For more information about how file storage location affects platform performance and stability during upgrades, see Configure File Storage in *Configuring TAS for VMs for Upgrades*.
- (**Optional**) External databases. When you install TAS for VMs, you can select internal or external databases for the BOSH Director and for TAS for VMs. VMware recommends using external databases in production deployments. An external database must be configured to use the UTC timezone.
- (**Optional**) External user stores. When you deploy TAS for VMs, you can select a SAML user store for Ops Manager or a SAML or LDAP user store for TAS for VMs, to integrate existing user accounts.
- The most recent version of the Cloud Foundry Command Line Interface (cf CLI).

Instance Number and Scaling Requirements

By default, TAS for VMs deploys the number of VM instances required to run a highly available configuration. If you are deploying a test or sandbox TAS for VMs that does not require HA, then you can scale down the number of instances in your deployment.

For information about the number of instances required to run a minimal, non-HA TAS for VMs deployment, see Scaling TAS for VMs.

Configuring TAS for VMs

This topic tells you how to configure VMware Tanzu Application Service for VMs (TAS for VMs) as part of deploying Ops Manager.

Prerequisites

Before you begin this procedure, you must:

- Ensure that you have successfully completed the steps to prepare your environment for Ops Manager and install and configure the BOSH Director. For more information, see the VMware Tanzu Operations Manager configuration topic for your IaaS.
- If you plan to install the Ops Manager IPsec add-on, you must do so before installing any other tiles. VMware recommends installing IPsec immediately after Ops Manager and before installing the TAS for VMs Runtime tile. For more information, see Installing the IPsec Add-On for Ops Manager.

Add TAS for VMs to Ops Manager

Before you can configure TAS for VMs, you must add the TAS for VMs tile to your Ops Manager Installation Dashboard.

To add the TAS for VMs tile to your Ops Manager Installation Dashboard:

1. If you have not already downloaded TAS for VMs, log in to VMware Tanzu Network and click **VMware Tanzu Application Service for VMs**.
2. From the **Releases** drop-down menu, select the release to install and choose one of the following:
    1. Click **VMware Tanzu Application Service for VMs** to download the Pivotal Application Service `.pivotal` file.
    2. Click **Small Footprint TAS for VMs** to download the Small Footprint PAS `.pivotal` file. For more information, see Getting Started with Small Footprint TAS for VMs.
3. Go to the Ops Manager Installation Dashboard.
4. Click **Import a Product** to add your tile to Ops Manager. For more information, see Adding and Deleting Products.
5. Click the **VMware Tanzu Application Service for VMs** tile in the Installation Dashboard.

Assign AZs and Networks

Note: For Azure environments, this configuration pane is **Assign Networks** and does not include AZ configuration.

In the **Assign AZ and Networks** pane, you assign jobs to your Availability Zones (AZs) and networks.

To configure the **Assign AZ and Networks** pane:

1. Select **Assign AZ and Networks**.
2. From the **Network** drop-down menu, select the network where you want to run TAS for VMs.
3. Click **Save**.

Configure domains

In the **Domains** pane, you configure a wildcard DNS record for both the apps domain and system domain.

To configure the **Domains** pane:

1. Select **Domains**.
2. Enter the name of your system domain in the **System domain** field. The system domain defines your target when you push apps to TAS for VMs, such as your load balancer or HAProxy. TAS for VMs assigns system components such as UAA and Apps Manager to subdomains under this domain.
3. Enter the name of your apps domain in the **Apps domain** field. The apps domain is the default domain that apps use for their hostnames. TAS for VMs hosts each app at subdomains under this domain. You can use the Cloud Foundry Command Line Interface (cf CLI) to add or delete subdomains assigned to individual apps.
4. Click **Save**.

For additional guidance based on your installation method, see the table below:

Installation Method	Guidance
Manual	Enter the domains you created when preparing your environment for Ops Manager.
Terraform	Enter the values for `sys_domain` and `apps_domain` from the Terraform output.

Note: VMware recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes.

Configure Networking

Networking pane, you configure security and routing services for your IaaS. To configure the **Networking** pane:

1. For **Gorouter IPs** and **HAProxy IPs**, see the guidance below:
    - For **AWS**, **Azure**, and **GCP**, leave these fields blank. You do not need to complete these fields when deploying TAS for VMs on these infrastructures.
    - For **OpenStack** and **vSphere**, the values you enter in the **Gorouter IPs** and **HAProxy IPs** fields depend on whether you are using HAProxy in your deployment. Use the table below to determine how to complete these fields:

        Note: If you choose to assign specific IP addresses in either the **Gorouter IPs** or **HAProxy IPs** field, ensure that these IP addresses are in the subnet that you configured for TAS for VMs in Ops Manager.

Using HAProxy?	Gorouter IPs field	HAProxy IPs field
No	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **Gorouter IPs** field. You should specify more than one IP address for high availability. The IP addresses must be within your subnet CIDR block. 3. Configure your load balancer to forward requests for the domains that you have configured for your deployment to these IP addresses.	Leave this field blank.
Yes	Leave this field blank.	1. Choose IP addresses from the subnet you configured in Ops Manager. 2. Enter these IP addresses in the **HAProxy IPs** field. You should specify more than one IP address for high availability. 3. Configure your load balancer to forward requests for the domains you have configured for your deployment to these IP addresses.

2. For **SSH Proxy IPs** and **TCP router IPs**, see the guidance below:
    - For **AWS**, **Azure**, and **GCP**, leave these fields blank. You do not need to complete these fields when deploying TAS for VMs on these infrastructures.
    - For **OpenStack** and **vSphere**:
        - (Optional) In **SSH Proxy IPs**, add the IP address for your Diego Brain, which accepts requests to SSH into app containers on port 2222.

- (Optional) In **TCP router IPs**, add the IP addresses you want to assign to the TCP routers. You enable this feature at the bottom of this pane.

  > **Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This deactivates TCP routing.

3. Under **Certificates and private keys for the Gorouter and HAProxy**, you must provide at least one certificate and private key name and certificate key pair for the Gorouter and HAProxy. The Gorouter and HAProxy are enabled to receive TLS communication by default. You can configure multiple certificates for the Gorouter and HAProxy.

   > **Note:** When providing custom certificates, enter them in this order: `wildcard`, `Intermediate`, `CA`. For more information, see Creating a .pem File for SSL Certificate Installations in the DigiCert documentation.

   1. Click **Add** to add a name for the certificate chain and its private key pair. This certificate is the default used by the Gorouter and HAProxy. You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a certificate generated by the Ops Manager CA. For the values to use, see Providing a Certificate for Your TLS Termination Point.

      > **Note:** If you configured Ops Manager Front End without a certificate, you can use this new certificate to complete Ops Manager configuration. To configure your Ops Manager Front End certificate, see Configure Front End in *Preparing to Deploy Ops Manager on GCP*.

   2. If you want to configure multiple certificates for the Gorouter and HAProxy, click **Add** and fill in the appropriate fields for each additional certificate key pair. For details about generating certificates in Ops Manager for your wildcard system domains, see Providing a Certificate for Your TLS Termination Point.

   > **Note:** Ensure that you add any certificates that you generate in this pane to your infrastructure load balancer.

4. (Optional) When validating client requests using mutual TLS, the Gorouter trusts multiple certificate authorities (CAs) by default. To configure HAProxy to trust the same CAs as the Gorouter, enter your CA certificates under **Certificate Authorities trusted by the Gorouter and HAProxy**. All CA certificates should be appended together into a single collection of PEM-encoded entries.

5. (Optional) To deactivate HTTP/2 ingress and egress for the Gorouter, clear the **Enables the HTTP/2 protocol for communicating with apps** checkbox. HTTP/2 is enabled by default. For more information about HTTP/2 routing, see Supporting HTTP/2.

6. Under **Select the range of TLS versions supported by the Gorouter and HAProxy**, select the range of TLS versions to use in Gorouter and HAProxy communications. The Gorouter and HAProxy support TLS v1.2 to TLS v1.3 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see TLS Cipher Suite Support in *Securing Traffic into TAS for VMs*.

7. (Optional) Under **Balancing algorithm used by the Gorouter**, select your load balancing algorithm for the Gorouter. VMware recommends the default option of **Round robin** for most use cases. Some use cases, such as those where apps have long-lived connections, may benefit from the least connection algorithm. For more information, see HTTP Routing.

8. Configure **Logging of client IPs in the Gorouter**. The **Log client IPs** option is set by default. To comply with the General Data Protection Regulation (GDPR), select one of these options To deactivate logging of client IP addresses:
    - If your load balancer exposes its own source IP address, select **Disable logging of X-Forwarded-For header only**.
    - If your load balancer exposes the source IP of the originating client, select **Disable logging of both source IP and X-Forwarded-For header**.

9. Under **TLS termination point**, configure how TAS for VMs handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment. The table below indicates which option to choose based on your deployment configuration:

Deployment Configuration	TLS Option	Additional Notes
◦ The load balancer is terminating TLS, and ◦ The load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	Infrastructure load balancer	Both HAProxy and the Gorouter forward the XFCC header when included in the request.
◦ The load balancer is configured to pass through the TLS handshake through TCP to the instances of HAProxy, and ◦ HAProxy instance count is more than `0`	HAProxy	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change:** If you select the **The Gorouter does not request client certificates** option in the **Gorouter behavior for client certificate validation** field, the XFCC header cannot be delivered to apps.

◦ The load balancer is configured to pass through the TLS handshake through TCP to instances of the Gorouter	Gorouter	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for SSH directly to the Diego Brain, consider reducing HAProxy instances to `0`. **Breaking Change:** If you select the **The Gorouter does not request client certificates** option in the **Gorouter behavior for client certificate validation** field, the XFCC header cannot be delivered to apps.

For a description of the behavior of each configuration option, see Forward Client Certificate to Apps in *HTTP Routing*.

10. To configure HAProxy to handle client certificates, select one of the following options in the **HAProxy behavior for client certificate validation** field:
    - **HAProxy does not request client certificates:** This option requires mutual authentication, which makes it incompatible with **TLS termination point** option **HAProxy**. HAProxy does not request client certificates, so the client does not provide them and no validation occurs. This is the default configuration.
    - **HAProxy requests but does not require client certificates:** The HAProxy requests client certificates in TLS handshakes and validates them when presented, but does not require them. This option is required if you want to enable mutual TLS app identity verification and TLS is terminated for the first time at HAProxy.

      Caution: Upon upgrade, TAS for VMs fails to receive requests if your load balancer is configured to present a client certificate in the TLS handshake with HAProxy but HAProxy has not been configured with the certificate authority used to sign it. To mitigate this issue, select **HAProxy does not request client certificates** or configure the HAProxy with the appropriate CA.

11. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Gorouter behavior for client certificate validation** field:
    - **The Gorouter does not request client certificates:** Client certificates are not requested, so the client does not provide them and validation of client certificates does not occur. This option is incompatible with the **TLS termination point** options **HAProxy** and **Gorouter** because these options require mutual authentication.
    - **The Gorouter requests but does not require client certificates:** The Gorouter requests client certificates in TLS handshakes and validates them when presented, but does not require them. This is the default configuration.

- **The Gorouter requires client certificates:** The Gorouter validates that the client certificate is signed by a Certificate Authority that the Gorouter trusts. If the Gorouter cannot validate the client certificate, the TLS handshake fails.

> ⚠ **Caution:** Requests to the platform fail upon upgrade if your load balancer is configured with client certificates and the Gorouter does not have the CA. To mitigate this issue, select **The Gorouter does not request client certificates**.

12. In the **TLS cipher suites for the Gorouter** field, review the TLS cipher suites for TLS handshakes between the Gorouter and front end clients such as load balancers or HAProxy. The default value for this field is `ECDHE-RSA-AES128-GCM-SHA256:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`.

    To modify the default configuration, use an ordered, colon-separated list of Golang-supported TLS cipher suites in the OpenSSL format. This field does not apply to TLS v1.3. For TLS v1.3, the Gorouter only uses a set of default ciphers, and this is not configurable.

    Operators should verify that the ciphers are supported by any clients or front end components that initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see TLS Cipher Suite Support in *Securing Traffic into TAS for VMs*.

    Verify that every client participating in TLS handshakes with the Gorouter has at least one cipher suite in common with the Gorouter.

    > **Note:** Specify cipher suites that are supported by the versions configured under **Select the range of TLS versions supported by the Gorouter and HAProxy**. For example, TLS v1.3 does not support configuring cipher suites. If you select **TLSv1.3 only**, you cannot configure cipher suites for the Gorouter or HAProxy.

    > **Note:** AWS Classic Load Balancers do not support TAS for VMs's default cipher suites. For more information about configuring your AWS load balancers and Gorouter, see TLS Cipher Suite Support by AWS Load Balancers in *Securing Traffic into TAS for VMs*.

13. In the **TLS cipher suites for HAProxy** field, review the TLS cipher suites for TLS handshakes between HAProxy and front end clients such as load balancers and the Gorouter. The default value for this field is: `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384`

    To modify the default configuration, use an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. Operators should verify that the ciphers are supported by any clients or front end components that initiate TLS handshakes with HAProxy.

Verify that every client participating in TLS handshakes with HAProxy has at least one cipher suite in common with HAProxy.

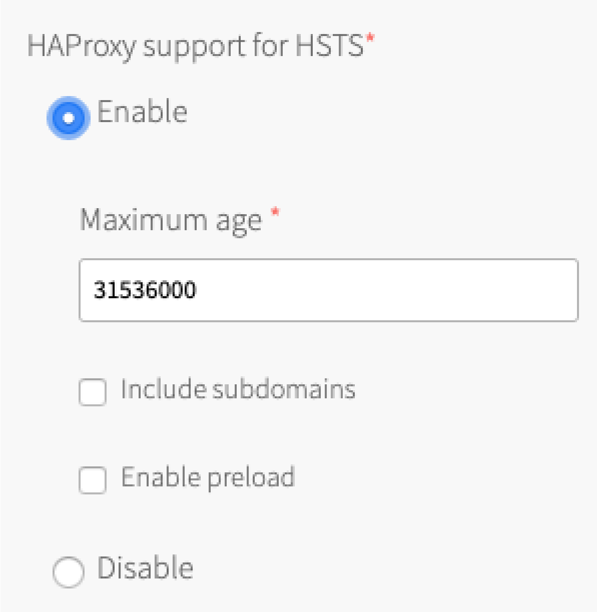
> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by the Gorouter and HAProxy** field.

14. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Enable** or **Disable** based on your deployment layout.
    - To enable communication between HAProxy and the Gorouter:
        1. Verify that **Enable** is selected.
        2. In the **Certificate authority for HAProxy back end** field, provide the CA that signed the certificate you configured in the **Certificates and private keys for the Gorouter and HAProxy** field.

            > **Note:** If you used the **Generate RSA Certificate** link to generate a certificate, then the CA to specify is the Ops Manager CA, which you can locate at the `/api/v0/certificate_authorities` endpoint in the Ops Manager API.

        3. Make sure that the Gorouter and HAProxy have TLS cipher suites in common in the **TLS cipher suites for the Gorouter** and **TLS cipher suites for HAProxy** fields.
           For more information, see Terminating TLS at the Load Balancer and Gorouter in *Securing Traffic into TAS for VMs*, Providing a Certificate for Your TLS Termination Point, and Using the Ops Manager API.
    - To use non-encrypted communication between HAProxy and the Gorouter:
        1. Select **Disable**.
        2. If you are not using HAProxy, set the number of HAProxy job instances to `0` in the **Resource Config** pane. For more information, see Scale Down and Deactivate Resources.
           For more information, see Terminating TLS at the Gorouter Only and Terminating TLS at the Load Balancer Only in *Securing Traffic into TAS for VMs*.
15. (Optional) To force browsers to use HTTPS when making requests to HAProxy, select **Enable** under **HAProxy support for HSTS** and complete these optional configuration

steps:



HAProxy support for HSTS*

● Enable

Maximum age *

31536000

□ Include subdomains

□ Enable preload

○ Disable

- Enter a **Maximum age** in seconds for the HSTS request. HAProxy forces HTTPS requests from browsers for the duration of this setting. The maximum age is one year, or 31536000 seconds.
- Enable the **Include subdomains** check box to force browsers to use HTTPS requests for all component subdomains.
- Select the **Enable preload** check box to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from HAProxy.

16. (Optional) If you want the Gorouter or HAProxy to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on the Gorouter and HAProxy** checkbox. When selected, the Gorouter and HAProxy do not listen on port 80.
17. (Optional) Select the **Disable insecure cookies on the Gorouter** checkbox to turn on the secure flag for cookies generated by the Gorouter.
18. (Optional) To deactivate the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the Gorouter** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see HTTP Headers for Zipkin Tracing in *HTTP Routing*.
19. (Optional) The **Enable the Gorouter to write access logs locally** checkbox is selected by default. VMware recommends deselecting this checkbox for high-traffic deployments, since logs may not be rotated fast enough and can fill up the disk.
20. 1. By default, TAS for VMs Gorouters accept all traffic for apps deployed to an isolation segment created by the Isolation Segment tile. To configure TAS for VMs Gorouters to reject requests for apps within isolation segments, enable the **Gorouters reject requests for isolation segments** checkbox. Do not enable this option without deploying Gorouters for each isolation segment. To allow Gorouters to accept traffic for specific isolation segments, check this box and list the Isolation Segment names

in the field titled “Always allow Gorouter to accept requests for these Isolation Segments”. For more information, see Installing Isolation Segment and Sharding Gorouters for Isolation Segments in *Routing for Isolation Segments*.

21. (Optional) By default, Gorouter support for the PROXY protocol is deactivated. To enable the PROXY protocol, select the **Enable support for PROXY protocol in the Gorouter** checkbox. When enabled, client-side load balancers that terminate TLS but do not support HTTP can pass along information from the originating client. Enabling this option may impact Gorouter performance. For more information about enabling the PROXY protocol in the Gorouter, see the *About HTTP Header Forwarding* sections in Securing Traffic into TAS for VMs.
22. In the **Route services** section, choose either **Enable** or **Disable**. Route services are a class of marketplace services that perform filtering or content transformation on app requests and responses. For more information, see Route Services and List Marketplace Services in *Managing Service Instances with the cf CLI*.
    1. If you enabled route services, you can also configure the **Bypass security checks for route service lookup** field. VMware recommends that you do not enable this field because it has potential security concerns. However, you may need to enable it if your load balancer requires mutual TLS from clients. For more information, see Configuring Route Service Lookup.
23. (Optional) If you want to limit the number of app connections to the back end, enter a value in the **Maximum connections per back end** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps. No value or a value of `0` sets no limit.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses App Metrics, you can also obtain this peak concurrent connection information from Network Metrics in *Monitoring and Troubleshooting Apps with App Metrics*. The default value is `500`.
24. Under **Keep-alive connections for the Gorouter**, select **Enable** or **Disable**. Keep-alive connections are enabled by default. For more information, see Keep-Alive Connections in *HTTP Routing*.
25. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Gorouter timeout to back ends** field.
26. (Optional) Use the **Front end idle timeout for the Gorouter and HAProxy** field to help prevent connections from your load balancer to the Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that the Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive connections.

    Set the value higher than the back end idle timeout for your load balancer to avoid the race condition where the load balancer sends a request before it discovers that the Gorouter or HAProxy has closed the connection.

See the table below for specific guidance and exceptions to this rule:

IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so VMware recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so VMware recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds. For GCP HTTP load balancers, VMware recommends a value greater than `600`. For GCP TCP load balancers, VMware recommends a value less than `600` to force the load balancer to send a TCP RST.
Other	Set the timeout value to be greater than that of the load balancer's back end idle timeout.

Note: Do not set a front end idle timeout lower than six seconds.

27. (Optional) Use **Gorouter drain timeout** to specify the amount of time, in seconds, that the Gorouter continues to serve existing connections before shutting down. After the timeout is reached, all remaining connections are terminated so the Gorouter can restart. Set the value lower than back end request timeout and idle timeout to reduce the drain time during deploys. The default value is `900`.
28. (Optional) Increase the value of **Load balancer unhealthy threshold** to specify the amount of time, in seconds, that the Gorouter continues to accept connections before shutting down. During this period, health checks may report the Gorouter as unhealthy, which causes load balancers to failover to other Gorouters. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a Gorouter instance unhealthy, given repeated failed health checks.
29. (Optional) Modify the value of **Load balancer healthy threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Gorouter instance started. This allows an external load balancer time to register the Gorouter instance as healthy.
30. (Optional) If app developers in your organization want certain HTTP headers on HTTP Requests to appear in their app logs with information from the Gorouter, specify them in the **HTTP headers to log** field with a comma-separated list. For example, to support app developers that deploy Spring apps to TAS for VMs, you can enter Spring-specific HTTP headers.
31. (Optional) If app developers in your organization want certain HTTP headers on HTTP Requests to appear in their app logs with information from the Gorouter, specify them in the **HTTP headers to log** field with a comma-separated list. For example, to support app developers that deploy Spring apps to TAS for VMs, you can enter Spring-specific HTTP headers. The **Max request header size in kb** property allows TAS for VMs to prevent denial-of-service attacks from requests with large headers. This value is the max size for all of a request's headers combined, including a request's header keys, header values, and all

values in the request line. A request whose headers exceed this value will receive a 431 status code. It is recommended to set this to 48kb, however, up to 1024kb is allowed for backwards compatibility. The value of this property must be between 1 and 1024. This property does not limit the size of the request body.

32. If you expect requests larger than the default maximum of 16.384 KB, enter a new value in bytes for **HAProxy request maximum buffer size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers. Requests larger than the maximum value result in a gateway error.
33. If your TAS for VMs deployment uses HAProxy and you want it to receive traffic only from specific sources, configure these fields:
    - **HAProxy protected domains:** Enter a comma-separated list of domains to protect from unknown source requests.
    - (Optional) **HAProxy trusted CIDRs:** Enter a space-separated list of CIDRs to limit which IP addresses from the **HAProxy protected domains** can send traffic to TAS for VMs.
34. The **Loggregator port** defaults to 443 if left blank. For AWS environments that are not using an Application Load Balancer, enter `4443`.
35. For **Container network interface plugin**, select one of these options:
    - **Silk:** This option is the default container network interface (CNI) for TAS for VMs.
    - **External:** Select this if you are deploying the VMware NSX-T Container Plug-in for TAS for VMs.

      If you select **External**, follow the instructions in Deploying TAS for VMs with NSX-T Networking in addition to the TAS for VMs configuration instructions in this topic.

      > **Caution:** The NSX-T integration only works for fresh installs of TAS for VMs. If your TAS for VMs tile is already deployed and running with Silk as its CNI, you cannot change the CNI plugin to NSX-T.
36. If you selected **Silk** in the previous step, review these fields:
    1. (Optional) For **App network maximum transmission unit**, enter an MTU value in bytes. The default value is `1454`. Some configurations may require a smaller MTU value. For example, networks that use GRE tunnels may require a smaller MTU.

       For Azure environments, VMware recommends entering `1400` or lower to prevent Azure from fragmenting the packets. For more information, see TCP/IP performance tuning for Azure VMs.
    2. (Optional) For **Overlay subnet**, enter an IP range for the overlay network. If you do not set a custom range, Ops Manager uses `10.255.0.0/16`. The overlay network IP range you configure must not conflict with any other IP addresses in your network.

       > **Caution:** Editing this property may cause container-to-container (C2C) networking downtime and security implications the next time

> you re-deploy TAS for VMs.

3. Enter a UDP port number in the **VXLAN tunnel endpoint port** box. This is the host port that receives VXLAN packets. If you do not set a custom port, Ops Manager uses 4789.

4.
    - For **Denied logging interval**, set the per-second rate limit for packets blocked by either a container-specific networking policy or by App Security Group (ASG) rules applied across the space, org, or deployment. This field defaults to `1`. For more information, see Policies in *Container-to-Container Networking* and Restricting App Access to Internal TAS for VMs Components.
    - For **UDP logging interval**, set the per-second rate limit for UDP packets sent and received. This field defaults to `100`.
    - To enable logging for app traffic, enable the **Log traffic for all accepted and denied app packets** checkbox. This increases log volume. For more information, see App Traffic Logging in *Configuring Logging in TAS for VMs*.

5. The **Enable Silk network policy enforcement** check box is selected by default. To deactivate Silk network policy enforcement between apps, deselect the check box. Deactivating the network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

> ⚠ **Caution:** Deactivating this feature allows all app containers to access any other app container with no restrictions.

37. For **DNS search domains**, enter the domains for your app containers to use as DNS search domains. Specify multiple DNS search domains as a comma-separated list.

38. For **Database connection timeout**, set the connection timeout for clients of the policy server and Silk databases, in seconds. The default value is `120`. You may need to increase this value if your deployment experiences timeout issues related to container-to-container networking.

39. TCP routing is deactivated by default. You should enable this feature if your DNS sends TCP traffic through a load balancer rather than directly to a TCP router. To enable TCP routing:

    1. Under **TCP routing**, select **Enable**.

    > **Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This deactivates TCP routing.

    2. For **TCP routing ports**, enter one or more ports to which the load balancer forwards requests. To support multiple TCP routes, VMware recommends allocating multiple ports. Do one of these steps:
        - To allocate a single port or range of ports, enter a single port or a range of ports.

> **Note:** If you configured AWS for Ops Manager manually, enter `1024-1123` which corresponds to the rules you created for `pcf-tcp-elb`.

- To allocate a list of ports:
  1. Enter a single port in the **TCP routing ports** field.
  2. After deploying TAS for VMs, follow the procedure in Router Groups: Modify TCP port reservations.

3. (Optional) For **TCP request timeout**, modify the default value of 300 seconds. This field determines when the TCP router closes idle connections from clients to apps that use TCP routes. You may want to increase this value to enable developers to push apps that require long-running idle connections with clients.

4. Follow these additional instructions based on your IaaS:

IaaS	Instructions
GCP	Specify the name of a GCP TCP load balancer in the **LOAD BALANCER** field of the **TCP Router** job in the **Resource Config** pane. You configure this later on in TAS for VMs. For more information, see Configure Resources.
AWS	Specify the name of a TCP ELB in the **LOAD BALANCER** field of the **TCP Router** job in the **Resource Config** pane. You configure this later on in TAS for VMs. For more information, see Configure Resources.
Azure	Specify the name of a Azure load balancer in the **LOAD BALANCER** field of the **TCP Router** job in the **Resource Config** pane. You configure this later on in TAS for VMs. For more information, see Configure Resources.
OpenStack and vSphere	1. Return to the top of the **Networking** pane. 2. In the **TCP router IPs** field, ensure that you have entered IP addresses that are within your subnet CIDR block. These are the same IP addresses you configured your load balancer with in the Pre-Deployment Steps in *Enabling TCP Routing*, unless you configured DNS to resolve the TCP domain name directly to an IP you have chosen for the TCP router.

40. (Optional) For additional security, enter headers that you want the Gorouter to remove from app responses in **Remove specified HTTP response headers**.

41. (Optional) In the **Sticky session cookies** field, enter one or more sticky session cookie names. The default cookie name is `JSESSIONID`. Some apps require a different cookie name. For example, Spring WebFlux requires `SESSION` for the cookie name. Gorouter uses these cookies to support session affinity, or *sticky sessions*. For more information, see Session Affinity in *HTTP Routing*.

42. Click **Save**.

Configure App Containers

In the **App Containers** pane, you enable microservice frameworks, private Docker registries, and other services that support your apps at the container level.

To configure the **App Containers** pane:

1. Select **App Containers**.
2. The **Enable custom buildpacks** checkbox governs the ability to pass a custom buildpack URL to the `-b` option of the `cf push` command. By default, this ability is enabled, letting developers use custom buildpacks when deploying apps. Deactivate this option by deselecting the checkbox. For more information about custom buildpacks, see Buildpacks.
3. The **Allow SSH access to app containers** checkbox controls SSH access to app instances. Select the checkbox to permit SSH access across your deployment, and deselecte it to prevent all SSH access. For more information about SSH access permissions at the space and app scope, see App SSH Overview.

   > **Note:** If you are using a load balancer instead of HAProxy, ensure that it has port 2222 open to enable SSH traffic.

   You can give SSH access to an app only if an admin assigns you a Space Developer role in the space where the app runs. For more information, see Manage App Space Roles in *Managing User Roles with Apps Manager*.
4. To enable SSH access for new apps by default in spaces that allow SSH, select the **Enable SSH when an app is created** checkbox. If you deselect this checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`.
5. Choose how the Gorouter verifies app identity to enable encryption and prevent misrouting under **Gorouter app identity verification**:

   > **Note:** This feature does not work if the **Disable SSL certificate verification for this environment** checkbox is enabled in the **Networking** pane.

   - **The Gorouter uses TLS to verify app identity**: Enables the Gorouter to verify app identity using TLS. This is the default option.
   - **The Gorouter and apps use mutual TLS to verify each other's identity**: Enables your apps and the Gorouter to verify each other's identity using mutual TLS (mTLS). This option deactivates TCP routing because app containers accept incoming communication only from the Gorouter.

   For more information, see Preventing Misrouting in *HTTP Routing*.
6. You can configure TAS for VMs to run app instances in Docker containers by providing a comma-separated list of their IP address ranges in the **Private Docker insecure registry allow list** field. For more information, see Using Docker Registries.
7. Select your preference for **Docker images disk cleanup scheduling on Diego Cell VMs**. If you choose **Clean up disk space once usage fills disk**, enter a value in MB for **Reserved disk space for other jobs**. This is the amount of space the garbage collection algorithm should keep free for other jobs. For more information about the configuration options and how to configure a reserved amount, see Configuring Diego Cell Disk Cleanup Scheduling.
8. The **Enable containerd delegation** checkbox governs whether or not Garden delegates container create and destroy operations to the containerd tool. By default, this option is

enabled and Garden uses containerd. Deactivate this option by deselecting the checkbox. For more information about the containerd tool, see containerd.

9. Enter a number in the **Max-in-flight container starts** field. This number configures the maximum number of started instances across the Diego Cells in your deployment. Entering `0` sets no limit. For more information, see Set a Maximum Number of Started Containers in *Configuring TAS for VMs for Upgrades*.

10. Under **NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow app developers to bind existing NFS volumes to their apps for shared file access. For more information, see Enabling Volume Services.

    > **Note:** In a fresh install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

11. (Optional) To configure LDAP for NFSv3 volume services:

    - For **LDAP service account user**, enter the username of the service account in LDAP that will manage volume services.
    - For **LDAP service account password**, enter the password for the service account.
    - For **LDAP server host**, enter the hostname or IP address of the LDAP server.
    - For **LDAP server port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
    - For **LDAP user search base**, enter the location in the LDAP directory tree from which any LDAP user search begins. The typical LDAP search base matches your domain name.
      For example, a domain named `cloud.example.com` typically uses the following LDAP user search base: `ou=Users,dc=example,dc=com`.
    - For **LDAP server CA certificate**, you can optionally enter a certificate if your LDAP server supports TLS and you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

    > **Note:** UAA can only parse one certificate entered into this field. If you enter multiple certificates, UAA only uses the first one you entered and ignores the rest. You only need to include one root certificate or self-signed certificate.

12. (Optional) To enable SMB volume services, select the **Enable SMB volume services** checkbox. Enabling SMB volume services allows developers to bind existing SMB shares to their apps. For more information, see Enabling Volume Services.

    > **Note:** If you enable SMB volume services, you must set the **SMB Broker Errand** to **On** in the **Errands** pane.

13. (Optional) Modify the **Default health check timeout**. The value configured for this field is the amount of time allowed between starting up an app and the first healthy response from the app. If the health check does not receive a healthy response within the configured timeout, then the app is declared unhealthy. The default timeout is `60` seconds and the maximum configurable timeout is `600` seconds.

    > ⚠ **Caution:** If you decrease the default health check timeout value below its current value, existing apps with startup times greater than the new value may fail to start up.

14. (Optional) To limit the number of log lines an app instance can generate per second, select **Enable** under **App log rate limit** and enter an integer for **Maximum app log lines per second**. At minimum, VMware recommends using the default limit of `100`. This feature is deactivated by default. For more information, see App Log Rate Limiting.

    > **Note:** In TAS for VMs, this rate is applied globally across all your apps. If only some of your apps require this setting, see App Containers in *Installing Isolation Segment*.

15. Click **Save**.

Configure App Developer Controls

In the **App Developer Controls** pane, you configure restrictions and default settings for your apps.

To configure the **App Developer Controls** pane:

1. Select **App Developer Controls**.
2. Enter the **Maximum file upload size** in MB. This is the maximum size of an app upload, including the buildpack.
3. Enter the **Maximum package size** in MB. This is the maximum total size of an app’s files.
4. Enter the **Default app memory** in MB. This is the default amount of memory allocated to a newly-pushed app if no value is specified with `cf push`.
5. Enter the **Default app memory quota per org** in MB. This is the default memory limit for all apps in an org. The specified limit only applies to the first installation of TAS for VMs. After the initial installation, operators can use the cf CLI to change the default value.
6. For **Maximum disk quota per app** in MB, enter in MB the maximum amount of disk allowed per app.

    > **Note:** If you allow developers to push large apps, TAS for VMs might have trouble placing them on Diego Cells. Additionally, in the event of a system upgrade or an outage that causes a rolling deploy, larger apps might not successfully redeploy if there is insufficient disk capacity. Monitor your deployment to ensure that your Diego Cells have sufficient disk to run your apps.

7. Enter the **Default disk quota per app** in MB. This is the amount of disk allocated by default to a newly-pushed app if no value is specified with `cf push`.

8. Enter the **Default service instance quota per org**. The specified limit only applies to the first installation of TAS for VMs. After the initial installation, operators can use the cf CLI to change the default value.

9. Enter the **Staging timeout** in seconds. When you stage an app droplet with the Cloud Controller, the server times out after the number of seconds you specify in this field.

10. For **Internal domains**, enter one or more domains that apps use for internal DNS service discovery. If you specify a domain using `cf push -d`, other TAS for VMs apps can reach the pushed app at `APP-NAME.INTERNAL-DOMAIN`. This value defaults to `apps.internal`.

11. Enable the **Allow space developers to manage network policies** check box to permit developers to manage their own network policies for their apps.

12. Click **Save**.

Configure App Security Groups

Setting appropriate ASGs is critical for a secure deployment. To acknowledge that you are responsible for setting the appropriate ASGs after the TAS for VMs deployment completes:

1. Select **App Security Groups**.

2. In the **Type “X” to acknowledge this requirement** field, enter `X`.

3. Click **Save**.

For more information about ASGs, see App Security Groups. For more information about setting ASGs, see Restricting App Access to Internal TAS for VMs Components.

Configure Authentication and Enterprise SSO

In the **Authentication and Enterprise SSO** pane, you configure your user store access.

To configure the **Authentication and Enterprise SSO** pane:

1. Select **Authentication and Enterprise SSO**.

2. To authenticate user sign-ons, your deployment can use one of three types of user database: the UAA server’s internal user store, an external SAML identity provider, or an external LDAP server.

    - To use the internal UAA, select **Internal UAA** and follow the procedure in Configuring UAA Password Policy to configure your password policy.

    - To connect to an external identity provider through SAML, select **SAML identity provider** and follow the procedure in Configure TAS for VMs to Use a SAML Identity Provider in *Configuring Authentication and Enterprise SSO for TAS for VMs.*

    - To connect to an external LDAP server, select **LDAP server** and follow the procedure in Configure LDAP as an Identity Provider for TAS for VMs in *Configuring Authentication and Enterprise SSO for TAS for VMs.*

3. Click **Save**.

Configure UAA

In the **UAA** pane, you configure the User Account and Authentication server.

To configure the **UAA** pane:

1. Select **UAA**.
2. Under **UAA database location**, select one of these options:
   - **Tanzu Application Service database:** Use the same database server that other TAS for VMs components use. This system database is configured in the **Databases** pane, and it can be either internal or external.
   - **Other external database:** Use a separate, dedicated database server for UAA.

   ⚠ **Caution:** Protect whichever database you use in your deployment with a password.

   Note: For GCP installations, VMware recommends using an external database on Google Cloud SQL.

3. (Optional) If you selected **Other external database**, complete these fields as follows. Each field includes additional guidance for specific IaaSes and installation methods.
   - For **Hostname**, enter the hostname of the database server.
     - **AWS Terraform:** Enter the value of `rds_address` in your Terraform output.
     - **GCP Terraform:** Enter the value of `sql_db_ip` from your Terraform output.
   - For **TCP port**, enter the port of the database server.
     - **AWS Terraform:** Enter the value of `rds_port` in the Terraform output.
     - **GCP and GCP Terraform:** Enter `3306`.
   - For **Username**, specify a unique username that can access this specific database on the database server.
     - **AWS Terraform:** Enter the value of `rds_username` from your Terraform output.
     - **GCP Terraform:** Enter the value of `tas_sql_username` from your Terraform output.
   - For **Password**, specify a password for the provided username.
     - **AWS Terraform:** Enter the value of `rds_password` from your Terraform output.
     - **GCP Terraform:** Enter the value of `tas_sql_password` from your Terraform output.
   - For **CA certificate**, enter a certificate to use for encrypting traffic to and from the database.

> **Note:** The **CA certificate** field only works if your external database hostname matches a name specified in the certificate. This is not true with GCP CloudSQL.

4. (Optional) Under **JWT issuer URI**, enter the URI that UAA uses as the issuer when generating tokens.

5. Under **SAML service provider credentials**, enter a certificate and private key to be used by UAA as a SAML service provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private key from your trusted Certificate Authority or generate a certificate. The domain `*.login.SYSTEM-DOMAIN` must be associated with the certificate, where `SYSTEM-DOMAIN` is the **System domain** you configured in the **Domains** pane.

   > **Note:** The Ops Manager Single Sign-On Service and Ops Manager Spring Cloud Services tiles require the `*.login.SYSTEM-DOMAIN`.

6. If the private key specified under **SAML service provider credentials** is password-protected, enter the password under **SAML service provider key password**.

7. (Optional) To override the default value, enter a custom SAML Entity ID in the **SAML Entity ID override** field. By default, the SAML Entity ID is `http://login.SYSTEM-DOMAIN`, where `SYSTEM-DOMAIN` is the **System domain** you configured in the **Domains** pane.

8. For **Signature algorithm**, choose an algorithm from the dropdown to use for signed requests and assertions. The default value is **SHA256**.

9. (Optional) In the **Apps Manager access token lifetime**, **Cloud Foundry CLI access token lifetime**, and **Cloud Foundry CLI refresh token lifetime** fields, change the lifetimes of tokens granted for Apps Manager and cf CLI login access and refresh. Most deployments use the defaults.

10. (Optional) In the **Global login session maximum timeout** and **Global login session idle timeout** fields, change the maximum number of seconds before a global login times out. These fields apply to:

    - **Default zone sessions:** Sessions in Apps Manager, Ops Manager Metrics, and other web UIs that use the UAA default zones.

    - **Identity zone sessions:** Sessions in apps that use a UAA identity zone, such as a Ops Manager Single Sign-On service plan.

11. (Optional) Customize the text prompts used for username and password from the cf CLI and Apps Manager login popup by entering values for **Username label** and **Password label**.

12. (Optional) The **Proxy IPs regular expression** field contains a pipe-separated set of regular expressions that UAA considers to be reverse proxy IP addresses. UAA respects the `x-forwarded-for` and `x-forwarded-proto` headers coming from IP addresses that match these regular expressions. To configure UAA to respond properly to Gorouter or HAProxy requests coming from a public IP address, append a regular expression or regular expressions to match the public IP address.

13. (Optional) Deactivate the **Client basic auth compatibility mode** checkbox to require URL encoding for UAA client basic authentication credentials. By default, the checkbox is enabled and URL encoding is not required. This represents the default behavior of UAA prior to UAA v74.0.0. URL encoding is defined by RFC6749. For more information, see RFC6749.

    > **Note:** To require URL encoding for certain UAA clients without deactivating compatibility mode, use the `X-CF-ENCODED-CREDENTIALS=true` HTTP header.

    > **Caution:** If you deactivate the **Client basic auth compatibility mode** checkbox, URL encoding is required for all UAA client apps in your deployment. To avoid breaking changes, ensure that all client apps support URL encoding before you deactivate the checkbox.

14. (Optional) If you are using Single Sign-On for VMware Tanzu Application Service and you want to honor the CORS policy for custom identity zones, clear the **Enforce system zone CORS policy across all identity zones** check box. This check box is enabled by default. If you use Single Sign-On, UAA creates custom identity zones. If you leave this check box selected, UAA ignores the CORS policy for custom identity zones and applies the system default identity zone CORS policy to all zones.

    > **Caution:** If you clear the **Enforce system zone CORS policy across all identity zones** checkbox, apps that are integrated with Single Sign-On might experience downtime because the default CORS policy of the custom identity zones is more restrictive. To prevent downtime, you must explicitly set the CORS policy of the custom identity zones according to the needs of your apps. For more information, see the Managing Service Plan Configurations in the Single Sign-On documentation.

15. Click **Save**.

Configure CredHub

In the **CredHub** pane, you configure the CredHub server.

To configure the **CredHub** pane:

1. Select **CredHub**.
2. Under **CredHub database location**, select the location of your CredHub database. TAS for VMs includes this CredHub database for services to store their service instance credentials.
    - **Tanzu Application Service database**: If you select this option, CredHub uses the same database as other TAS for VMs components, whether the database is internal or external. You configure this database in the **Databases** pane.
    - **Other external database**: If you select this option, configure these fields:

        - **Hostname:** Enter the IP address of your database server.
        - **TCP port:** Enter the port of your database server, such as `3306`.
        - **Username:** Enter a unique username that can access your CredHub database on the database server.
        - **Password:** Enter the password for the provided username.
        - **Database CA certificate:** This certificate is used when encrypting traffic to and from the database.
        - **Disable hostname verification:** Activate this check box to deactivate hostname verification for TLS connections to the targeted database server. You cannot deactivate hostname verification for TLS connections to Postgres databases. You must select this option if you are configuring an external database on GCP or Azure.

> **Note:** If you deploy TAS for VMs on AWS using Terraform, you can locate these values in your Terraform output:
>
> - **Hostname:** `rds_address`
> - **TCP port:** `rds_port`
> - **Username:** `rds_username`
> - **Password:** `rds_password`

3. (Optional) Under **KMS plugin providers**, specify one or more Key Management Service (KMS) providers:
    - **Instance name:** Enter the name of the KMS provider.
    - **Endpoint:** Enter the Unix socket address for the KMS plugin.
    - **Primary:** Enable the **Primary** check box to indicate the primary KMS provider for your environment. You must mark one provider as primary. Do not mark more than one provider as primary.
4. Under **Internal encryption provider keys**, specify one or more keys to use for encrypting and decrypting the values stored in the CredHub database.
    - If you plan to use internal encryption, configure these fields:
        - **Name:** Enter any key name.
        - **Key:** Enter a randomly generated value that is at least 20 characters long. This key is used for encrypting all data.
        - **Primary:** This check box marks the key you specified above as the primary encryption key. You must mark one key as primary. Do not mark more than one key as primary.
    - If you plan to use a hardware security module (HSM) as your encryption provider, configure these fields:
        - **Name:** Enter a key name that already exists on your HSM or a new key name. For each new key name, CredHub generates a key in **HSM provider**

        partition that you configure below.

        - **Key:** Enter a placeholder value under **Key**. CredHub does not use this key for encryption. However, you cannot leave the **Key** field blank.
        - **Primary:** This check box is used for marking the key you specified above as the primary encryption key. You must mark one key as primary. Do not mark more than one key as primary.

> **Note:** For information about using additional keys for key rotation, see Rotating Runtime CredHub Encryption Keys.

5. (Optional) To configure CredHub to use an HSM, configure these fields:
    - **HSM provider partition:** Enter the name of the HSM provider partition.
    - **HSM provider partition password:** Enter a password to use to access the HSM provider partition.
    - **HSM provider client certificate:** Enter the client certificate for the HSM. For more information, see Create and Register HSM Clients in *Preparing CredHub HSMs for Configuration*.
    - Under **HSM provider encryption keys**, specify one or more keys to use for encrypting and decrypting the values stored in the HSM provider. For each key, configure these fields:
        - **Name:** Enter the name of the encryption key.
        - **Primary:** This check box is used for marking the key you specified above as the primary encryption key. You must mark one key as primary. Do not mark more than one key as primary.
    - In the **HSM provider servers** section, click **Add** to add an HSM server. You can add multiple HSM servers. For each HSM server, configure these fields:
        - **Host address:** Enter the hostname or IP address of the HSM server.
        - **Port:** Enter the port of the HSM server. If you do not know your port address, enter `1792`.
        - **Partition serial number:** Enter the serial number of the HSM partition.
        - **HSM certificate:** Enter the certificate for the HSM server. The HSM presents this certificate to CredHub to establish a two-way TLS connection.
6. If your deployment uses any Ops Manager services that support storing service instance credentials in CredHub and you want to enable this feature, enable the **Secure service instance credentials** check box. For more information about using CredHub for securing service instance credentials, see Securing Service Instance Credentials with Runtime CredHub.
7. Click **Save**.
8. Go to the **Resource Config** pane.
9. Under the **Job** column of the **CredHub** row, ensure that the number of instances is set to `2`. This is the minimum instance count required for high availability.

10. To configure UAA so it recognizes a proxy, complete the following fields:
    - **List of proxy servers:** Enter a comma-separated list of router IP addresses for the first group of HTTP/TCP backends.
    - **http proxy:** Enter the `http_proxy` for all http requests across VMs.
    - **https proxy:** Enter the `https_proxy` for all http requests across VMs
    - **no proxy:** Enter a comma-separated list of either domains, IP addresses or DNS suffixes.
11. Click **Save**.

Configure Databases

In the **Databases** pane, you can configure TAS for VMs to use an internal MySQL database provided with Ops Manager, or you can configure an external database provider for the databases required by TAS for VMs.

> **Note:** If you are performing an upgrade, do not modify your existing internal database configuration, or you might lose data. You must migrate your existing data first before changing the configuration. For additional upgrade information, see Upgrading Ops Manager.

Internal Database Configuration

> **Note:** For GCP installations, VMware recommends selecting **External** and using Google Cloud SQL. Only use internal MySQL for non-production or test installations on GCP.

To configure internal databases for your deployment:

1. Under **System databases location**, select **Internal databases - MySQL - Percona XtraDB Cluster**.
2. Click **Save**.

To configure high availability for your internal MySQL databases, see Configure Internal MySQL.

External System Database Configuration

To configure external databases for your deployment:

1. Ensure that you have a database instance with the following databases created. The steps vary depending on your database type. For an example procedure, see Creating Databases for TAS for VMs.
    - `account`
    - `app_usage_service`
    - `autoscale`

   - `ccdb`
   - `credhub`
   - `diego`
   - `locket`
   - `networkpolicyserver`
   - `nfsvolume`
   - `notifications`
   - `routing`
   - `silk`
   - `uaa`
2. In the TAS for VMs tile, select **Databases**.
3. Under **System databases location**, select **External databases**.

   > **Note:** If you configure external databases, you cannot configure an internal database in the **UAA** pane.

4. For **Hostname**, enter the hostname of the database server. If you are installing TAS for VMs using Terraform, this value corresponds to the following variable:
   - **AWS Terraform**: `rds_address`
   - **GCP Terraform**: `sql_db_ip`
5. The **Enable hostname validation** checkbox is selected by default. When **Enable hostname validation** is selected and you enable TLS for your external databases, TAS for VMs verifies the hostname of the external database for communication between TAS for VMs and the external database.

   > **Caution:** If your deployment uses a GCP or Azure external database for TAS for VMs that is TLS-enabled, you must deselect the **Enable hostname validation** checkbox. For more information, see Deactivate Hostname Validation for External Databases on GCP and Azure in *Upgrade Preparation Checklist for Ops Manager v2.12*.

   > **Note:** The **Enable hostname validation** checkbox does not affect communcation between TAS for VMs components and external CredHub databases. To activate or deactivate hostname validation for the CredHub external database, see Configure CredHub.

6. For **TCP port**, enter the port of the database server.
   - If you are using GCP CloudSQL, enter `3306`.

   - If you are installing TAS for VMs on AWS using Terraform, enter the value for `rds_port`.

7. Each component that requires a relational database has two corresponding fields: one for the database username and one for the database password. For each set of fields, specify a unique username that can access this specific database on the database server and a password for the provided username. If you are installing TAS for VMs using Terraform, these values correspond to the following variables:

   - **AWS Terraform:** `rds_username` and `rds_password`
   - **GCP Terraform:** `pas_sql_username` and `pas_sql_password`

   > **Note:** Ensure that the networkpolicyserver database user has the `ALL PRIVILEGES` permission.

   > **Note:** If you are configuring an external database on GCP or Azure and want to use this database for CredHub, ignore the **CredHub database username** and **CredHub database password** fields. Enter the credentials in the **CredHub** configuration pane instead.

8. (Optional) To enable TLS for your external databases, paste your CA certificate in the **Database CA certificate** field.

   > **Note:** TLS is not currently supported for databases that do not include a matching hostname in their server certificate, such as Azure and GCP, unless you deactivate hostname verification in TAS for VMs. You can deactivate hostname verification for external GCP and Azure databases by deselecting the **Enable hostname validation** checkbox described above and selecting the **Disable hostname verification** checkbox in the **CredHub** pane. For more information, see Connection Options for External Applications in the GCP documentation. To configure the **Disable hostname verification** checkbox, see Configure CredHub.

9. Click **Save**.

(Optional) Configure Internal MySQL

In the **Internal MySQL** pane, you configure the internal MySQL clusters for TAS for VMs. Only configure this section if you selected **Internal databases - MySQL - Percona XtraDB Cluster** in the **Databases** pane.

To configure the **Internal MySQL** pane:

1. Select **Internal MySQL**.
2. In the **Replication canary time period** field, specify how frequently the canary checks for replication failure. Leave the default of 30 seconds or modify the value based on the needs

of your deployment. Lower numbers cause the canary to run more frequently, which means that the canary reacts more quickly to replication failure but adds load to the database.

3. In the **Replication canary read delay** field, specify how long the canary waits, in seconds, before verifying that data is replicating across each MySQL node. Leave the default of 20 seconds or modify the value based on the needs of your deployment. Clusters under heavy load can experience a small replication lag as write-sets are committed across the nodes.
4. In the **Email address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.
5. The **Allow command history** checkbox is enabled by default. To prohibit the creation of command line history files on the MySQL nodes, deselect this checkbox.
6. To allow admin and read-only admin users to connect from any remote host, enable the **Allow remote admin access** checkbox. When the checkbox is deselected, admins must `bosh ssh` into each MySQL VM to connect as the MySQL super user.

   > **Note:** Network configuration and ASG restrictions may still limit a client's ability to establish a connection with the databases.

7. In the **Cluster probe timeout** field, enter the maximum amount of time, in seconds, that a new node searches for existing cluster nodes. If left blank, the default value is 10 seconds.
8. In the **Maximum connections** field, enter the maximum number of connections allowed to the database. The default value is `3500`.
9. To log audit events for internal MySQL, select **Enable** under **Server activity logging**.
   - In the **Event types** field, you can enter the events you want the MySQL service to log. By default, this field includes `connect` and `query`, which tracks who connects to the system and what queries are processed. For more information about which events the Percona MySQL server can log, see Audit Log Plugin in the Percona documentation.

   > **Note:** Internal MySQL audit logs are not forwarded to the syslog server because they can contain personally identifying information (PII) and secrets. You can use the `download-logs` script to retrieve the logs, which each MySQL cluster node VM stores in `/var/vcap/store/mysql\_audit\_logs/`. For more information, see Script to download MySQL logs for TAS for VMs or Tile HA Clusters in the VMware Tanzu Knowledge Base.

10. In the **Load balancer healthy threshold** field, enter the amount of time, in seconds, to wait until reporting that the MySQL Proxy instance has started. This allows an external load balancer time to register the instance as healthy. The default value is 0 seconds.
11. In the **Load balancer unhealthy threshold** field, enter the amount of time, in seconds, that the MySQL Proxy continues to accept connections before shutting down. During this period, the health check reports the MySQL Proxy instance as unhealthy to cause load balancers to fail over to other proxies. You must enter a value greater than or equal to the

maximum time it takes your load balancer to consider a proxy instance unhealthy, given repeated failed health checks. The default value is 30 seconds.

12. To enable MySQL Proxy to listen on port 3336, select the **Enable inactive MySQL port** check box. When you run MySQL in high availability mode, this feature allows you to connect to a MySQL node that is not currently serving traffic so that you can run auditing and reporting queries without affecting performance.
13. Click **Save**.

For more information about how to monitor the node health of your MySQL Proxy instances, see Using the MySQL Proxy.

Configure File Storage

In **File Storage**, you determine where you want to place the file storage for your Cloud Controller.

To configure the **File Storage** pane:

1. Select **File Storage**.
2. To set the maximum number of recent valid packages that your app can store, not including the package for the current droplet, enter a value in the **Maximum valid packages per app** field. VMware recommends leaving the default value of `2`. However, you can lower the value if you have strict storage requirements and want to use less disk space.
3. To set the maximum number of recent valid droplets that your app can store, not including the package for the current droplet, enter a value in the **Maximum staged droplets per app** field. VMware recommends leaving the default value of `2`. However, you can lower the value if you have strict storage requirements and want to use less disk space.
4. Under **File storage backup level**, select what you want to back up from your blobstores:
    - **Complete file storage backup** includes the entire blobstore. This includes droplets, packages, and buildpacks.
    - **Exclude droplets from file storage backup** excludes droplets from your backup. If you choose this option, you must re-stage all apps during a restore.
    - **Exclude droplets and packages from file storage backup** excludes droplets and packages from your backup. If you choose this option, you must re-push all apps during a restore.

    For more information about the advantages and disadvantages of excluding droplets and packages, see File Storage Backup Level.
5. For **Cloud Controller filesystem**, see Configuring File Storage for TAS for VMs.

(Optional) Configure System Logging

In the **System Logging** pane, you can configure system logging in TAS for VMs to forward log messages from TAS for VMs component VMs to an external service. VMware recommends forwarding logs to an external service for use in troubleshooting. If you do not fill these fields, platform logs are not forwarded but remain available on the component VMs and for download through Ops Manager.

> **Note:** This procedure explains how to configure system logging for TAS for VMs component VMs. To forward logs from Ops Manager tiles to an external service, you must also configure system logging in each tile. For more information about configuring system logging, see the documentation for the given tiles.

To configure the **System Logging** pane:

1. Select **System Logging**.
2. For **Address**, enter the hostname or IP address of the syslog server.
3. For **Port**, enter the port of the syslog server. The default port for a syslog server is 514.

   > **Note:** The host must be reachable from the TAS for VMs network and accept UDP or TCP connections. Ensure the syslog server listens on external interfaces.

4. For **Transport protocol**, select a transport protocol for log forwarding.
5. For **Encrypt syslog using TLS?**, select **Yes** to use TLS encryption when forwarding logs to a remote server.
   1. For **Permitted peer**, enter either the name or SHA1 fingerprint of the remote peer.
   2. For **Destination certificate**, enter the TLS CA certificate for the remote server.
6. Select the **Enable Cloud Controller security event logging** checkbox to include security events in the log stream. This feature logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).
7. Enable the **Use TCP for file forwarding local transport** checkbox to transmit logs over TCP. This prevents log truncation, but may cause performance issues.
8. The **Do not forward debug logs** checkbox is enabled by default. To forward `DEBUG` syslog messages to an external service, deactivate the checkbox.

   > **Note:** Some TAS for VMs components generate a high volume of `DEBUG` syslog messages. Enabling the **Do not forward debug logs** checkbox prevents TAS for VMs components from forwarding the `DEBUG` syslog messages to external services. However, TAS for VMs still writes the messages to the local disk.

9. For **Custom rsyslog configuration**, enter a custom syslog rule. For more information about adding custom syslog rules, see Customizing Platform Log Forwarding.
10. For **Timestamp format for component logs**, select one of the following:

    > **Breaking Change:** If you change the timestamp format for logs, you might need to update any external monitoring configuration. For more information, see Timestamp Format for Component Logs Replaces Timestamp Format for Diego Logs in *TAS for VMs Breaking Changes*.

   - (Recommended) **Converge to human-readable RFC3339 format:** Every TAS for VMs component that supports RFC3339 timestamps uses this format in their logs. The RFC3339 format uses Coordinated Universal Time (UTC) and provides up to nine points of precision. For more information, see RFC3339. For a current list of components that use the RFC3339 timestamp format, see VMware Tanzu Application Service for VMs v2.10 Release Notes.

     For example:
       - 2019-11-21T22:16:18.750673404Z
       - 2019-11-21T22:16:18.750000000Z
   - **Maintain previous format:** TAS for VMs component logs use their previous timestamp format. VMware only recommends this option if you have scripts that require logs that use the previous timestamp format.

11. Configure how TAS for VMs emits app logs and app metrics for ingestion in your deployment. The options include:
    - Use existing Firehose integrations for app metric and app log ingestion.
    - Preserve existing Firehose integrations for app metrics, but use an alternate method for app log ingestion.
    - Deactivate all Firehose integrations and use alternate methods for both app log and app metric ingestion.

      The following table provides the configuration procedures for each option. For more information about each field, see the Field Descriptions table below.

Option:	Configuration Procedure:
Use existing Firehose app log and metrics integrations	1. Select **Enable V1 Firehose**. 2. Select **Enable V2 Firehose**. 3. Deselect **Enable Log Cache Syslog Ingestion**. 4. Deselect **Disable logs in Firehose**. 5. (Optional) Configure **Aggregate log and metric drain destinations**.
Preserve existing Firehose integrations for app metrics, but use an alternate method for app log ingestion	⚠ **Caution:** Do not use this option if your deployment depends on partner log integrations. 1. Select **Enable V1 Firehose**. 2. Select **Enable V2 Firehose**. 3. Select **Enable Log Cache Syslog Ingestion**. 4. Select **Disable logs in Firehose**. 5. Configure **Aggregate log and metric drain destinations**.

Deactivate all Firehose integrations and use alternate methods for both app log and app metric ingestion	⚠ **Caution:** Do not use this option if your deployment depends on any of these: ◦ Service tile metrics ◦ Healthwatch or App Metrics ◦ Partner log or metric integrations 1. Deselect **Enable V1 Firehose**. 2. Deselect **Enable V2 Firehose**. 3. Select **Enable Log Cache syslog ingestion**. 4. Deselect **Disable logs in Firehose**. 5. Configure **Aggregate log and metric drain destinations**. 6. Deactivate the Smoke Test errand. Otherwise, the TAS for VMs deployment fails. To deactivate errands, see Configure Errands.

Field Descriptions:

The following table provides more details on field values:

Field Name	Description
Enable V1 Firehose	Enabled by default. When enabled, app logs and app metrics flow to the Loggregator V1 Firehose.
Enable V2 Firehose	Enabled by default. When enabled, app logs and app metrics flow to the Loggregator V2 Firehose. If you deselect the checkbox To deactivate the V2 Firehose, you must also deactivate the V1 Firehose.
Enable Log Cache syslog ingestion	Deactivated by default. Configures Log Cache to ingest app logs and app metrics through the syslog server instead of the Reverse Log Proxy. If you deactivate the V1 Firehose, you must activate Log Cache syslog ingestion to receive service tile metrics.
Default loggregator drain metadata	Enabled by default. When enabled, TAS for VMs sends all metadata in app and aggregate syslog drains. Deactivating this option can reduce logging to external databases by up to 50 percent.
Disable logs in Firehose	Deselected by default. Prevents the Firehose from emitting app logs but still allows the Firehose to emit app metrics. Deactivating logs in Firehose helps reduce the load on TAS for VMs by allowing you to scale down Doppler and Traffic Controller VMs.

Aggregate log and metric drain destinations	Aggregate drains forward all app logs on your foundation to the endpoints that you provide in this field. Enter a comma-separated list of syslog endpoints for aggregate log drains. Specify the endpoints in the format: `syslog://HOSTNAME:PORT`. To use TLS for sending logs, specify `syslog-tls://HOSTNAME:PORT` or `https://HOSTNAME:PORT`. In TAS for VMs v2.10, aggregate drains no longer forward metrics by default. You can choose to forward app metrics and TAS for VMs component VM metrics by adding `?include-metrics-deprecated=true` to the endpoints. For example, `syslog://myhost:514?include-metrics-deprecated=true`. **Breaking Change:** If you are upgrading from TAS for VMs v2.9 and want the aggregate drain to continue forwarding app and component VM metrics, you must add `?include-metrics-deprecated=true` to the endpoint. For more information, see Aggregate Syslog Drains Contain Logs Only in the *TAS for VMs Breaking Changes*.

12. To send BOSH system metrics to your logging endpoint more or less frequently, change the value for **System metrics scrape interval**. The default value is `1m`, which sends BOSH system metrics to your logging endpoint once per minute. To send metrics more or less frequently, change the value. For example, enter `2m` to send metrics every two minutes or `10s` to send metrics every ten seconds. The minimum recommended value is five seconds.

13. Click **Save**.

To configure Ops Manager for system logging, see Settings Page in *Using the Ops Manager Interface*.

(Optional) Configure Custom Branding and Apps Manager

In the **Custom Branding** pane, you can customize the appearance of the TAS for VMs login portal and Apps Manager. In the **Apps Manager** pane, you can configure the functionality of Apps Manager, as well as how it displays the pages for your apps in the Marketplace.

To configure the **Custom Branding** and **Apps Manager** panes:

1. Select **Custom Branding**.

2. Configure the fields in the **Custom Branding** pane. For more information about the **Custom Branding** configuration settings, see Custom-Branding Apps Manager.

3. Click **Save**.

4. Select **Apps Manager**.

5. Select the **Enable invitations** checkbox to enable invitations in Apps Manager. Space Managers can invite new users for a given space, Org Managers can invite new users for a given org, and Admins can invite new users across all orgs and spaces. For more information, see Invite New Users in *Managing User Roles with Apps Manager*.

6. Enable the **Display Marketplace service plan prices** checkbox to display the prices for your services plans in the Marketplace.

7. Enter the **Supported currencies as JSON** to appear in the Marketplace. Use the format `{ "CURRENCY-CODE": "SYMBOL" }`. This defaults to `{ "usd": "$", "eur": "€" }`.

8. Enter a **Product name** and **Marketplace name** to configure page names.
9. Enter a **Marketplace URL** to point the **Marketplace** link in Apps Manager to your own marketplace.
10. In **Secondary navigation links**, enter link text and URLs to configure secondary navigation links in the **Apps Manager** and **Marketplace** pages. You may configure up to 10 links in the Apps Manager secondary navigation.
11. (Optional) In **Apps Manager buildpack**, enter the name of the buildpack you want to use to deploy the Apps Manager app. This buildpack must be for static content. If you do not specify a buildpack, TAS for VMs uses the detection process to determine a single buildpack to use. For more information about the detection process, see Buildpack Detection in *How Buildpacks Work*.
12. (Optional) In **Search Server buildpack**, enter the name of the buildpack you want to use to deploy the Search Server app. This must be a node-based buildpack. If you do not specify a buildpack, TAS for VMs uses the detection process to determine a single buildpack to use. For more information about the detection process, see Buildpack Detection in *How Buildpacks Work*.
13. (Optional) In **Invitations buildpack**, enter the name of the buildpack you want to use to deploy the Invitations app. This must be a node-based buildpack. If you do not specify a buildpack, TAS for VMs uses the detection process to determine a single buildpack to use. For more information about the detection process, see Buildpack Detection in *How Buildpacks Work*.
14. The **Apps Manager memory usage** field sets the memory limit with which to deploy the Apps Manager app. Use this field to increase the memory limit if the app fails to start with an `out of memory` error. Enter the value in MB. To use the default value of 128 MB, leave this field blank.
15. The **Search Server memory usage** field sets the memory limit with which to deploy the Search Server app. Use this field to increase the memory limit if the app fails to start with an `out of memory` error. Enter the value in MB. To use the default value of 256 MB, leave this field blank.
16. The **Invitations memory usage** field sets the memory limit with which to deploy the Invitations app. Use this field to increase the memory limit if the app fails to start with an `out of memory` error. Enter the value in MB. To use the default value of 256 MB, leave this field blank.
17. The **Apps Manager polling interval** field provides a temporary fix if Apps Manager usage degrades Cloud Controller response times. In this case, you can use this field to reduce the load on the Cloud Controller and ensure Apps Manager remains available while you troubleshoot the Cloud Controller. VMware recommends that you do not keep this field modified as a long-term fix because it can degrade Apps Manager performance. Optionally, you can:
    - Increase the polling interval above the default of 30 seconds. Enter the value in seconds.

> **Note:** If you enter a value between `0` and `30`, the field is automatically set to `30`.

  - Deactivate polling by entering `0`. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.

18. The **App details polling interval** field provides an additional way to reduce the load on the Cloud Controller when the **Apps Manager polling interval** field is not sufficient. This field controls the rate at which Apps Manager polls for data when a user views the **Overview** page of an app. VMware recommends that you do not keep this field modified as a long-term fix because it can degrade Apps Manager performance. Optionally, you can:
    - Increase the polling interval above the default of 10 seconds. Enter the value in seconds.

      > **Note:** If you enter a value between `0` and `30`, the field is automatically set to `10`.

    - Deactivate polling by entering `0`. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.
19. To enable multi-foundation support for Apps Manager, enter a JSON object containing all the foundations you want Apps Manager to manage in **Multi-foundation configuration (beta)**. Enabling multi-foundation support allows you to manage orgs, spaces, apps, and service instances from multiple TAS for VMs foundations from a single Apps Manager interface. For more information, see Configuring Multi-Foundation Support in Apps Manager.
20. For **Redirect URIs**, enter a comma-separated list of the URI for each additional foundation on which you enabled multi-foundation support.
21. Click **Save**.

(Optional) Configure Email Notifications

In the **Email Notifications** pane, you can enable end-user self-registration. TAS for VMs uses SMTP to send invitations and confirmations to Apps Manager users. If you do not need this service, leave this pane blank and deactivate the **Notifications** and **Notifications UI** errands in the **Errands** pane.

To configure the **Email Notifications** pane:

1. Select **Email Notifications**.
2. For **From email**, enter the email address from which email notifications are sent.
3. For **SMTP server address**, enter the SMTP address of the server that sends emails.
4. For **SMTP server port**, enter the port of the SMTP server that sends email notifications.

> **Note:** For GCP, you must use port 2525. Ports 25 and 587 are not allowed on GCP Compute Engine.

5. For **SMTP server credentials**, enter the username and password for the SMTP server that sends email notifications.
6. Select the **SMTP enable automatic STARTTLS** checkbox if you want your SMTP server to automatically create a secure TLS connection when sending emails.
7. Verify your authentication requirements with your email admin and use the **SMTP authentication mechanism** dropdown to select **None**, **Plain**, or **CRAMMD5**. If you have no SMTP authentication requirements, select **None**.
8. If you selected **CRAMMD5** as your authentication mechanism, enter a secret in the **SMTP CRAMMD5 secret** field.
9. Click **Save**.

> **Note:** If you do not configure the SMTP settings using this form, the admin must create orgs and users using the cf CLI. For more information, see Creating and Managing Users with the cf CLI.

(Optional) Configure App Autoscaler

In the **App Autoscaler** pane, you configure the App Autoscaler service. To use App Autoscaler, you must create an instance of the service and bind it to an app. To create an instance of App Autoscaler and bind it to an app, see Set Up App Autoscaler in *Scaling an App Using App Autoscaler*.

To configure the **App Autoscaler** pane:

1. Select **App Autoscaler**.
2. For **Autoscaler instance count**, enter the number of instances of the App Autoscaler service you want to deploy. The default value is `3`. For high availability, set this number to `3` or higher. Larger environments might require more instances than the default number. We recommend one App Autoscaler instance for every 10 apps using App Autoscaler.
3. For **Autoscaler API instance count**, enter the number of instances of the App Autoscaler API you want to deploy. The default value is `1`. Larger environments may require more instances than the default number.
4. For **Metric collection interval**, enter how many seconds of data collection you want App Autoscaler to evaluate when making scaling decisions. The minimum interval is 60 seconds, and the maximum interval is 3600 seconds. The default value is `120`. Increase this number if the metrics you use in your scaling rules are emitted less frequently than the existing metric collection interval.
5. For **Scaling interval**, enter in seconds how frequently App Autoscaler evaluates an app for scaling. The minimum interval is 15 seconds, and the maximum interval is 120 seconds. The default value is `35`.

6. To enable verbose logging for App Autoscaler, select the **Enable verbose logging** checkbox. Verbose logging is deactivated by default. Activate this checkbox to see more detailed logs. Verbose logs show specific reasons why App Autoscaler scaled the app, including information about minimum and maximum instance limits, App Autoscaler's status. For more information about App Autoscaler logs, see App Autoscaler Events and Notifications in *Scaling an App Using App Autoscaler*.
7. If you do not want the Autoscaler API to reuse HTTP connections, select the **Disable API connection pooling** checkbox. This may be necessary if your front end idle timeout for the Gorouter is set to a low value, such as 1 second. For more information, see Configuring Front End Idle Timeout for Gorouter and HAProxy.
8. To enable email notifications of Autoscaler events, select the **Enable email notifications** checkbox. For more information about managing email notifications, see App Autoscaler Events and Notifications in *Scaling an App Using App Autoscaler*.
9. Click **Save**.

Configure Cloud Controller

In the **Cloud Controller** pane, you configure the Cloud Controller.

To configure the **Cloud Controller** pane:

1. Select **Cloud Controller**.
2. Enter your **Cloud Controller database encryption key** if all of the following are true:
   - You deployed TAS for VMs previously.
   - You then stopped TAS for VMs or it crashed.
   - You are re-deploying TAS for VMs with a backup of your Cloud Controller database.

     For more information, see Backing Up and Restoring Ops Manager.
3. **Cloud Foundry API rate limiting** prevents API consumers from overwhelming the platform API servers. Limits are imposed on a per-user or per-client basis and reset on an hourly interval.

   To deactivate Cloud Foundry API rate limiting, select **Disable** under **Cloud Foundry API rate limiting**. To enable Cloud Foundry API rate limiting:

   1. Under **Cloud Foundry API rate limiting**, select **Enable**.
   2. For **General limit**, enter the number of requests a user or client is allowed to make over an hour-long interval for all endpoints that do not have a custom limit. The default value is `2000`.
   3. For **Unauthenticated limit**, enter the number of requests an unauthenticated client is allowed to make over an hour-long interval. The default value is `100`.
4. (Optional) Enter in seconds your **Database connection validation timeout**. By default, the setting is `3600` seconds or 60 minutes. You can enter `-1` to cause Cloud Controller to make an additional query to the database whenever connections are checked out from the pool. Choosing `-1` has performance implications.

5. (Optional) Enter in seconds your **Database read timeout**. By default, the setting is `3600` seconds or 60 minutes.
6. (Optional) Enter in days the **Age of audit events pruned from Cloud Controller database**.
7. (Optional) Enter in days the **Age of completed tasks pruned from Cloud Controller database**.
8. (Optional) Rotate the Cloud Controller database (CCDB) encryption key using the **Encryption key ledger** field. For more information, see Rotating the Cloud Controller Database Encryption Key.
9. Click **Save**.

Configure Smoke Tests

In the **Smoke Tests** pane, you configure the org and space where smoke tests are run. In the org and space you configure in the **Smoke Tests** pane, the Smoke Tests errand pushes an app to a Ops Manager org to run basic functionality tests against the TAS for VMs tile after an installation or update. In the **Errands** pane, you can choose whether or not to run the Smoke Tests errand.

To configure the **Smoke Tests** pane:

1. Select **Smoke Tests**.
2. If you have a shared apps domain, select **A temporary space within the system org**, which creates a temporary space within the `system` org for running smoke tests and deletes the space afterwards. Otherwise, select **A specified org and space** and complete the fields to configure where TAS for VMs pushes an app to run smoke tests:
    - For **Org**, enter the org TAS for VMs should use when pushing an app to run smoke tests.
    - For **Space**, enter the space TAS for VMs should use when pushing an app to run smoke tests.
3. Click **Save**.

(Optional) Configure Advanced Features

The **Advanced Features** pane includes new functionality that may have certain constraints. Although these features are fully supported, VMware recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

> **Note:** Due to the risk of app failure and the deployment-specific nature of disk and memory use, VMware has no recommendation for how much, if any, memory or disk space to overcommit.

To enable overcommit:

1. Select **Advanced Features**.
2. Enter in MB the total desired amount of Diego Cell memory in the **Diego Cell memory capacity** field. See the **Diego Cell** row in the **Resource Config** tab for the current Diego Cell memory capacity settings that this field overrides.
3. Enter in MB the total desired amount of Diego Cell disk capacity in the **Diego Cell disk capacity** field. See the **Diego Cell** row in the **Resource Config** tab for the current Diego Cell disk capacity settings that this field overrides.
4. Click **Save**.

> **Note:** Entries made to each of these two fields set the total amount of resources allocated, not the overage.

App Graceful Shutdown Period

If your apps require a longer period of time to finish in-flight jobs and gracefully shut down, you can increase the graceful shutdown period. By default, this graceful shutdown period is set to 10 seconds.

When TAS for VMs requests a shutdown of an app, the processes in the container have a period of time to gracefully shut down before the processes are forcefully terminated. For more information, see Shutdown in *App Container Lifecycle*.

If you significantly increase the value of the graceful shutdown period, platform upgrades and updates might become slower. This is because each Diego Cell uses the graceful shutdown period when it is cleaning up evacuated app instances and waits for each app to gracefully shut down.

VMware recommends using isolation segments to separate apps that have different shutdown requirements to ensure Diego Cell update times are reliable. For more information, see Installing Isolation Segment.

> **Note:** You must ensure that **App graceful shutdown period** has the same value in all environments that have deployed apps. This is to avoid unexpected behavior.

To increase the app graceful shutdown period:

1. Select **Advanced Features**.
2. Enter an integer in the **App graceful shutdown period** field. This value is the period of time in seconds the platform should wait for an app instance to exit after it is signaled to gracefully shut down. The default and minimum value is `10`.
3. Click **Save**.

Non-RFC-1918 Private Network Allow List

Some private networks require extra configuration so that internal file storage (WebDAV) can communicate with other TAS for VMs processes.

The **Non-RFC-1918 private network allow list** field is provided for deployments that use a non-RFC 1918 private network. This is typically a private network other than `10.0.0.0/8`, `172.16.0.0/12`, or `192.168.0.0/16`.

Most TAS for VMs deployments do not require any modifications to this field.

To add your private network to the allow list:

1. Select **Advanced Features**.
2. Append a new `allow` rule to the existing contents of the **Non-RFC-1918 private network allow list** field. Include the word `allow`, the network CIDR range to allow, and a semi-colon (`;`) at the end. For example:

   ```
   allow 172.99.0.0/24;
   ```

3. Click **Save**.

CF CLI Connection Timeout

The **CF CLI connection timeout** field allows you to override the default five-second timeout of the Cloud Foundry Command Line Interface (cf CLI) used within your TAS for VMs deployment. This timeout affects the cf CLI command used to push TAS for VMs errand apps such as Notifications, Autoscaler, and Apps Manager.

Set the value of this field to a higher value, in seconds, if you are experiencing domain name resolution timeouts when pushing errands in TAS for VMs.

To modify the value of the CF CLI connection timeout:

1. Select **Advanced Features**.
2. Enter a value, in seconds, to the **CF CLI connection timeout** field.
3. Click **Save**.

(Optional) Enable TLS for Internal System Database

You can enable TLS for clients of the internal system database. This feature is in beta, and the checkbox is deselected by default. For more information about the internal system database, see Managing Internal Databases.

To enable TLS for clients of the internal system database:

1. Select **Advanced Features**.
2. Select the **Enable TLS for internal system database (beta)** checkbox.
3. Click **Save**.

Database Connection Limits

You can configure the maximum number of concurrent database connections that Diego and container networking components can have.

To configure the maximum number of concurrent database connections:

1. Select **Advanced Features**.
2. Enter a value in each field beginning with **Maximum number of open connections...** for a given component. The placeholder values for each field are the default values.
3. Click **Save**.

When there are not enough connections available, such as during a time of heavy load, components may experience degraded performance and sometimes failure. To resolve or prevent this, you can increase and fine-tune database connection limits for the component.

> ⚠ **Caution:** Decreasing the value of this field for a component may affect the amount of time it takes for it to respond to requests.

(Optional) Rolling App Deployments

You can deactivate rolling app deployments. For more information, see Rolling App Deployments.

To deactivate rolling app deployments:

1. Select **Advanced Features**.
2. Select the **Disable rolling app deployments** checkbox.
3. Click **Save**.

Maximum Number of Envelopes Stored in Log Cache Per Source

By default, Log Cache keeps 100,000 envelopes per source. An envelope wraps an event and adds metadata. For sources that produce more than 100,000 envelopes, this default may not provide a long enough duration for you to specify a time period for a historical query.

To set the maximum number of envelopes stored per source above the default:

1. Select **Advanced Features**.
2. Enter the **Maximum number of envelopes stored in Log Cache per source**.
3. Click **Save**.

Usage Service Data Retention Period

By default, Usage Service deletes granular data after 365 days.

To configure this retention period:

1. Select **Advanced Features**.
2. Enter the number of days of granular data you want to retain in the **Usage service cutoff age** field.
3. Click **Save**.

To avoid performance or data migration issues, VMware recommends that you do not retain data for longer than 365 days. Configuring this field does not affect monthly summary records.

For more information, see Usage Data Retention in *Reporting App, Task, and Service Instance Usage*.

(Optional) Configure Metric Registrar

> **Note:** The Metric Registrar is bundled with TAS for VMs, and you configure it in the TAS for VMs tile. You do not install and configure Metric Registrar as a separate product tile.

In the Metric Registrar pane, you configure the Metric Registrar. Metric Registrar allows the conversion of structured logs into metrics. It also scrapes metrics endpoints and forwards the metrics to Loggregator. If enabled, VMware recommends also enabling the Metric Registrar Smoke Test errand.

Deactivate the Metric Registrar

The Metric Registrar is enabled by default.

To deactivate the Metric Registrar:

1. Select **Metric Registrar**.
2. Clear the **Enable Metric Registrar** checkbox.
3. Click **Save**.

Edit Default Scraping Interval

The scraping interval defines how often the Metric Registrar polls custom metric endpoints. The default is 35 seconds.

To edit the Metric Registrar scraping interval:

1. Select **Metric Registrar**.
2. Edit the **Endpoint scraping interval** field.
3. Click **Save**.

Add Blocked Tags

To prevent the Metric Registrar from consuming the value of a metric or event tag, you can add the tag to the **Blocked tags** field. For example, if you tag your metrics with a `customer_id`, you may want to add `customer_id` to the list of blocked tags.

By default, the following tags are blocked to prevent interference with other products like App Metrics that use and rely on such tags.

- `deployment`
- `job`
- `index`

- `id`

To prevent the Metric Registrar from consuming the value of a metric or event tag:

1. Select **Metric Registrar**.
2. Add the desired tags to the **Blocked tags** field in a comma-separated list.
3. Click **Save**.

Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of TAS for VMs. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

By default, Ops Manager always runs all errands.

In the **Errands** pane, you can change these run rules. For each errand, you can select **On** to run it always or **Off** to never run it.

For more information about how Ops Manager manages errands, see Managing Errands in Ops Manager.

> **Note:** Several errands, such as App Autoscaler and Notifications, deploy apps that provide services for your deployment. When one of these apps is running, selecting **Off** for the corresponding errand on a subsequent installation does not stop the app.

- **Smoke Test Errand** verifies that your deployment can:
  - Push, scale, and delete apps
  - Create and delete orgs and spaces

  > **Caution:** If both the V2 and V1 Firehoses are deactivated, then deactivate the Smoke Test Errand. Otherwise, the deploy fails.

- **Usage Service Errand** deploys the Usage Service app, which Apps Manager depends on.
- **Apps Manager Errand** deploys Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the cf CLI. After Apps Manager has been deployed, VMware recommends setting this errand to **Off** for subsequent TAS for VMs deployments. For more information about Apps Manager, see Getting Started with Apps Manager.
- **Notifications Errand** deploys an API for sending email notifications to your TAS for VMs platform users.

  > **Note:** The Notifications app requires that you configure SMTP with a username and password, even if you set the value of **SMTP Authentication**

> **Mechanism** to `none`. To configure SMTP with a username and password, see Configure Email Notifications.

- **Notifications UI Errand** deploys a dashboard for users to manage notification subscriptions.
- **App Autoscaler Errand** pushes the App Autoscaler app, which enables you to configure your apps to automatically scale in response to changes in their usage load. For more information, see Scaling an App Using App Autoscaler.
- **App Autoscaler Smoke Test Errand** runs smoke tests against App Autoscaler.
- **NFS Broker Errand** pushes the NFS Broker app, which supports NFS Volume Services for TAS for VMs. For more information, see Enable NFS Volume Services in *Enabling Volume Services*.
- The **Metric Registrar Smoke Test Errand** verifies that the Metric Registrar can access custom metrics that an app emits and convert them into Loggregator metrics.

  > **Note:** The **Metric Registrar Smoke Test** errand runs only if the Metric Registrar is deployed. For more information about configuring the Metric Registrar, see Metric Registrar and Custom App Metrics.

- **SMB Broker Application Errand** pushes the SMB Broker app, which supports SMB Volume Services for TAS for VMs. For more information, see Enable SMB Volume Services in *Enabling Volume Services*.
- **Rotate CC Database Key** translates sensitive data in the Cloud Controller database to the currently specified **Primary** key. You specify this key in the **Encryption key ledger** field of the **Cloud Controller** pane.

Configure Resources

In the **Resource Config** pane, you must associate load balancers with the VMs in your deployment to enable traffic. For more information, see Configure Load Balancing for TAS for VMs.

(Optional) Scale Down and Deactivate Resources

> **Note:** The **Resource Config** pane has fewer VMs if you are installing Small Footprint TAS for VMs. For more information, see Getting Started with Small Footprint TAS for VMs.

> **Note:** Small Footprint TAS for VMs does not default to a highly available configuration. It defaults to the minimum configuration. To make Small Footprint TAS for VMs highly available, scale the **Compute**, **Router**, and **Database** VMs to `3` instances and scale the **Control** VM to `2` instances.

TAS for VMs defaults to a highly available resource configuration. However, you may need to perform additional procedures to make your deployment highly available. For more information, see High Availability in TAS for VMs and Scaling TAS for VMs.

If you do not want a highly available resource configuration, you must scale down your instances manually by navigating to the **Resource Config** section and using the dropdowns under **Instances** for each job.

By default, TAS for VMs also uses an internal filestore and internal databases. If you configure TAS for VMs to use external resources, you can deactivate the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

To deactivate specific VMs in Ops Manager:

1. Select **Resource Config**.
2. If you configured TAS for VMs to use an external S3-compatible filestore, enter `0` in **Instances** in the **File Storage** field.
3. If you selected **External** when configuring the UAA, System, and CredHub databases, edit these fields:
    - **MySQL Proxy**: Enter `0` in **Instances**.
    - **MySQL Server**: Enter `0` in **Instances**.
    - **MySQL Monitor**: Enter `0` in **Instances**.
4. If you deactivated TCP routing, enter `0` **Instances** in the **TCP Router** field.
5. If you are not using HAProxy, enter `0` **Instances** in the **HAProxy** field.
6. Click **Save**.

Download Stemcell

This step is only required if your Ops Manager deployment does not already have the stemcell version TAS for VMs requires. For more information about importing stemcells, see Importing and Managing Stemcells.

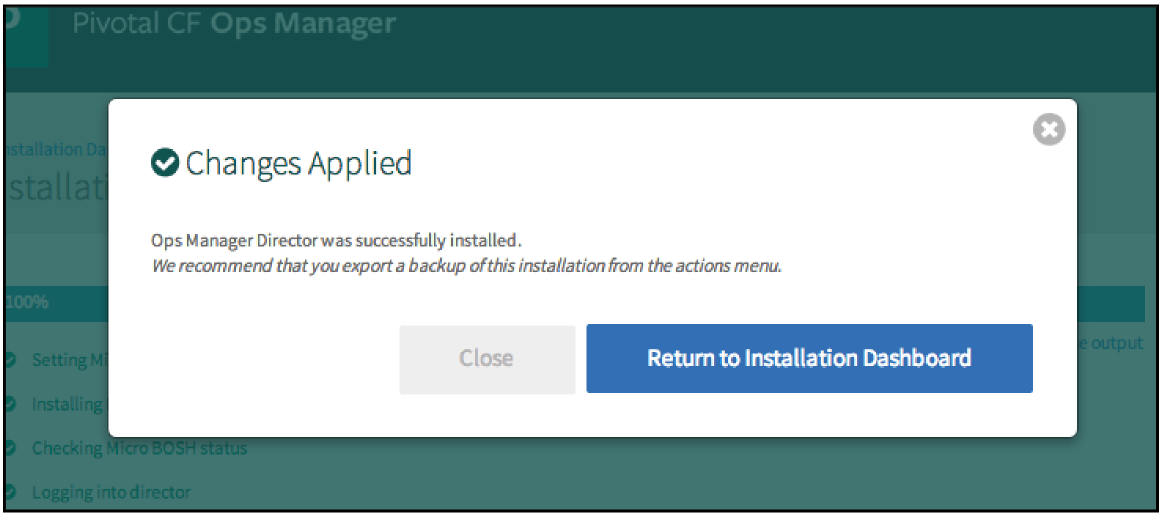
To download the stemcell version TAS for VMs requires:

1. Go to the Stemcell product page on VMware Tanzu Network. You may have to log in.
2. Download the appropriate stemcell version targeted for your IaaS.
3. In the Ops Manager Installation Dashboard, navigate to **Stemcell Library**.
4. Click **Import Stemcell** to import the downloaded stemcell `.tgz` file.
5. When prompted, enable the Ops Manager product checkbox to stage your stemcell.
6. Click **Apply Stemcell to Products**.

Complete the TAS for VMs Installation

To complete your installation of TAS for VMs:

1. Click the **Installation Dashboard** link to return to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**, then **Apply Changes**. The install process generally requires a minimum of 90 minutes to complete. The following image shows the Changes Applied window that displays when the installation process successfully completes.



Ops Manager Director was successfully installed.
We recommend that you export a backup of this installation from the actions menu.

Quick Start TAS for VMs Configuration

This topic describes how to minimally configure VMware Tanzu Application Service for VMs (TAS for VMs) for evaluation or testing purposes. It does not include optional configurations such as external databases or external file storage.

For production deployments, VMware recommends following the procedure in Configuring TAS for VMs.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed the steps to prepare your environment for Ops Manager and install and configure the BOSH Director.

Add TAS for VMs to VMware Tanzu Operations Manager

To add TAS for VMs to Ops Manager:

1. If you have not already downloaded TAS for VMs, log in to VMware Tanzu Network and click **VMware Tanzu Application Service for VMs**.
2. From the **Releases** drop-down menu, select the release to install and choose one of the following:
    1. Click **VMware Tanzu Application Service for VMs** to download the TAS for VMs `.pivotal` file.
    2. Click **Small Footprint TAS for VMs** to download the Small Footprint TAS for VMs `.pivotal` file. For more information, see Getting Started with Small Footprint TAS for VMs.
3. Go to the Ops Manager Installation Dashboard.
4. Click **Import a Product** to add your tile to Ops Manager. For more information, see Adding and Deleting Products.
5. Click the TAS for VMs tile.

Configure TAS for VMs

To install TAS for VMs with minimal configuration:

1. Follow the procedure in Assign AZs and Networks in *Configuring TAS for VMs*.
2. Follow the procedure in Configure Domains in *Configuring TAS for VMs*.
3. Select **Networking**.
4. Under **Certificates and private keys for the Gorouter and HAProxy**, you must provide at least one certificate and private key name and certificate key pair for the Gorouter and HAProxy. The Gorouter and HAProxyu are enabled to receive TLS communication by default. You can configure multiple certificates for the Gorouter and HAProxy.

   > **Note:** When providing custom certificates, enter them in this order: `wildcard`, `Intermediate`, `CA`. For more information, see Creating a .pem File for SSL Certificate Installations in the DigiCert documentation.

   1. Click **Add** to add a name for the certificate chain and its private key pair. This certificate is the default used by the Gorouter. You can either provide a certificate signed by a Certificate Authority (CA) or click **Generate RSA Certificate** to generate a self-signed certificate in Ops Manager.

   > **Note:** If you configured Ops Manager Front End without a certificate, you can use this new certificate to complete your Ops Manager configuration. To configure your Ops Manager Front End certificate, see Configure Front End in *Preparing to Deploy Ops Manager on GCP*.

   > **Note:** Ensure that you add any certificates that you generate in this pane to your infrastructure load balancer.

5. Deactivate the **HAProxy forwards all requests to the Gorouter over TLS** check box. By default, TAS for VMs does not deploy HAProxy.
6. Setting appropriate ASGs is critical for a secure deployment. To acknowledge that you are responsible for setting the appropriate ASGs after the TAS for VMs deployment completes:
   1. Select **App Security Groups**.
   2. In the **Type “X” to acknowledge this requirement** field, enter `X`.
   3. Click **Save**.

      For more information about ASGs, see App Security Groups. For more information about setting ASGs, see Restricting App Access to Internal TAS for VMs Components.
7. Under **SAML service provider credentials**, enter a certificate and private key for the User Account and Authentication (UAA) server to use as a SAML service provider for signing outgoing SAML authentication requests. You can provide an existing certificate and private

key from your trusted CA or generate a self-signed certificate. The domain `*.login.SYSTEM-DOMAIN` must be associated with the certificate, where `SYSTEM-DOMAIN` is the system domain you configured in the **Domains** pane.

> **Note:** The Single Sign-On for VMware Tanzu service and Spring Cloud Services for VMware Tanzu tiles require the `*.login.SYSTEM-DOMAIN`.

8. Select **UAA**.
9. If the private key specified under **SAML service provider credentials** is password-protected, enter the password under **SAML service provider key password**.
10. Select **CredHub**.
11. Under **Internal encryption provider keys**, specify one or more keys to use for encrypting and decrypting the values stored in the CredHub database:
    - **Name:** This is the name of the encryption key.
    - **Key:** This key is used for encrypting all data. The key must be at least 20 characters long.
    - **Primary:** This check box is used for marking the key you specified above as the primary encryption key. You must mark one key as **Primary**. Do not mark more than one key as **Primary**.
12. Go to the **Internal MySQL** pane.
13. In the **Email address** field, enter the email address where the MySQL service sends alerts when the cluster experiences a replication issue or when a node is not allowed to auto-rejoin the cluster.
14. Select **Resource Config**.
15. In the **Resource Config** pane, you must associate load balancers with the VMs in your deployment to enable traffic. For more information, see Configure Load Balancing for TAS for VMs.

Complete the TAS for VMs Installation

To complete the TAS for VMs installation:

1. Click the **Installation Dashboard** link to return to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**, then **Apply Changes**.

TAS for VMs on vSphere Requirements

This topic tells you about the requirements for installing VMware Tanzu Application Service for VMs (TAS for VMs) on vSphere.

For general resource requirements, see TAS for VMs Resource Requirements.

vSphere Requirements

The following are requirements for installing TAS for VMs that are specific to vSphere.

Minimum Resource Requirements for Ops Manager Deployment with TAS for VMs

The following are the minimum resource requirements for maintaining a Ops Manager deployment with TAS for VMs on vSphere:

- vSphere v7.0, v6.7, or v6.5
- Disk space: 2 TB recommended
- Memory: 120 GB
- Two public IP addresses: One for TAS for VMs and one for Ops Manager
- vCPU cores: 82
- Overall CPU: 28 GHz
- vSphere editions: Enterprise Plus or Platinum. These editions include Distributed Resource Scheduler (DRS), which is required by TAS for VMs.
- Ops Manager must have HTTPS access to vCenter and ESX hosts on TCP port 443.
- A configured vSphere cluster:
    - If you activate vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to **Partially automated** or **Fully automated**. If you set the Automation level to **Manual**, the BOSH automated installation will fail with a `power_on_vm` error when BOSH attempts to create virtual machines (VMs).
    - Deactivate hardware virtualization if your vSphere hosts do not support VT-X/EPT. If you are unsure whether the VM hosts support VT-x/EPT, you must deactivate this setting. If you leave this setting enabled and the VM hosts do not support VT-x/EPT, then each VM requires manual intervention in vCenter to continue powering on without the Intel virtualized VT-x/EPT. Refer to the vCenter help topic at Configuring Virtual Machines > Setting Virtual Processors and Memory > Set Advanced Processor Options for more information.
- If you configure an external load balancer, an HTTP keep-alive connection timeout greater than five seconds ote">**Note:** If you are deploying Ops Manager behind a firewall, see <a href="https://docs.vmware.com/en/VMware-Tanzu-Operations-Manager/2.10/vmware-tanzu-ops-manager/install-config_firewall.html

Preparing Your Firewall for Deploying Ops Manager.

Minimum Resource Requirements for Ops Manager Deployment with Small Footprint TAS for VMs

The following are the minimum resource requirements for maintaining a Ops Manager deployment with Small Footprint TAS for VMs on vSphere:

- 1 vSphere cluster/1 AZ
- 2 resource pools: 1 for NSX-T components and 1 for TAS for VMs or TKGI
- 3 hosts minimum for vSphere HA (4 hosts for vSphere VSAN)

- Shared storage (Such as iSCSI, NFS, and VSAN)
- 1 NSX Manager
- 3 NSX controllers
- 4 large edge VMs in a cluster
- Ops Manager: Ops Manager, the BOSH Director, and Small Footprint TAS for VMs

Instance Number and Scaling Requirements

By default, TAS for VMs deploys the number of VM instances required to run a highly available configuration of Ops Manager. If you are deploying a test or sandbox Ops Manager that does not require HA, then you can scale down the number of instances in your deployment.

For information about the number of instances required to run a minimal, non-HA Ops Manager deployment, see Scaling TAS for VMs.

vSphere Virtual Disk Types

When you create a virtual machine in VMware vSphere, vSphere creates a new virtual hard drive for that virtual machine. The virtual hard drive is contained in a virtual machine disk (VMDK). The disk format you choose for the new virtual hard drive can have a significant impact on performance.

You can choose one of three formats when creating a virtual hard drive:

- Thin Provisioned
- Thick Provisioned Lazy Zeroed
- Thick Provisioned Eager Zeroed

Thin Provisioned

Advantages:

- Fastest to provision
- Allows disk space to be over-committed to VMs

Disadvantages:

- Slowest performance due to metadata allocation overhead and additional overhead during initial write operations
- Over-commitment of storage can lead to application disruption or downtime if resources are actually used
- Does not support clustering features

When vSphere creates a thin provisioned disk, it only writes a small amount of metadata to the datastore. It does not allocate or zero out any disk space. At write time, vSphere first updates the allocation metadata for the VMDK, then zeros out the block or blocks, then finally writes the data. Because of this overhead, thin provisioned VMDKs have the lowest performance of the three disk formats.

Thin provisioning allows you to overcommit disk spaces to VMs on a datastore. For example, you could put 10 VMs, each with a 50 GB VMDK attached to it, on a single 100 GB datastore, as long as the sum total of all data written by the VMs never exceeded 100 GB. Thin provisioning allows administrators to use space on datastores that would otherwise be unavailable if using thick provisioning, possibly reducing costs and administrative overhead.

Thick Provisioned Lazy Zeroed

Advantages:

- Faster to provision than Thick Provisioned Eager Zeroed
- Better performance than Thin Provisioned

Disadvantages:

- Slightly slower to provision than Thin Provisioned
- Slower performance than Thick Provisioned Eager Zero
- Does not support clustering features

When vSphere creates a thick provisioned lazy zeroed disk, it allocates the maximum size of the disk to the VMDK, but does nothing else. At the initial access to each block, vSphere first zeros out the block, then writes the data. Performance of a thick provisioned lazy zeroed disk is not as good a thick provisioned eager zero disk because of this added overhead.

Thick Provisioned Eager Zeroed

Advantages:

- Best performance
- Overwriting allocated disk space with zeros reduces possible security risks
- Supports clustering features such as Microsoft Cluster Server (MSCS) and VMware Fault Tolerance

Disadvantages:

- Longest time to provision

When vSphere creates a thick provisioned eager zeroed disk, it allocates the maximum size of the disk to the VMDK, then zeros out all of that space.

Example: If you create an 80 GB thick provisioned eager zeroed VMDK, vSphere allocates 80 GB and writes 80 GB of zeros.

By overwriting all data in the allocated space with zeros, thick provisioned eager zeroed eliminates the possibility of reading any residual data from the disk, thereby reducing possible security risks.

Thick provisioned eager zeroed VMDKs have the best performance. When a write operation occurs to a thick provisioned eager zeroed disk, vSphere writes to the disk, with none of the additional overhead required by thin provisioned or thick provisioned lazy zeroed formats.

Deploying TAS for VMs with NSX-T Networking

This topic describes how to install VMware Tanzu Application Service for VMs (TAS for VMs) on vSphere with NSX-T internal networking, using the VMware NSX-T Container Plug-in for Ops Manager.

> **Note:** These instructions have been updated to use the NSX-T Policy API, the next-generation interface for integrating with the NSX-T networking and security framework.

Overview

TAS for VMs uses a Container Network Interface (CNI) plugin to support secure and direct internal communication between containers. This plugin can be:

- The internal Silk plugin that comes packaged with TAS for VMs, or
- On vSphere, NSX-T Container Plug-in (NCP), which installs as the VMware NSX-T Container Plug-in for Ops Manager tile in Ops Manager.
- On vSphere, the NSX-T Container Plug-in (NCP), which is installed as the VMware NSX-T Container Plug-in for Ops Manager tile in Ops Manager.

Prerequisites

Before deploying TAS for VMs with NSX-T networking, you must have the following:

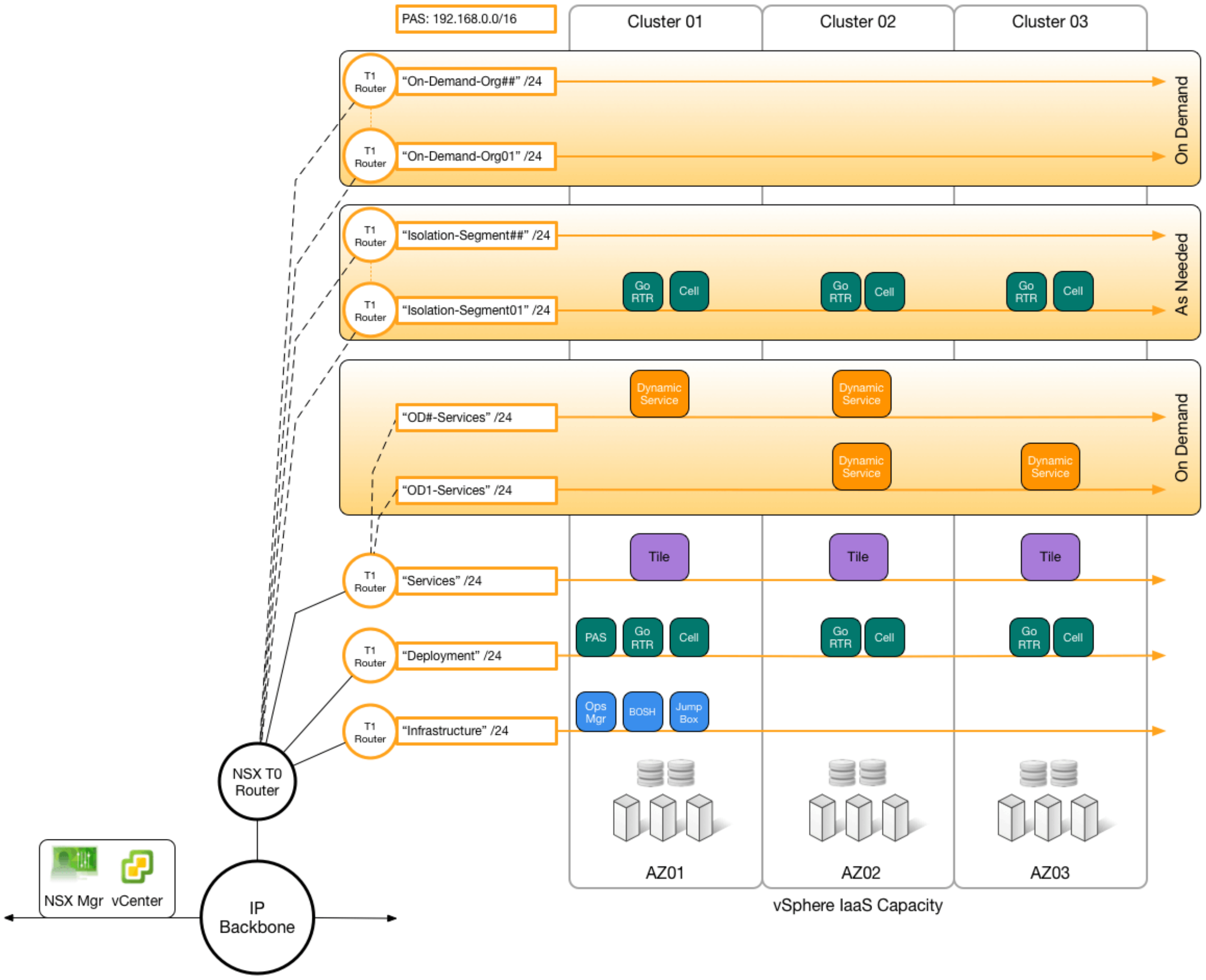
- An NSX-T environment with NSX-T components installed and configured, including a Tier-0 gateway (configured in Active/Passive HA mode) attached to a North-South Transport Zone and a NSX-T edge cluster. The NSX-T version must support the versions of NCP and TAS for VMs you intend to use. Verify the compatibility between NSX-T, NCP and TAS for VMs with the following documentation:
    - Product Interoperability Matrix: TAS for VMs, VMware NSX-T, and VMware NSX-T Container Plug-in for Ops Manager for supported version combinations.
    - VMware NSX-T Data Center Documentation. In particular, review the *NSX Container Plug-in (NCP) Release Notes* and *NSX-T Data Center Installation Guide* for the versions of NCP and NSX-T that you want to install.
- BOSH and Ops Manager installed and configured on vSphere. For more information, see Deploying Ops Manager on vSphere and Configuring BOSH Director on vSphere.
- The VMware NSX-T Container Plug-in for Ops Manager tile is downloaded from VMware Tanzu Network and imported to the Ops Manager **Installation Dashboard**. For information about downloading and importing VMware Tanzu products to the Installation Dashboard, see Add and Import Products in *Adding and Deleting Products*.
- The TAS for VMs tile is downloaded from VMware Tanzu Network and imported to the Ops Manager Installation Dashboard. The TAS for VMs tile must be in one of these states:
    - Configured but not currently deployed. You did not click **Review Pending Changes**, then **Apply Changes** on this version of TAS for VMs.

- Deployed previously, with the **Container network interface plugin** field set to **External** in the **Networking** pane of the TAS for VMs tile.

> **Note:** Deploying TAS for VMs with its container network interface (CNI) set to **Silk** configures Diego Cells to use an internally-managed container network. Subsequently switching the CNI interface to **External** NSX-T leads to errors.

Architecture

The following diagram shows how to deploy an NSX-T machine to run TAS for VMs across multiple vSphere hardware clusters. NSX-T runs a Tier-0 (T0) gateway and multiple Tier-1 (T1) gateways, each connecting to a network within Ops Manager. Each vSphere hardware column cluster corresponds to an Availability Zone in Ops Manager:



When a developer pushes an app to a new org for the first time, the NSX-T plugin triggers NSX-T to create a new T1 gateway and allocate an address range for the org, on demand.

Install and Configure TAS for VMs and NSX-T

Installing NSX-T to run with TAS for VMs requires:

1. Configure NSX-T to Integrate with TAS for VMs

2. Enable NSX-T Mode in the BOSH Director
3. Configure TAS for VMs for External Container Networking
4. Install and Configure the NSX-T Tile

Set Up NSX-T to Integrate with TAS for VMs

To set up NSX-T to integrate with TAS for VMs, complete these procedures:

- Configure NATs
- Configure Gateways
- Configure Segments
- Configure A Load Balancer

Configure NAT Rules

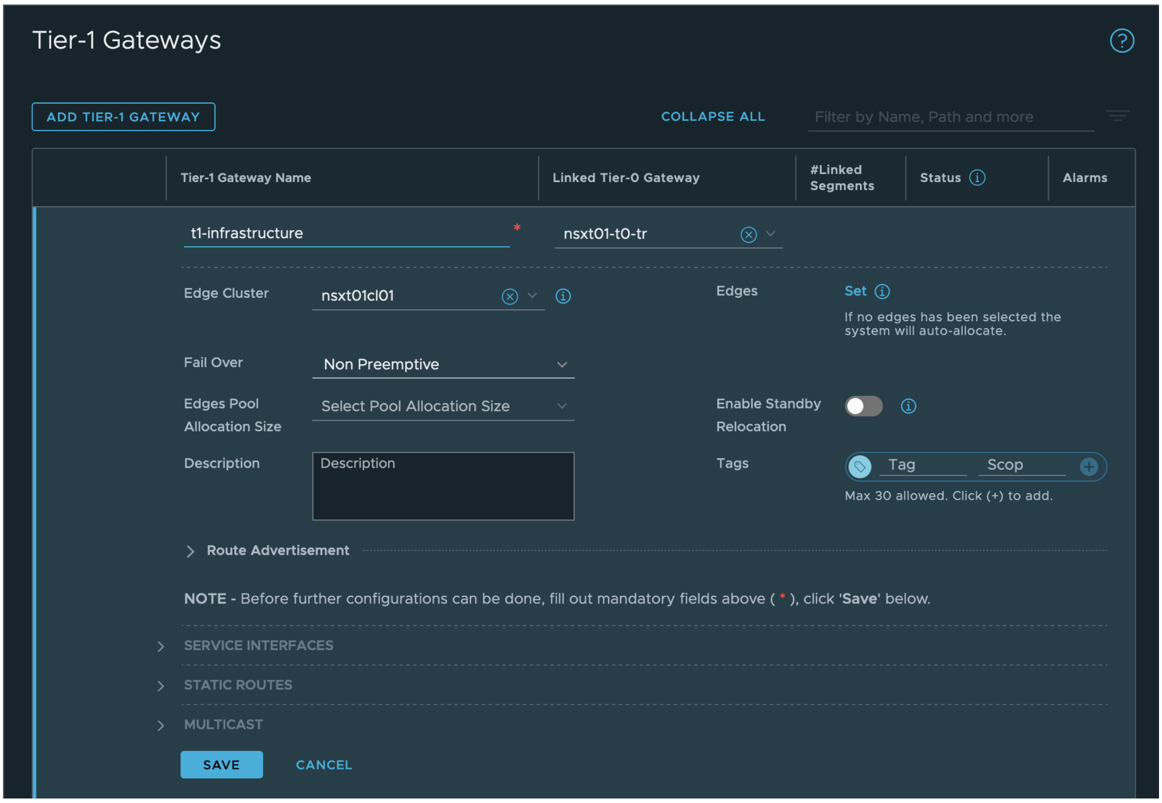
To configure Network Address Translation (NAT) rules:

1. Create network address translation (NAT) rules to communicate with Ops Manager:
    1. Go to **Networking**.
    2. Go to the **NAT** pane.
    3. Select your T0 gateway.
    4. Click **ADD NAT RULE**.
    5. Add a rule for destination NAT (DNAT) with:

Configure Gateways

To configure Tier-1 (T1) gateways:

1. Create T1 gateways for TAS for VMs, to connect from the T0 gateway. For each Ops Manager network, Infrastructure, Deployment, and so on, create a T1 gateway as follows:
    1. In the NSX-T Manager UI, go to **Networking**, then **Tier-1 Gateways**.
    2. Click **ADD TIER-1 GATEWAY**.
    3. Configure the gateway. Include the Edge Cluster, as it is required to enable the Load Balancer. The Infrastructure network gateway configuration looks similar to the following screenshot:



Tier-1 Gateway Name: t1-infrastructure *
Linked Tier-0 Gateway: nsxt01-t0-tr

Edge Cluster: nsxt01cl01

Edges: Set
If no edges has been selected the system will auto-allocate.

Fail Over: Non Preemptive

Edges Pool Allocation Size: Select Pool Allocation Size

Enable Standby Relocation

Description: Description

Tags: Tag Scop
Max 30 allowed. Click (+) to add.

NOTE - Before further configurations can be done, fill out mandatory fields above (*), click '**Save**' below.

2. Advertise the routes of the T1 gateways to the T0 gateway, so the T0 gateway can correctly route incoming requests based on their destination IP addresses:

    1. Edit your T1 Gateway and go to **Route Advertisement**.
    2. Activate **All Connected Segments & Service Ports**.
    3. Activate **All LB VIP Routes**. (necessary if Load Balancing service is configured).

3. Allocate an IP block for TAS for VMs orgs.

    1. From the NSX-T Manager, navigate to **Networking**, then **IP Address Pools**, click "IP ADDRESS BLOCKS tab, and click **ADD IP ADDRESS BLOCK**.
    2. Enter a name (for example, `TAS for VMs-container-ip-block`). This IP block name is also used in the **VMware NSX-T** tile in the **NCP** section under **IP Blocks of Container Networks**.
    3. Enter a description, such as `Subnets are allocated from this pool to each newly-created org`.
    4. Enter a CIDR to allocate an address block large enough to accommodate all TAS for VMs apps. A `/14` CIDR is large enough for ~1,000 Orgs with ~250 apps each. If you are planning such a large foundation, see VMware NSX-T TAS for VMs limits in the VMware documentation.

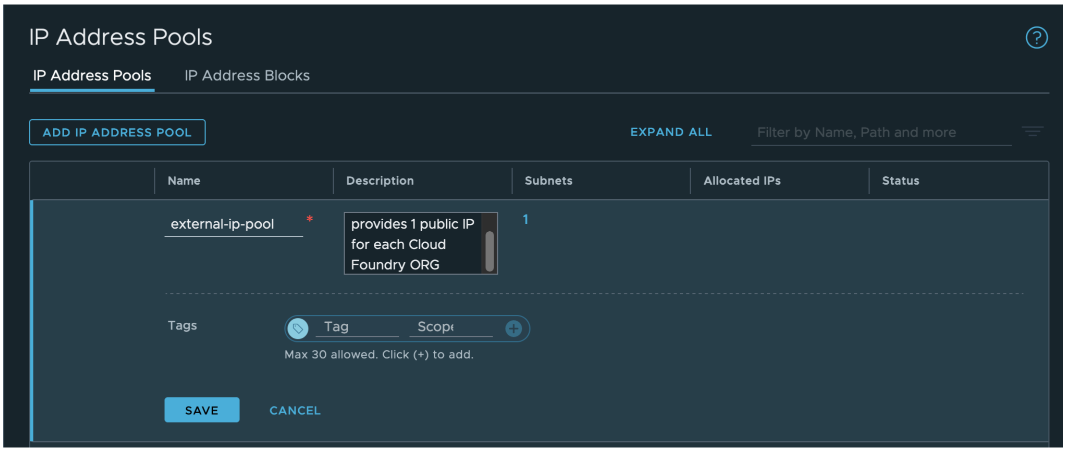
4. Allocate an IP block for TAS for VMs orgs.

    1. From the NSX-T Manager, go to **Networking**, then **IP Address Pools**, click the **IP Address Blocks** tab, and click **Add IP Address Block**.
    2. Enter a name (for example, `TAS for VMs-container-ip-block`). This IP block name is also used in the **VMware NSX-T** tile in the **NCP** section under **IP Blocks of Container Networks**.

3. Enter a description, such as `Subnets are allocated from this pool to each newly-created org`.
4. Enter a CIDR to allocate an address block large enough to accommodate all TAS for VMs apps. A `/14` CIDR is large enough for approximately 1,000 Orgs with about 250 apps each. If you are planning a large foundation, see VMware NSX-T TAS for VMs limits in the VMware documentation.

Configure Segments

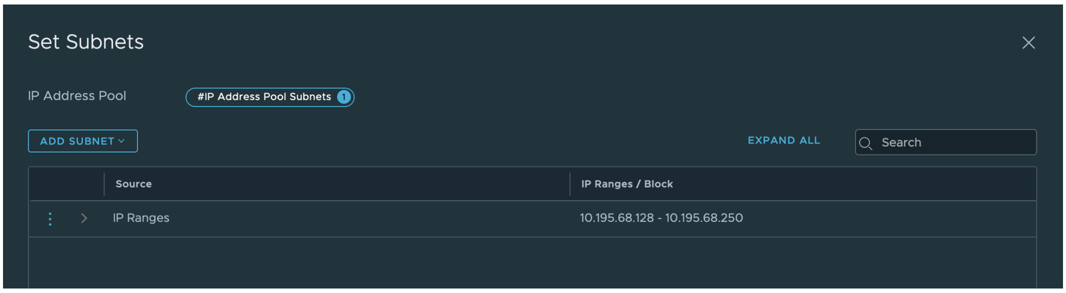
1. Create an external SNAT IP pool:
   1. Go to **Networking**, then **IP Address Pools**.
   2. Click the **IP Address Pools** tab and then click **Add IP Address Pool**.
   3. Enter a name (for example, `external-ip-pool`).
   4. Enter a description (for example, “IP pool that provides 1 public IP for each TAS for VMs Org”). Later, you enter this pool name, on the **VMware NSX-T** tile, in the **NCP** section, under **IP Pools used to provide External (NAT) IP Addresses to Org Networks**.
   5. Set a subnet of externally-routable IP addresses for future NAT IP addresses.



IP Address Pools

Name	Description	Subnets	Allocated IPs	Status
external-ip-pool *	provides 1 public IP for each Cloud Foundry ORG	1		

Tags: Tag / Scope — Max 30 allowed. Click (+) to add.



Set Subnets

IP Address Pool: #IP Address Pool Subnets 1

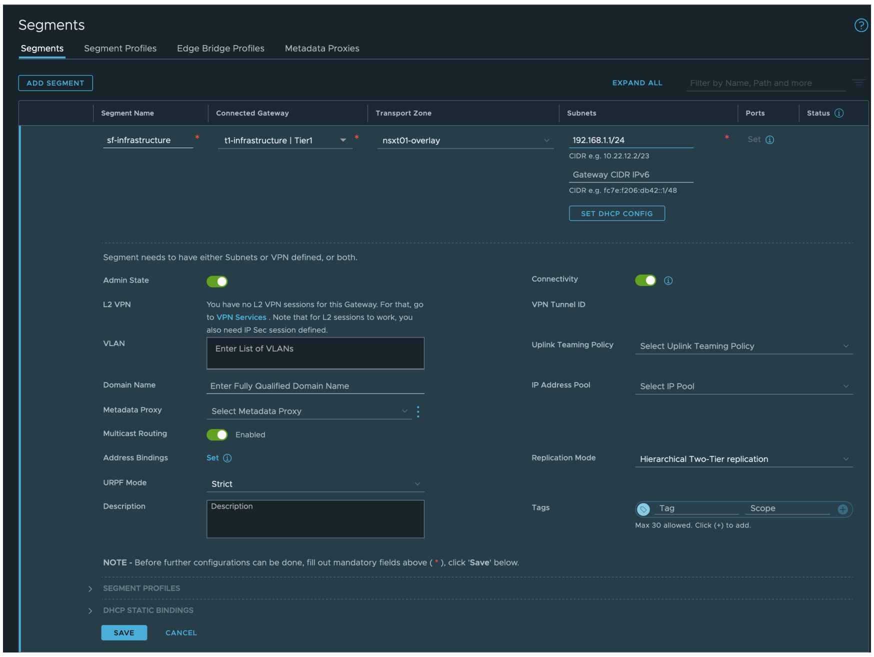
Source	IP Ranges / Block
IP Ranges	10.195.68.128 - 10.195.68.250

2. In vSphere, create segments that correspond to the networks that Ops Manager uses.
   1. Log in to the **NSX-T Manager Dashboard**.
   2. Go to **Networking**.
   3. Go to the **Segments** pane.
   4. For each of these networks...

        - **Infrastructure** (BOSH and Ops Manager, defined in the **Assign AZs and Networks** pane of the BOSH Director tile)
        - **Deployment** (TAS for VMs, defined in the **Assign AZs and Networks** pane of the TAS for VMs tile)
        - **Services** and **Dynamic Services** (marketplace services and on-demand services, also defined in the TAS for VMs tile)
        - **Isolation Segment** (optional, defined in the **Assign AZs and Networks** pane of the Isolation Segment tile) …do the following:
            1. Click **ADD SEGMENT**.
            2. Enter a name for the segment.
            3. Enter a Gateway to connect to.
            4. Click **SAVE**.

3. In vSphere, create segments that correspond to the networks that Ops Manager uses.
    1. Log in to the **NSX-T Manager Dashboard**.
    2. Go to **Networking**.
    3. Go to the **Segments** pane.
    4. For each of these networks…
        - **Infrastructure** (BOSH and Ops Manager, defined in the **Assign AZs and Networks** pane of the BOSH Director tile)
        - **Deployment** (TAS for VMs, defined in the **Assign AZs and Networks** pane of the TAS for VMs tile)
        - **Services** and **Dynamic Services** (marketplace services and on-demand services, also defined in the TAS for VMs tile)
        - **Isolation Segment** (optional, defined in the **Assign AZs and Networks** pane of the Isolation Segment tile) …do the following:
            1. Click **ADD SEGMENT**.
            2. Enter a name for the segment
            3. Enter a Gateway to connect to.

4. Click **SAVE**.



Segment Name	Connected Gateway	Transport Zone	Subnets	Ports	Status
sf-infrastructure *	t1-infrastructure | Tier1 *	nsxt01-overlay	192.168.1.1/24 * CIDR e.g. 10.22.12.2/23 Gateway CIDR IPv6 CIDR e.g. fc7e:f206:db42::1/48	Set	

Segment needs to have either Subnets or VPN defined, or both.

Admin State: on

Connectivity: on

L2 VPN: You have no L2 VPN sessions for this Gateway. For that, go to VPN Services . Note that for L2 sessions to work, you also need IP Sec session defined.

VPN Tunnel ID

VLAN: Enter List of VLANs

Uplink Teaming Policy: Select Uplink Teaming Policy

Domain Name: Enter Fully Qualified Domain Name

IP Address Pool: Select IP Pool

Metadata Proxy: Select Metadata Proxy

Multicast Routing: Enabled

Address Bindings: Set

Replication Mode: Hierarchical Two-Tier replication

URPF Mode: Strict

Description: Description

Tags: Tag Scope

Max 30 allowed. Click (+) to add.

NOTE - Before further configurations can be done, fill out mandatory fields above (*), click '**Save**' below.

Configure Load Balancer

1. Create Active Monitors (health checks) for use by the virtual servers later.
    1. In the NSX-T Manager UI, go to **Networking**, then **Load Balancing**, and click the **Monitors** tab.
    2. Create the health monitor for web load balancing:
    3. Click **Add Active Monitor**.
    4. Select **HTTP**.
        - **Name**: `tas-web-monitor`
        - **Monitoring Port**: 8080
        - **Monitoring Port**: 8080
    5. Configure **Additional Properties**:
        - **HTTP Request URL**: `/health`
        - **HTTP Response Code**: 200
    6. Click **Save**.
2. Create the health monitor for TCP load balancing:
    1. Click **ADD ACTIVE MONITOR**.
    2. Select **HTTP**.
        - **Name**: `tas-tcp-monitor`
        - **Monitoring Port**: 80
    3. Configure **Additional Properties**:
        - **HTTP Request URL**: `/health`
        - **HTTP Response Code**: 200

    4. Click **Save**.

3. Create the health monitor for SSH load balancing:
    1. Click **ADD ACTIVE MONITOR**.
    2. Select **TCP**:
        - **Name**: `tas-ssh-monitor`
        - **Monitoring Port**: 2222
    3. Click **Save**.
4. Create Server Pools (collections of VMs that handle traffic) for use by the virtual servers.
    1. In the NSX-T Manager UI, go to **Networking**, then **Load Balancing**, and click the **Server Pools** tab.
    2. Create the server pool for web load balancing:
    3. Click **Add Server Pool** to add a new pool.
        - **Name**: `tas-web-pool`
    4. Enter **SNAT Translation**: `Automap`
    5. Click **Select Members**:
        - **Membership Type**: `Static`
    6. Click **Active Monitor Set**:
        - Select `tas-web-monitor`
        - Click **Apply**
    7. Click **Save**.
5. Create the server pool for TCP load balancing:
    1. Click **Add Server Pool** to add a new pool.
        - **Name**: `tas-tcp-pool`
    2. Enter **SNAT Translation**: `Disabled`
    3. Click **Select Members**:
        - **Membership Type**: `Static`
    4. Click **Active Monitor Set**:
        - Select `tas-tcp-monitor`
        - Click **Apply**
    5. Click **Save**.
6. Create the server pool for SSH load balancing:
    1. Click **ADD SERVER POOL** to add a new pool.
        - **Name**: `tas-ssh-pool`
    2. Enter **SNAT Translation**: `Disabled`
    3. Click **Select Members**:
        - **Membership Type**: `Static`

    4. Click **Active Monitor Set**:
        - Select `tas-ssh-monitor`
        - Click **Apply**
    5. Click **Save**.
7. Create the load balancer. In the NSX-T Manager UI, go to **Networking**, then **Load Balancing**, and click the **Load Balancers** tab.
    1. Click **Add Load Balancer**.
        1. Enter the fields:
            - **Name**: `tas-lb`
            - **Load Balancer Size**: Select `Small` unless you have a larger Foundation.
            - **Attachment**: `t1-deployment` Attach your load balancer to the Tier 1 gateway fronting your deployment instances.
        2. Click **Save**
        3. Click **Yes** when prompted **Want to continue configuring this Load Balancer?**
        4. Click **Virtual Servers Set**
8. Click **Add Active Monitor**.
9. Select **HTTP**:
    - **Name**: `tas-web-monitor`
    - **Monitoring Port**: 8080.
10. Configure **Additional Properties**:
    - **HTTP Request URL**: `/health`
    - **HTTP Response Code**: 200.
11. Click **Save**.

To create the health monitor for TCP load balancing:

1. Click **Add Active Monitor**.
2. Select **HTTP**:
    - **Name**: `tas-tcp-monitor`
    - **Monitoring Port**: 80.
3. Configure **Additional Properties**:
    - **HTTP Request URL**: `/health`
    - **HTTP Response Code**: 200
4. Click **Save**.

To create the health monitor for SSH load balancing:

1. Click **Add Active Monitor**.

2. Select **TCP**:
    - **Name**: `tas-ssh-monitor`
    - **Monitoring Port**: 2222.
3. Click **Save**.

To create Server Pools (collections of VMs which handle traffic) for use by the virtual servers:

1. In the NSX-T Manager UI, go to **Networking**, then **Load Balancing**.
2. Click the **Server Pools** tab.

To create the server pool for web load balancing:

1. Click **Add Server Pool** : **Name**: `tas-web-pool`
2. Enter **SNAT Translation**: `Automap`
3. Click **Select Members**: **Membership Type**: `Static`
4. Click **Active Monitor Set**: `tas-web-monitor`
5. Click **Apply**
6. Click **Save**.

To create the server pool for TCP load balancing:

1. Click **Add Server Pool** to add a new pool.
    - **Name**: `tas-tcp-pool`
        1. Enter **SNAT Translation**: `Disabled`
        2. Click **Select Members**: **Membership Type**: `Static`
        3. Click **Active Monitor Set**: `tas-tcp-monitor`
        4. Click **Apply**
        5. Click **Save**.
2. Create the server pool for SSH load balancing:
    1. Click **Add Server Pool Name**: `tas-ssh-pool`
    2. Enter **SNAT Translation**: `Disabled`
    3. Click **Select Members**: **Membership Type**: `Static`
    4. Click **Active Monitor Set**: `tas-ssh-monitor`
    5. Click **Apply**.
    6. Click **Save**.

To create the load balancer:

1. In the NSX-T Manager UI, go to **Networking**, then **Load Balancing**.
2. Click the **Load Balancers** tab.
3. Click **Add Load Balancer**.
    1. Enter the fields:
        - **Name**: `tas-lb`

        - **Load Balancer Size**: Select `Small` unless you have a larger Foundation.
        - **Attachment**: `t1-deployment` Attach your load balancer to the Tier 1 gateway fronting your deployment instances.
    2. Click **Save**.
    3. Click Yes, when prompted with, **Want to continue configuring this Load Balancer?**.

To create the virtual server that forwards unencrypted web (HTTP) traffic to the foundation.

> **Note:** For foundations requiring end-to-end encryption, do not enable the virtual server on port 80. If it must be enabled, configure it to redirect traffic to the encrypted port (443).

1. Click **Virtual Servers Set**.
2. Click **Add Virtual Server**.
3. Select **L4 TCP**.
    - **Name**: `tas-web-vs`
    - **Application Profile**: `default-tcp-lb-app-profile`
    - **IP Address**: use the address of the DNS record of `*.system.YOUR-SYSTEM-DOMAIN.com`
    - **Port**: `80,443`
    - **Server Pool**: `tas-web-pool`
4. Click **Save**.

To create the virtual server that forwards traffic to apps with custom tcp ports to the foundation:

1. Click **Add Virtual Server**.
2. Select **L4 TCP**.
    - **Name**: `tas-tcp-vs`
    - **Application Profile**: `default-tcp-lb-app-profile`
    - **IP Address**: Use the address of the DNS record of `tcp.apps.YOUR-SYSTEM-DOMAIN.com`
    - **Port**: Use the same ports as configured in the **TAS for VMs Tile**, then **Networking**, and **TCP Routing Ports**. For example: `1024-1123,5900`
    - **Server Pool**: `tas-tcp-pool`
3. Click **Save**.

To create the virtual server that forwards SSH traffic to the foundation:

1. Click **Add Virtual Server**.
2. Select **L4 TCP**.
    - **Name**: `tas-ssh-vs`

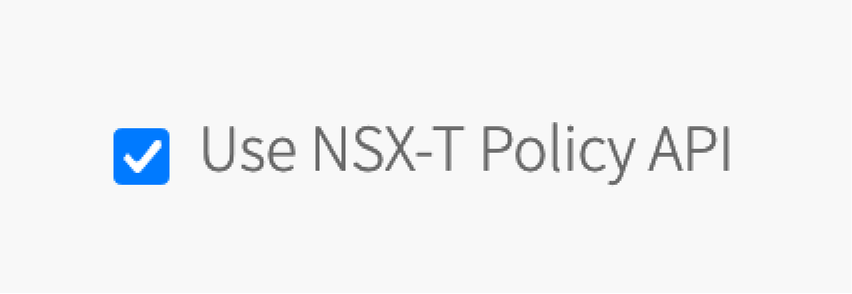
   - **Application Profile**: `default-tcp-lb-app-profile`
   - **IP Address**: Use the address of the DNS record of `ssh.system.YOUR-SYSTEM-DOMAIN.com`.
   - **Port**: `2222`
   - **Server Pool**: `tas-ssh-pool`

3. Click **Save**.

Enable NSX-T Mode in the BOSH Director

To enable NSX-T mode in the BOSH Director:

1. From the Ops Manager **Installation Dashboard**, open the **BOSH Director for vSphere** tile.
2. In the **vCenter Config** pane, click the pencil icon for the vCenter Config you want to edit.
3. Select **NSX Networking** below.
4. Configure BOSH Director authentication to the NSX Manager by following the **NSX Networking** instructions in the Step 2: Configure vCenter section of *Configuring BOSH Director on vSphere*.
5. Verify that the **Use NSX-T Policy API** option is selected.

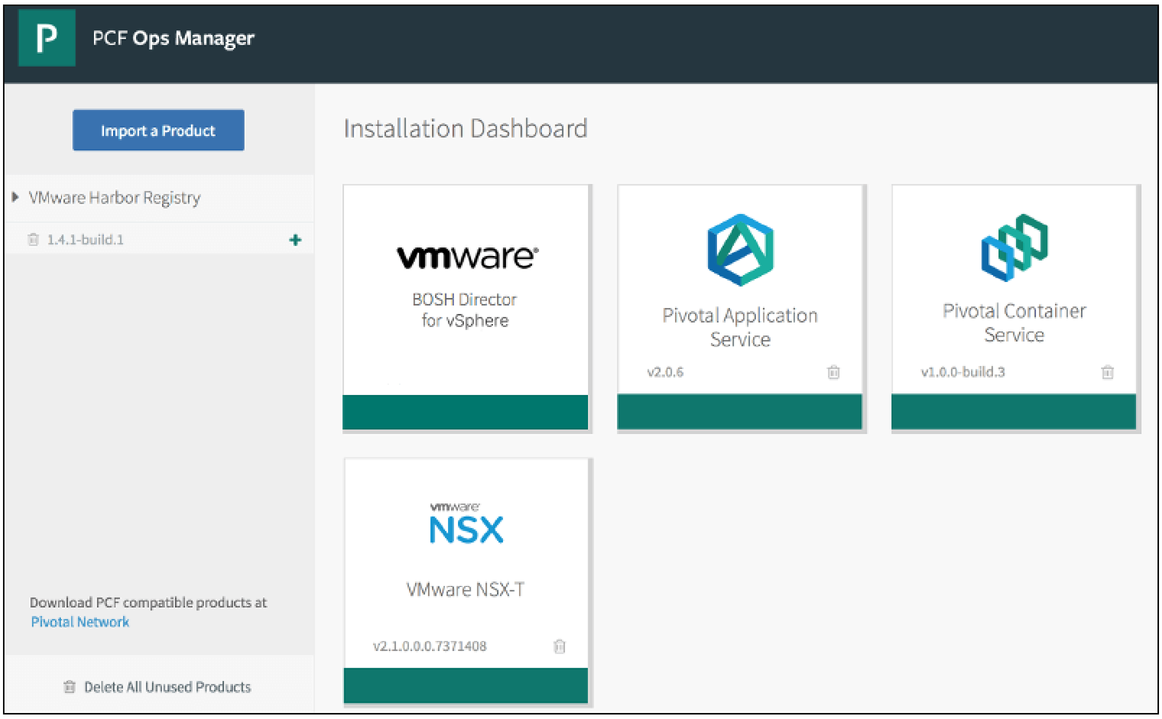


☑ Use NSX-T Policy API

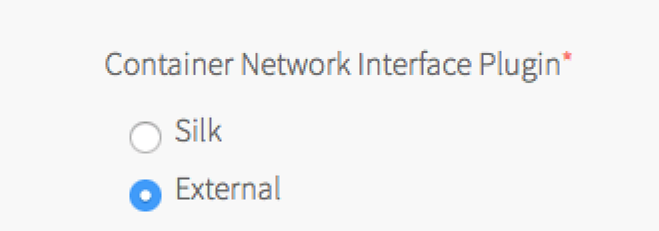
Configure TAS for VMs for External Container Networking

1. If you have not already done so, download the TAS for VMs tile from VMware Tanzu Network and import it to the **Installation Dashboard**.

   For instructions, see Add and Import Products.



2. Configure TAS for VMs, following the directions in Configuring TAS for VMs. When you configure **Networking**, select **External** under **Container networking interface plugin**.



Container Network Interface Plugin*

○ Silk

● External

3. Configure TAS for VMs to add router, diego_brain, and tcp_router instances to the corresponding NSX-T server pools upon deployment.

    1. Open the TAS for VMs tile > **Resource Config** pane.

    2. Click the arrow next to each job to reveal the **NSX-T CONFIGURATION** column.

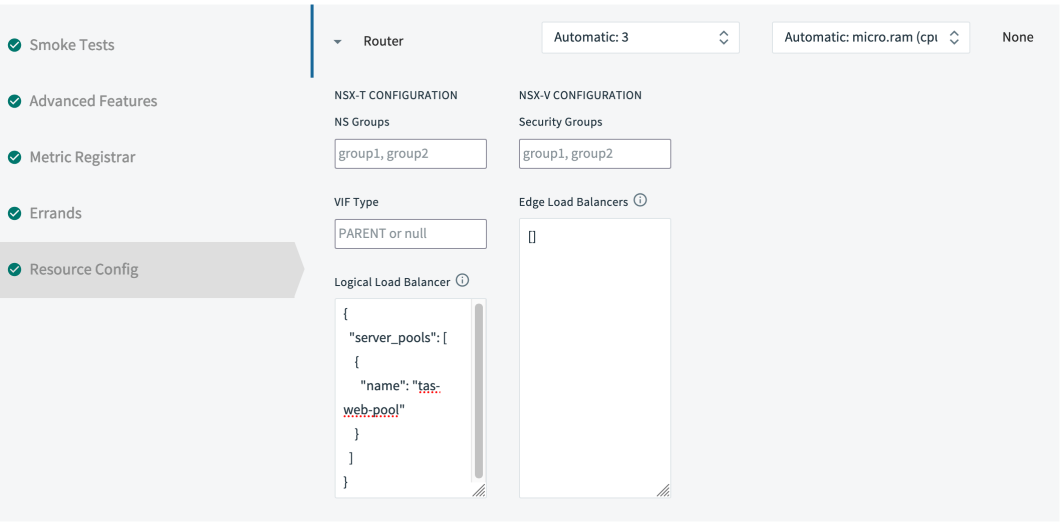
    3. Under **Logical Load Balancer**, fill in the JSON `server_pools` list with the NSX-T Server Pool these instance should be added to upon deployment.

        - router -> tas-web-pool

        - diego_brain -> tas-ssh-pool

        - tcp_router -> tas-tcp-pool

4. Click **Save**



Smoke Tests
Advanced Features
Metric Registrar
Errands
Resource Config

Router: Automatic: 3 | Automatic: micro.ram (cpu | None

NSX-T CONFIGURATION

NS Groups: group1, group2

VIF Type: PARENT or null

Logical Load Balancer:

```
{
  "server_pools": [
   {
     "name": "tas-web-pool"
   }
  ]
}
```

NSX-V CONFIGURATION

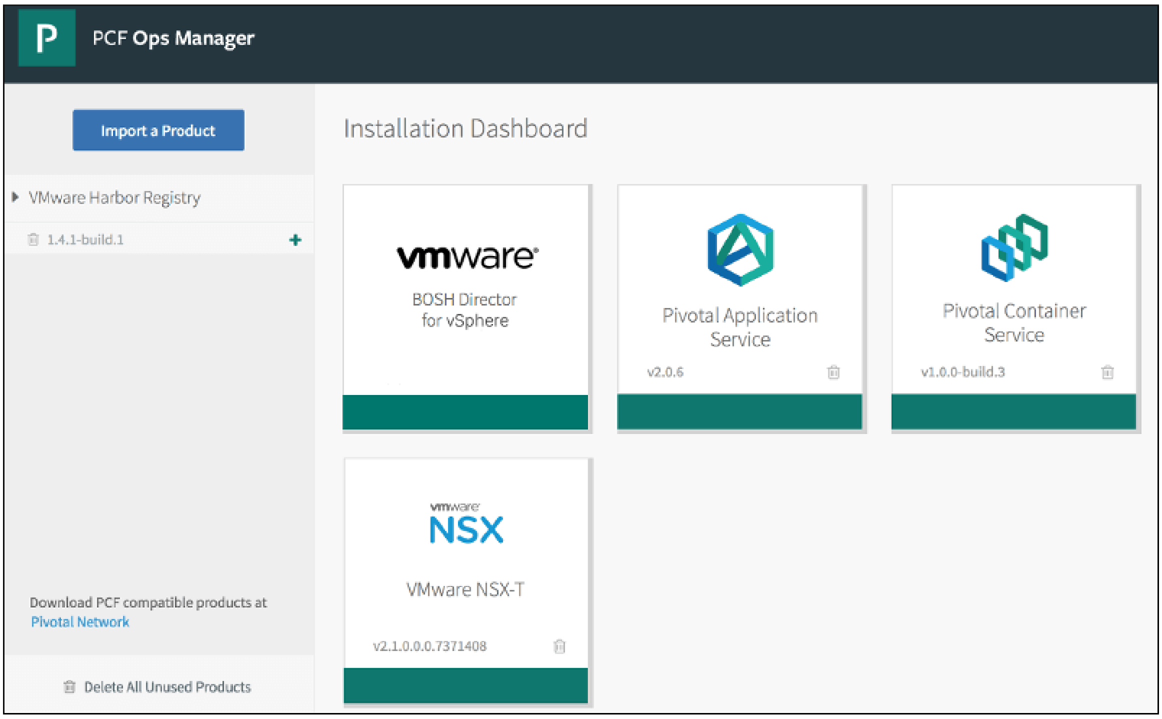
Security Groups: group1, group2

Edge Load Balancers: []

Install and Configure the NSX-T Container Plug-In

To install and configure the tile:

1. If you have not already done so, download the VMware NSX-T Container Plug-in for Ops Manager tile from VMware Tanzu Network and import it to the **Installation Dashboard**. For instructions, see Add and Import Products.



PCF Ops Manager
Import a Product
VMware Harbor Registry
1.4.1-build.1
Installation Dashboard
vmware
BOSH Director for vSphere
Pivotal Application Service
v2.0.6
Pivotal Container Service
v1.0.0-build.3
vmware NSX
VMware NSX-T
v2.1.0.0.0.7371408
Download PCF compatible products at Pivotal Network
Delete All Unused Products

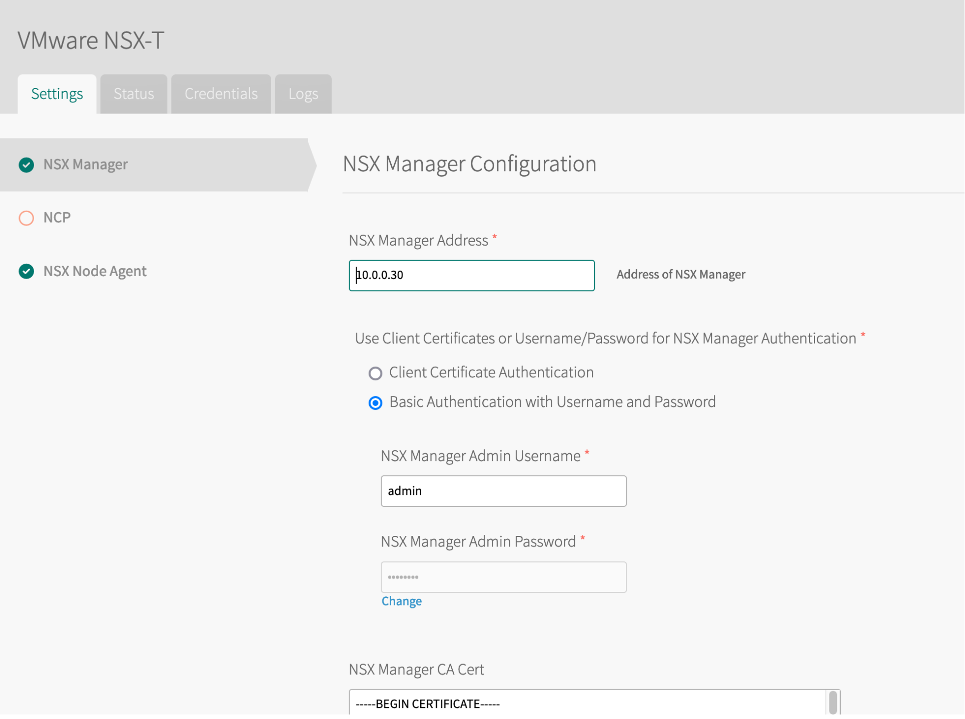
1. If you are using VMware Workspace ONE Access, formerly called VMware Identity Manager (vIDM), then select **Client Certificate Authentication**.

2. Otherwise, select **Basic Authentication with Username and Password** and enter **NSX Manager Admin Username** and **Admin Password** credentials in the fields.

3. **NSX Manager CA Cert**: Obtain this certificate from NSX-T Manager as follows:

    1. `ssh` into NSX-T Manager using the admin account that you created when you deployed NSX-T Manager.

2. From the NSX-T Manager command line, run `get certificate api` to retrieve the certificate.



NSX Manager Configuration

NSX Manager Address *
10.0.0.30
Address of NSX Manager

Use Client Certificates or Username/Password for NSX Manager Authentication *

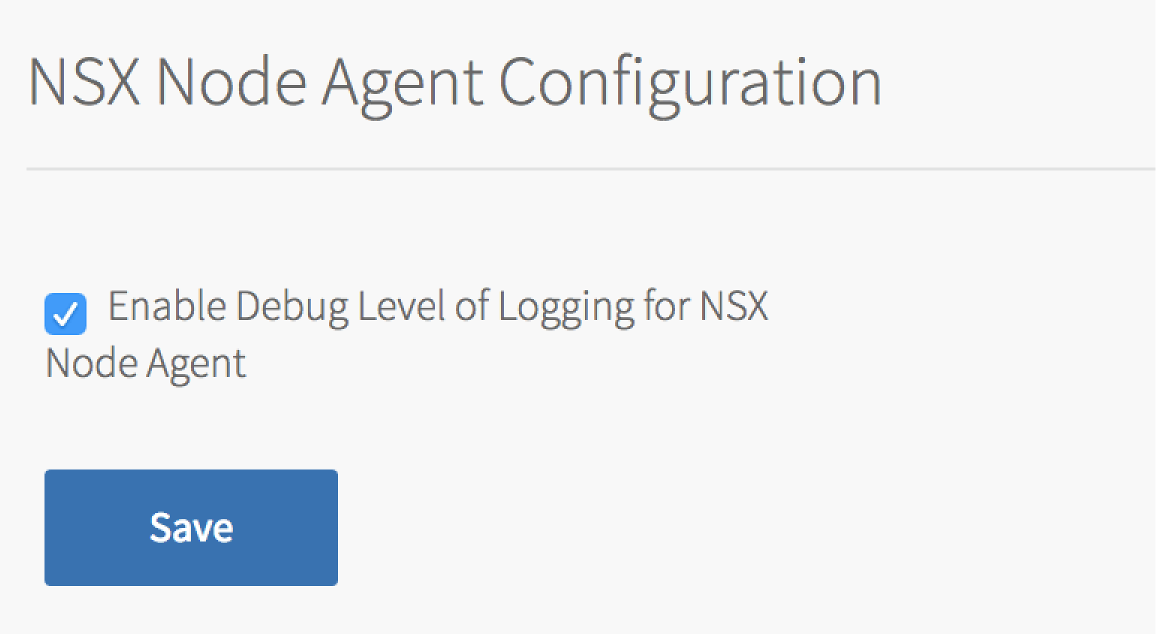
○ Client Certificate Authentication
● Basic Authentication with Username and Password

NSX Manager Admin Username *
admin

NSX Manager Admin Password *
••••••••
Change

NSX Manager CA Cert
-----BEGIN CERTIFICATE-----

2. Open and configure the **NCP** (NSX-T Container Plugin) pane as follows:
   - **TAS for VMs Foundation Name**: If unsure, use `TAS for VMs`. If multiple foundations co-exist on the same NSX-T Manager, choose a unique string, such as `TAS for VMs-beta`. NCP creates artifacts, such as T1 gateways and prefixes their names with this string for easy identification.
   - **Overlay Transport Zone**: A uniquely identifying string for the **Transport Zone** that you chose when you created segments for each network. This can be the name of the transport zone if no other zones in NSX-T share the same name, or else the UUID for the transport zone.
   - **Tier-0 Router**: A uniquely identifying string for the T0 gateway. This can be the tag string that you gave the gateway in NSX-T Manager if no other T0 gateways in NSX-T share the same name, or else the UUID for the gateway.
   - **IP Blocks of Container Networks**: Use the same IP block created Configure Gateways.
   - **Subnet Prefix of Container Networks**: Subnet mask to set the address range size for apps in a single org. Defaults to `24`. This number must be higher than the mask for all TAS for VMs orgs in the NSX-T Manager **New IP Block** pane, to define each org's fraction of the total TAS for VMs address space.
   - **IP Pools used to provide External (NAT) IP Addresses to Org Networks**: Use the same IP Pool created iin Configure Gateways.
   - **Enable NSX-T Policy API**: Enable this checkbox to use the new Policy API.
3. In the **NSX Node Agent** pane, enable the **Enable Debug Level of Logging for NSX Node Agent** checkbox.



NSX Node Agent Configuration

☑ Enable Debug Level of Logging for NSX Node Agent

4. Click **Save** and return to the **Installation Dashboard**.

5. After you have configured both the **TAS for VMs** tile and the **VMware NSX-T** tile, click **Review Pending Changes**, then **Apply Changes** to deploy TAS for VMs with NSX-T networking.

Upgrade TAS for VMs with NSX-T Networking

After you have deployed TAS for VMs with NSX-T, you may need to upgrade either Ops Manager, TAS for VMs, the NSX-T Container Plug-in or NSX-T Data Center. If you upgrade one of these components, you may need to upgrade the other components as well.

For example, if you want to upgrade NSX-T Data Center, you may need to upgrade the NSX-T Container Plug-in first.

To upgrade TAS for VMs with NSX-T Networking:

1. Plan the upgrade by determining the compatibility of NCP, NSX-T and TAS for VMs by checking the following documentation:
    - See Product Interoperability Matrix: TAS for VMs, VMware NSX-T, and VMware NSX-T Container Plug-in for Ops Manager for supported version combinations.
    - See VMware NSX-T Data Center Documentation. In particular, review the *NSX Container Plug-in (NCP) Release Notes* and *NSX-T Data Center Installation Guide* for the versions of NCP and NSX-T that you want to install.
2. Download the desired version of VMware NSX-T Container Plug-in for Ops Manager tile from VMware Tanzu Network.
3. In Ops Manager, import the new version of the tile to the **Installation Dashboard**. For instructions, see Adding and Importing Products.
4. Click **Review Pending Changes** and review your changes.
5. Click **Apply Changes**.

6. Continue with the upgrade of Ops Manager, TAS for VMs, or NSX-T Data Center. For more information, see Upgrade NCP in a Ops Manager Environment in the VMware NSX-T Data Center documentation.

Deploying TAS for VMs to AVS

This topic describes how to install VMware Tanzu Application Service for VMs (TAS for VMs) on Azure VMware Solution (AVS) with NSX-T internal networking, using the VMware NSX-T Container Plug-in.

For more information about AVS, see the Azure VMware Solution documentation.

Overview

AVS provides you with private clouds that contain vSphere clusters built from dedicated bare-metal Azure infrastructure. AVS is a VMware-validated solution with ongoing validation and testing of enhancements and upgrades. Microsoft manages and maintains private cloud infrastructure and software.

All provisioned private clouds have vCenter Server, vSAN, vSphere, and NSX-T.

Deploying TAS for VMs to AVS is similar to traditional TAS for VMs deployments on vSphere, but there are a few minor differences due to the way that AVS deploys vSphere components.

To install NSX-T to run with TAS for VMs:

1. Configure NSX-T to Integrate with TAS for VMs
2. Enable NSX-T Mode in the BOSH Director
3. Configure TAS for VMs for External Container Networking
4. Install and Configure the NSX-T Container Plugin

Prerequisites

Before deploying TAS for VMs with NSX-T networking, you must have:

- BOSH and Ops Manager installed and configured on vSphere. For more information, see Deploying Ops Manager on vSphere and Configuring BOSH Director on vSphere.
- The VMware NSX-T Container Plug-in tile downloaded from VMware Tanzu Network and imported to the Ops Manager Installation Dashboard. For information about downloading and importing VMware Tanzu products to the Installation Dashboard, see Adding and Importing Products.
- The TAS for VMs tile downloaded from VMware Tanzu Network and imported to the Ops Manager Installation Dashboard. The TAS for VMs tile must be configured but not deployed yet. Configure the tile for the first time, but do not click **Review Pending Changes** or **Apply Changes**.
- The URLs and user credentials for your AVS private cloud vCenter and NSX-T Manager. See Connect to the local vCenter of your private cloud in the AVS documentation.

Configure NSX-T to Integrate with TAS for VMs

To set up NSX-T to integrate with TAS for VMs, complete these procedures:

- Regenerate NSX-T Management TLS Certificate with a Valid SAN
- Configure Logical Switches
- Configure Routers
- Configure Load Balancer

Regenerate NSX-T Management TLS Certificate with a Valid SAN

The AVS-deployed NSX-T Manager includes a self-signed TLS certificate with an invalid Subject Alternative Name (SAN). This causes issues when connecting from Ops Manager and the BOSH Director, so you must create a new certificate using the NSX-T Manager's fully qualified domain name (FQDN) or IP address as the new SAN.

For instructions, see Generate and Register the NSX-T Management TLS Certificate and Private Key in the VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) documentation.

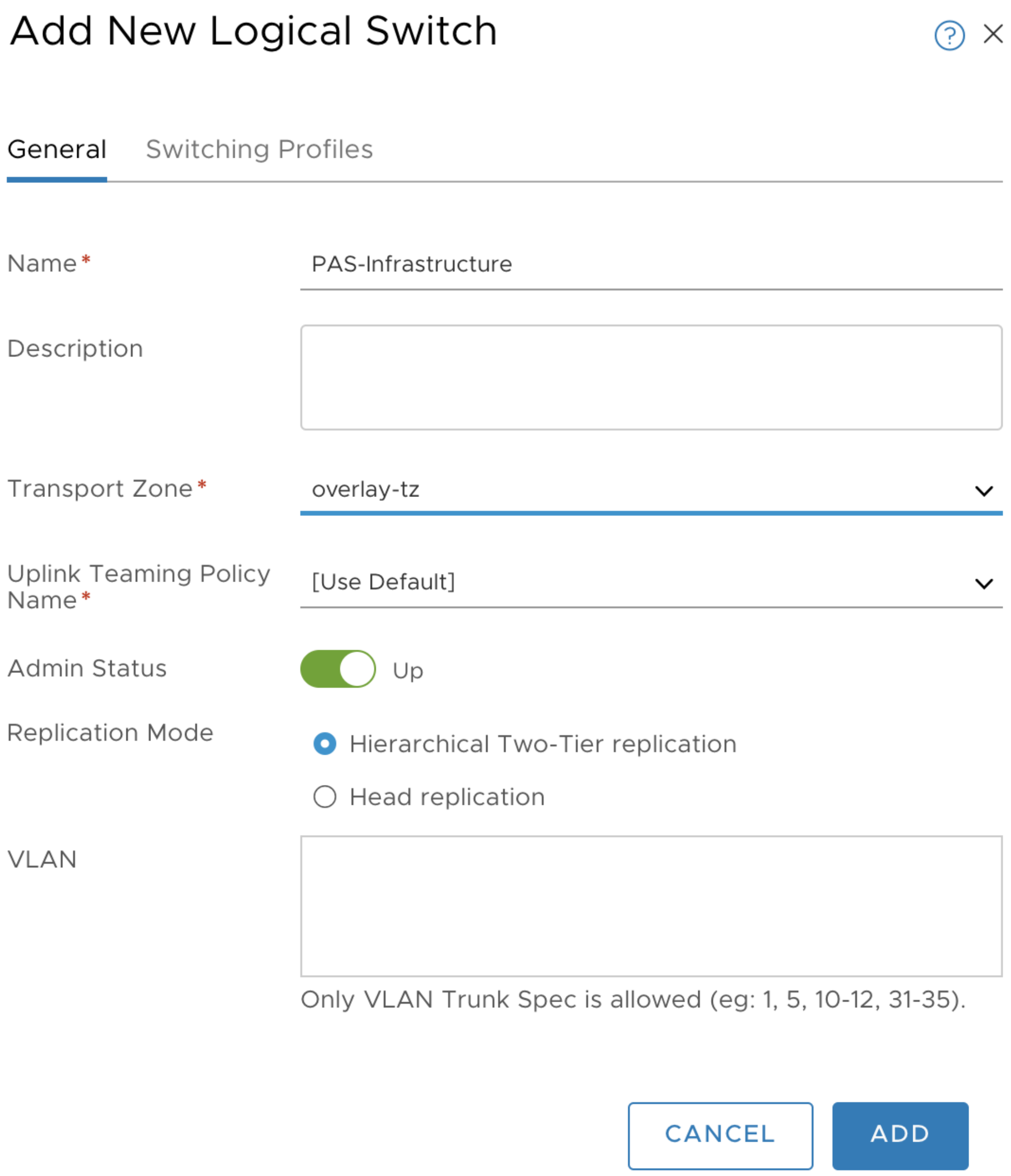
Configure Logical Switches

To configure logical switches:

1. In vSphere, create logical network switches to correspond to the networks that your foundation uses.
2. Log in to the **NSX-T Manager Dashboard**.
3. Go to **Advanced Networking & Security**.
4. Go to the **Switching** pane.
5. For each of these networks:
    - **Infrastructure** (BOSH and Ops Manager, defined in the **Assign AZs and Networks** pane of the BOSH Director tile)
    - **Deployment** (TAS for VMs, defined in the **Assign AZs and Networks** pane of the TAS for VMs tile)
    - **Services** and **Dynamic Services** (marketplace services and on-demand services, also defined in the TAS for VMs tile)
    - **Isolation Segment** (optional, defined in the **Assign AZs and Networks** pane of the Isolation Segment tile)

    Do the following:

    1. Click **+Add**.
    2. Enter a name for the logical switch (such as `TAS for VMs-Infrastructure`, `TAS for VMs-Deployment`).
    3. Click **Add**.



Add New Logical Switch

Name* PAS-Infrastructure

Description

Transport Zone* overlay-tz

Uplink Teaming Policy Name* [Use Default]

Admin Status Up

Replication Mode
● Hierarchical Two-Tier replication
○ Head replication

VLAN

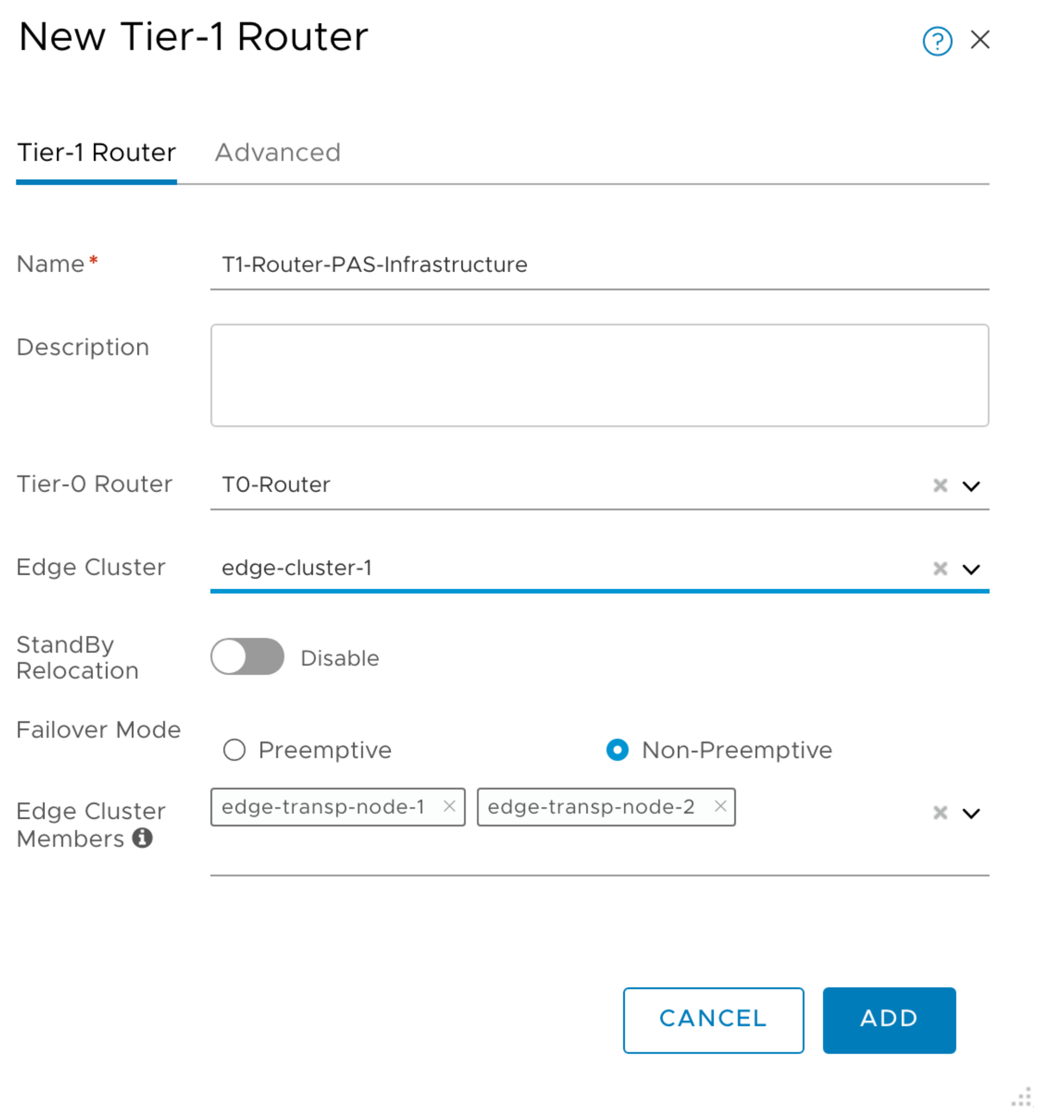
Only VLAN Trunk Spec is allowed (eg: 1, 5, 10-12, 31-35).

Configure Routers

To configure routers:

1. Create T1 routers for TAS for VMs, to connect from the T0 router. For each TAS for VMs network, Infrastructure, Deployment, and so on, create a T1 router as follows:
2. In the NSX-T Manager UI, navigate to **Advanced Networking & Security** > **Routing** > **Routers**.
    1. Click **+ADD** > **Tier-1 Router**.

2. Configure the router. Include the Edge Cluster and Edge Cluster Members; they are required to enable the Load Balancer. The Infrastructure network router configuration might look like the following diagram:



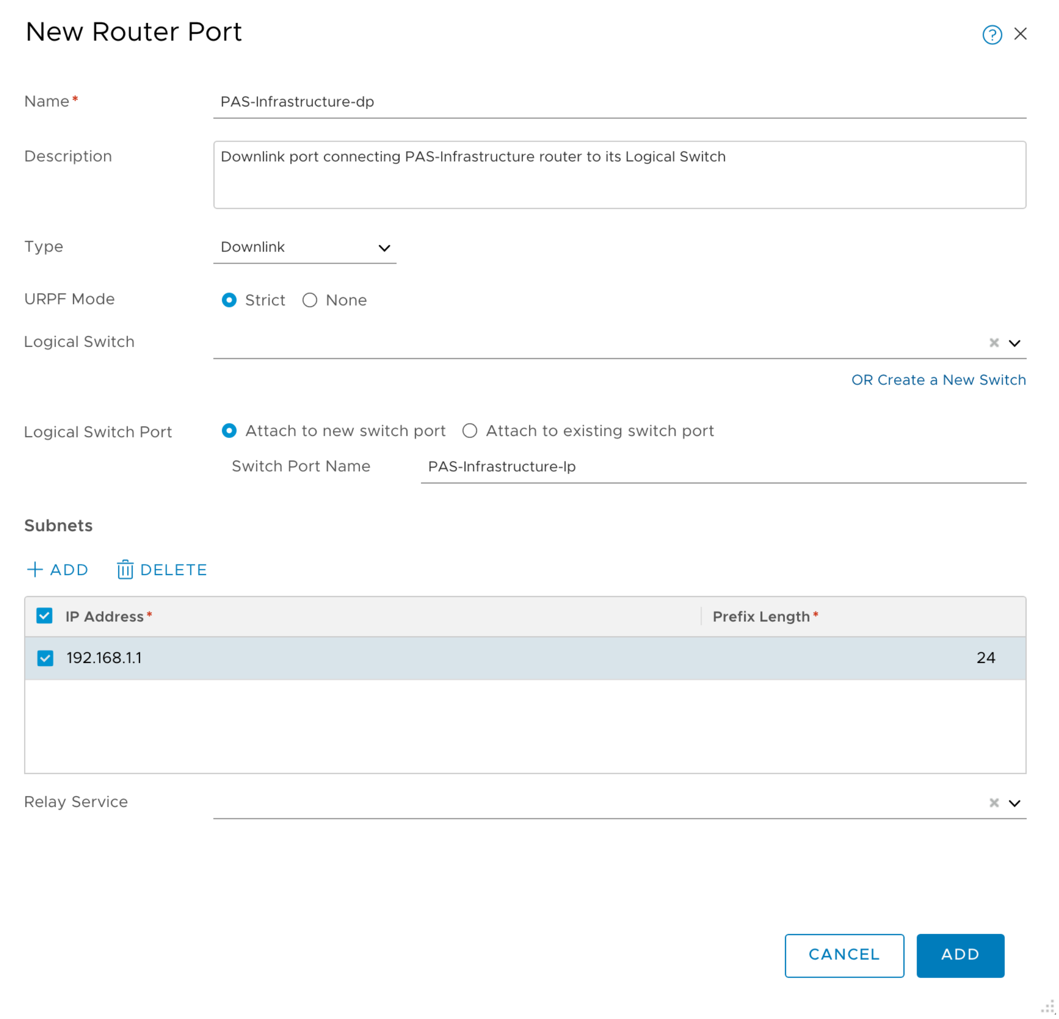
New Tier-1 Router

Name*	T1-Router-PAS-Infrastructure
Description	
Tier-0 Router	T0-Router
Edge Cluster	edge-cluster-1
StandBy Relocation	Disable
Failover Mode	○ Preemptive ● Non-Preemptive
Edge Cluster Members	edge-transp-node-1 edge-transp-node-2

3. Create T1 router downlink ports for TAS for VMs. For each T1 router you created, add a **New Router Port** as follows, to allow traffic in and out:

    1. In the NSX-T Manager UI, select the T1 router.
    2. In **Configuration** > **Router Ports**, click **+ADD** to add a new router port.
    3. For **Logical Switch**, enter the name of the logical switch you defined for the network in **Add New Logical Switch**, above.
    4. For **IP Address**, use the first IP of the appropriate subnet. In this example, 192.168.1.0/24 is set aside for Infrastructure (Ops Manager and BOSH Director), and 192.168.2.0/24 for the Deployment, so 192.168.1.1 and 192.168.2.1 are used

respectively.



New Router Port

Name*: PAS-Infrastructure-dp

Description: Downlink port connecting PAS-Infrastructure router to its Logical Switch

Type: Downlink

URPF Mode: ● Strict ○ None

Logical Switch:

OR Create a New Switch

Logical Switch Port: ● Attach to new switch port ○ Attach to existing switch port

Switch Port Name: PAS-Infrastructure-lp

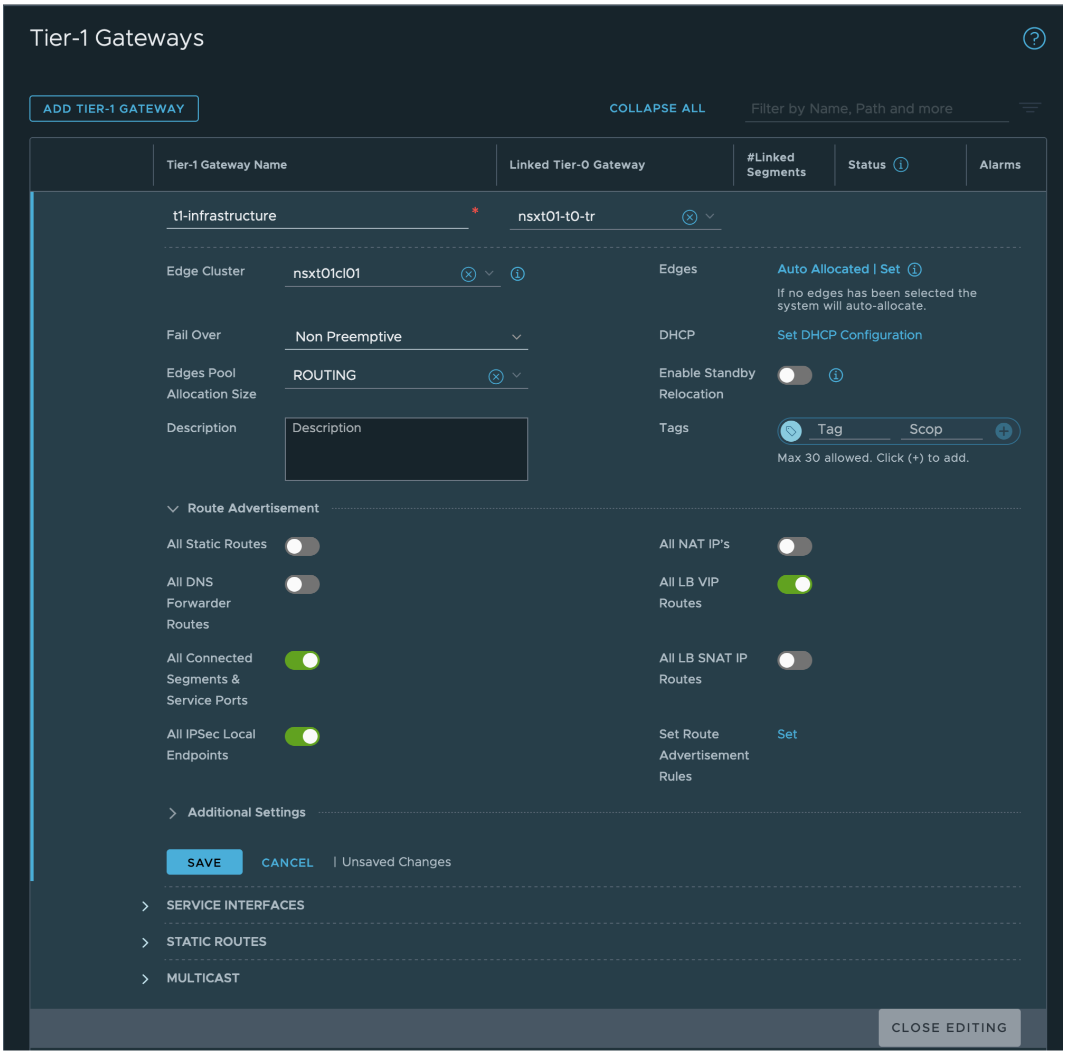
Subnets

+ ADD DELETE

☑	IP Address*	Prefix Length*
☑	192.168.1.1	24

Relay Service:

4. Advertise the routes of the T1 routers to the T0 router, so the T0 router can correctly route incoming requests based on their destination IP address:
    1. Select your T1 Router and navigate to **Routing**, then **Route Advertisement**.
    2. Under **Edit Route Advertisement Configuration**, enable route advertisement by setting **Status** to **Enabled**.
    3. Set **Advertise All Connected Routes** to **Yes**.
    4. Set **Advertise All LB VIP Routes** to **Yes** (necessary if Load Balancing service is configured).



Tier-1 Gateway Name: t1-infrastructure

Linked Tier-0 Gateway: nsxt01-t0-tr

Edge Cluster: nsxt01cl01

Edges: Auto Allocated | Set

If no edges has been selected the system will auto-allocate.

Fail Over: Non Preemptive

DHCP: Set DHCP Configuration

Edges Pool Allocation Size: ROUTING

Enable Standby Relocation: off

Description: Description

Tags: Tag Scop

Max 30 allowed. Click (+) to add.

All Static Routes: off

All NAT IP's: off

All DNS Forwarder Routes: off

All LB VIP Routes: on

All Connected Segments & Service Ports: on

All LB SNAT IP Routes: off

All IPSec Local Endpoints: on

Set Route Advertisement Rules: Set

5. Allocate an IP block for TAS for VMs orgs.

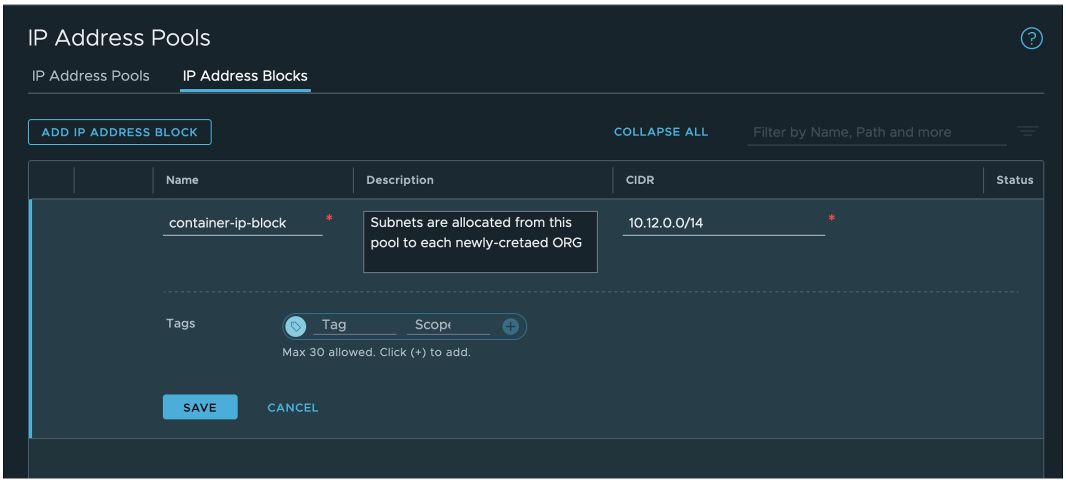
   1. From the NSX-T Manager, navigate to **Advanced Networking & Security** > **Networking** > **IPAM** and click **+ADD**.

   2. Enter a name (for example, `TAS for VMs-container-ip-block`). This IP block name is also used in the **VMware NSX-T** tile in the **NCP** section under **IP Blocks of Container Networks**.

   3. Enter a description, such as `Subnets are allocated from this pool to each newly-created org`.

   4. Enter a CIDR to allocate an address block large enough to accommodate all TAS for VMs apps. A `/14` CIDR is large enough for ~1,000 Orgs with ~250 apps each. If you are planning such a large foundation, see VMware NSX-T TAS for VMs limits in the

VMware documentation.



Name: container-ip-block

Description: Subnets are allocated from this pool to each newly-cretaed ORG

CIDR: 10.12.0.0/14

Configure Load Balancer

To configure a load balancer:

1. Create Active Health Monitors (health checks) for use by the virtual server later on:
    1. In the NSX-T Manager UI, navigate to **Advanced Networking & Security** > **Networking** > **Load Balancing** > **Monitors** > **Active Health Monitors**.
    2. Create the health monitor for web load balancing:
    3. Click **+ADD**.
    4. Enter **Monitor Properties**:
        - **Name**: `pas-web-monitor`
        - **Health Check Protocol**: `LbHttpMonitor`
        - **Monitoring Port**: 8080
    5. Click **Next**.
    6. Enter **Health Check Parameters**:
        - **HTTP Method**: `GET`
        - **HTTP Request URL**: `/health`
        - **HTTP Response Code**: 200
    7. Click **Finish**.
    8. Create the health monitor for TCP load balancing:
    9. Click **+ADD**.
    10. Enter **Monitor Properties**:
        - **Name**: `pas-tcp-monitor`
        - **Health Check Protocol**: `LbHttpMonitor`
        - **Monitoring Port**: 80
    11. Click **Next**.
    12. Enter **Health Check Parameters**:

       - **HTTP Method**: `GET`
       - **HTTP Request URL**: `/health`
       - **HTTP Response Code**: 200
   13. Create the health monitor for SSH load balancing:
   14. Click **+ADD**.
   15. Enter **Monitor Properties**:
       - **Name**: `pas-ssh-monitor`
       - **Health Check Protocol**: `LbTcpMonitor`
       - **Monitoring Port**: 2222
   16. Click **Next**, then click **Finish**.

2. Create server pools (collections of VMs which handle traffic) for use by the virtual server:
   1. In the NSX-T Manager UI, navigate to **Advanced Networking & Security** > **Networking** > **Load Balancing** > **Server Pools**.
   2. Create the server pool for web load balancing:
   3. Click **+ADD** to add a new pool.
   4. Enter **General Properties**:
      - **Name**: `pas-web-pool`
   5. Click **Next**.
   6. Enter **SNAT Translation**:
      - **Translation Mode**: `IP List`
      - Enter a range of available IPs for SNAT translation. By default, ports from `4000` to `64000` are for all configured SNAT IP addresses. Allocate enough IPs to handle your traffic load. Without enough IPs, the SNAT port is exhausted.
   7. Click **Next**.
   8. Enter **Pool Members**:
      - **Membership Type**: `Static`
   9. Click **Next**.
   10. Enter **Health Monitors**:
       - **Active Health Monitor**: `pas-web-monitor`
   11. Click **Finish**.
   12. Create the server pool for TCP load balancing:
   13. Click **+ADD** to add new pool.
   14. Enter **General Properties**:
       - **Name**: `pas-tcp-pool`
   15. Click **Next**.
   16. Enter **SNAT Translation**:

- **Translation Mode**: `Transparent`

17. Click **Next**.
18. Enter **Pool Members**:
    - **Membership Type**: `Static`
19. Click **Next**.
20. Enter **Health Monitors**:
    - **Active Health Monitor**: `pas-tcp-monitor`
21. Click **Finish**.
22. Create the server pool for SSH load balancing:
23. Click **+ADD** to add new pool.
24. Enter **General Properties**:
    - **Name**: `pas-ssh-pool`
25. Click **Next**.
26. Enter **SNAT Translation**:
    - **Translation Mode**: `Transparent`
27. Click **Next**.
28. Enter **Pool Members**:
    - **Membership Type**: `Static`
29. Click **Next**.
30. Enter **Health Monitors**:
    - **Active Health Monitor**: `pas-ssh-monitor`
31. Click **Finish**.

Create virtual servers.

1. In the NSX-T Manager UI, navigate to **Advanced Networking & Security** > **Networking** > **Load Balancing** > **Virtual Servers**.
2. Create the virtual server which forwards unencrypted web (HTTP) traffic to the foundation:

> **Note:** Foundations requiring end-to-end encryption should not enable the virtual server on port 80, or, if enabled, should configure it to redirect traffic to the encrypted port (443).

1. Click **+Add**.
2. Enter **General Properties**:
    - **Name**: `pas-web-vs`
    - **Application Type**: `Layer 4 (TCP)`
    - **Application Profile**: `nsx-default-lb-fast-tcp-profile`
3. Click **Next**.

4. Enter **Virtual Server Identifiers**:
    - **IP Address**: use the address of the DNS record of `*.system.YOUR-SYSTEM-DOMAIN.com`
    - **Port**: `80,443`
5. Enter **Server Pool and Rules**:
    - **Default Server Pool**: `pas-web-pool`
6. Click **Next** several times, then click **Finish**.

Create the virtual server which forwards traffic to apps with custom ports to the foundation.

1. Click **+Add** to add a new virtual server.
2. Enter **General Properties**:
    - **Name**: `pas-tcp-vs`
    - **Application Type**: `Layer 4 (TCP)`
    - **Application Profile**: `nsx-default-lb-fast-tcp-profile`
    - Click **Next**.
    - Enter **Virtual Server Identifiers**:
    - **IP Address**: use the address of the DNS record of `tcp.apps.YOUR-SYSTEM-DOMAIN.com`
    - **Port**: use the same ports as configured in the **TAS for VMs Tile** > **Networking** > **TCP Routing Ports**, e.g. `1024-1123,5900`
    - Click **Next**.
    - Enter **Server Pool and Rules**:
        - **Default Server Pool**: `pas-tcp-pool`
    - Click **Next**, then click **Finish**.

Create the virtual server which forwards SSH traffic to the foundation:

1. Click **+Add** to add a new virtual server.
2. Enter **General Properties**:
    - **Name**: `pas-ssh-vs`
    - **Application Type**: `Layer 4 (TCP)`
    - **Application Profile**: `nsx-default-lb-fast-tcp-profile`
    - Click **Next**.
    - Enter **Virtual Server Identifiers**:
    - **IP Address**: use the address of the DNS record of `ssh.system.YOUR-SYSTEM-DOMAIN.com`
    - **Port**: `2222`
    - Click **Next**.
    - Enter **Server Pool and Rules**:

    - **Default Server Pool**: `pas-ssh-pool`
  - Click **Next**, then click **Finish**.

Create the load balancer.

1. In the NSX-T Manager UI, navigate to **Advanced Networking & Security** > **Networking** > **Load Balancing** > **Load Balancers**.
   1. Click **+Add**.
   2. Enter the fields:
      - **Name**: `pas-lb`
      - **Load Balancer Size**: Choose `Small` unless you have a larger Foundation
   3. Click **OK**.
   4. Select `pas-lb`.
   5. Click **Actions** > **Attach to a Virtual Server**, and then select `pas-web-vs`. Repeat this procedure for the Virtual Servers `pas-tcp-vs` and `pas-ssh-vs`.
   6. Click **Action** > **Attach to a Logical Router**, and then select `T1-Router-TAS for VMs-Deployment`.

Enable NSX-T Mode in the BOSH Director

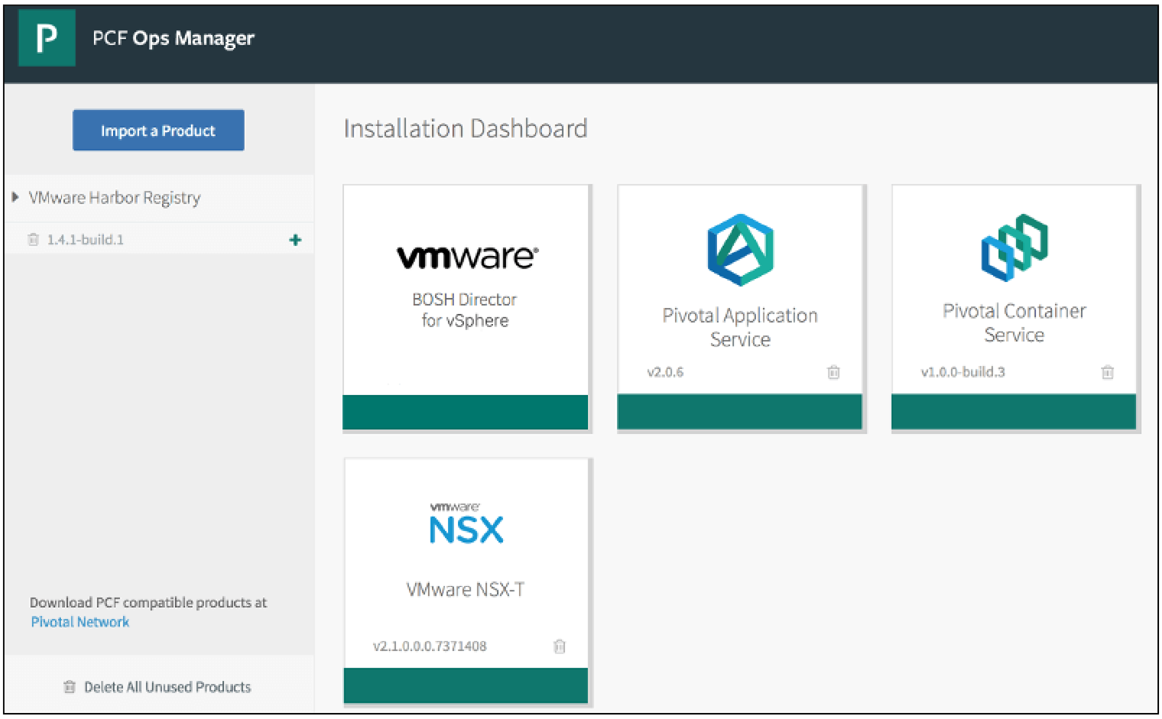
To enable NSX-T mode in the BOSH Director:

1. From the Ops Manager **Installation Dashboard**, open the **BOSH Director** tile.
2. In the **vCenter Configs** pane, click the pencil icon for the vCenter Config you want to edit.
3. Select **NSX Networking** below.
4. Configure BOSH Director authentication to the NSX Manager by following the **NSX Networking** instructions in the Step 2: Configure vCenter section of *Configuring BOSH Director on vSphere*.

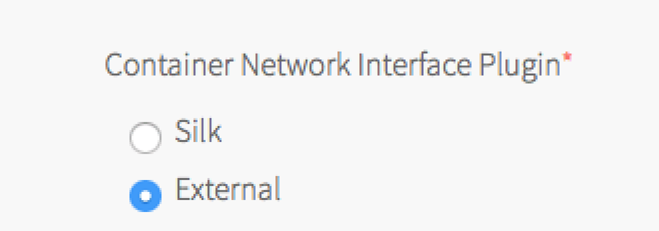
Configure TAS for VMs for External Container Networking

To configure TAS for VMs for external container networking:

1. If you have not already done so, download the TAS for VMs tile from VMware Tanzu Network and import it to the **Installation Dashboard**. For instructions, see Adding and Importing Products.



2. Configure TAS for VMs, following the directions in Configuring TAS for VMs. When you configure **Networking**, select **External** under **Container networking interface plugin**.



Container Network Interface Plugin*

○ Silk

● External

3. Update the server pool membership for the NSX-T load balancers:

   1. Open the **BOSH Director for vSphere** tile > **Resource Config** pane.

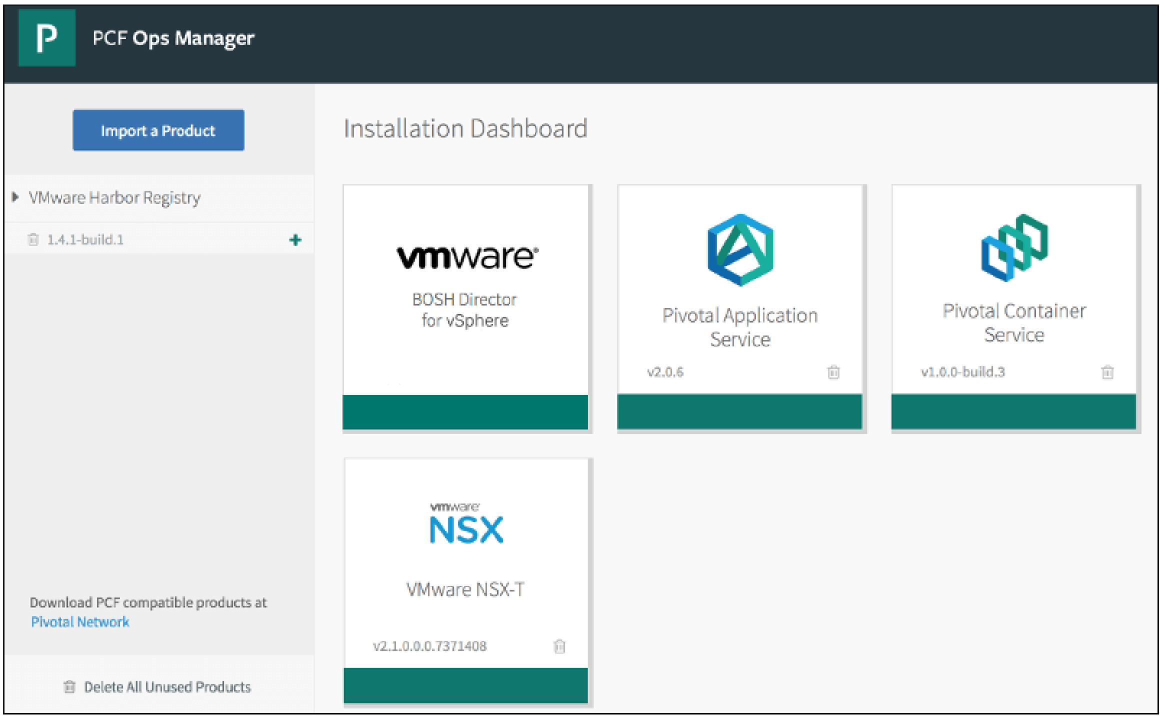
   2. Click the arrow next to each job to reveal the **NSX-T CONFIGURATION** column. See Step 10: Resource Config Pane in *Configuring BOSH Director on vSphere*.

   3. Under **Logical Load Balancer**, enter a JSON-formatted structure to defining a list of `server_pools` a VM extension for each of the three server pools: `pas-web-pool`, `pas-tcp-pool`, and `pas-ssh-pool`.

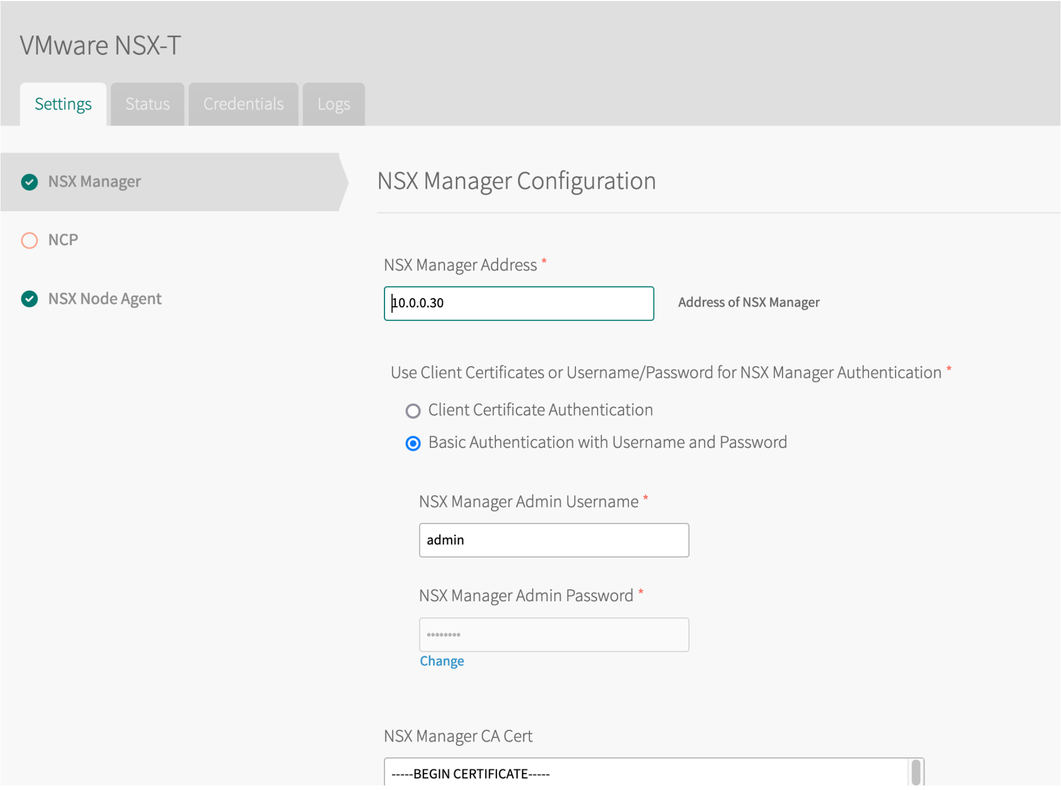
Install and Configure the NSX-T Container Plug-In

To install and configure the NSX-T Container Plug-In tile:

1. If you have not already done so, download the VMware NSX-T Container Plug-in tile from VMware Tanzu Network and import it to the **Installation Dashboard**. For instructions, see Adding and Importing Products.


PCF Ops Manager
Import a Product
VMware Harbor Registry
1.4.1-build.1
Download PCF compatible products at Pivotal Network
Delete All Unused Products
Installation Dashboard
vmware
BOSH Director for vSphere
Pivotal Application Service
v2.0.6
Pivotal Container Service
v1.0.0-build.3
vmware NSX
VMware NSX-T
v2.1.0.0.0.7371408

2. Click the **VMware NSX-T** tile to open its **Settings** tab, and configure the **NSX Manager** pane as follows:
   - **NSX Manager Address**: The NSX-T Manager host address or IP address.
   - **Use Client Certificates or Username/Password**: Configure this setting as follows:
     1. If you are using VMware Workspace ONE Access, formerly called VMware Identity Manager (vIDM), then select **Client Certificate Authentication**.
     2. Otherwise, select **Basic Authentication with Username and Password** and enter **NSX Manager Admin Username** and **Admin Password** credentials in the fields underneath.
   - **NSX Manager CA Cert**: Obtain this certificate from NSX-T Manager as follows:
     1. SSH into NSX-T Manager using the admin account that you created when you deployed NSX-T Manager.
     2. From the NSX-T Manager command line, run `get certificate api` to retrieve the certificate.



NSX Manager Configuration

NSX Manager Address *

10.0.0.30

Address of NSX Manager

Use Client Certificates or Username/Password for NSX Manager Authentication *

○ Client Certificate Authentication

● Basic Authentication with Username and Password

NSX Manager Admin Username *

admin

NSX Manager Admin Password *

••••••••

Change

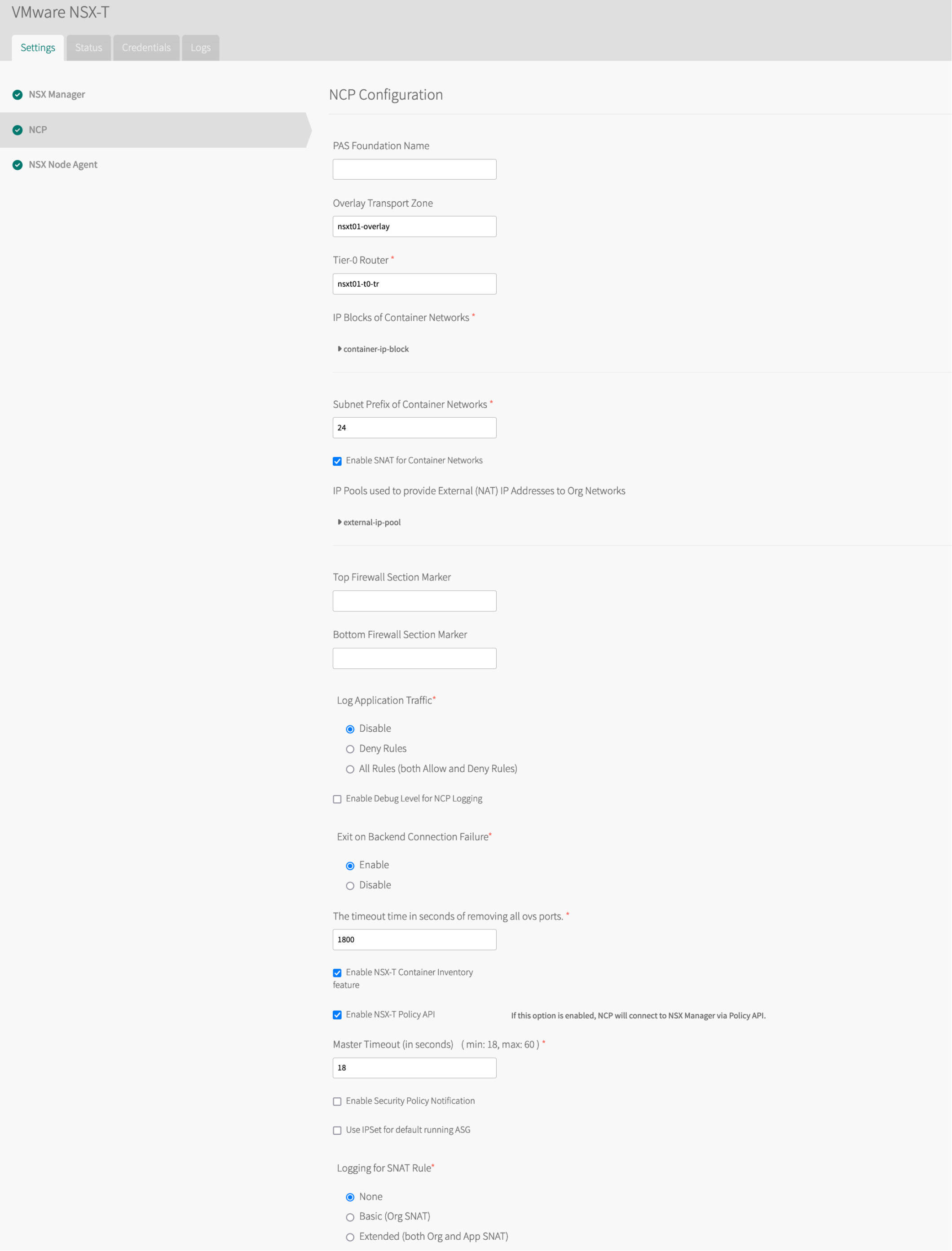
NSX Manager CA Cert

-----BEGIN CERTIFICATE-----

3. Open and configure the **NCP** (NSX-T Container Plugin) pane as follows:

   - **TAS for VMs Foundation Name**: If unsure, use `TAS for VMs`. If multiple foundations co-exist on the same NSX-T Manager, choose a unique string, such as `TAS for VMs-beta`. NCP creates artifacts, such as T1 routers and prefixes their names with this string for easy identification.
   - **Overlay Transport Zone**: A uniquely identifying string for the **Transport Zone** that you chose when you created logical switches for each network. This can be the name of the transport zone if no other zones in NSX-T share the same name, or else the UUID for the transport zone.
   - **Tier-0 Router**: A uniquely identifying string for the T0 router. This can be the tag string that you gave the router in NSX-T Manager if no other T0 routers in NSX-T share the same name, or else the UUID for the router.
   - **Subnet Prefix of Container Networks**: Subnet mask to set the address range size for apps in a single org. Defaults to `24`. This number must be higher than the mask for all TAS for VMs orgs in the NSX-T Manager **New IP Block** pane, to define each org's fraction of the total TAS for VMs address space.

- **Enable SNAT for Container Network**: Enable this checkbox.



VMware NSX-T

NCP Configuration

PAS Foundation Name

Overlay Transport Zone: nsxt01-overlay

Tier-0 Router *: nsxt01-t0-tr

IP Blocks of Container Networks *

▸ container-ip-block

Subnet Prefix of Container Networks *: 24

☑ Enable SNAT for Container Networks

IP Pools used to provide External (NAT) IP Addresses to Org Networks

▸ external-ip-pool

Top Firewall Section Marker

Bottom Firewall Section Marker

Log Application Traffic*

● Disable
○ Deny Rules
○ All Rules (both Allow and Deny Rules)

□ Enable Debug Level for NCP Logging

Exit on Backend Connection Failure*

● Enable
○ Disable

The timeout time in seconds of removing all ovs ports. *: 1800

☑ Enable NSX-T Container Inventory feature

☑ Enable NSX-T Policy API — If this option is enabled, NCP will connect to NSX Manager via Policy API.

Master Timeout (in seconds) (min: 18, max: 60) *: 18

□ Enable Security Policy Notification

□ Use IPSet for default running ASG

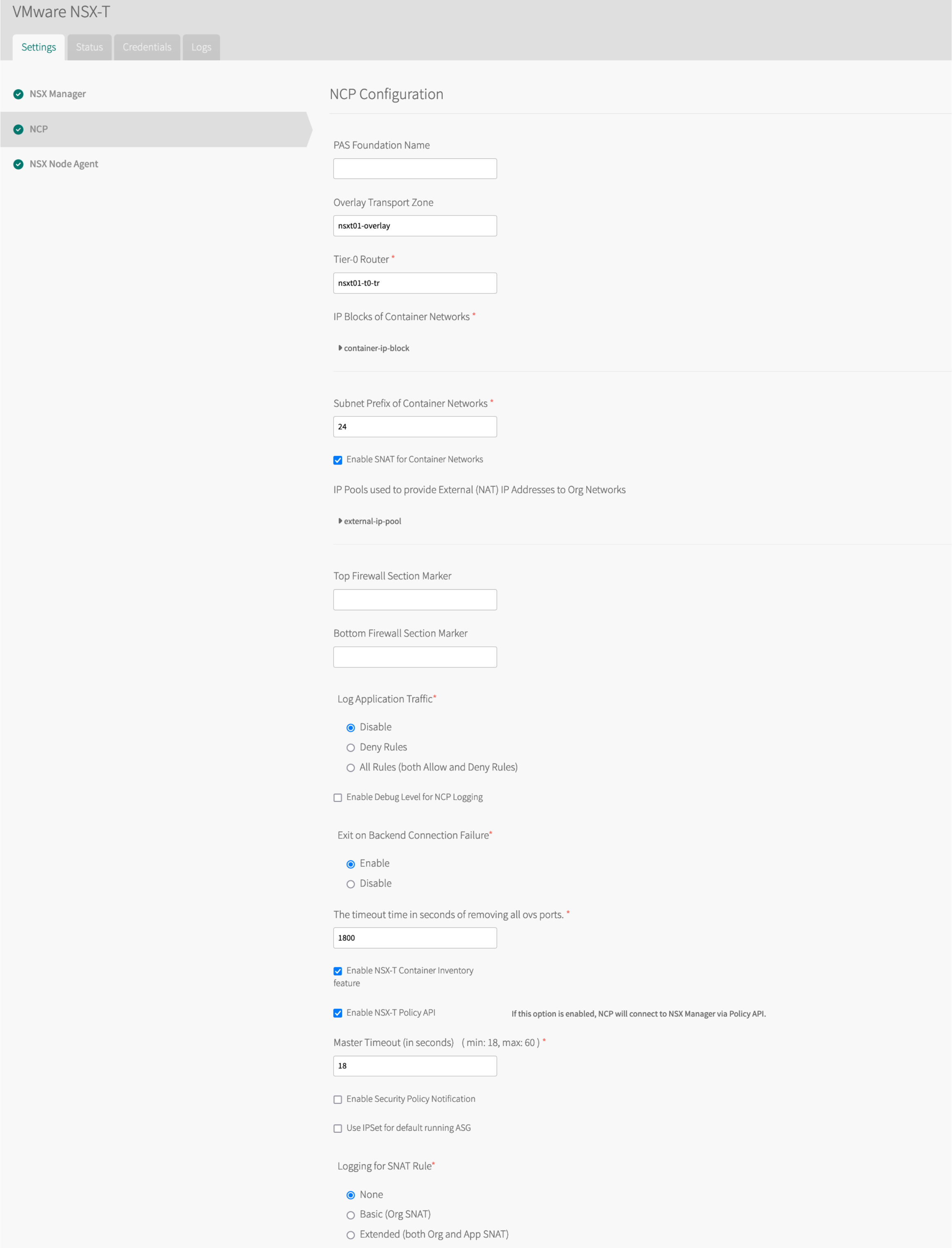
Logging for SNAT Rule*

● None
○ Basic (Org SNAT)
○ Extended (both Org and App SNAT)

- **TAS for VMs Foundation Name**: If unsure, use `TAS for VMs`. If multiple foundations co-exist on the same NSX-T Manager, choose a unique string, such as `TAS for VMs-beta`. NCP creates artifacts, such as T1 routers and prefixes their names with this string for easy identification.
- **Overlay Transport Zone**: A uniquely identifying string for the **Transport Zone** that you chose when you created logical switches for each network. This can be the name of the transport zone if no other zones in NSX-T share the same name, or else the UUID for the transport zone.
- **Tier-0 Router**: A uniquely identifying string for the T0 router. This can be the tag string that you gave the router in NSX-T Manager if no other T0 routers in NSX-T

share the same name, or else the UUID for the router.

- **Subnet Prefix of Container Networks**: Subnet mask to set the address range size for apps in a single org. Defaults to `24`. This number must be higher than the mask for all TAS for VMs orgs in the NSX-T Manager **New IP Block** pane, to define each org's fraction of the total TAS for VMs address space.
- **Enable SNAT for Container Network**: Select this check box.



NCP Configuration

PAS Foundation Name

Overlay Transport Zone: nsxt01-overlay

Tier-0 Router *: nsxt01-t0-tr

IP Blocks of Container Networks *

▸ container-ip-block

Subnet Prefix of Container Networks *: 24

☑ Enable SNAT for Container Networks

IP Pools used to provide External (NAT) IP Addresses to Org Networks

▸ external-ip-pool

Top Firewall Section Marker

Bottom Firewall Section Marker

Log Application Traffic*

● Disable
○ Deny Rules
○ All Rules (both Allow and Deny Rules)

□ Enable Debug Level for NCP Logging

Exit on Backend Connection Failure*

● Enable
○ Disable

The timeout time in seconds of removing all ovs ports. *: 1800

☑ Enable NSX-T Container Inventory feature

☑ Enable NSX-T Policy API — If this option is enabled, NCP will connect to NSX Manager via Policy API.

Master Timeout (in seconds) (min: 18, max: 60) *: 18

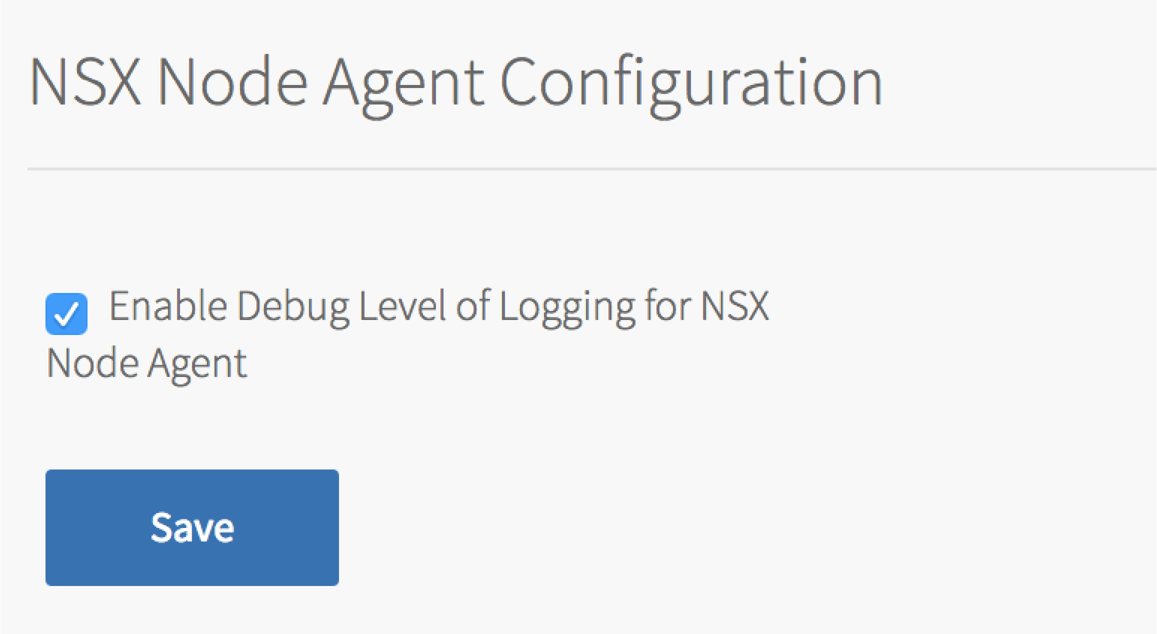
□ Enable Security Policy Notification

□ Use IPSet for default running ASG

Logging for SNAT Rule*

● None
○ Basic (Org SNAT)
○ Extended (both Org and App SNAT)

4. In the **NSX Node Agent** pane, select the **Enable Debug Level of Logging for NSX Node Agent** check box.



NSX Node Agent Configuration

☑ Enable Debug Level of Logging for NSX Node Agent

5. Click **Save** and return to the **Installation Dashboard**.
6. After you have configured both the **TAS for VMs** tile and the **VMware NSX-T** tile, click **Review Pending Changes**.
7. Click **Apply Changes** to deploy TAS for VMs with NSX-T networking.

Deploying TAS for VMs to VCF

This topic describes how to install VMware Tanzu Application Service for VMs (TAS for VMs) to VMware Cloud Foundation (VCF).

Overview

VCF is the hybrid cloud platform for managing VMs and orchestrating containers. You can install and configure Ops Manager and VMware Tanzu Application Service for VMs (TAS for VMs) after launching the vSphere client through VCF.

Prerequisites

Before you deploy TAS for VMs on VCF:

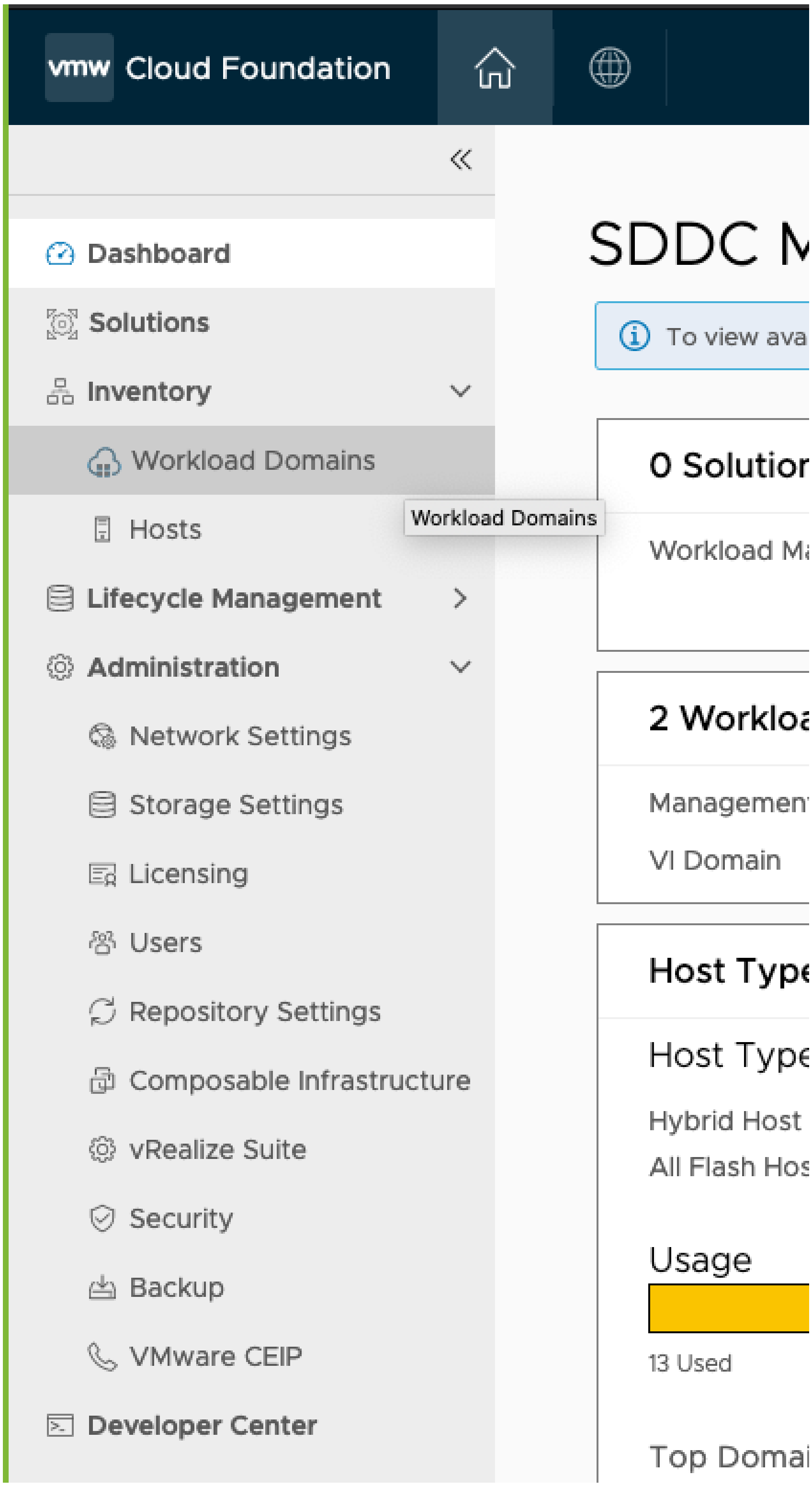
- Provision a VI workload domain on a Software Defined Data Center (SDDC). See About VI Workload Domains.
- Download Ops Manager for vSphere from VMware Tanzu Network.
- Download TAS for VMs from VMware Tanzu Network.
- Install the govc CLI to your local workstation or jump host. See govmomi on GitHub.

Prepare to Deploy TAS for VMs to VCF

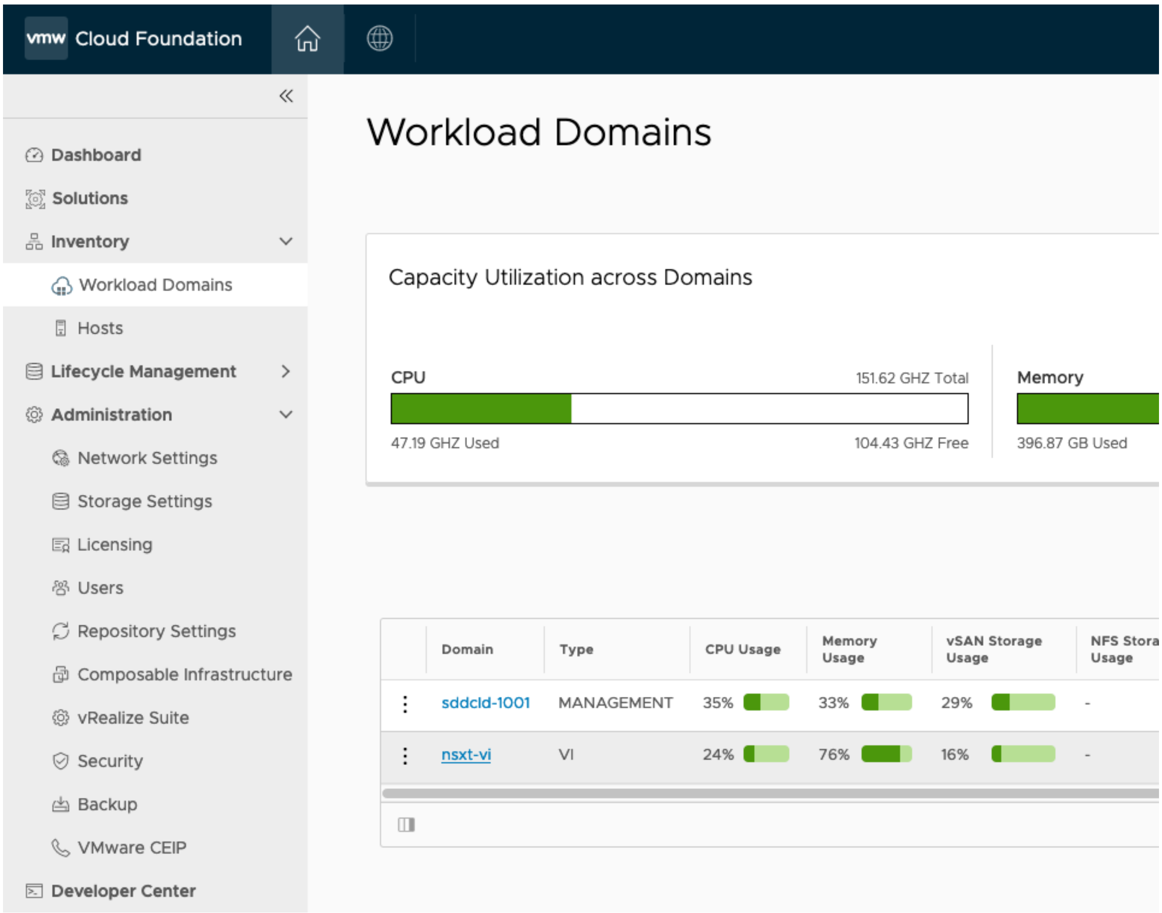
To prepare VCF before you install TAS for VMs:

Collect your vCenter information:

1. In SDDC Manager, select **Workload Domains**.



2. Select the VI domain where you want to install Ops Manager and TAS for VMs.



Workload Domains

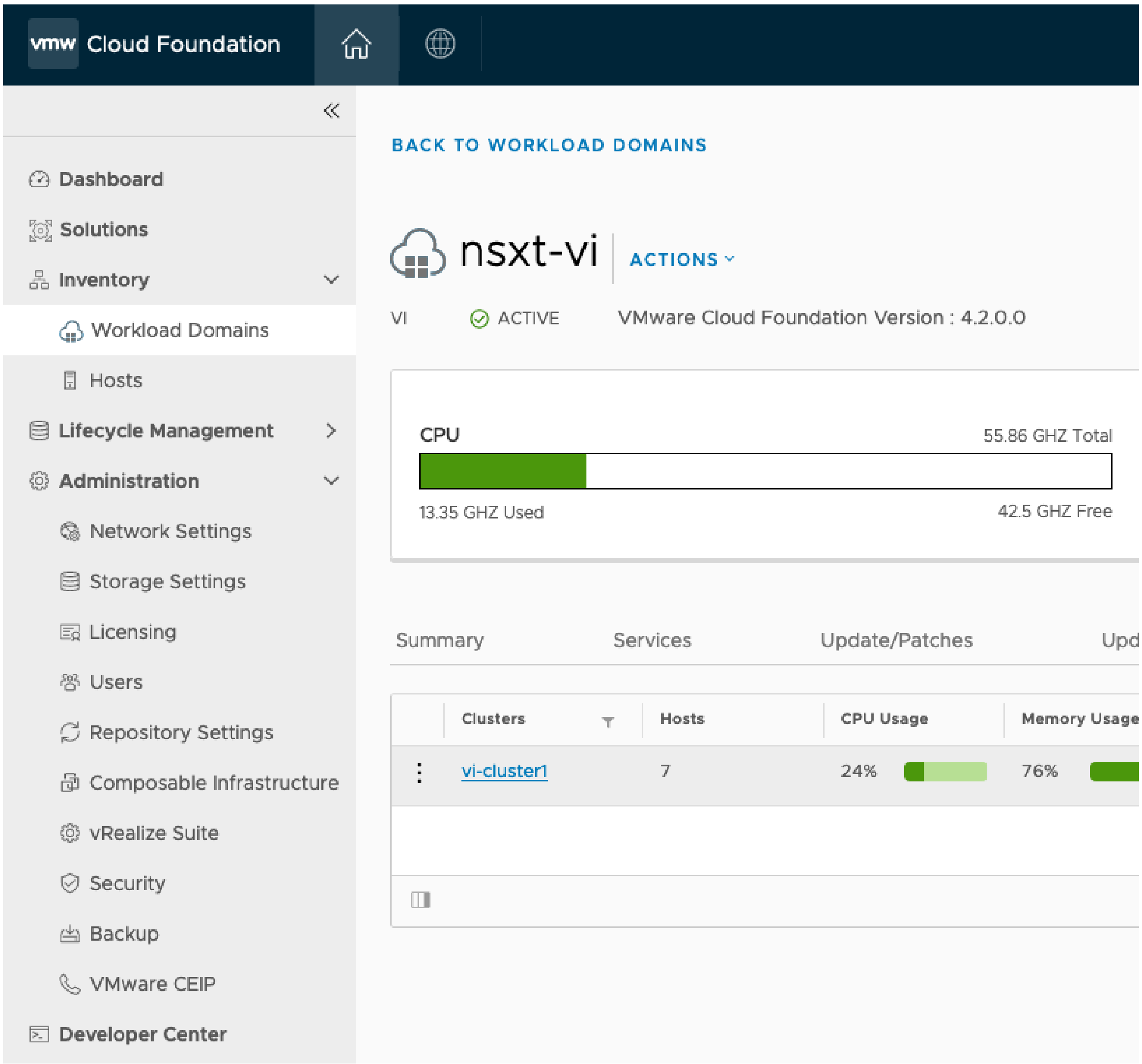
Capacity Utilization across Domains

CPU 151.62 GHZ Total
47.19 GHZ Used 104.43 GHZ Free

Memory
396.87 GB Used

	Domain	Type	CPU Usage	Memory Usage	vSAN Storage Usage	NFS Stora Usage
⋮	sddcld-1001	MANAGEMENT	35%	33%	29%	-
⋮	nsxt-vi	VI	24%	76%	16%	-

3. Select the cluster where you want to install Ops Manager and TAS for VMs.



nsxt-vi

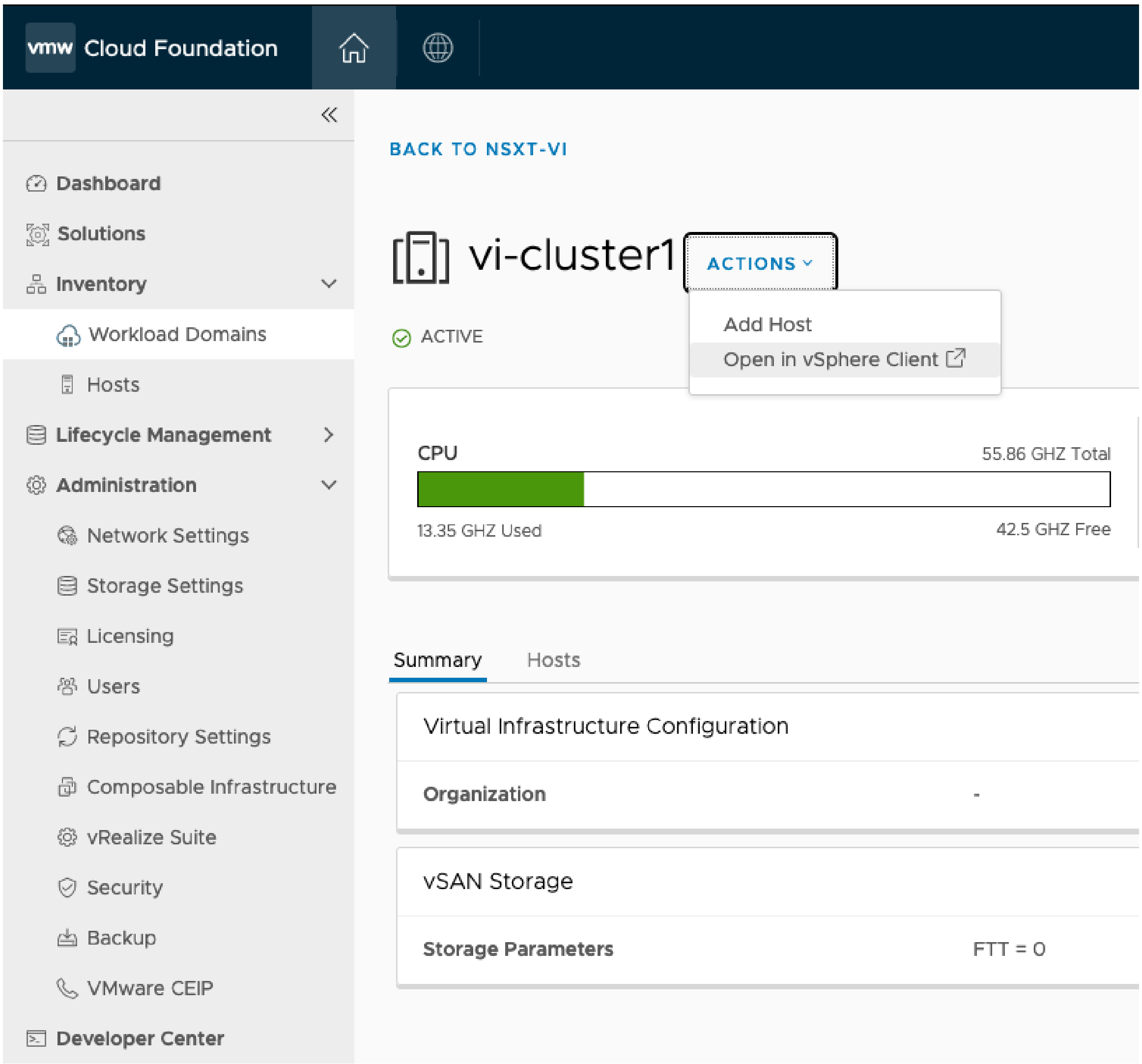
VI ACTIVE VMware Cloud Foundation Version : 4.2.0.0

CPU 55.86 GHZ Total

13.35 GHZ Used 42.5 GHZ Free

Clusters	Hosts	CPU Usage	Memory Usage
vi-cluster1	7	24%	76%

4. From the **Actions** menu, select **Open in vSphere Client**.



vi-cluster1

ACTIVE

CPU 55.86 GHZ Total

13.35 GHZ Used 42.5 GHZ Free

Virtual Infrastructure Configuration

Organization -

vSAN Storage

Storage Parameters FTT = 0

If you are using NSX-T, select **Workload Domains** and select the **Summary** tab. Record the IP address listed for **NSX Manager IP Address**.

Next Step

After you complete the procedures in this topic, continue to Installing Ops Manager on vSphere.

After you configure Ops Manager, you can install and configure TAS for VMs.

Deploying TAS for VMs to VMC

This topic tells you how to install VMware Tanzu Application Service for VMs (TAS for VMs) to VMware Cloud (VMC) on Amazon Web Services (AWS).

Overview

VMC provides Software Defined Data Centers (SDDCs) that run on AWS with a high-level management console to configure networking rules. This topic tells you the procedures you need for configuring and installing TAS for VMs on VMC using public IP addresses for ingress.

To install and configure TAS for VMs on VMC:

1. Install Ops Manager
2. Configure BOSH Director
3. Install and Configure TAS for VMs

Prerequisites

Before you install and configure TAS for VMs on VMC:

- Create a VMC account. See VMware Cloud.
- Deploy an SDDC. See Deploy an SDDC from the VMC Console.
- Enable browser access to vCenter from your SDDC. See Enable Browser Access to vCenter.
- Download Ops Manager for vSphere from VMware Tanzu Network.
- Download TAS for VMs from VMware Tanzu Network.
- Install the govc CLI to your local workstation or jump host. See govmomi on GitHub.

(Optional) If you plan to connect to the VMC SDDC using a VPN or Direct Connect Virtual Interfaces (VIFs):

- To use a VPN to connect your network to the VMC SDDC, configure the VPN connection before you install TAS for VMs. See Configure a VPN Connection Between Your SDDC and On-Premises Data Center.
- To use VIFs to connect your network to the VMC SDDC, configure the VIFs before you install TAS for VMs. See Configure AWS Direct Connect Between Your SDDC and On-Premises Data Center.

Enable Browser Access to vCenter

By default, ingress to vCenter is not enabled. If you are not using a VPN or other network, you must create a firewall rule in SDDC to allow vCenter access from your workstation's public IP address.

To enable browser access to vCenter:

1. Go to the VMC console.
2. Select **SDDCs**.
3. Click **View Details** on your datacenter tile.
4. Select **Networking & Security**, and then select **Security**.
5. Click **Gateway Firewall**.
6. Click **Management Gateway**.
7. Click **+ Add Rule**.
    - **Name**: Enter `vCenter Browser Inbound Rule`.
    - **Sources**:
        1. Click **User Defined Groups**.

      2. Click **Add Group**.
          - For **Group name**, enter `public ip address`.
          - Click **Set Members**.
          - Select **IP Addresses**.
          - Enter your public IP address. For example, `66.170.99.1`.

            > **Note**: You can locate your public IP address by navigating to `https://ifconfig.me/`.

          - Press **Enter**.
          - Click **Apply**.
      3. Click **Save**.
   - **Destinations**: Select **vCenter** and then click **Apply**.
   - **Services**: Enter `HTTPS, ICMP, SSO`.

8. Click **Publish**.

Record vCenter Credentials

To gather login credentials for the vCenter instance in your SDDC:

1. Go to the VMC console.
2. Select **SDDCs**.
3. Click **Open vCenter** on your datacenter tile.
4. A dialog box appears with the text: **Before you can log into vCenter, you must open network access to vCenter through the management gateway.** Select an option for opening network access.
5. Click **Show Credentials**.
6. Record the credentials.

After you enable browser access and record your vCenter credentials, you can access and authenticate with your cluster's vCenter server.

Install Ops Manager

To install Ops Manager on VMC:

1. Find the VMC pre-installed network to use for the BOSH infrastructure network:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**.
    5. Select **Network**, and then select **Segments**.

6. Record the segment name for **sddc-cgw-nework-1**.

7. Record the subnet **192.168.1.1/24**. SDDC combines the gateway IP address and CIDR into a single value.

2. If nothing has changed, create a file named `options.json` with the following contents:

```
{
  "NetworkMapping": [
    {
      "Name": "Network 1",
      "Network": "sddc-cgw-network-1"
    }
  ],
  "PropertyMapping": [
    {
      "Key": "ip0",
      "Value": "192.168.1.10"
    },
    {
      "Key": "netmask0",
      "Value": "255.255.255.0"
    },
    {
      "Key": "gateway",
      "Value": "192.168.1.1"
    },
    {
      "Key": "DNS",
      "Value": "8.8.8.8"
    },
    {
      "Key": "ntp_servers",
      "Value": "time.google.com"
    }
  ]
}
```

3. Replace the following example text in the code below, then run the commands to upload the Ops Manager file to VMC:

   - `EXAMPLE-PASSWORD`: the vCenter password you recorded in Record vCenter Credentials.
   - `www.example.com`: your vCenter URL.
   - `PATH-TO-OPS-MANAGER`: the path to your Ops Manager OVA file.

```
export GOVC_DATACENTER=SDDC-Datacenter
export GOVC_DATASTORE=WorkloadDatastore
export GOVC_RESOURCE_POOL=Compute-ResourcePool

export GOVC_URL='cloudadmin@vmc.local':'EXAMPLE-PASSWORD©'@www.example.com

govc library.create tas
govc library.import tas PATH-TO-OPS-MANAGER
govc library.deploy -options options.json /tas/ops-manager-vsphere-2.##.#-buil
```

```
d.### ops-manager
govc pool.create /SDDC-Datacenter/host/Cluster-1/Resources/az{1,2,3}
```

> **Note:** VMware recommends using `govc library.*` commands instead of `govc import.ova`.
>
> The `govc import.ova` commands depend on access to the ESXi hosts, which is not configured by default in VMC. Using these commands can cause errors like the following: `govc: Post "https://10.2.32.4/nfc/5224a51f-114e-4627-8ca8-547c2e2e9488/disk-0.vmdk": dial tcp 10.2.32.4:443: i/o timeout`

4. Log in to vCenter:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **Open vCenter** on your datacenter tile.
    4. Select **Show Credentials**, and then click **Open vCenter**.
    5. Find the Ops Manager VM.
    6. Open the **Hosts & Clusters** view.
    7. Open the `Compute-ResourcePool` and select the `ops-manager` VM.
    8. Click ▶ to power on the Ops Manager VM.
5. Create public IP addresses for Ops Manager and HAProxy:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**.
    5. Select **Public IPs**, and then select **Request New IP**.
        1. For **Notes**, enter `ops-manager` and click **Save**.
        2. Record the IP address. For example, `54.190.190.190`.
    6. Click **Request New IP**.
        1. For **Notes**, enter `HAProxy` and click **Save**.
        2. Record the IP address. For example, `54.180.180.180`.
6. Create a second network segment to use as the BOSH deployment network:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**.
    5. Select **Network**, and then select **Segments**.

    6. Click **Add Segment**:
        1. For **Segment Name**, enter `bosh-network`.
        2. For **Type**, enter `routed`.
        3. For **Subnets**, enter a subnet. For example, `192.168.2.1/24`.
7. Assign public IP addresses to Ops Manager and HAProxy:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**.
    5. Select **NAT**, and then click **Add NAT Rule**.
        1. For **Rule Name**, enter `ops-manager`.
        2. For **Public IP**, enter the IP address you created for Ops Manager in a previous step. For example, `54.190.190.190`.
        3. For **Internal IP**, enter `192.168.1.10`.
        4. Click **Save**.
    6. Select **Add NAT Rule**.
        1. For **Rule Name**, enter `HAProxy`.
        2. For **Public IP**, enter the IP address you created for HAProxy in a previous step. For example, `54.180.180.180`.
        3. For **Internal IP**, enter `192.168.2.2`.
        4. Click **Save**.
8. Add firewall rules that allow ingress to Ops Manager and HAProxy:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**, and then select **Security**.
    5. Select **Gateway Firewall**, and then **Compute Gateway**.
    6. Click **Add Rule**.
    7. Select the newly-created rule:
        1. For **Rule Name**, enter `opsman-ingress`.
        2. For **Sources**, select **Any**.
        3. Configure **Destinations**:
            - Click **Add Group**.
                - For **Group name**, enter `OM`.
                - Click **Set Members**.
                - Select **IP Addresses**.

            - Enter the Ops Manager internal IP address, `192.168.1.10`.
            - Press **Enter**.
            - Click **Apply**.
         - Click **Save**.
         - Click **Apply**.
         - Click **Publish**.
   8. Click **Add Rule**.
   9. Select the newly-created rule:
      1. For **Rule Name**, enter `HAProxy-ingress`.
      2. For **Sources**, select **Any**.
      3. Configure **Destinations**:
         - Click **Add Group**.
            - For **Group name**, enter `HAProxy`.
            - Click **Set Members**.
            - Select **IP Addresses**.
            - Enter the HAProxy internal IP address, `192.168.2.2`.
            - Press **Enter**.
            - Click **Apply**.
         - Click **Save**.
         - Click **Apply**.
         - Click **Publish**.

9. Add a firewall rule that allows egress for the 192.168.1.x and 192.168.2.x subnets:
   1. Go to the VMC console.
   2. Select **SDDCs**.
   3. Click **View Details** on your datacenter tile.
   4. Select **Networking & Security**, and then select **Security**.
   5. Select **Gateway Firewall**, and then select **Compute Gateway**.
   6. Click **Add Rule**.
   7. Select the newly-created rule:
      1. For **Rule Name**, enter `tas-egress`.
      2. Edit **Sources**.
         - Click **Add Group**.
            - For **Group name**, enter `tas`.
            - Click **Set Members**.
            - Select **IP Addresses**.

            - Enter the first subnet CIDR, `192.168.1.0/24`.
            - Press **Enter**.
            - Enter the second subnet CIDR, `192.168.2.0/24`.
            - Press **Enter**.
            - Click **Apply**.
        - Click **Save**.
    3. Click **Apply**.
    4. For **Destinations**, select **Any**.
8. Click **Publish**.

10. Add a firewall rule that allows ingress to vCenter from the TAS for VMs control plane:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**, and then select **Security**.
    5. Click **Gateway Firewall**, and then select **Management Gateway**.
    6. Click **Add Rule**.
    7. Select the newly-created rule:
        1. For **Rule Name**, enter `vCenter Inbound Rule`.
        2. Edit **Sources**:
            - Click **User Defined Groups**.
            - Click **Add Group**.
                - For **Group name**, enter `Ops Manager public IP`.
                - Click **Set Members**.
                - Select **IP Addresses**.
                - Enter your Ops Manager public IP address. For example, `54.190.190.190`.
                - Press **Enter**.
                - Click **Apply**.
            - Click **Save**.
            - Click **Add Group**.
                - For **Group name**, enter `Workloads Compute NAT public IP`.
                - Click **Set Members**.
                - Select **IP Addresses**.
                - Enter your Workloads Compute NAT public IP address. For example, `44.232.216.160`.

> **Note**: You can locate your Workloads Compute NAT public IP address in the Networking & Security Overview.

                - Press **Enter**.
                - Click **Apply**.
            - Click **Save**.
        3. Edit **Destinations**: Select **vCenter** and then click **Apply**.
        4. Edit **Services**: Enter `HTTPS, ICMP, SSO`.
    8. Click **Publish**.

11. Add a firewall rule that allows ingress to ESXi from the TAS for VMs control plane:
    1. Go to the VMC console.
    2. Select **SDDCs**.
    3. Click **View Details** on your datacenter tile.
    4. Select **Networking & Security**, and then select **Security**.
    5. Click **Gateway Firewall**, and then select **Management Gateway**.
    6. Click **Add Rule**.
    7. Select the newly-created rule:
        1. For **Rule Name**, enter `ESXi Inbound Rule`.
        2. Edit **Sources**:
            - Click **User Defined Groups**.
            - Click **Add Group**.
                - For **Group name**, enter `Ops Manager private IP`.
                - Click **Set Members**.
                - Select **IP Addresses**.
                - Enter the Ops Manager private IP address, `192.168.1.10`.
                - Press **Enter**.
                - Click **Apply**.
            - Click **Save**.
            - Click **Add Group**.
                - For **Group name**, enter `BOSH Director private IP`.
                - Click **Set Members**.
                - Select **IP Addresses**.
                - Enter the BOSH Director private IP address, `192.168.1.11`.
                - Press **Enter**.

            - Click **Apply**.
        - Click **Save**.
    3. Edit **Destinations**: Select **vCenter** and then click **Apply**.
    4. Edit **Services**: Enter `HTTPS, ICMP, SSO`.
8. Click **Publish**.

Configure BOSH Director

> **Note:** The procedure in this section contains only the configuration information that is specific to VMC. For more information about configuring BOSH on vSphere, see Configuring BOSH Director on vSphere.

To configure BOSH Director for VMC:

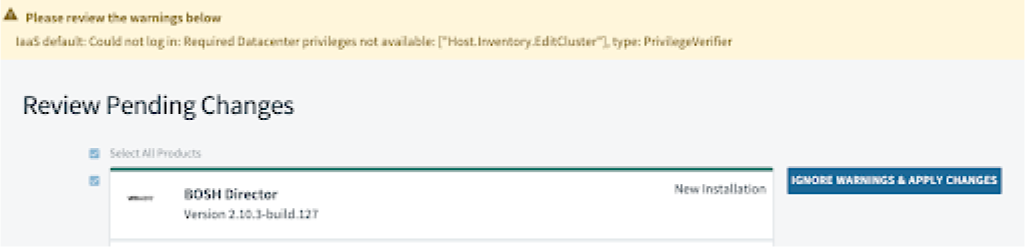
1. Log in to Ops Manager:
    - Go to the IP address you configured for Ops Manager. For example, https://54.190.190.190/.
    - Enter a **Username**, **Password**, and **Password confirmation** to create an Admin user.
    - Enter a **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore and is not recoverable.

> **Note:** When configuring the BOSH Director, do not configure NSX-T networking; instead, select Standard vCenter Networking.

1. Select the **BOSH Director for vSphere** tile and configure BOSH as follows:
    - **Settings → vCenter Config**
        - **vCenter Host**: your vCenter URL. For example, `vcenter.sddc-35-162-72-214.vmwarevmc.com`.
        - **vCenter Username**: `cloudadmin@vmc.local`
        - **vCenter Password**: Enter the password you gathered from the SDDC.
        - **Datacenter Name**: Enter `SDDC-Datacenter`.
        - **Virtual Disk Type**: Select **Thin**.
        - **Ephemeral Datastore Names**: Enter `WorkloadDatastore`.
        - **Persistent Datastore Names**: Enter `WorkloadDatastore`.
        - Select **Standard vCenter Networking**.
    - **Settings → Director Config**
        - **NTP Servers**: Enter an NTP server of your choice. For example, `time.google.com`.
    - **Settings → Create Availability Zones**

        - **az1**:
            - **Name**: `az1`
            - **Clusters**:
                - **Cluster**: `Cluster-1`
                - **Resource Pool**: `az1`
        - **az2**:
            - **Name**: `az2`
            - **Clusters**:
                - **Cluster**: `Cluster-1`
                - **Resource Pool**: `az2`
        - **az3**:
            - **Name**: `az3`
            - **Clusters**:
                - **Cluster**: `Cluster-1`
                - **Resource Pool**: `az3`
    - **Settings → Create Networks → Networks**
        - **infra**:
            - **Name**: `infra`
            - **Subnets**:
                - **vSphere Network Name**: `sddc-cgw-network-1`
                - **CIDR**: `192.168.1.0/24`
                - **Reserved IP Ranges**: `192.168.1.1-192.168.1.10`
                - **DNS**: `8.8.8.8`
                - **Gateway**: `192.168.1.1`
                - **Availability Zones**: `az1, az2, az3`
        - **deployment**:
            - **Name**: `deployment`
            - **Subnets**:
                - **vSphere Network Name**: `bosh-network`
                - **CIDR**: `192.168.2.0/24`
                - **Reserved IP Ranges**: `192.168.2.1`
                - **DNS**: `8.8.8.8`
                - **Gateway**: `192.168.2.1`
                - **Availability Zones**: `az1, az2, az3`
    - **Settings → Assign AZs and Networks**
        - **Singleton Availability Zone**: `az1`

        - **Network**: `infra`
    - **Settings → Security**
        - **Include Tanzu Ops Manager Root CA in Trusted Certs**: Select the checkbox.
2. After you finish configuration, click **Apply Changes**. The following warning appears:



⚠ Please review the warnings below

IaaS default: Could not log in: Required Datacenter privileges not available: ["Host.Inventory.EditCluster"], type: PrivilegeVerifier

Review Pending Changes

   Click **Ignore Warnings & Apply Changes**. `EditCluster` permission is not required.

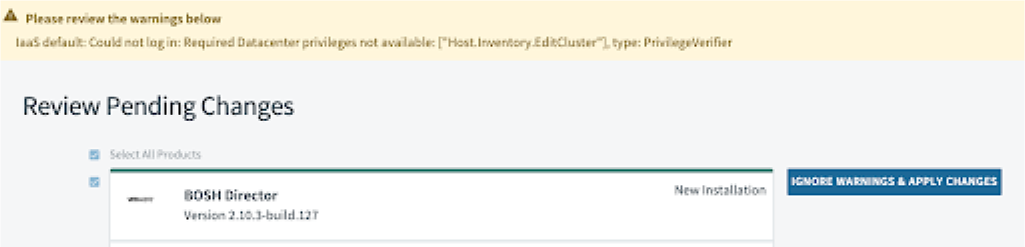
Install TAS for VMs

To configure TAS for VMs for VMC:

1. Configure the **TAS for VMs** tile as follows:
    - **Settings → Assign AZs and Networks**
        - **Network**: Select `deployment`.
        - Click **Save**.
    - **Settings → Domains**
        - Set up a wildcard domain and ensure that it maps to the HAProxy public IP address you configured earlier. For example:
            - **System domain**: `sys.54.180.180.180.nip.io`
            - **Apps domain**: `run.54.180.180.180.nip.io`
    - **Settings → Networking**
        - Set the HAProxy IP address to the one you specified in the NAT rule above. This is the first available IP address in the `deployment` network.
            - **HAProxy IPs**: `192.168.2.2`
        - Generate a certificate for the Gorouter and HAProxy:
            - **Certificates and private keys for the Gorouter and HAProxy**: Click **Add**.
                - Name: `haproxy cert`
                - Click **Generate RSA Certificate**.
                - Add `*.` domains for your system and apps domain above, separated by a comma. For example, `*.**sys.54.180.180.180.nip.io,*.run.54.180.180.180.nip.io`.
        - Set HAProxy as the TLS termination point:
            - **TLS Termination**: **HA Proxy**

        - Disable TLS forwarding for HAProxy:
            - **HAProxy forwards all requests to the Gorouter over TLS**: Click **Disable**.
    - **Settings → UAA**
        - For UAA configuration, generate a SAML certificate for *.login.SYSTEM-DOMAIN:
            - **→ SAML service provider credentials**
                - Click **Generate RSA Certificate**.
                - Enter your domain. For example, `*.login.sys.54.180.180.180.nip.io`
                - Click **Generate**.
    - **Settings → Resource Config**:
        - Scale the HAProxy instances up to 1.
        - **→ HAProxy**
            - Set Instances to **1**.
            - Click **Save**.

2. After you finish configuration, click **Apply Changes**. The following warning appears:



⚠ Please review the warnings below

IaaS default: Could not log in: Required Datacenter privileges not available: ["Host.Inventory.EditCluster"], type: PrivilegeVerifier

   Click **Ignore Warnings & Apply Changes**. `EditCluster` permission is not required.

Installing TAS for VMs [Windows]

In this section:

- Overview
- Product Architecture
- Installing and Configuring TAS for VMs [Windows]
- Downloading or Creating Windows Stemcells
- Creating a Windows Stemcell for vSphere Using stembuild
- Windows Diego Cells in Isolation Segments
- Upgrading TAS for VMs [Windows] and Windows Stemcells
- Migrating Apps to TAS for VMs [Windows]
- Troubleshooting Windows Diego Cells

TAS for VMs [Windows] Overview

VMware Tanzu Application Service for VMs [Windows] enables operators to provision, operate, and manage Windows stemcells on Ops Manager.

After operators install the TAS for VMs [Windows] tile on the Ops Manager Installation Dashboard, developers can push .NET apps to Windows Diego Cells using the Cloud Foundry Command Line Interface (cf CLI).

For the TAS for VMs [Windows] release notes, see VMware Tanzu Application Service for VMs [Windows] v2.12 Release Notes.

Requirements

To install the TAS for VMs [Windows] tile, you must have:

- Ops Manager v2.12 and VMware Tanzu Application Service for VMs v2.12 deployed to vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), or Azure.
- A Windows stemcell. For information about obtaining or creating a Windows stemcell, see Downloading or Creating Windows Stemcells.

The minimum resource requirements for each Windows Diego Cell are:

- Disk size: 64 GB (100 GB recommended)
- Memory: 16 GB
- CPUs: 4

Contents

About TAS for VMs [Windows]

Installation Guide

Admin Guide

Developer Guide

- Changing Windows Stacks
- Tips for .NET Developers

Limitations

The following limitations affect Windows containers in TAS for VMs [Windows]:

- Windows Server Container Limitations
- TAS for VMs [Windows] Limitations

Windows Server Container Limitations

Windows Server Core Containers are used in TAS for VMs [Windows]. The following are known Windows Server Container limitations:

- Windows Server Core containers support the host VM's timezone only. In TAS for VMs [Windows], the host and container time-zone is always UTC.
- Windows Server 2019 Core images support only Lucida Console font. No other fonts are supported, and no others can be installed.

TAS for VMs [Windows] Limitations

TAS for VMs [Windows] has the following limitations:

- TAS for VMs [Windows] can be installed in an environment with the NSX Container Plugin (NCP), but NCP cannot be used with Windows containers in TAS for VMs [Windows]. For more information about NCP, see What's New in the VMware documentation.
- Developers cannot push Docker or other OCI-compatible images to Windows Diego Cells.
- Container-to-container networking is not available for Windows-hosted apps.
- OpenStack is currently not supported for TAS for VMs [Windows]. Contact your VMware representative for information about OpenStack deployments.
- SMB is supported, but Volume Services is not supported.
- Due to Known Issue #244 in Windows Server OS, apps hosted on TAS for VMs [Windows] cannot route traffic when deployed with the IPsec. For more information, see Traffic to containers via NAT stops working when using IPsec to encrypt network connections on GitHub.

Product Architecture

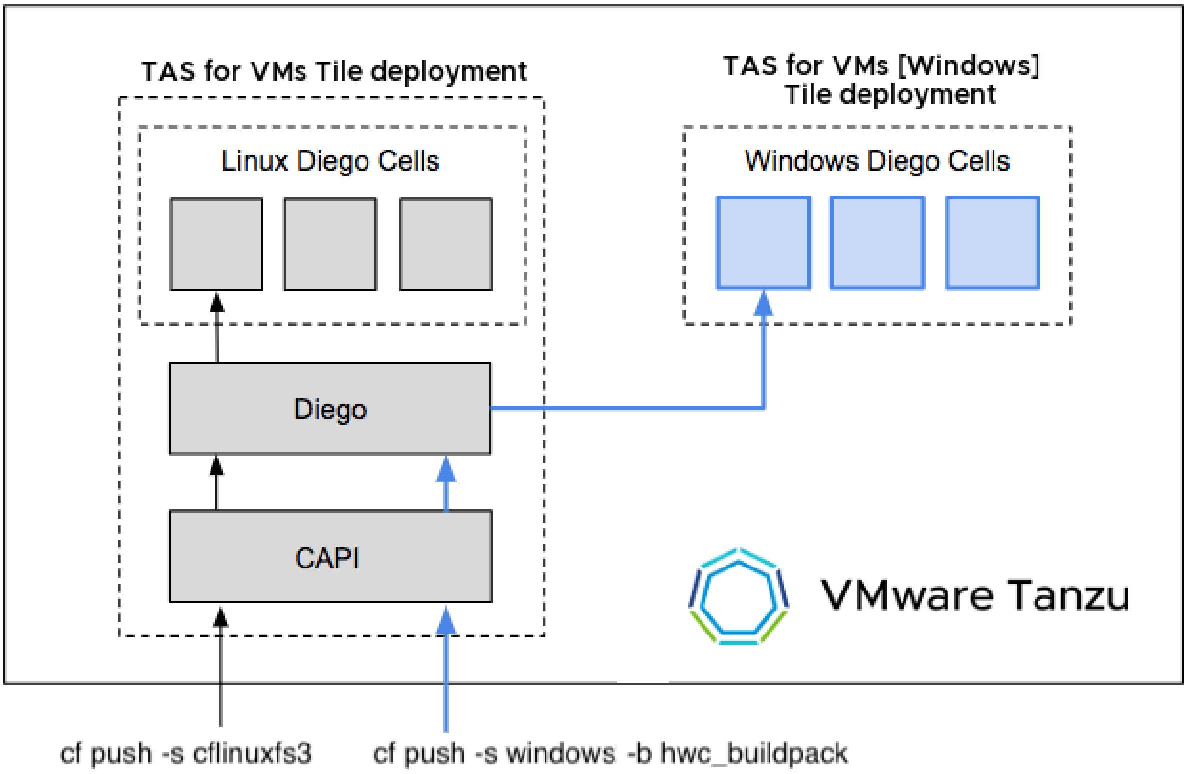
This topic describes the architecture of Windows Diego Cells that VMware Tanzu Application Service for VMs [Windows] deploys to run containerized .NET apps and the stemcells that it supplies to BOSH as the operating system for the Windows Diego Cell VMs.

Overview

Operators, who want to run Windows Diego Cells in Ops Manager to enable developers to push .NET apps, can deploy the TAS for VMs [Windows] tile.

Deploying this tile creates a separate BOSH deployment populated with the Garden Windows release, which runs on a Windows Diego Cell built from a Windows stemcell.

After the Windows Diego Cell is running, developers can specify a Windows stack when pushing .NET apps from the command line. Ops Manager passes the app to the Windows Diego Cell in the TAS for VMs [Windows] BOSH deployment. The diagram below illustrates the process.


TAS for VMs Tile deployment
Linux Diego Cells
Diego
CAPI
TAS for VMs [Windows] Tile deployment
Windows Diego Cells
VMware Tanzu
cf push -s cflinuxfs3
cf push -s windows -b hwc_buildpack

About Windows Cells

App instances in Ops Manager run inside containers. Garden is the API that creates and manages these containers. An implementation equivalent to that on Linux Diego Cells provides this infrastructure on Windows Diego Cells, utilizing native Windows Server containers.

By installing the TAS for VMs [Windows] tile, operators create a Windows Diego Cell from a stemcell that contains the Windows Server operating system. Garden on Windows uses Windows containers to isolate resources on Windows Diego Cells that Ops Manager manages alongside Linux Diego Cells.

For more information about installing the TAS for VMs [Windows] tile, see Installing and Configuring TAS for VMs [Windows].

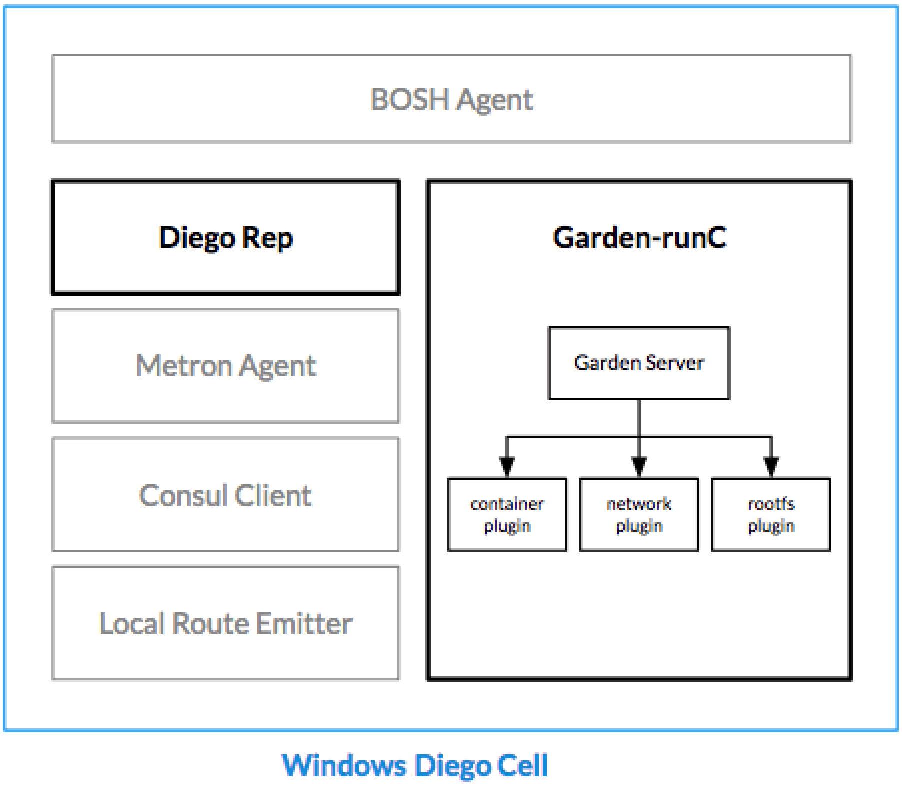
For more information about Garden, see Garden. For more information about Diego Cells, see Diego Cell in *TAS for VMs Components*. For more information about stemcells, see What is a Stemcell? in the BOSH documentation. For more information about Windows containers, see Windows and containers in the Microsoft documentation.

Components

A Windows Diego Cell includes the following components:

- Guardian: Implements the Garden API on Windows.
- Loggregator Agent: Forwards app logs, errors, and metrics to the Loggregator system. For more information about the Loggregator Agent, see Step 6: Sends Logs to Loggregator in *Diego Components and Architecture*. For more information about Loggregator, see Loggregator Architecture.
- BOSH Agent: Executes instructions from the BOSH Director. For more information, see Agent in *Components of BOSH* in the BOSH documentation.
- Diego Rep: Runs and manages Tasks and Long Running Processes. For more information, see Tasks and Long-Running Processes in *How Diego Balances App Processes*.

The following diagram illustrates the architecture of a Windows Diego Cell:



Container Implementation

Garden on Windows uses the following runtime plugins to create and manage Windows containers for VMware Tanzu Application Service for VMs:

- Container plugin `winc`: Creates Open Container Initiative (OCI)-compliant containers, executes processes in the containers, and sets their CPU and RAM limits. For more information about OCI compliance, see the Open Container Initiative website.
- Network plugin `winc-network`: Creates a network compartment for the container, applies its DNS settings, and defines its inbound and outbound network access rules.

- Rootfs image plugin `groot`: Sets up the container filesystem volume and uses the File Server Resource Manager (FSRM) API to define its disk usage quotas. For more information about the FSRM, see File Server Resource Manager (FSRM) overview in the Microsoft documentation.

About Windows Stemcells

A stemcell is a customized operating system image containing the filesystem for BOSH-managed VMs. When deployed, the operating system includes the BOSH Agent process, which is dedicated to communicating with the orchestrating VM, the BOSH Director. The BOSH Agent executes and monitors BOSH jobs on its VM.

Deployments of Windows Server on Ops Manager currently use a stemcell containing Windows Server 2019.

For more information about obtaining or creating a stemcell for TAS for VMs [Windows], see Downloading or Creating Windows Stemcells.

Installing and Configuring TAS for VMs [Windows]

This topic describes how to install and configure the VMware Tanzu Application Service for VMs [Windows] tile.

Overview

The TAS for VMs [Windows] tile installs Windows Diego Cells in your Ops Manager deployment.

The TAS for VMs [Windows] tile inherits settings from the VMware Tanzu Application Service for VMs (TAS for VMs) tile and also includes additional configuration settings.

To install, configure, and deploy TAS for VMs [Windows]:

- In an environment with internet access:
    1. Complete the prerequisites. See Prerequisites.
    2. Download and install the TAS for VMs [Windows] tile. See Install the Tile.
    3. Configure required settings for the tile. See Configure the Tile.
    4. Configure resources for the tile. See Configure Tile Resources.
    5. Upload the Windows stemcell to the tile. See Upload the Stemcell.
    6. Deploy the tile. See Deploy the Tile.
- In an air-gapped environment, complete the steps in Install and Configure TAS for VMs [Windows] in an Air-Gapped Environment below.

Prerequisites

Before you install and configure the TAS for VMs [Windows] tile, ensure that you meet the requirements to use the **Windows FS Injector** tool. For more information, see Windows FS Injector Prerequisites.

Windows FS Injector Prerequisites

You use the **Windows FS Injector** tool to install the TAS for VMs [Windows] tile. The **Windows FS Injector** tool requires:

- The `git` and `tar` executables must be in your `%PATH%`. If `git` and `tar` are not in your `%PATH%`, either add your `git` and `tar` executable locations to your existing `%PATH%` configuration, or copy the `git.exe` and `tar.exe` executables to a directory in your `%PATH%`.
- Your installation environment must allow the **Windows FS Injector** tool access to all of the following URLs:
  - https://network.pivotal.io, for downloading the tile and injector
  - https://s3.amazonaws.com/
  - https://registry.hub.docker.com/
  - https://production.cloudflare.docker.com
  - https://go.microsoft.com/
  - https://winlayers.cdn.mscr.io
  - https://mcr.microsoft.com
  - https://msecnd.net, or any domain within the Microsoft Windows Azure Content Delivery Network For more information about the Windows Azure Content Delivery Network, see Introducing the Windows Azure Content Delivery Network in the Microsoft Azure documentation.

> **Note:** To ensure the authenticity of Microsoft container images, Microsoft does not permit the distribution of its base images. This includes Microsoft container images consumed through Docker Hub, which are actually delivered by an Microsoft CDN endpoint.

Install the Tile

To install the TAS for VMs [Windows] tile:

1. Go to the VMware Tanzu Application Service for VMs [Windows] page on VMware Tanzu Network.
2. Download the **VMware Tanzu Application Service for VMs [Windows]** product file.
3. Download the **Windows FS Injector** tool for your workstation OS. The Injector tool, `winfs-injector`, is an executable binary that adds the Windows Server container base image into the product file. This step requires internet access and can take up to 20 minutes.

   > **Note:** You need the `git` and `tar` executables in your `%PATH%` to run `winfs-injector.exe`. For example, copy `git.exe` and `tar.exe` to a directory in your `%PATH%`.

4. Add the Windows Server container base image to the product file:

```
winfs-injector ^
  --input-tile PASW-DOWNLOAD-PATH ^
  --output-tile PASW-IMPORTABLE-PATH
```

Where: * `PASW-DOWNLOAD-PATH` is the path and filename to the TAS for VMs [Windows] product file you downloaded. * `PASW-IMPORTABLE-PATH` is the desired output path for the importable product file.
For example:

```
C:\Users\admin\> winfs-injector ^
--input-tile c:\temp\pas-windows-2.9.0-build.1.pivotal ^
--output-tile c:\temp\pas-windows-2.9.0-build.1-INJECTED.pivotal
```

This step can take up to 20 minutes to complete.

5. Navigate to the Ops Manager Installation Dashboard and click **Import a Product**.
6. To add the TAS for VMs [Windows] tile to the **Import a Product** product list, select the importable `PASW-IMPORTABLE-PATH` file on your workstation.
7. To add the TAS for VMs [Windows] tile to your staging area, click **+** under the **VMware Tanzu Application Service for VMs [Windows]** product listing.

Configure the Tile

The following sections describe how to configure the settings for the TAS for VMs [Windows] tile.

Assign Availability Zones and Networks

In **Assign AZ and Networks**, you assign jobs to your Availability Zones (AZs) and networks.

To configure AZs and networks:

1. Click the TAS for VMs [Windows] tile.
2. Select **Assign AZs and Networks** or **Assign Networks**. The name of the pane varies depending on your IaaS.
3. Select the first AZ under **Place singleton jobs**. Ops Manager runs any job with a single instance in this AZ.
4. Select all AZs under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the AZs that you specify.

   > **Note:** For production deployments, VMware recommends at least three AZs for a highly available installation.

5. From the **Network** dropdown, choose the runtime network that you created when configuring the BOSH Director tile.
6. Click **Save**.

Configure VMs

In **VM Options**, you configure settings for accessing your VMs.

To configure VM access:

1. Select **VM Options**.
2. Select one of the following for **Manage administrator password**:
    - To randomize the admin password, select **Use random password**. If you select this option, the admin password is not retrievable by an operator. This is the default selection.
    - To set the same admin password for every Windows Diego Cell, select **Set the password** and enter a password in **Password**. If you select this option, this password can be used to access any Windows Diego Cell.
3. (Optional) To start the Microsoft beta port of the OpenSSH daemon on port 22 on all VMs, select the **Enable BOSH-native SSH support on all VMs (beta)** check box. If you select this option, users can SSH into Windows VMs with the `bosh ssh` command and enter a CMD terminal as an admin user. They can then run `powershell.exe` to start a PowerShell session.
4. (Optional) To configure a Key Management Service (KMS) that your volume-licensed Windows Diego Cell can register with:
    1. Under **Key Management Service**, select **Enable**.
    2. For the **Host** field, enter the KMS hostname.
    3. For the **Port** field, enter the port number. The default port number is 1688.
5. Click **Save**.

Configure Smoke Tests

In **Smoke Tests**, you specify the org and space where smoke tests are run.

In the org and space that you specify, the Smoke Test errand pushes an app to the org. The app runs basic functionality tests against your TAS for VMs [Windows] deployment after an installation or update.

The Smoke Test errand is on by default. You can turn off the Smoke Test errand in the **Errands** pane. For more information, see Configure Errands.

To configure smoke tests:

1. Select **Smoke Tests**.
2. If you have a shared apps domain, select **A temporary space within the system org**, which creates a temporary space within the system org for running smoke tests and deletes the space afterwards. Otherwise, select **A specified org and space** and complete these fields to configure where TAS for VMs [Windows] pushes an app to run smoke tests:
    - For **Org**, enter the org TAS for VMs [Windows] should use when pushing an app to run smoke tests.
    - For **Space**, enter the space TAS for VMs [Windows] should use when pushing an app to run smoke tests.

- For **Domain**, enter the domain TAS for VMs [Windows] should use when pushing an app to run smoke tests.

Configure Advanced Features

Advanced Features includes new functionality that might have certain constraints. Although these features are fully supported, VMware recommends caution when using them in production environments.

The following sections describe how to configure these advanced features.

Diego Cell Memory and Disk Overcommit

If your apps do not use the full allocation of disk space and memory set in **Resource Config**, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each host VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Windows Diego Cell**.

> **Note:** Due to the risk of app failure and the deployment-specific nature of disk and memory use, VMware has no recommendation for how much, if any, memory or disk space to overcommit.

To enable overcommit:

1. Select **Advanced Features**.
2. Enter in MB the total desired amount of Diego Cell memory in the **Diego Cell memory capacity** field. See the **Diego Cell** row in **Resource Config** for the current Diego Cell memory capacity settings that this field overrides.
3. Enter in MB the total desired amount of Diego Cell disk capacity in the **Diego Cell disk capacity** field. See the **Diego Cell** row in **Resource Config** for the current Diego Cell disk capacity settings that this field overrides.
4. Click **Save**.

> **Note:** Entries made to each of these two fields set the total amount of resources allocated, not the overage.

TLS Connections from the Gorouter to Apps (Beta)

You can choose the method the Gorouter uses to verify app identity. Verifying app identity using TLS or mutual TLS (mTLS) enables encryption between the Gorouter and app containers and guards against misrouting during control plane failures. This feature is disabled by default.

For more information about Gorouter route consistency modes, see Preventing Misrouting in *HTTP Routing*.

To configure app identity verification:

1. Select **Advanced Features**.
2. Under **TLS connections from the Gorouter to apps (beta)**, select one of the following options:
    - **Disable route integrity and mutual TLS**: Disables app identity verification and mTLS.
    - **The Gorouter uses TLS to verify app identity**: Enables the Gorouter to verify app identity using TLS. This is the default option.
    - **The Gorouter and apps use mutual TLS to verify each other's identity**: Enables the Gorouter and your apps to verify each other's identity using TLS. Before you enable this option, be aware of the following:
        - This option disables TCP routing because app containers accept incoming communication only from the Gorouter.
        - If you enable mTLS in the TAS for VMs [Windows] tile, you must also enable mTLS in the **App Containers** pane of the TAS for VMs tile.
        - You need v2.3 or later of both TAS for VMs and Isolation Segment. The Gorouter and Diego Cell components in Pivotal Cloud Foundry v2.2 and earlier do not support mTLS handshakes.
3. Click **Save**.

Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of TAS for VMs [Windows]. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

By default, Ops Manager runs all errands.

In **Errands**, you can change these run rules. For each errand, you can select **On** to run it each time Ops Manager installs or uninstalls a product, or **Off** to never run it.

For more information about how Ops Manager manages errands, see Managing Errands in Ops Manager.

To configure errands:

1. Select **Errands**.
2. To ensure that you receive the most up-to-date HWC buildpack, set the **Install HWC Buildpack Errand** to **On**.
3. To ensure that a smoke test is run against your TAS for VMs [Windows] installation, set the **Smoke Test Errand** to **On**.
4. Click **Save**.

> **Note:** This beta feature checks only that the client certificate is signed by the expected CA using mTLS. It does not include SAN (Subject Alternative Name) checks of the presented client certificates.

(Optional) Configure Isolation Segments

To deploy your TAS for VMs [Windows] app workloads to an isolation segment, select **App Containers** and follow the procedure in Assign a Tile to an Isolation Segment in *Windows Diego Cells in Isolation Segments*.

(Optional) Configure System Logging

To configure Windows Diego Cells to send vm logs to an external syslog server, select **System Logging** and follow the procedure in Forwarding Logs to a Syslog Server in *Troubleshooting Windows Diego Cells*.

(Optional) Configure DNS Search Domains

To configure DNS search domains for your app containers:

1. Click the VMware Tanzu Application Service for VMs tile in the **Installation Dashboard**.
2. Select the **Networking** pane.
3. In the **DNS search domains** field, enter DNS search domains as a comma-separated list.
4. Click **Save**.

Configure Tile Resources

In **Resource Config**, you must associate load balancers with the VMs in your deployment to enable traffic.

To configure your tile resources:

1. Select **Resource Config**.
2. Use the dropdowns to configure **Windows Diego Cell**. The table below shows the recommended Windows Diego Cell disk size for your IaaS:

IaaS	Recommended Windows Diego Cell Disk Size
AWS	100 GB
Azure	150 GB
GCP	150 GB
vSphere	100 GB

> **Note:** Windows stemcells in the v2019.x line support ephemeral disks.

3. Provision your **Master Compilation Job** with at least 100 GB of disk space.
4. Click **Save**.

Upload the Stemcell

After configuring resources for the TAS for VMs [Windows] tile, you must upload the Windows stemcell to the tile.

To upload the stemcell:

1. In the TAS for VMs [Windows] tile, select **Stemcell Library**.
2. Retrieve the stemcell that you downloaded or created in Downloading or Creating a Windows Stemcell.
3. Follow the procedure in Importing and Managing Stemcells to upload the Windows stemcell to TAS for VMs [Windows].

> **Note:** If you use vSphere, you must create your own stemcell. The default root disk size of Windows stemcells v2019.x line is 30 GB. VMware recommends setting the root disk size of your Windows stemcell for vSphere to 30 GB. For more information, see Creating a Windows Stemcell for vSphere Using stembuild.

Deploy the Tile

After uploading the Windows stemcell to the TAS for VMs [Windows] tile, you are ready to deploy the tile.

To deploy the TAS for VMs [Windows] tile:

1. Go to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**.
3. Select the TAS for VMs [Windows] tile and review the changes. For more information, see Reviewing Pending Product Changes.
4. Click **Apply Changes**.

(Optional) Create More Tiles

To run Windows Diego Cells in multiple isolation segments, you must create and configure additional TAS for VMs [Windows] tiles. For more information, see Windows Diego Cells in Isolation Segments.

Install and Configure TAS for VMs [Windows] in an Air-Gapped Environment

To install, configure, and deploy TAS for VMs [Windows] in an air-gapped environment:

1. Complete the steps in Prepare a Windows Rootfs Image in a Private Registry below.
2. Complete the steps in Install the Tile above with the following exceptions:
   - To add the Windows Server container base image to the product file, replace step 4’s internet-enabled `winfs-injector` command line with the `winfs-injector` procedure in Add the Windows Server Container Base Image to the Product File below.

3. Configure required settings for the tile. See Configure the Tile above.
4. Configure resources for the tile. See Configure Tile Resources above.
5. Upload the Windows stemcell to the tile. See Upload the Stemcell above.
6. Deploy the tile. See Deploy the Tile above.

Prepare a Windows Rootfs Image in a Private Registry

To create a TAS for VMs [Windows] tile, a windows file-system container image is typically fetched from a Docker registry. An administrator can fetch the windows file-system image from either cloudfoundry/windows2016fs the publicly hosted DockerHub repository, or a privately hosted container image registry.

To prepare a windows file-system container image in a private registry:

1. Create an accessible Windows Server 2019 machine in your environment.
2. Install Docker on this Windows Server 2019 machine.
3. Configure this Windows machine's Docker daemon to allow non-redistributable artifacts to be pushed to your private registry. For information about configuring your Docker daemon, see Allow push of nondistributable artifacts in the Docker documentation.
4. Open a command line on the Windows machine.
5. To download the windows file-system container image, run the following command:

   ```
   docker pull cloudfoundry/windows2016fs:2019
   ```

6. To tag the Windows container image, run the following command:

   ```
   docker tag cloudfoundry/windows2016fs:2019  REGISTRY-ROOT/cloudfoundry/windows2016fs:2019
   ```

   Where `REGISTRY-ROOT` is your private registry's URI.

7. To upload the Windows Container image to your accessible private registry, run the following command:

   ```
   docker push IMAGE-URI
   ```

   Where `IMAGE-URI` is the URI to the Windows rootfs image in your private registry. Your image URI should follow the pattern: `my.private.registry/cloudfoundry/windows2016fs:2019`.

Add the Windows Server Container Base Image to the Product File

To add the Windows Server container base image to the product file in an air-gapped environment, run the following:

```
winfs-injector ^
  --input-tile PASW-DOWNLOAD-PATH ^
  --output-tile PASW-IMPORTABLE-PATH ^
  --registry PASW-REGISTRY-URI
```

Where:

- `PASW-DOWNLOAD-PATH` is the path and filename to the PASW product file you downloaded.
- `PASW-IMPORTABLE-PATH` is the desired output path for the importable product file.
- `PASW-REGISTRY-URI` is the uri to the container registry hosting your cloudfoundry/windows2016fs image.
  For example:

```
C:\Users\admin> winfs-injector ^
--input-tile c:\temp\pas-windows-2.6.0-build.1.pivotal ^
--output-tile c:\temp\pas-windows-2.6.0-build.1-INJECTED.pivotal ^
--registry https://my.registry.com
```

For information about troubleshooting `winfs-injector`, see Missing Local Certificates for Windows File System Injector in *Troubleshooting Windows Diego Cells*.

Downloading or Creating Windows Stemcells

This topic describes how to download or create the stemcell that VMware Tanzu Application Service for VMs [Windows] needs to create VMs on an infrastructure.

Note: Traffic between the Gorouter and Windows stemcells is not encrypted with TLS.

Windows Stemcells

A BOSH stemcell is a versioned OS image wrapped with IaaS-specific packaging. BOSH uses a stemcell to supply the basic OS for any Windows VM it creates. A typical BOSH stemcell for Windows contains the following:

- A bare minimum Windows OS skeleton.
- A BOSH Agent.
- OS Configuration files and common utilities.

For more information about stemcells, see What is a Stemcell? in the Cloud Foundry BOSH documentation.

Windows Stemcell Compatibility Matrix

Refer to the compatibility matrix below before deploying and installing the TAS for VMs [Windows] tile:

TAS for VMs [Windows] Version	WindowsFS Version	Stemcell 2019.43 and earlier	Stemcell 2019.44 for Azure and Stembuild vSphere	Stemcell v2019.44 for other IAASes	Stemcell 2019.45
v2.7.35 and earlier	v2.31.0 and earlier	Compatible	Not compatible	Compatible	Not compatible

TAS for VMs [Windows] Version	WindowsFS Version	Stemcell 2019.43 and earlier	Stemcell 2019.44 for Azure and Stembuild vSphere	Stemcell v2019.44 for other IAASes	Stemcell 2019.45
v2.7.37 and later	v2.33.2	Compatible	Compatible	Compatible	Not compatible
v2.10.19 and earlier	v2.31.9 and earlier	Compatible	Not compatible	Compatible	Not compatible
v2.10.21 and later	v2.33.2	Compatible	Compatible	Compatible	Not compatible
v2.11.8 and earlier	v2.31.0 and earlier	Compatible	Not compatible	Compatible	Not compatible
v2.11.10 and later	v2.33.2	Compatible	Compatible	Compatible	Not compatible
v2.12.3 and earlier	v2.31.0 and earlier	Compatible	Not compatible	Compatible	Not compatible
v2.12.4 and later	v2.33.2	Compatible	Compatible	Compatible	Not compatible

Depending on your IaaS, you can create or download a stemcell using these methods:

- **Azure**: To activate and download the Azure Windows Stemcell, see Configuring the Azure Light Stemcell.
- **Google Cloud Platform (GCP)**: Download the **GCP light Stemcell for Windows 2019 Server** from the Stemcells (Windows) page on VMware Tanzu Network.
- **vSphere**: To create a stemcell, see Creating a Windows Stemcell for vSphere Using stembuild.
- **Amazon Web Services (AWS)**: Download the **AWS light Stemcell for Windows 2019 Server** from the Stemcells (Windows) page on VMware Tanzu Network.

Configuring the Azure Light Stemcell

On Azure, the stemcell exists as an offering in the Azure Marketplace. To start using the Azure light stemcell:

- Enable the Azure Light Stemcell
- Download the Azure Light Stemcell

Enable the Azure Light Stemcell

To use the Azure light stemcell, you must first accept the corresponding Microsoft license agreement and enable the stemcell for your non-trial Azure Subscription through the Azure Marketplace.

Enable Your Azure Subscriptions

To enable the Azure subscriptions with which you want to use your stemcell:

1. Go to the Microsoft Azure portal and log in.
2. From the options on the left side of the page, click **+ Create a resource**.
3. In the **Search the Marketplace** bar, search for **BOSH Stemcell for Windows Server 2019**.
4. Select **BOSH Stemcell for Windows Server 2019** from your search results. A description of the Azure stemcell offering appears.
5. Below the description, at the bottom of the page, click the blue banner that reads **Want to deploy programmatically? Get started**.
6. Review the Terms of Use on the "Configure Programmatic Deployment" page that appears.
7. Under **Choose the subscriptions**, click **Enable** for each Azure subscription with which you want to use the stemcell.
8. Click **Save**.

Accept the License Agreement

To accept the license agreement for your stemcell:

1. To log in to to Azure with the Azure CLI, use the following command:

   ```
   az login
   ```

2. To list the available Windows stemcells and find latest Windows stemcell `urn`, run the following Azure CLI command:

   ```
   az vm image list --all --publisher pivotal --sku 2019
   ```

   For example:

   ```
   az vm image list --all --publisher pivotal --sku 2019
   Offer                     Publisher    Sku    Urn
   Version
   ------------------------  -----------  -----  ------------------------
   -------------------------  ------------
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.1.050001  2019.1.050001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.2.022001  2019.2.022001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.3.041001  2019.3.041001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.4.044001  2019.4.044001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.5.006001  2019.5.006001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.6.025001  2019.6.025001
   bosh-windows-server-2019  pivotal      2019   pivotal:bosh-windows-serv
   er-2019:2019:2019.7.012001  2019.7.012001
   ```

```
bosh-windows-server-2019  pivotal     2019   pivotal:bosh-windows-serv
er-2019:2019:2019.8.028001  2019.8.028001
```

In the example above, `pivotal:bosh-windows-server-2019:2019:2019.8.028001` is the latest Windows stemcell `urn`.

3. To accept the terms for the selected Windows stemcell for each Azure subscription with which you want to use the stemcell, run the following command:

   ```
   az vm image accept-terms --subscription SUBSCRIPTION_ID --urn URN-ID
   ```

   Where:

   - `SUBSCRIPTION_ID` is an Azure subscription with which you want to use the stemcell.
   - `URN-ID` is the `urn` for the latest Windows stemcell.

Download the Azure Light Stemcell

To download the Azure light stemcell for use in Ops Manager:

1. Go to the Stemcells (Windows) page on VMware Tanzu Network.
2. Download the **Azure Light Stemcell for Windows 2019 Server**.

For information about how to deploy and configure the TAS for VMs [Windows] tile, see Installing and Configuring TAS for VMs [Windows].

Creating a Windows Stemcell for vSphere Using stembuild

This topic describes how to create a BOSH stemcell for Windows on vSphere using stembuild.

Overview of Windows Stemcell Creation

A BOSH stemcell is a versioned operating system image.

You must create a BOSH stemcell for Windows before you can deploy the following products on vSphere:

- VMware Tanzu Application Service for VMs [Windows] (TAS for VMs [Windows])
- VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) with Windows workers in Kubernetes clusters

To create a Windows stemcell for vSphere, you create a base Windows VM from a volume-licensed ISO and subsequently maintain that base template with all Windows-recommended security updates, but without the BOSH dependencies.

The Windows VM with security updates serves as the base for all future stemcells produced from clones of that base VM. This enables you to build new stemcells without having to run Windows updates from scratch each time. You can also use a “snapshot” feature to maintain an updated Windows image that does not contain the BOSH dependencies.

VMware recommends installing any available critical updates and then rebuilding the stemcell from a clone of the original VM.

The BOSH stemcell that you create in this topic is based on Windows Server 2019. If you already have a BOSH stemcell for Windows on vSphere, see Monthly Stemcell Upgrades below.

For more information, see Best Practices for Stembuild for Tanzu Application Service & Tanzu Kubernetes Grid Integrated Edition in *VMware Tanzu Tech Tutorials*.

Overview of Stembuild

Stembuild is a binary that you use to build BOSH stemcells for Windows Server 2019.

Stembuild creates a BOSH stemcell from a base Windows image. The stembuild CLI has two commands, construct and package, which you run against a Windows Server 2019 VM. Step 4: Construct the BOSH Stemcell and Step 5: Package the BOSH Stemcell below explain how to run these commands.

Before using stembuild to create a stemcell, you need to create a Windows Server 2019 VM and update the VM with the latest Windows updates. Step 1: Create a Base VM for the BOSH Stemcell, Step 2: Configure the Base VM, and Step 3: Clone the Base VM below explain how to prepare the Windows Server 2019 VM.

Prerequisites

Before you create a BOSH stemcell for Windows on vSphere, you must have:

- A vSphere environment. To ensure the VM hardware used by the stemcell is compatible with your deployment environment's ESXi/ESX host and vCenter Server versions, see ESXi/ESX hosts and compatible virtual machine hardware versions list (2007240) in the VMware Knowledge Base.
- An ISO for a Windows Server 2019 Server Core installation, build number: 17763, from Microsoft Developer Network (MSDN) or Microsoft Volume Licensing Service Center (VLSC). The Windows Server 2019 ISO must be a clean, base ISO file. You can use an evaluation copy for testing, but VMware does not recommend an evaluation copy for production, becauase the licensing expires. For more information, see the Windows Server documentation or the Microsoft Volume Licensing Service Center website.

  > **Note:** A clean ISO file has no custom scripts or tooling. For example, the ISO must have no logging or antivirus tools installed.

- A download of the `stembuild` command line interface (CLI) from a 2019.x release in Stemcells (Windows) on VMware Tanzu Network.
  - To build a Windows stemcell for TAS for VMs [Windows], use the vSphere `stembuild` CLI for Windows corresponding to both the operating system of your local host and the stemcell version that you want to build.
  - To build a Windows stemcell for TKGI, use a Windows stemcell and `stembuild` CLI version listed as compatible in the **Product Snapshot** table in the release notes for your TKGI version:
    - **TKGI v1.8**: Release Notes

        - **PKS v1.7**: Release Notes
        - **PKS v1.6**: Release Notes
- Microsoft Local Group Policy Object Utility (LGPO) downloaded to the same folder as your `stembuild` CLI.
- The minimum vCenter user permissions required to use `stembuild` for vSphere stemcells, specifically:
    - `VirtualMachine.GuestOperations.Modify`
    - `VirtualMachine.GuestOperations.Execute`
    - `VirtualMachine.GuestOperations.Query`
    - `VirtualMachine.Config.AddRemoveDevice`
    - `VirtualMachine.Interact.SetCDMedia`
    - `VApp.Export`
    - `System.Anonymous`*
    - `System.Read`*
    - `System.View`*

Permissions marked with an `*` are generated upon creating a new user in vCenter and cannot be set within the vCenter UI.

Step 1: Create a Base VM for the BOSH Stemcell

This section describes how to create, configure, and verify a base VM for Windows from a volume-licensed ISO.

Upload the Windows Server 2019 ISO

To upload the Windows Server 2019 ISO to vSphere:

1. Log in to the vSphere Web Client.

    Note: The instructions in this topic are based on vSphere v6.0.

2. Click **Storage** and select a datastore.
3. Select or create a folder where you want to upload the Windows Server 2019 ISO.
4. Click **Upload a File** and select the Windows Server 2019 ISO.

You can use the `scp` utility instead of the vSphere Web Client to copy the file directly to the datastore server.

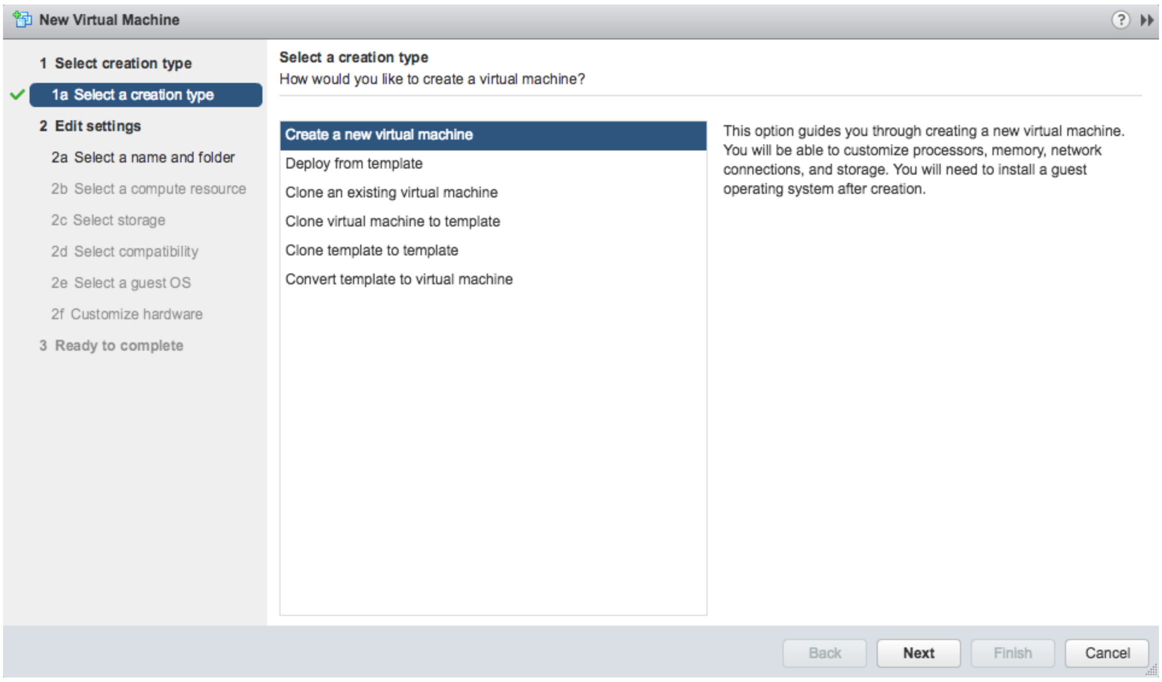
Create and Customize a Base VM

To create and customize a base VM:

1. In the vSphere Web Client, click the **VMs and Templates** view to display the inventory objects.

2. Right-click an object and select **New Virtual Machine** > **New Virtual Machine**.

3. On the **Select a creation type** page, select **Create a new virtual machine** and click **Next**.



New Virtual Machine
1 Select creation type
1a Select a creation type
2 Edit settings
2a Select a name and folder
2b Select a compute resource
2c Select storage
2d Select compatibility
2e Select a guest OS
2f Customize hardware
3 Ready to complete
Create a new virtual machine
Deploy from template
Clone an existing virtual machine
Clone virtual machine to template
Clone template to template
Convert template to virtual machine
Back
Next
Finish
Cancel

Select a creation type

How would you like to create a virtual machine?

This option guides you through creating a new virtual machine. You will be able to customize processors, memory, network connections, and storage. You will need to install a guest operating system after creation.

4. On the **Select a name and folder** page:

    1. Enter a name for the VM.

    2. Select a location for the VM.

    3. Click **Next**.

5. On the **Select a compute resource** page, select a compute resource to run the VM and click **Next**.

6. On the **Select storage** page:

    1. Select a **VM Storage Policy**.

    2. Select the destination datastore for the VM configuration files and virtual disks.

    3. Click **Next**.

7. On the **Select compatibility** page, for the **Compatible with** configuration setting, select **ESXi 6.0 and later** and click **Next**.

8. On the **Select a guest OS** page:

    1. For **Guest OS Family**, select **Windows**.

    2. For **Guest OS Version**, select **Microsoft Windows Server 2019**. If **Microsoft Windows Server 2019** is not available, select **Microsoft Windows Server 2016**.

    3. Click **Next**.

9. On the **Customize hardware** page, configure the VM hardware using the information below and click **Next**.

    1. For **New Hard disk**, specify 30 GB or greater.

    2. For **New CD\DVD Drive**:

3. Select **Datastore ISO File**.

4. Select the Windows Server 2019 ISO file you uploaded to your datastore and click **OK**.

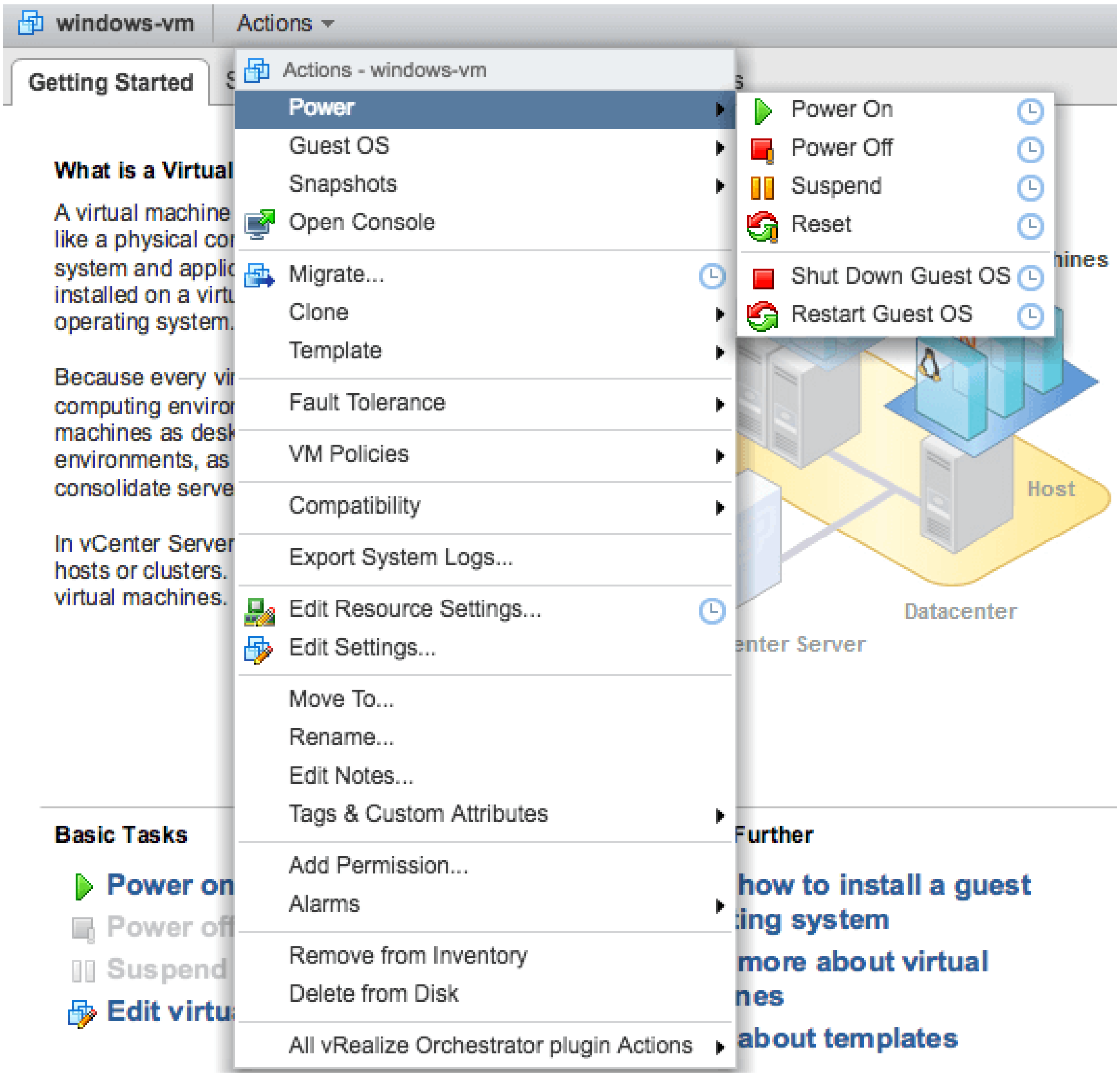
5. Enable the **Connect At Power On** check box.

10. Review the configuration settings on the **Ready to complete** page and click **Finish**.

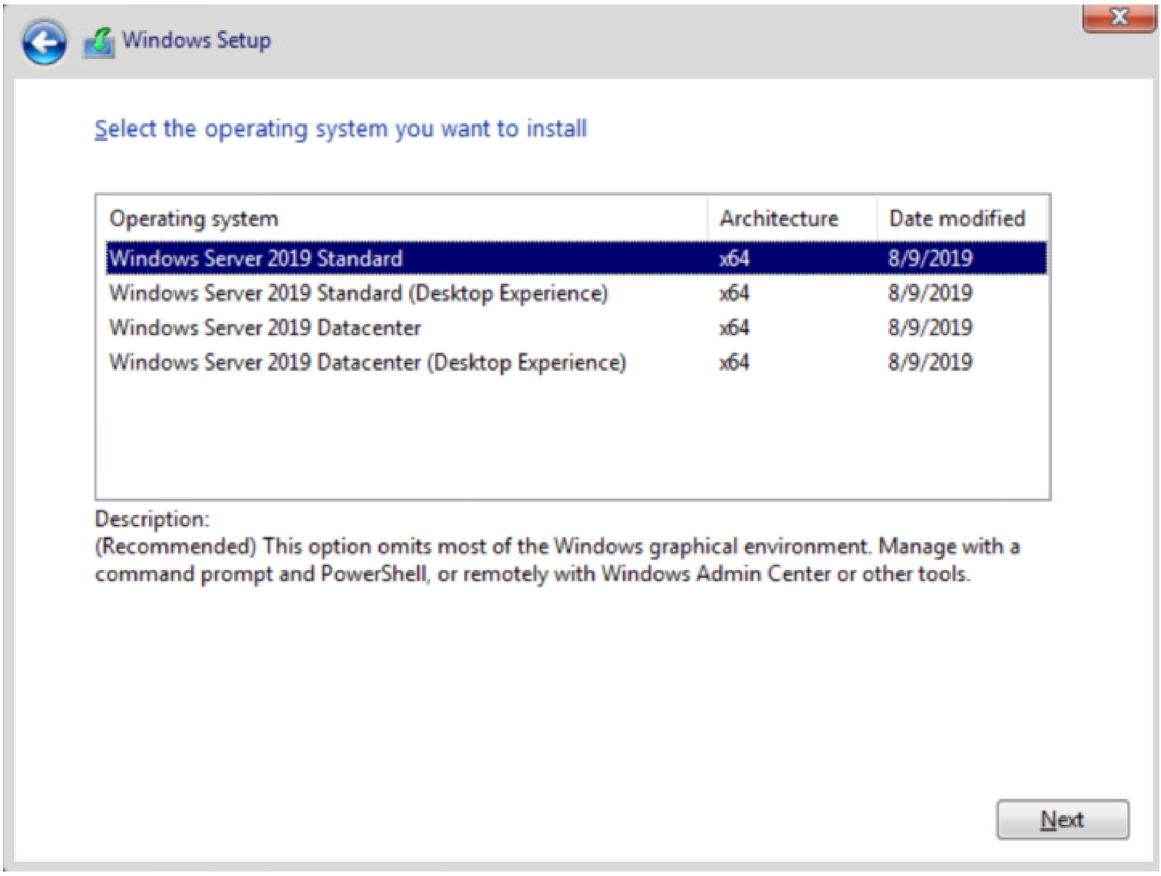
Install Windows Server

To install Windows Server on the base VM:

1. After creating the VM, click **Power** > **Power On** in the **Actions** tab for your VM.



2. Select **Windows Server 2019 Standard**.



Select the operating system you want to install

Operating system	Architecture	Date modified
Windows Server 2019 Standard	x64	8/9/2019
Windows Server 2019 Standard (Desktop Experience)	x64	8/9/2019
Windows Server 2019 Datacenter	x64	8/9/2019
Windows Server 2019 Datacenter (Desktop Experience)	x64	8/9/2019

Description:
(Recommended) This option omits most of the Windows graphical environment. Manage with a command prompt and PowerShell, or remotely with Windows Admin Center or other tools.

3. Select **Custom installation**.

4. Complete the installation process and enter a password for the Administrator user.

Verify OS

To verify that you are using the correct OS version, run the following PowerShell command on the base VM:

```
[System.Environment]::OSVersion.Version
```

The output should display the following:

```
[System.Environment]::OSVersion.Version
Major    Minor    Build    Revision
----     ----     -----    --------
10        0       17763    0
```

Install VMware Tools

To install VMware Tools on the base VM:

1. In the vSphere Web Client, right-click the base VM and select **Guest OS** > **Install VMware Tools**.

2. When prompted, choose **Mount**.

3. Log in to the the VM.

4. Navigate to the `D:` drive.

5. To install the VMware Tools, run:

   ```
   setup64.exe
   ```

6. Wait for a pop-up dialog.

   > **Note:** The VMware Tools install window might display behind the command line window. To avoid this, shrink or minimize the command line window while you are waiting.

7. When prompted, follow the instructions to finish the install.

8. To complete the installation, restart the VM.

Step 2: Configure the Base VM

Install all the Windows Server updates so that you have the latest, most secure, version of the Windows Server operating system.

To configure the base VM network settings and install Windows updates:

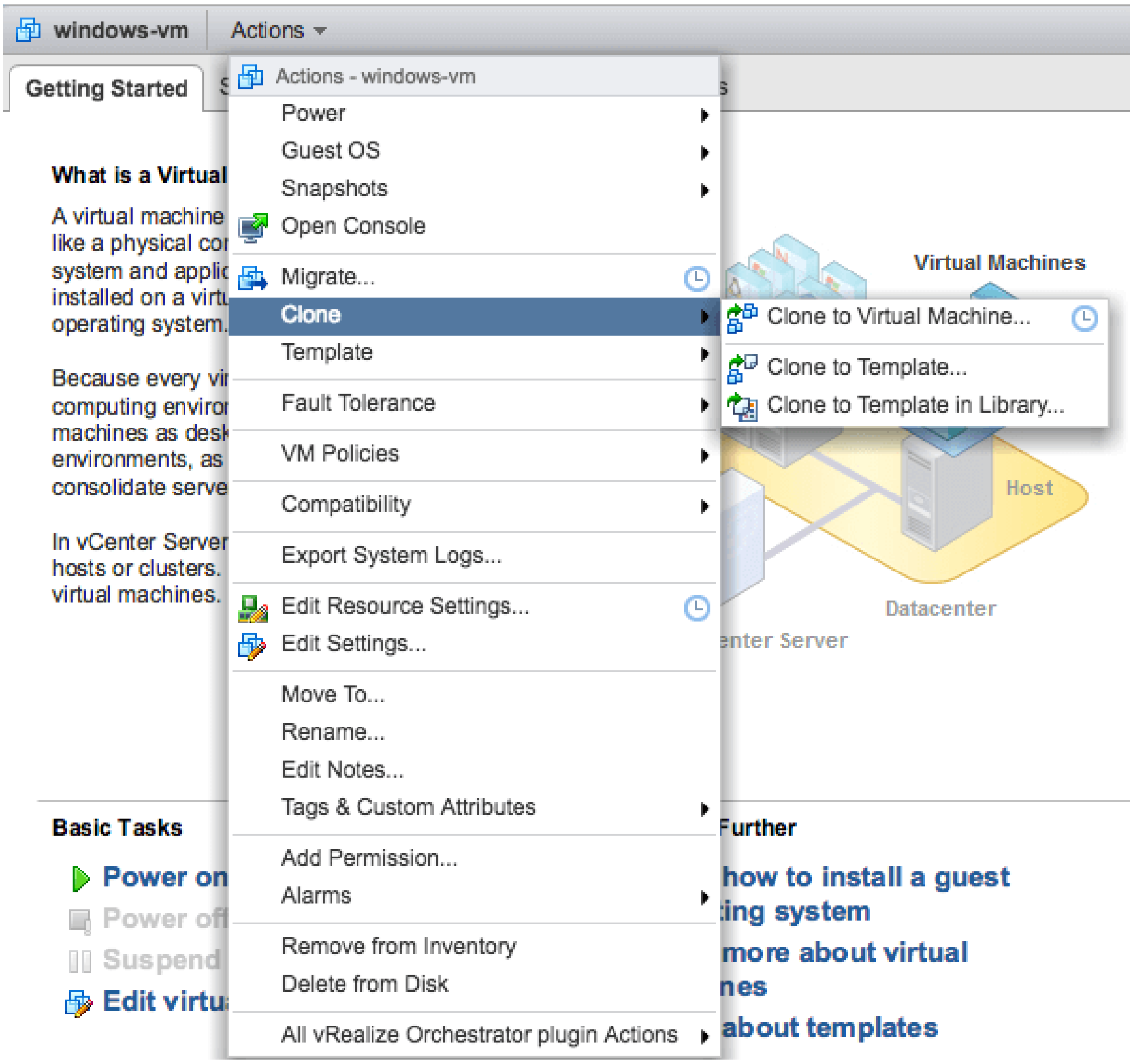
1. From the vSphere Web Client, right-click the base VM and select **Open Remote Console**.
2. On the command line, enter `sconfig` to run the **SConfig** utility.
3. On the **Server Configuration** page, enter `8` for **Network Settings**.
4. On the **Network Settings** pane:
   1. Under **Available Network Adapters**, enter the number that corresponds to your network adapter.
   2. Enter `1` to set the network adapter address.
   3. Enter `S` to set a static IP address.
   4. Enter a static IP address.
   5. Enter a subnet mask or leave the field blank to use the default value.
   6. Enter the default gateway.
   7. Enter `2` to set a DNS server.
   8. Enter a DNS server and click **OK** on the resulting message box.
   9. (Optional) Enter a secondary DNS server.
   10. Enter `4` to return to the main menu.
5. On the **Server Configuration** page, enter `6` for **Download and Install Updates**.
6. Enter `A` to search for all updates.
7. For **Select an option**, enter `A` to install all updates. You might need to restart the base VM while installing the updates, if so re-run **Download and Install Updates** after reboot until no more updates are found.

8. From the vSphere Web Client select **Actions > Edit Settings** on the VM.
9. In the **CD/DVD drive 1** row disable the **Connected** check box. Do not remove the CD/DVD drive.
10. Restart the VM.
11. After the VM restarts verify that you can ping the IP address you assigned to the base VM.

Step 3: Clone the Base VM

To clone the base VM:

1. From the vSphere Web Client, power down the base VM. This is important because your base VM and the clone VM you create share the same IP address.
2. Right-click the base VM.
3. Select **Clone** > **Clone to Virtual Machine**. This clone is your target VM.



4. Save the base VM. You run Windows updates on this VM for future stemcells.
5. Take snapshots of both your base and target VMs. If there is an issue when you run stembuild, use these snapshots to revert to a clean state. For more information, see Managing snapshots in vSphere Web Client in the *VMware Knowledge Base*.

6. To start the target VM, click **Power** > **Power On** in the **Actions** tab for your VM.

Step 4: Construct the BOSH Stemcell

To create your BOSH stemcell, complete the following tasks:

1. If you are running govc, a CLI for vSphere operations, unset any `GOVC_` environment variables before you run `stembuild construct`.
   For each of the following environment variables, run `echo $GOVC_VAR` to record its current setting, and then `export GOVC_VAR=` to temporarily unset it:
   - `GOVC_USERNAME`
   - `GOVC_INSECURE`
   - `GOVC_PASSWORD`
   - `GOVC_URL`

   Explanation: Local `GOVC` parameters that are set to configure `govc` take precedence over command-line parameters that `stembuild construct` passes in to `govc`, which can cause `stembuild construct` to fail.
2. Collect the following information from the vCenter Web Client **Inventory** > **VMs and Templates** tab:
   - Target VM IP address
   - Target VM username
   - Target VM password
   - vCenter inventory path to the target VM in the `/YOUR-DATA-CENTER/vm/YOUR-FOLDER/YOUR-VM` format, where:
     - `YOUR-DATA-CENTER` is the name of the data center.
     - `vm` is a static string.
     - `YOUR-FOLDER` is the name of the folder that contains the VM. If the target VM is not in a folder, use the `/YOUR-DATA-CENTER/vm/YOUR-VM` format instead.
     - `YOUR-VM` is the name of the target VM.
   - vCenter username
   - vCenter password
   - vCenter URL

   > **Note:** The target VM must be routable from your local host. Before running the `construct` command, ensure you are logged out of the target VM.

3. To construct the BOSH stemcell, run the following command from the folder where you downloaded the CLI:

```
./STEMBUILD-BINARY construct  ^
  -vm-ip 'TARGET-VM-IP'  ^
  -vm-username TARGET-USERNAME  ^
  -vm-password 'TARGET-VM-PASSWORD'  ^
  -vcenter-url VCENTER-URL  ^
  -vcenter-username VCENTER-USERNAME  ^
  -vcenter-password 'VCENTER-PASSWORD'  ^
  -vm-inventory-path 'INVENTORY-PATH'  ^
  -vcenter-ca-certs 'CUSTOM-CERTS-PATH'
```

Where:

- `STEMBUILD-BINARY` is the `stembuild` file for the version of your local host operating system and the version of the stemcell that you want to build. For example, `stembuild-windows-2019-2`.
- `TARGET-VM-IP` is the IP address of your target VM.
- `TARGET-USERNAME` is the username of an account with administrator privileges.
- `TARGET-VM-PASSWORD` is the password for the administrator account. The password must be enclosed in single quotes.
- `VCENTER-URL` is the URL of your vCenter.
- `VCENTER-USERNAME` is the username of your account in vCenter.
- `VCENTER-PASSWORD` is your password. The password must be enclosed in single quotes.
- `INVENTORY-PATH` is the vCenter inventory path to the target VM.
- `CUSTOM-CERTS-PATH` is the file path to custom CA certificates for the destination vCenter. The `-vcenter-ca-certs` flag is optional.

For more information, see Authenticate into a Destination vCenter Using CA Certificates below.

For example:

```
./STEMBUILD-BINARY construct -vm-ip '192.0.2.254' -vm-username user001
-vm-password 'P1a2s3Sword5' -vcenter-url example.com -vcenter-username
user002 -vcenter-password 'P1a2s3Sword5' -vm-inventory-path '/datacente
r/vm/folder/test-vm'
2020-01-29T08:52:26.4523812-08:00 Successfully created stemcell version
file.
Finished executing setup script.
WinRM has been disconnected so the VM can reboot. Preparing the VM to b
e shutdown.
2020-01-29T16:53:27.505532+00:00 Still preparing VM...
2020-01-29T16:54:27.94085+00:00 Still preparing VM...
2020-01-29T16:55:28.374568+00:00 Still preparing VM...
Stembuild construct has finished running and the VM has now been shutdo
wn. Run 'stembuild package' to finish building the stemcell.
```

Note: Your `stembuild construct` operation may take up to an hour to complete. Although the WinRM connection terminates during `construct`

> execution, `construct` is still running. Do not attempt to re-run the `construct` command.

The following is an example of the messages displayed as `stembuild construct` runs and completes successfully:

```
2020-01-01T00:00:00 Successfully created stemcell version file.
Finished executing setup script.
WinRM has been disconnected so the VM can reboot. Preparing the VM to be shutdown.
2020-01-01T00:01:00 Still preparing VM...
2020-01-01T00:02:00 Still preparing VM...
2020-01-01T00:03:00 Still preparing VM...
Stembuild construct has finished running and the VM has now been shutdown. Run 'stembuild package' to finish building the stemcell.
```

4. (Optional) To monitor the status of your `construct` job complete the following tasks:
    1. Log in to the target VM.
    2. Start PowerShell.
    3. Run:

       ```
       Get-Content -Path "C:\provision\log.log" -Wait
       ```

5. Restore the values of any `govc` variables you unset previously.

For more information about `stembuild construct`, see stembuild construct in the Cloud Foundry stembuild GitHub repository.

Step 5: Package the BOSH Stemcell

To package the BOSH stemcell:

1. Gather the vCenter Web Client **VMs and Templates** information that you recorded in the previous step:
    - vCenter inventory path to the target VM in the `/YOUR-DATA-CENTER/vm/YOUR-FOLDER/YOUR-VM` format, where:
        - `YOUR-DATA-CENTER` is the name of the data center.
        - `vm` is a static string.
        - `YOUR-FOLDER` is the name of the folder that contains the VM. If the target VM is not in a folder, use the `/YOUR-DATA-CENTER/vm/MY-VM` format instead.
        - `YOUR-VM` is the name of the target VM.
    - vCenter username
    - vCenter password
    - vCenter URL

2. Shut down the VM. If you do not shut down the VM before you continue to the next step, the package command fails stating that the storage location for the VM could not be read.
3. To package the BOSH stemcell, run the following PowerShell command from your local host:

   ```
   ./STEMBUILD-BINARY package ^
     -vcenter-url VCENTER-URL ^
     -vcenter-username VCENTER-USERNAME ^
     -vcenter-password VCENTER-PASSWORD ^
     -patch-version PATCH-VERSION ^
     -vm-inventory-path 'INVENTORY-PATH'  ^
     -vcenter-ca-certs 'CUSTOM-CERTS-PATH'
   ```

   Where:

   - `STEMBUILD-BINARY` is the `stembuild` file for the version of your local host operating system and the version of the stemcell that you want to build. For example, `stembuild-windows-2019-2`.
   - `VCENTER-URL` is the URL of your vCenter.
   - `VCENTER-USERNAME` is the username of your account in vCenter.
   - `VCENTER-PASSWORD` is your password. The password must be enclosed in single quotes.
   - `PATCH-VERSION` is the patch version for the stemcell being built.
   - `PATCH-VERSION` can be specified as an unquoted version name, for example `2019.12.3`, or as a quoted patch number, for example `“3”`.
   - `INVENTORY-PATH` is the vCenter inventory path to the target VM.
   - `CUSTOM-CERTS-PATH` is the file path to custom CA certificates for the destination vCenter. The `-vcenter-ca-certs` flag is optional.For more information, see Authenticate into a Destination vCenter Using CA Certificates below.

   Note: This command creates a stemcell on your local host in the folder where you ran the command and might take up to 30 minutes to complete.

For more information about `stembuild package`, see stembuild package in the Cloud Foundry stembuild GitHub repository.

Step 6: Upload the BOSH Stemcell to Ops Manager

To upload the BOSH stemcell to Ops Manager:

1. In Ops Manager, navigate to **Stemcell Library**.
2. Upload your BOSH stemcell.
3. Deploy the TAS for VMs [Windows] or TKGI tile.

For more information about `stembuild`, see Stembuild in the Cloud Foundry stembuild GitHub repository.

Monthly Stemcell Upgrades

After Microsoft releases operating system updates, you should upgrade your BOSH stemcell. Microsoft typically releases Windows updates on the second Tuesday of each month.

To upgrade your BOSH stemcell:

1. Install Windows Updates on the base VM.
2. Clone the Base VM.
3. Construct the BOSH Stemcell.
4. Package the BOSH Stemcell.
5. Replace the existing stemcell in the Ops Manager stemcell library with this new stemcell.
6. Deploy the TAS for VMs [Windows] or TKGI tile.

Known Issues

Authentication Error with Special Characters in stembuild Commands

Symptom

You authenticate with vCenter and see this error:

```
./out/stembuild: ServerFaultCode: Cannot complete login due to an incorrect u
ser name or password.
vcenter_client - unable to validate url: vcenter.example.com
```

Explanation

`stembuild` uses govc libraries. These libraries cannot parse the special characters `/`, `#`, and `:`. This results in errors when authenticating with vCenter.

You might also experience this issue on Windows if your password includes a single quote character, `'`. This also affects the Inventory path if it contains a single quote or a space.

Workaround

If your vCenter username or password contains `/`, `#`, or `:`, or `'` on Windows, set these environment variables:

- For **Linux**:

  ```
  export GOVC_USERNAME=VCENTER-USERNAME
  export GOVC_PASSWORD=VCENTER-PASSWORD
  ```

- For **Windows**:

  ```
  set GOVC_USERNAME=VCENTER-USERNAME
  set GOVC_PASSWORD=VCENTER-PASSWORD
  ```

```
set GOVC_PATH=VCENTER-INVENTORY-PATH
```

Where:

- `VCENTER-USERNAME` is your vCenter account username. For example, `johndoe`.
- `VCENTER-PASSWORD` is your vCenter account password. For example, `pass#word`.
- `VCENTER-INVENTORY-PATH` is the location of your VM in the cluster inventory.

If you use other special characters, add single quotes around the input parameters, or set them in an environment variable as described above.

For example:

- For **Linux**:

  ```
  ./STEMBUILD-BINARY package \
    -vcenter-url VCENTER-URL \
    -vcenter-username 'admin@' \
    -vcenter-password VCENTER-PASSWORD \
    -patch-version PATCH-VERSION \
    -vm-inventory-path 'INVENTORY-PATH' \
    -vcenter-ca-certs 'CUSTOM-CERTS-PATH'
  ```

  Where:

  - `STEMBUILD-BINARY` is the `stembuild` file for the version of your local host operating system and the version of the stemcell that you want to build. For example, `stembuild-windows-2019-2`.
  - `VCENTER-URL` is the URL of your vCenter.
  - `VCENTER-PASSWORD` is your password. The password must be enclosed in single quotes.
  - `PATCH-VERSION` is the patch version for the stemcell being built.
  - `PATCH-VERSION` can be specified as an unquoted version name, for example `2019.12.3`, or as a quoted patch number, for example `“3”`.
  - `INVENTORY-PATH` is the vCenter inventory path to the target VM.
  - `CUSTOM-CERTS-PATH` is the file path to custom CA certificates for the destination vCenter. The `-vcenter-ca-certs` flag is optional. For more information, see Authenticate into a Destination vCenter Using CA Certificates below.

- For **Windows**:

  ```
  set GOVC_PASSWORD=VCENTER-PASSWORD
  ./STEMBUILD-BINARY package ^
    -vcenter-url VCENTER-URL ^
    -vcenter-username %GOVC_USERNAME% ^
    -vcenter-password %GOVC_PASSWORD% ^
    -patch-version PATCH-VERSION ^
    -vm-inventory-path %GOVC_PATH% ^
    -vcenter-ca-certs 'CUSTOM-CERTS-PATH'
  ```

  Where:

- VCENTER-PASSWORD is your vCenter account password. For example, A_Strange!PAssword@Here#1.
- STEMBUILD-BINARY is the stembuild file for the version of your local host operating system and the version of the stemcell that you want to build. For example, stembuild-windows-2019-2.
- VCENTER-URL is the URL of your vCenter.
- PATCH-VERSION is the patch version for the stemcell being built. PATCH-VERSION can be specified as an unquoted version name, for example 2019.12.3, or as a quoted patch number, for example "3".
- CUSTOM-CERTS-PATH is the file path to custom CA certificates for the destination vCenter. The -vcenter-ca-certs flag is optional. For more information, see Authenticate into a Destination vCenter Using CA Certificates below.

Note: Windows environment variables do not automatically override vCenter command line parameters, so you must specify the environment variables in the vCenter command as shown above.

Authenticate into a Destination vCenter Using CA Certificates

Symptom

You are running stembuild from a location outside of your destination vCenter and stembuild is unable to authenticate.

Explanation

You are running stembuild from a location with different vCenter permissions than your target VM.

Solution

You must provide stembuild with the CA certificates needed to authenticate into your destination vCenter.

To stage CA certificates for stembuild to use to access a target VM in the destination vCenter:

1. Open your destination vCenter's homepage.
2. Go to the bottom of the grey box on the right side of the vCenter homepage.
3. Select **Download trusted root CA certificates**.
4. Copy the downloaded CA certificates to a path accessible to stembuild.

To use the staged CA certificates with stembuild, include the optional -vcenter-ca-certs flag in your stembuild command as shown in the example construct and package command lines above.

Windows Diego Cells in Isolation Segments

This topic describes how to run Windows Diego Cells in isolation segments, which are compartmentalized resource pools for Diego Cells.

Diego Cells in one isolation segment share routing, computing, and logging resources with other Diego Cells in the same segment, and do not use resources from other isolation segments.

Overview

To run Windows Diego Cells in multiple isolation segments, you must create and install multiple VMware Tanzu Application Service for VMs [Windows] tiles and configure each to run in a different isolation segment.

To create multiple copies of the TAS for VMs [Windows] tile, see Replicate a Tile.

To associate a TAS for VMs [Windows] tile with an isolation segment, so that its Diego Cells run in that segment, see Assign a Tile to an Isolation Segment.

Prerequisites

To run a Windows Diego Cell in a particular isolation segment, the isolation segment must first already exist in the Cloud Controller database (CCDB) before the Windows Diego Cell can be assigned to it.

To create an isolation segment in the CCDB, see Create an Isolation Segment in *Managing Isolation Segments*.

Replicate a Tile

To make multiple copies of the TAS for VMs [Windows] tile that you can assign to different isolation segments, you must use the Replicator tool.

To use the Replicator tool:

1. Download the Replicator tool from the VMware Tanzu Application Service for VMs [Windows] page on VMware Tanzu Network.
2. Navigate to the directory where you downloaded the Replicator tool.
3. Replicate the tile by running:

   ```
   replicator-windows -name "TILE-NAME" -path PATH-TO-ORIGINAL -output PATH-TO-COPY
   ```

   Where:

   - `TILE-NAME` is a unique name for the new TAS for VMs [Windows] tile. The name must be ten or fewer characters in length and contain only alphanumeric characters, dashes, underscores, and spaces.
   - `PATH-TO-ORIGINAL` is the absolute path to the original `.pivotal` TAS for VMs [Windows] tile you downloaded from VMware Tanzu Network.
   - `PATH-TO-COPY` is the absolute path for the copy of the `.pivotal` TAS for VMs [Windows] tile that the Replicator tool produces.

   For example:

   ```
   C:\Users\admin\> replicator-windows -name "Shiny Tile" ^
   -path c:\downloads\TileOriginal.pivotal ^
   -output c:\temp\TileCopy.pivotal
   ```

4. Install and configure the Windows isolation segment, using the new `.pivotal` file and, following the procedures in *Installing and Configuring TAS for VMs [Windows]*, starting with the **Import a Product** step of Install the Tile.

Assign a Tile to an Isolation Segment

To assign a TAS for VMs [Windows] tile to an isolation segment:

1. Select **App Containers**.
2. Under **Segment name**, enter the name for the isolation segment to associate the tile with. If you are creating a new isolation segment, ensure that this name is unique across your deployment.
3. (Optional) To limit the number of log lines each app instance in your isolation segment can generate per second, select **Enable** under **App log rate limit (beta)** and enter an integer for **Maximum app log lines per second**. At minimum, VMware recommends using the default limit of `100`. This feature is disabled by default. Enabling this feature prevents app instances from overloading the Loggregator Agent with logs, so the Loggregator Agent does not drop logs for other app instances. Enabling this feature also prevents apps from reporting inaccurate app metrics in the cf CLI or increasing the CPU usage on the Diego Cell VM. You can use this value to monitor the `AppInstanceExceededLogRateLimitCount` metric for the number of app instances that exceed the log rate limit.
4. Click **Save**.

Upgrading TAS for VMs [Windows] and Windows Stemcells

This topic describes how to upgrade the VMware Tanzu Application Service for VMs [Windows] tile and update the Windows stemcell.

For information about migrating apps to TAS for VMs [Windows] from TAS for VMs [Windows] 2012 R2, see Migrating Apps to TAS for VMs [Windows].

Rotating Credentials

The VMware Tanzu Application Service for VMs (TAS for VMs) tile handles all of the credentials for TAS for VMs [Windows].

To rotate your credentials in TAS for VMs [Windows], you must re-deploy TAS for VMs and TAS for VMs [Windows]. To re-deploy TAS for VMs and TAS for VMs [Windows]:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click **Review Pending Changes**.
3. Select the TAS for VMs and TAS for VMs [Windows] tiles and review the changes. For more information, see Reviewing Pending Product Changes.
4. Click **Apply Changes**.

Upgrade TAS for VMs [Windows]

To upgrade TAS for VMs [Windows]:

1. Download the latest TAS for VMs [Windows] version from the VMware Tanzu Application Service for VMs [Windows] page on VMware Tanzu Network. For more information about upgrading your Ops Manager products, see Upgrading Ops Manager Products in *Upgrading TAS for VMs and Other Ops Manager Products*.
2. If necessary, configure the product. For more information about configuring TAS for VMs [Windows], see Installing and Configuring TAS for VMs [Windows].
3. Click **Review Pending Changes**.
4. Select the TAS for VMs [Windows] tile and review the changes. For more information, see Reviewing Pending Product Changes.
5. Click **Apply Changes**.

Upgrade the Windows Stemcell

To upgrade your Windows stemcell:

1. Retrieve the stemcell by following the steps for your IaaS:
    - For vSphere, you must build your own stemcell. For more information, see Creating a Windows Stemcell for vSphere in the BOSH documentation.
    - For AWS, GCP, and Azure, go to Stemcells (Windows) on VMware Tanzu Network.
2. Go to **Stemcell Library**.
3. Click **Import Stemcell** to import the stemcell file.
4. When prompted, enable the Ops Manager product check box to stage your stemcell.
5. Click **Apply Stemcell to Products**.

For more information about Windows stemcells, see Downloading or Creating Windows Stemcells.

Migrating Apps to TAS for VMs [Windows]

This topic describes the process of migrating apps running on VMware Tanzu Application Service for VMs [Windows] 2012 R2 stemcells to run on TAS for VMs [Windows] stemcells, which run Windows Server 2019.

> **Note**: TAS for VMs [Windows] 2012 R2 has reached its end of availability (EoA) and is no longer generally available.

VMware recommends you use the blue-green deployment method for high availability. For more information about blue-green deployments, see Using Blue-Green Deployment to Reduce Downtime and Risk.

Step 1: Install and Deploy the TAS for VMs [Windows] Tile

To install and deploy the TAS for VMs [Windows] tile, follow steps 1 and 2 of Installing and Configuring TAS for VMs [Windows].

Step 2: Push App to TAS for VMs [Windows] Diego Cells

To redeploy a running app with zero downtime using the blue-green method:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI) by running:

   ```
   cf login
   ```

2. Choose your org and space.
3. Navigate to the location of your app.
4. To find the name of the existing TAS for VMs [Windows] 2012 R2 app you are migrating to TAS for VMs [Windows] Diego, run:

   ```
   cf apps
   ```

5. Create a name for the replacement TAS for VMs [Windows] app. VMware recommends you append `-green` to your existing app name, using the following format:

   ```
   APP-NAME-green
   ```

   Where `APP-NAME` is the name of the existing TAS for VMs [Windows] Diego 2012 R2 app.

6. To push your app to a TAS for VMs [Windows] Diego Cell using your newly created name, run:

   ```
   cf push APP-NAME-GREEN -s windows -b BUILDPACK -n HOSTNAME --no-start --no-route
   ```

   Where:

   - `APP-NAME-GREEN` is the newly created “green” name for your app.
   - `BUILDPACK` is your custom buildpack. Specify the buildpack either by name or GitHub URL with an optional branch or tag.
   - `HOSTNAME` is the name of your app’s subdomain. For example, if `example.com` is your domain and you want the URL to your app to be `http://my-app.example.com`, then specify `my-app` as the `HOSTNAME`.

   For example:

   ```
   C:\Users\admin\> cf push ExampleApp-green -s windows ^
   -b https://github.com/cloudfoundry/binary-buildpack.git ^
   -n my-app --no-start --no-route
   ```

   > **Note:** The `--no-start` and `--no-route` parameters included in this `cf push` command are required for the this procedure. `--no-start` is used to create the instance VMs and not start the app, and `--no-route` is used to prevent the push command from automatically mapping a route to the app. For more information about `cf push`, see cf push in the Cloud Foundry CLI Reference Guide.

> **Note:** The `windows` stack is a renaming of the old `windows2016` stack. These stacks are identical and differ in name only. If the `windows` stack is available, specify `-s windows`. Otherwise, specify `-s windows2016`. In TAS for VMs [Windows] v2.5 and later, the `windows2016` stack is deprecated. For more information, see Deprecation of the windows2016 Stack in *VMware Tanzu Application Service for VMs [Windows] v2.5 Release Notes*.

7. To configure the router so all incoming requests go to both `APP-NAME` and `APP-NAME-green`, run:

   ```
   cf map-route APP-NAME-green DOMAIN -n HOSTNAME
   ```

   Where:

   - `APP-NAME-green` is the name of the new “green” version of your app.
   - `DOMAIN` is your domain name. For example, `example.com`.
   - `HOSTNAME` is the name of your app’s subdomain. For example, if `example.com` is your domain and you want the URL to your app to be `http://my-app.example.com`, then specify `my-app` as the `HOSTNAME`.

   > **Note:** When using the `cf map-route` command, you must specify domain name after specifying app name.

   For additional information about `cf map-route`, see [cf map-route](http://cli.cloudfoundry.org/en-US/cf/map-route.html) in the Cloud Foundry CLI Reference Guide.

8. To start the green app, run:

   ```
   cf start APP-NAME-green
   ```

   Where `APP-NAME-green` is the name of the new “green” version of your app.

9. To confirm that both your `APP-NAME` and `APP-NAME-green` apps are running, run:

   ```
   cf apps
   ```

   If you experience a problem, see Troubleshooting App Deployment and Health.

10. To unmap the original app’s route, run:

    ```
    cf unmap-route APP-NAME DOMAIN -n HOSTNAME
    ```

    Where:

    - `APP-NAME` is the name of the existing version of your app you want to replace with `APP-NAME-green`.
    - `DOMAIN` is your domain name, for example, `example.com`.

  - `HOSTNAME` is the name of your app's subdomain. For example, if `example.com` is your domain and the existing app is currently accessed using the URL `http://my-app.example.com`, then specify `my-app` as the `HOSTNAME`.

For additional information about `cf-unmap-route`, see cf unmap-route in the Cloud Foundry CLI Reference Guide.

Step 3: Delete App from TAS for VMs [Windows] 2012 R2 Server Diego Cells

To delete the original app:

1. Run:

   ```
   cf delete APP-NAME
   ```

   Where `APP-NAME` is the name of the app that you have replaced with `APP-NAME-green`.

   Note: To also delete any mapped routes, run the command with the `-r` flag.

   For additional information about `cf delete`, see [cf delete](http://cli.cloudfoundry.org/en-US/cf/delete.html) in the Cloud Foundry CLI Reference Guide.

Step 4: (Optional) Uninstall Old Tile

Once you have migrated all of your apps and you are no longer using the TAS for VMs [Windows] 2012 R2 tile, a Ops Manager operator uninstall the tile.

To uninstall the tile:

1. From the Installation Dashboard, click the trash icon on the tile to remove that product. In the **Delete Product** dialog box that appears, click **Confirm**.
2. Click **Review Pending Changes**.
3. Select the TAS for VMs [Windows] tile and review the changes. For more information, see Reviewing Pending Product Changes.
4. Click **Apply Changes**.

   Note: After you delete a product, the product tile is removed from the installation and the Installation Dashboard. However, the product appears in the Available Products view.

Troubleshooting Windows Diego Cells

This topic describes how to troubleshoot Windows Diego Cells deployed by VMware Tanzu Application Service for VMs [Windows] (TAS for VMs [Windows]).

Installation Issues

This section describes issues that may occur during the installation process.

Missing Local Certificates for Windows File System Injector

Symptom

You run the `winfs-injector` and see the following error about certificates:

```
Get https://auth.docker.io/token?service=registry.docker.io&
scope=repository:cloudfoundry/windows2016fs:pull: x509:
failed to load system roots and no roots provided
```

Explanation

Local certificates are needed to communicate with Docker Hub.

Solution

Install the necessary certificates on your local machine. On Ubuntu, you can install certificates with the `ca-certificates` package.

Outdated Version for Windows File System Injector

Symptom

You run the `winfs-injector` and see the following error about a missing file or directory:

```
open ...windows2016fs-release/VERSION: no such file or directory
```

Explanation

You are using an outdated version of the `winfs-injector`.

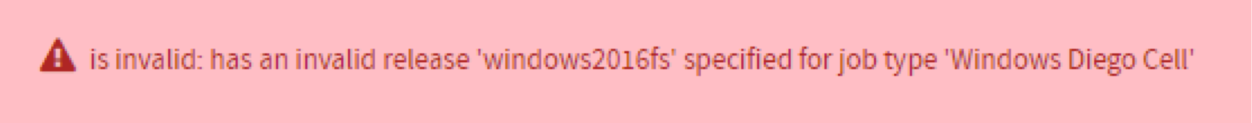
Solution

From the VMware Tanzu Application Service for VMs [Windows] page on VMware Tanzu Network, download the recommended version of **File System Injector tool** for the tile.

Missing Container Image

Symptom

You click the **+** icon in Ops Manager to add the TAS for VMs [Windows] tile to the Installation Dashboard and see the following error:



⚠ is invalid: has an invalid release 'windows2016fs' specified for job type 'Windows Diego Cell'

Explanation

The product file that you are trying to upload does not contain the Windows Server container base image.

Solution

1. Delete the product file listing from Ops Manager by clicking its trash can icon under **Import a Product**.

2. Follow the TAS for VMs [Windows] installation instructions to run the `winfs-injector` tool locally on the product file. This step adds the Windows Server container base image to the product file, requires internet access, and can take up to 20 minutes. For more information, see Install the Tile in *Installing and Configuring TAS for VMs [Windows]*.
3. Click **Import a Product** to upload the injected product file.
4. Click the **+** icon next to the product listing to add the TAS for VMs [Windows] tile to the Installation Dashboard.

Upgrade Issues

This section describes issues that may occur during the upgrade process.

Failure to Create Containers When Upgrading with Shared Microsoft Base Image

Symptom

The pre-start script for the `windowsfs` job fails, and the upgrade fails with the following output:

```
Task 308031 | 13:47:04 | Preparing deployment: Preparing deployment (00:00:03)

Task 308031 | 13:47:11 | Preparing package compilation: Finding packages to compile (0
0:00:00)

Task 308031 | 13:47:21 | Updating instance windows_diego_cell: windows_diego_cell/44c5
841f-7580-4e9c-9856-89fcbe08ab0d (2) (canary) (00:00:35)

L Error: Action Failed get_task: Task 59ba76d1-14c5-4d7b-681c-08b9ec4bd64d result: 1 o
f 10 pre-start scripts failed. Failed Jobs: windows1803fs. Successful Jobs: set_kms_ho
st, groot, loggregator_agent_windows, bosh-dns-windows, rep_windows, winc-network-180
3, set_password, enable_ssh, enable_rdp.

Task 308031 | 13:47:56 | Error: Action Failed get_task: Task 59ba76d1-14c5-4d7b-681c-0
8b9ec4bd64d result: 1 of 10 pre-start scripts failed. Failed Jobs: windows1803fs. Succ
essful Jobs: set_kms_host, groot, loggregator_agent_windows, bosh-dns-windows, rep_win
dows, winc-network-1803, set_password, enable_ssh, enable_rdp.
```

Otherwise, the post-start script for the `rep_windows` job fails, and the upgrade fails with the following output:

```
Task 8192 | 21:12:30 | Updating instance windows2019-cell: windows2019-cell/bd6d70b9-e
d1f-412f-9d49-8045627f4ab3 (0) (canary) (00:17:24)
                     L Error: Action Failed get_task: Task a9555020-1a3b-40c7-677c-d6f
c392ce135 result: 1 of 3 post-start scripts failed. Failed Jobs: rep_windows. Successf
ul Jobs: route_emitter_windows, bosh-dns-windows.
Task 8192 | 21:29:55 | Error: Action Failed get_task: Task a9555020-1a3b-40c7-677c-d6f
c392ce135 result: 1 of 3 post-start scripts failed. Failed Jobs: rep_windows. Successf
ul Jobs: route_emitter_windows, bosh-dns-windows.
```

Explanation

When upgrading between versions of Windows rootfs that have a shared Microsoft base layer, TAS for VMs [Windows] may fail to create containers.

Solution

For available workarounds, see Failure to create containers when upgrading with shared Microsoft base image in the Knowledge Base.

Forwarding Logs to a Syslog Server

You can use Windows Diego Cell logs to troubleshoot Windows Diego Cells. Windows Diego Cells generate the following types of logs:

- **BOSH job logs**, such as `rep_windows` and `consul_agent_windows`. These logs stream to the syslog server configured in the **System Logging** pane of the TAS for VMs [Windows] tile, along with other Ops Manager component logs. The names of these BOSH job logs correspond to the names of the logs emitted by Linux Diego Cells.
- **Windows event logs**. These logs stream to the syslog server configured in the **System Logging** pane of the TAS for VMs [Windows] tile.

To forward Windows logs to an external syslog server:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs [Windows] tile.
3. Select **System Logging**.
4. Under **Enable syslog for VM logs?**, select **Enable**.
5. Under **Address**, enter the hostname or IP address of your syslog server.
6. Under **Port**, enter the port of your syslog server. The default port is `514`.

   > **Note:** The host must be reachable from the TAS for VMs network. Ensure that your syslog server listens on external interfaces.

7. Under **Protocol**, select the transport protocol to use when forwarding logs.
8. Under **tls_enabled**, select enabled if you are using tcp and want tls.
9. Under **ca_cert**, add the certificate to validate connections to external server if using tls.
10. Enable the **Enable system metrics** check box. For a list of the VM metrics that the System Metric Agent emits, see System Metrics Agent in the System Metrics repository on GitHub.
11. Click **Save**.

Download Windows Diego Cell Logs

To download Windows Diego Cell logs:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs [Windows] tile.
3. Click the **Status** tab.
4. Under the **Logs** column, click the download icon for the Windows Diego Cell for which you want to retrieve logs.

5. Click the **Logs** tab.
6. When the logs are ready, click the filename to download them.
7. Unzip the file to examine the contents. Each component on the Diego Cell has its own logs directory:
    - `/consul_agent_windows/`
    - `/garden-windows/`
    - `/metron_agent_windows/`
    - `/rep_windows/`

Troubleshoot Windows Compilation VMs

BOSH automatically deletes a compilation VM after the compilation VM fails. In a vSphere environment, use one of the procedures below to troubleshoot your Windows stemcell v2019.7 and later compilation VM issues:

- Troubleshoot a Slowly-Deleted Windows Compilation VM
- Troubleshoot a Quickly-Deleted Windows Compilation VM

Troubleshoot a Slowly-Deleted Windows Compilation VM

The easiest method to troubleshoot a Windows compilation VM is to SSH into the VM before BOSH deletes it.

To troubleshoot a compilation VM from an `ssh` session:

1. Open the vSphere UI.
2. Open two different BOSH CLI terminal sessions.
3. Open Ops Manager.
4. Enable the two following settings in Ops Manager:
    - Select **Keep Unreachable Director VMs** from **BOSH Director** tile > **Director config**.
    - Select **Enable BOSH-native SSH support on all VMs** from **TAS for VMs [Windows]** tile > **VM options**.
5. Click **Apply Changes** against the TAS for VMs [Windows] tile.
6. From the first BOSH CLI terminal, monitor the BOSH task:

    ```
    watch -n 5 "bosh -d TAS-WINDOWS-DEPLOYMENT is --details | grep compilation"
    ```

    Where `TAS-WINDOWS-DEPLOYMENT` is the name of your TAS for VMs [Windows] deployment.
7. Wait until the compilation VM CID is up.
8. From the second BOSH CLI terminal, SSH to the Windows compilation VM:

    ```
    bosh -d TAS-WINDOWS-DEPLOYMENT ssh COMPILATION-NAME
    ```

Where:

- `TAS-WINDOWS-DEPLOYMENT` is the name of your TAS for VMs [Windows] deployment.
- `COMPILATION-NAME` is the name of your Windows compilation VM.

9. To prevent BOSH from deleting the compilation VM after the compilation VM fails, search for the compilation VM CID in the vSphere UI and rename it. You can now troubleshoot in this session.
10. After troubleshooting, delete the VM manually.

Troubleshoot a Quickly-Deleted Windows Compilation VM

In some situations, the Windows compilation VM might be deleted very quickly, making it impossible to SSH into the VM before BOSH deletes it.

To troubleshoot a quickly-deleted compilation VM:

1. Download an Ubuntu desktop image from Ubuntu Releases Xenial.
2. Upload the Ubuntu desktop image into your vSphere datastore.
3. Open the vSphere UI.
4. Open a BOSH CLI terminal session.
5. Open Ops Manager.
6. Enable the two following settings in Ops Manager:
   - Select **Keep Unreachable Director VMs** from **BOSH Director** tile > **Director config**.
   - Select **enable BOSH-native SSH support on all VMs** from **TAS for VMs [Windows]** tile > **VM options**.
7. Click **Apply Changes** in Ops Manager.
8. From the BOSH CLI terminal, monitor the BOSH task:

   ```
   watch -n 5 "bosh -d TAS-WINDOWS-DEPLOYMENT is --details | grep compilation"
   ```

   Where `TAS-WINDOWS-DEPLOYMENT` is the name of your TAS for VMs [Windows] deployment.
9. Wait until the compilation VM CID is up.
10. From the vSphere UI:
    1. Locate the compilation VM CID in the vSphere UI.
    2. To prevent BOSH from deleting the compilation VM after the compilation VM fails, rename the compilation VM.
    3. On the Windows compilation VM, go to **Edit settings** > **add a device CD/DVD drive** > **browse Datastore ISO file**, and select the Ubuntu desktop iso -> select **Connect at Power ON**.
    4. Go to **Edit settings** -> **VM options** tab -> **Boot Options**.
    5. Increase the **Boot Delay** to `10000 milliseconds`.

    6. Select **Force BIOS Setup**.
    7. Select **Start/Restart** to restart the VM.

11. On the BIOS setup screen, boot with the CD-ROM Drive.
12. After Ubuntu desktop starts, select **try Ubuntu** and launch a terminal.
13. In the terminal, run:

    ```
    sudo fdisk -l
    sudo mkdir /mnt/windows
    sudo mount /dev/sda1 /mnt/windows
    ```

14. You can now troubleshoot inside this session by exploring the contents of the Windows VMs file system in `/mnt/windows`.
15. After troubleshooting, delete the VM manually.

Installing Isolation Segment

In this section:

- Installing Isolation Segment
- Managing Isolation Segments
- Routing for Isolation Segments

Installing Isolation Segment

This topic tells you how to install the Isolation Segment tile, which allows operators to isolate deployment workloads into dedicated resource pools called *isolation segments*.

Installing the tile installs a single isolation segment. However, you can install multiple isolation segments using the Replicator tool documented in Step 4.

After installing the tile, you must follow the procedure in Register an Isolation Segment in *Managing Isolation Segments* to create the isolation segment in the Cloud Controller database (CCDB). The topic also includes information about managing an isolation segment.

For more information about how isolation segments work, see Isolation Segments in *TAS for VMs Security*.

Step 1: (Optional) Configure Routing

By default, the VMware Tanzu Application Service for VMs (TAS for VMs) Gorouter handles traffic for your isolation segment. However, you can deploy a dedicated Gorouter for your isolation segment instead. For information about configuring and managing routing for isolation segments, see Routing for Isolation Segments.

To deploy a dedicated Gorouter:

1. Add a load balancer in front of the TAS for VMs Gorouter. The steps to do this depend on your IaaS, but the setup of the load balancer must mirror the setup of the load balancer for the TAS for VMs Gorouter that you configured in the TAS for VMs tile.

2. Create a wildcard DNS entry for traffic routed to any app in the isolation segment. For example, `*.iso.example.com`.

3. Attach the wildcard DNS entry to the load balancer you created.

Step 2: Install the Tile

To install the Isolation Segment tile:

1. Download the product file from the Isolation Segment page of VMware Tanzu Network.

2. Go to your Ops Manager URL in a browser to log in to the Ops Manager Installation Dashboard.

3. Click **Import a Product** and select the downloaded product file.

4. Under **Isolation Segment** in the left column, click the **+** icon.

Step 3: Configure the Tile

Click the orange **Isolation Segment** tile to start the configuration process.

Assign AZs and Networks

In the **Assign AZ and Networks** pane, you assign jobs to your Availability Zones (AZs) and networks.

To configure the **Assign AZ and Networks** pane:

1. Select **Assign AZs and Networks**.

2. Select an AZ for your singleton jobs, and one or more AZs to balance other jobs in.

3. Select a network. This network does not need to be the same network where you deployed TAS for VMs. For most deployments, operators can create unique networks in which to deploy the Isolation Segment tile. These networks can maintain network reach with the Diego components because the Diego Cells can reach the Diego Brain and Diego Database VMs.

4. Click **Save**.

Configure Networking

In the **Networking** pane, you configure security and routing services for your IaaS.

To configure the **Networking** pane:

1. Select **Networking**.

2. (Optional) Under **Gorouter IPs**, enter one or more static IP addresses for the Gorouters that handle this isolation segment. These IP addresses must be within the subnet CIDR block that you defined in the Ops Manager network configuration for your isolation segment. If you have a load balancer, configure it to point to these IP addresses.

   Note: Entering the static IP addresses is not necessary for deployments running on a public IaaS such as AWS, GCP, or Azure because users specify the IaaS load balancer in the **Resource Config** pane of the Isolation Segment tile.

3. If you want to use HAProxy for this isolation segment, enter at least one address in the **HAProxy IPs** field. You should specify more than one IP address for high availability. Then configure your load balancer to forward requests for the domains you have set up for your deployment to these IP addresses. For more information, see Configuring SSL/TLS Termination at HAProxy.

   > **Note:** If you rely on HAProxy for a feature in TAS for VMs and you want isolated networking for this isolation segment, you may want to deploy the HAProxy provided by the Isolation Segment tile.

4. Under **Certificates and private keys for the Gorouter and HAProxy**, you must provide at least one certificate and private key name and certificate key pair for the Gorouter and HAProxy. The Gorouter and HAProxy are enabled to receive TLS communication by default. You can configure multiple certificates for the Gorouter and HAProxy.

   > **Note:** When providing custom certificates, enter them in this order: `wildcard`, `Intermediate`, `CA`. For more information, see Creating a .pem File for SSL Certificate Installations in the DigiCert documentation.

   1. Click **Add** to add a name for the certificate chain and its private key pair. This certificate is the default used by the Gorouter and HAProxy. You can either provide a certificate signed by a Certificate Authority (CA) or click on the **Generate RSA Certificate** link to generate a certificate generated by the Ops Manager CA. For the values to use, see Providing a Certificate for Your TLS Termination Point.

      > **Note:** If you configured Ops Manager Front End without a certificate, you can use this new certificate to complete Ops Manager configuration. To configure your Ops Manager Front End certificate, see Configure Front End in *Preparing to Deploy Ops Manager on GCP*.

   2. If you want to configure multiple certificates for the Gorouter and HAProxy, click **Add** and fill in the appropriate fields for each additional certificate key pair. For details about generating certificates in Ops Manager for your wildcard system domains, see Providing a Certificate for Your TLS Termination Point.

   > **Note:** Ensure that you add any certificates that you generate in this pane to your infrastructure load balancer.

5. (Optional) When validating requests using mutual TLS to back ends and route services, the Gorouter trusts multiple CAs by default. You can use the following fields to configure which CA certificates Gorouter trusts:

- For backwards compatibility with older versions of Isolation Segment:
    1. Under **Certificate Authorities trusted by the Gorouter for client requests**, select **Trust certs provided in "Certificate Authorities trusted by the Gorouter for back ends and route services"**.
- If you want the Gorouter to trust a particular set of CAs only:
    1. Under **Certificate Authorities trusted by the Gorouter for client requests**, select **Only trust the following Certificate Authorities**.
    2. Enter the CAs in the field that appears. When this option is selected, the Gorouter only trusts the provided CAs for client requests. It does not trust any well-known CAs that are provided with the stemcell.

6. Under **Select the range of TLS versions supported by the Gorouter and HAProxy**, select the range of TLS versions to use in Gorouter and HAProxy communications. The Gorouter and HAProxy support TLS v1.2 to TLS v1.3 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see TLS Cipher Suite Support in *Securing Traffic into TAS for VMs*.
7. (Optional) Under **Balancing algorithm used by the Gorouter**, select your load balancing algorithm for the Gorouter. VMware recommends the default option of **Round robin** for most use cases. Some use cases, such as those where apps have long-lived connections, may benefit from the least connection algorithm. For more information, see HTTP Routing.
8. Configure **Logging of client IPs in the Gorouter**. The **Log client IPs** option is set by default. To comply with the General Data Protection Regulation (GDPR), select one of these options To deactivate logging of client IP addresses:
    - If your load balancer exposes its own source IP address, select **Disable logging of X-Forwarded-For header only**.
    - If your load balancer exposes the source IP of the originating client, select **Disable logging of both source IP and X-Forwarded-For header**.
9. Under **TLS termination point**, configure how TAS for VMs handles `x-forwarded-client-cert` (XFCC) HTTP headers based on where TLS is terminated for the first time in your deployment. The table below indicates which option to choose based on your deployment configuration:

Deployment Configuration	TLS Option	Additional Notes
◦ The load balancer is terminating TLS, and ◦ The load balancer is configured to put the client certificate from a mutual authentication TLS handshake into the X-Forwarded-Client-Cert HTTP header	Infrastructure load balancer	Both HAProxy and the Gorouter forward the XFCC header when included in the request.

○ The load balancer is configured to pass through the TLS handshake through TCP to the instances of HAProxy, and ○ HAProxy instance count is more than `0`	HAProxy	HAProxy sets the XFCC header with the client certificate received in the TLS handshake. The Gorouter forwards the header. **Breaking Change:** If you select the **The Gorouter does not request client certificates** option in the **Gorouter behavior for client certificate validation** field, the XFCC header cannot be delivered to apps.
○ The load balancer is configured to pass through the TLS handshake through TCP to instances of the Gorouter	Gorouter	The Gorouter strips the XFCC header if it is included in the request and forwards the client certificate received in the TLS handshake in a new XFCC header. If you have deployed instances of HAProxy, app traffic bypasses those instances in this configuration. If you have also configured your load balancer to route requests for SSH directly to the Diego Brain, consider reducing HAProxy instances to `0`. **Breaking Change:** If you select the **The Gorouter does not request client certificates** option in the **Gorouter behavior for client certificate validation** field, the XFCC header cannot be delivered to apps.

For a description of the behavior of each configuration option, see Forward Client Certificate to Apps in *HTTP Routing*.

10. To configure HAProxy to handle client certificates, select one of the following options in the **HAProxy behavior for client certificate validation** field:
    - **HAProxy does not request client certificates:** This option requires mutual authentication, which makes it incompatible with **TLS termination point** option **HAProxy**. HAProxy does not request client certificates, so the client does not provide them and no validation occurs. This is the default configuration.
    - **HAProxy requests but does not require client certificates:** The HAProxy requests client certificates in TLS handshakes and validates them when presented, but does not require them. This option is required if you want to enable mutual TLS app identity verification and TLS is terminated for the first time at HAProxy.

      > **Caution:** Upon upgrade, TAS for VMs fails to receive requests if your load balancer is configured to present a client certificate in the TLS handshake with HAProxy but HAProxy has not been configured with the certificate authority used to sign it. To mitigate this issue, select **HAProxy does not request client certificates** or configure the HAProxy with the appropriate CA.

11. To configure Gorouter behavior for handling client certificates, select one of the following options in the **Gorouter behavior for client certificate validation** field:
    - **The Gorouter does not request client certificates:** Client certificates are not requested, so the client does not provide them and validation of client certificates does not occur. This option is incompatible with the **TLS termination point** options **HAProxy** and **Gorouter** because these options require mutual authentication.
    - **The Gorouter requests but does not require client certificates:** The Gorouter requests client certificates in TLS handshakes and validates them when presented, but does not require them. This is the default configuration.
    - **The Gorouter requires client certificates:** The Gorouter validates that the client certificate is signed by a Certificate Authority that the Gorouter trusts. If the Gorouter cannot validate the client certificate, the TLS handshake fails.

    > **Caution:** Requests to the platform fail upon upgrade if your load balancer is configured with client certificates and the Gorouter does not have the CA. To mitigate this issue, select **The Gorouter does not request client certificates**.

12. In the **TLS cipher suites for the Gorouter** field, review the TLS cipher suites for TLS handshakes between the Gorouter and front end clients such as load balancers or HAProxy. The default value for this field is `ECDHE-RSA-AES128-GCM-SHA256:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`.

    To modify the default configuration, use an ordered, colon-separated list of Golang-supported TLS cipher suites in the OpenSSL format. This field does not apply to TLS v1.3. For TLS v1.3, the Gorouter only uses a set of default ciphers, and this is not configurable.

    Operators should verify that the ciphers are supported by any clients or front end components that initiate TLS handshakes with the Gorouter. For a list of TLS ciphers supported by the Gorouter, see TLS Cipher Suite Support in *Securing Traffic into TAS for VMs*.

    Verify that every client participating in TLS handshakes with the Gorouter has at least one cipher suite in common with the Gorouter.

    > **Note:** Specify cipher suites that are supported by the versions configured under **Select the range of TLS versions supported by the Gorouter and HAProxy**. For example, TLS v1.3 does not support configuring cipher suites. If you select **TLSv1.3 only**, you cannot configure cipher suites for the Gorouter or HAProxy.

13. In the **TLS cipher suites for HAProxy** field, review the TLS cipher suites for TLS handshakes between HAProxy and front end clients such as load balancers and the Gorouter. The default value for this field is:
    `DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-`

SHA256:ECDHE-RSA-AES256-GCM-SHA384

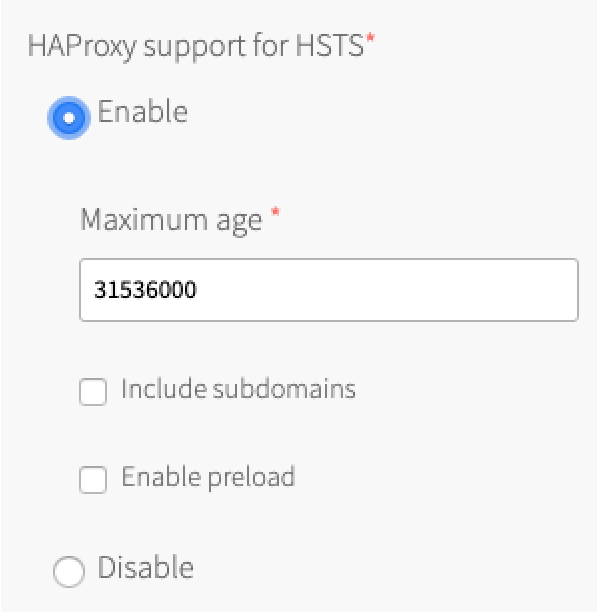
To modify the default configuration, use an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. Operators should verify that the ciphers are supported by any clients or front end components that initiate TLS handshakes with HAProxy. Verify that every client participating in TLS handshakes with HAProxy has at least one cipher suite in common with HAProxy.

> **Note:** Specify cipher suites that are supported by the versions configured in the **Minimum version of TLS supported by the Gorouter and HAProxy** field.

14. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Enable** or **Disable** based on your deployment layout.
    - To enable communication between HAProxy and the Gorouter:
        1. Verify that **Enable** is selected.
        2. In the **Certificate authority for HAProxy back end** field, provide the CA that signed the certificate you configured in the **Certificates and private keys for the Gorouter and HAProxy** field.

            > **Note:** If you used the **Generate RSA Certificate** link to generate a certificate, then the CA to specify is the Ops Manager CA, which you can locate at the `/api/v0/certificate_authorities` endpoint in the Ops Manager API.

        3. Make sure that the Gorouter and HAProxy have TLS cipher suites in common in the **TLS cipher suites for the Gorouter** and **TLS cipher suites for HAProxy** fields.
           For more information, see Terminating TLS at the Load Balancer and Gorouter in *Securing Traffic into TAS for VMs*, Providing a Certificate for Your TLS Termination Point, and Using the Ops Manager API.
    - To use non-encrypted communication between HAProxy and the Gorouter:
        1. Select **Disable**.
        2. If you are not using HAProxy, set the number of HAProxy job instances to `0` in the **Resource Config** pane. For more information, see Scale Down and Deactivate Resources.
           For more information, see Terminating TLS at the Gorouter Only and Terminating TLS at the Load Balancer Only in *Securing Traffic into TAS for VMs*.
15. (Optional) To force browsers to use HTTPS when making requests to HAProxy, select **Enable** under **HAProxy support for HSTS** and complete these optional configuration



HAProxy support for HSTS*

● Enable

Maximum age *

31536000

□ Include subdomains

□ Enable preload

○ Disable

steps:

- Enter a **Maximum age** in seconds for the HSTS request. HAProxy forces HTTPS requests from browsers for the duration of this setting. The maximum age is one year, or 31536000 seconds.
- Enable the **Include subdomains** check box to force browsers to use HTTPS requests for all component subdomains.
- Select the **Enable preload** check box to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from HAProxy.

16. (Optional) If you want the Gorouter or HAProxy to reject any HTTP (non-encrypted) traffic, select the **Disable HTTP on the Gorouter and HAProxy** checkbox. When selected, the Gorouter and HAProxy do not listen on port 80.
17. (Optional) Select the **Disable insecure cookies on the Gorouter** checkbox to turn on the secure flag for cookies generated by the Gorouter.
18. (Optional) To deactivate the addition of Zipkin tracing headers on the Gorouter, deselect the **Enable Zipkin tracing headers on the Gorouter** checkbox. Zipkin tracing headers are enabled by default. For more information about using Zipkin trace logging headers, see HTTP Headers for Zipkin Tracing in *HTTP Routing*.
19. (Optional) The **Enable the Gorouter to write access logs locally** checkbox is selected by default. VMware recommends deselecting this checkbox for high-traffic deployments, since logs may not be rotated fast enough and can fill up the disk.
20. (Optional) By default, Gorouter support for the PROXY protocol is deactivated. To enable the PROXY protocol, select the **Enable support for PROXY protocol in the Gorouter** checkbox. When enabled, client-side load balancers that terminate TLS but do not support HTTP can pass along information from the originating client. Enabling this option may impact Gorouter performance. For more information about enabling the PROXY protocol in the Gorouter, see the *About HTTP Header Forwarding* sections in Securing Traffic into TAS for VMs.

21. **Bypass security checks for route service lookup** has potential security concerns, but may be needed for backwards compatibility. For more information, see Route Services.

22. (Optional) If you want to limit the number of app connections to the back end, enter a value in the **Maximum connections per back end** field. You can use this field to prevent a poorly behaving app from all the connections and impacting other apps. No value or a value of `0` sets no limit.

    To choose a value for this field, review the peak concurrent connections received by instances of the most popular apps in your deployment. You can determine the number of concurrent connections for an app from the `httpStartStop` event metrics emitted for each app request.

    If your deployment uses App Metrics, you can also obtain this peak concurrent connection information from Network Metrics in *Monitoring and Troubleshooting Apps with App Metrics*. The default value is `500`.

23. Under **Keep-alive connections for the Gorouter**, select **Enable** or **Disable**. Keep-alive connections are enabled by default. For more information, see Keep-Alive Connections in *HTTP Routing*.

24. (Optional) To accommodate larger uploads over connections with high latency, increase the number of seconds in the **Gorouter timeout to back ends** field.

25. (Optional) Increase the value of **Load balancer unhealthy threshold** to specify the amount of time, in seconds, that the Gorouter continues to accept connections before shutting down. During this period, health checks may report the Gorouter as unhealthy, which causes load balancers to failover to other Gorouters. Set this value to an amount greater than or equal to the maximum time it takes your load balancer to consider a Gorouter instance unhealthy, given repeated failed health checks.

26. (Optional) Modify the value of **Load balancer healthy threshold**. This field specifies the amount of time, in seconds, to wait until declaring the Gorouter instance started. This allows an external load balancer time to register the Gorouter instance as healthy.

27. (Optional) If app developers in your organization want certain HTTP headers on HTTP Requests to appear in their app logs with information from the Gorouter, specify them in the **HTTP headers to log** field with a comma-separated list. For example, to support app developers that deploy Spring apps to TAS for VMs, you can enter Spring-specific HTTP headers.

28. (Optional) If app developers in your organization want certain HTTP headers on HTTP Requests to appear in their app logs with information from the Gorouter, specify them in the **HTTP headers to log** field with a comma-separated list. For example, to support app developers that deploy Spring apps to TAS for VMs, you can enter Spring-specific HTTP headers. The **Max request header size in kb** property allows TAS for VMs to prevent denial-of-service attacks from requests with large headers. This value is the max size for all of a request’s headers combined, including a request’s header keys, header values, and all values in the request line. A request whose headers exceed this value will receive a 431 status code. It is recommended to set this to 48kb, however, up to 1024kb is allowed for

backwards compatibility. The value of this property must be between 1 and 1024. This property does not limit the size of the request body.

29. If you expect requests larger than the default maximum of 16.384 KB, enter a new value in bytes for **HAProxy request maximum buffer size**. You may need to do this, for example, to support apps that embed a large cookie or query string values in headers. Requests larger than the maximum value result in a gateway error.
30. If your TAS for VMs deployment uses HAProxy and you want it to receive traffic only from specific sources, configure these fields:
    - **HAProxy protected domains:** Enter a comma-separated list of domains to protect from unknown source requests.
    - (Optional) **HAProxy trusted CIDRs:** Enter a space-separated list of CIDRs to limit which IP addresses from the **HAProxy protected domains** can send traffic to TAS for VMs.
31. For **DNS search domains**, enter DNS search domains as a comma-separated list. Your containers append these search domains to hostnames to resolve them into full domain names.
32. The **Enable Silk policy enforcement** checkbox is selected by default. To deactivate Silk network policy enforcement between apps, deselect the checkbox. Deactivating network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.
33. (Optional) In the **Sticky session cookies** field, enter one or more sticky session cookie names. The default cookie name is `JSESSIONID`. Some apps require a different cookie name. For example, Spring WebFlux requires `SESSION` for the cookie name. Gorouter uses these cookies to support session affinity, or *sticky sessions*. For more information, see Session Affinity in *HTTP Routing*.
34. (Optional) For additional security, enter headers that you want the Gorouter to remove from app responses in **Remove specified HTTP response headers**.
35. Click **Save**.

Configure App Containers

In the **App Containers** pane, you enable microservice frameworks, private Docker registries, and other services that support your apps at the container level.

To configure the **App Containers** pane:

1. Select **App Containers**.
2. Choose how the Gorouter verifies app identity to enable encryption and prevent misrouting under **Gorouter app identity verification**:
    - **The Gorouter uses TLS to verify app identity**: Enables the Gorouter to verify app identity using TLS. This is the default option.
    - **The Gorouter and apps use mutual TLS to verify each other's identity**: Enables your apps and the Gorouter to verify each other's identity using mutual TLS (mTLS).

This option deactivates TCP routing because app containers accept incoming communication only from the Gorouter.

For more information, see Preventing Misrouting in *HTTP Routing*.

3. (Optional) You can configure TAS for VMs to run app instances in Docker containers by provided a comma-separated list of their IP address ranges in the **Private Docker insecure registry allow list** field. For more information, see Using Docker Registries.
4. Select your preference for disk cleanup scheduling. If you select **Clean up disk space once usage fills disk**, enter a value in MB for **Reserved disk space for other jobs**. This is the amount of space the garbage collection algorithm must keep free for other jobs. For more information about the configuration options and how to configure a reserved amount, see Configuring Diego Cell Disk Cleanup Scheduling.
5. The **Enable containerd delegation** check box governs whether or not Garden delegates container create and destroy operations to the containerd tool. By default, this option is enabled and Garden uses containerd. Deactivate this option by disabling the check box. For more information about the containerd tool, see containerd.
6. Under **NFSv3 volume services**, select **Enable** or **Disable**. NFS volume services allow app developers to bind existing NFS volumes to their apps for shared file access. For more information, see Enabling Volume Services.

Note: In a fresh install, NFSv3 volume services is enabled by default. In an upgrade, NFSv3 volume services is set to the same setting as it was in the previous deployment.

7. (Optional) To configure LDAP for NFSv3 volume services:
   - For **LDAP service account user**, enter the username of the service account in LDAP that manages volume services.
   - For **LDAP service account password**, enter the password for the service account.
   - For **LDAP server host**, enter the hostname or IP address of the LDAP server.
   - For **LDAP server port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
   - For **LDAP user search base**, enter the location in the LDAP directory tree from which any LDAP user search begins. The typical LDAP search base matches your domain name.
     For example, a domain named `cloud.example.com` typically uses the following LDAP user search base: `ou=Users,dc=example,dc=com`.
   - (Optional) For **LDAP server CA certificate**, you can enter a certificate if your LDAP server supports TLS and you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

Note: UAA can only parse one certificate entered into this field. If you enter multiple certificates, UAA only uses the first one you entered and ignores the rest. You only need to include one root certificate or self-signed certificate.

8. (Optional) To limit the number of log lines each app instance in your isolation segment can generate per second, select **Enable** under **App log rate limit (beta)** and enter the **Maximum app log lines per second**. At minimum, VMware recommends using the default limit of `100`. This feature is deactivated by default. Disabling this feature may cause an app instance to overload the Loggregator Agent with logs, which may result in the Loggregator Agent dropping logs for other app instances. Otherwise, it may report inaccurate app metrics in the cf CLI or increase the CPU usage on the Diego Cell VM. If you configure a log rate limit, you can monitor the `AppInstanceExceededLogRateLimitCount` metric for the number of app instances that exceed the log rate limit.
9. Click **Save**.

Configure System Logging

In the **System Logging** pane, you can configure system logging in TAS for VMs to forward log messages from TAS for VMs component VMs to an external service. VMware recommends forwarding logs to an external service for use in troubleshooting. If you do not fill these fields, platform logs are not forwarded but remain available on the component VMs and for download through Ops Manager.

To configure the **System Logging** pane:

1. Select **System Logging**.
2. Select an option under **Configure syslog for system components?**. **No** is selected by default. This setting only affects Diego Cell, Gorouter, and HAProxy components within the isolation segment. This setting does not affect shared TAS for VMs system components.
3. To use syslog, select **Yes**.
4. Enter the address of your external syslog aggregation service in the **Address** field. The address can be a hostname or IP address.
5. Enter a port number in the **Port** field.
6. Select a protocol from the **Transport protocol** menu. This is the protocol the system uses to transmit logs to syslog.
7. (Optional) To transmit logs over TLS, select the **Enable TLS** check box.
8. Enter a **Permitted peer**.
9. Paste the certificate for your TLS certificate authority (CA) in the **TLS CA certificate** field.
10. (Optional) Enable the **Use TCP for file forwarding local transport** check box to transmit logs over TCP. This prevents log truncation, but may cause performance issues.
11. (Optional) The **Do not forward debug logs** check box is enabled by default. To forward `DEBUG` syslog messages to an external service, deselect the check box.

Note: Some Isolation Segment components generate a high volume of `DEBUG` syslog messages. Enabling the **Do not forward debug logs** check box prevents TAS for VMs components from forwarding the `DEBUG` syslog messages to external services. However, Isolation Segment still writes the messages to the local disk.

12. (Optional) To specify a custom syslog rule, enter it in the **Custom rsyslog configuration** field in RainerScript syntax. For more information about customizing syslog rules, see Customizing Syslog Rules. For more information about RainerScript, see the RainerScript documentation.
13. Select the **Enable system metrics** check box to emit system-level metrics about all VMs in the deployment. For a list of the VM metrics that the System Metric Agent emits, see System Metrics Agent in GitHub. When you activate this check box, ensure that you open port `9100` for the isolation segment. For more information, see Configure Firewall Rules in *Routing for Isolation Segments*.
14. Click **Save**.

Configure Advanced Features

The **Advanced Features** pane includes new functionality that may have certain constraints. Although these features are fully supported, VMware recommends caution when using them in production environments.

Diego Cell Memory and Disk Overcommit

If your apps do not use the full allocation of disk space and memory set in the **Resource Config** tab, you might want use this feature. These fields control the amount to overcommit disk and memory resources to each Diego Cell VM.

For example, you might want to use the overcommit if your apps use a small amount of disk and memory capacity compared to the amounts set in the **Resource Config** settings for **Diego Cell**.

Note: Due to the risk of app failure and the deployment-specific nature of disk and memory use, VMware has no recommendation for how much, if any, memory or disk space to overcommit.

To enable overcommit:

1. Select **Advanced Features**.
2. Enter in MB the total desired amount of Diego Cell memory in the **Diego Cell memory capacity** field. See the **Diego Cell** row in the **Resource Config** tab for the current Diego Cell memory capacity settings that this field overrides.
3. Enter in MB the total desired amount of Diego Cell disk capacity in the **Diego Cell disk capacity** field. See the **Diego Cell** row in the **Resource Config** tab for the current Diego Cell disk capacity settings that this field overrides.
4. Click **Save**.

Note: Entries made to each of these two fields set the total amount of resources allocated, not the overage.

App Graceful Shutdown Period

If your apps require a longer period of time to finish in-flight jobs and gracefully shut down, you can increase the graceful shutdown period. By default, this graceful shutdown period is set to 10 seconds.

When TAS for VMs requests a shutdown of an app, the processes in the container have a period of time to gracefully shut down before the processes are forcefully terminated. For more information, see Shutdown in *App Container Lifecycle*.

If you significantly increase the value of the graceful shutdown period, platform upgrades and updates might become slower. This is because each Diego Cell uses the graceful shutdown period when it is cleaning up evacuated app instances and waits for each app to gracefully shut down.

VMware recommends using isolation segments to separate apps that have different shutdown requirements to ensure Diego Cell update times are reliable.

Note: You must ensure that **App graceful shutdown period** has the same value in all environments that have deployed apps. This is to avoid unexpected behavior.

To increase the app graceful shutdown period:

1. Select **Advanced Features**.
2. Enter an integer in the **App graceful shutdown period** field. This value is the period of time in seconds the platform should wait for an app instance to exit after it is signaled to gracefully shut down. The default and minimum value is `10`.
3. Click **Save**.

SMB Volume Services

Enabling SMB volume services allows developers to bind existing SMB shares to their apps for shared file access.

To enable SMB volume services:

1. Select **Advanced Features**.
2. Select the **Enable SMB volume services** checkbox.
3. Click **Save**.
4. In the **Errands** pane of the TAS for VMs tile, set the **SMB Broker Errand** to **On**.
5. Click **Save**.

For more information about SMB volume services, see Enabling Volume Services.

Configure Compute and Networking Isolation

In the **Compute and Networking Isolation** pane, you can activate or deactivate compute isolation and configure Gorouter sharding for isolation segments.

Compute isolation is when an app runs on compute resources, such as clusters, resource pools, and servers, that are isolated from other resources by a network partition or firewall.

Networking isolation, or Gorouter sharding, is when traffic to an app instance uses a dedicated route that is isolated from routes used by other app instances by a network partition or firewall rule.

In the Isolation Segment tile, you cannot deactivate compute isolation and enable networking isolation at the same time. This is because networking isolation configures the route by which traffic goes to the Isolation Segment tile. Both compute and networking isolation are enabled by default.

Options for Configuring Compute and Networking Isolation

The table below describes the ways VMware recommends configuring the **Compute and Networking Isolation** pane and the **Gorouter isolation segment request handling** property in the **Networking** pane of the TAS for VMs tile:

Use case	Compute isolation	Gorouter sharding mode	Gorouters reject requests for isolation segments (TAS for VMs tile)	Other requirements	Scaling recommendations
Use compute isolation and network isolation	Enabled	Isolation segment only	Reject requests for all isolation segments	Isolation Segment has its own load balancer and uses a different domain from TAS for VMs	Apps in the isolation segment can have their own Diego Cells and Gorouters.
Use compute isolation and extra Isolation Segment Gorouters	Enabled	Isolation segment only	Accept requests for all isolation segments	Isolation Segment has its own load balancer and uses the same domain as TAS for VMs	Apps in the isolation segment can have their own Diego Cells and Gorouters.
Use compute isolation only	Enabled	No isolation segment	Accept requests for all isolation segments	Isolation Segment uses TAS for VMs load balancer and domain	Scale Isolation Segment Gorouters to 0.
Use network isolation in some cases	See other rows for guidance	See other rows for guidance	Accept requests for some isolation segments	See other rows for guidance	See other rows for guidance

The use cases from the previous table are described here:

- **Use compute isolation and network isolation**: The most common use case is to enable compute isolation, networking isolation, and the **Gorouters reject requests for isolation segments** check box in the **Networking** pane of the TAS for VMs tile. This configuration routes traffic between an isolation segment and a block of compute resources that is dedicated solely to that isolation segment. It also ensures that no traffic is routed through the TAS for VMs Gorouters.
- **Use compute isolation and extra Isolation Segment Gorouters**: To configure traffic for your isolated apps to pass through the Gorouters for both Isolation Segment and TAS for VMs, activate compute isolation and networking isolation, and deactivate the **Gorouters**

reject requests for isolation segments check box in TAS for VMs. You can also use either the same apps domain you configured in the **Domains** pane of the TAS for VMs tile, or a different domain that you configure with the DNS for the load balancer for the Isolation Segment Gorouters. This configuration is useful if you want to configure isolated compute resources for your apps but do not have additional networking restrictions.

- **Use compute isolation only**: To configure the TAS for VMs load balancer and apps domain route requests from your isolated apps, you can enable compute isolation, deactivate networking isolation, and deactivate the **Gorouters reject requests for isolation segments** check box in TAS for VMs. This configuration enables you to move Diego Cells out of TAS for VMs and into isolation segments, so you can upgrade TAS for VMs first and your Isolation Segment tiles as time permits. This allows you to upgrade a large foundation in pieces, making it easier to upgrade within scheduled maintenance windows. This configuration is useful if your foundation has additional compute resource blocks, but no additional Gorouters for Isolation Segment configured in the **Resource Config** pane.
- **Use network isolation in some cases**: For this use case, select “Accept some” for the **Gorouter isolation segment request handling** radio button in the **Networking** pane of the TAS for VMs tile. Then provide a comma-separated list of Isolation Segment names that Gorouters accept traffic to. See other cases for guidance on how to configure each Isolation Segment.

> ⚠ **Caution:** If you configure the **Compute and Networking Isolation** pane and the **Gorouters reject requests for isolation segments** check box in the TAS for VMs tile in a manner other than the use cases described previously, the smoke tests fail. To deploy the tile anyway, you must deactivate the Smoke Test errand in the **Errands** pane of the TAS for VMs tile. VMware does not recommend disabling smoke tests. To configure the Smoke Test errand, see Configure Errands.

Configure Pane

To configure the **Compute and Networking Isolation** pane:

1. Select **Compute and Networking Isolation**.
2. Under **Compute isolation**, activate or deactivate compute isolation:
    - To deactivate compute isolation, select **Disable**. Select this option if you do not require compute isolation and you are deploying the Isolation Segment tile for the purpose of making upgrades more manageable. For example, if you have a TAS for VMs tile with a large volume of Diego Cells. This approach helps separate the upgrade of the TAS for VMs control plane from the upgrade of the Diego Cells. Further, it splits the upgrade of the Diego Cells into smaller groups.
    - To activate compute isolation, select **Enable** and enter a placement tag for your Diego Cells in the **Segment name** field. This placement tag identifies the Diego Cells that run in your isolation segment and must be unique across your Ops Manager deployment. You use this placement tag when following the steps in Register an Isolation Segment in *Managing Isolation Segments* to create the isolation segment in the CCDB.

> **Note:** If you deactivate compute isolation, you must set **Gorouter sharding mode** to **No isolation segment**. Otherwise, the Isolation Segment tile fails to schedule the apps in your isolation segment.

3. Under **Gorouter sharding mode**, activate or deactivate networking isolation by selecting a sharding mode. For more information, see Sharding Gorouters for Isolation Segments in *Routing for Isolation Segments*.
   - **Isolation segment only**: The Gorouters for the TAS for VMs tile acknowledge requests only for apps deployed within the Diego Cells of the tile. All other requests fail.
   - **No isolation segment**: The Gorouters for the TAS for VMs tile reject requests for any isolation segment. Choose this option to add a group of Gorouters for the TAS for VMs tile, such as when you want a private point of entry for the system domain or to configure a group of Gorouters to receive requests from your corporate intranet.

Configure Smoke Tests

In the **Smoke Tests** pane, you specify the org and space where smoke tests are run. In the org and space that you specify, the Smoke Test errand pushes the app to a Ops Manager org. The app runs basic functionality tests against both the TAS for VMs and Isolation Segment tiles after an installation or update.

The Smoke Test errand verifies that your deployment can:

- Push, scale, and delete apps
- Create and delete orgs and spaces

The Smoke Test errand is on by default. You can turn off the Smoke Test errand in the **Errands** pane. For more information, see Configure Errands.

To configure the **Smoke Tests** pane:

1. Select **Smoke Tests**.
2. (Optional) In **Domain**, enter the name of the apps domain that Isolation Segment uses when pushing an app to run smoke tests.
   - To configure routing from the isolation segment to either the TAS for VMs or Isolation Segment Gorouters, enter the apps domain you configured in the **Domains** pane of the TAS for VMs tile.
   - To configure separate routing for the isolation segment, enter a different apps domain.
3. If you have a shared apps domain, select **A temporary space within the system org**, which creates a temporary space within the system org for running smoke tests and deletes the space afterwards. Otherwise, select **A specified org and space** and complete these fields to configure where Isolation Segment pushes an app to run smoke tests:
   - For **Org**, enter the org Isolation Segment uses when pushing an app to run smoke tests.

- For **Space**, enter the space Isolation Segment uses when pushing an app to run smoke tests.

4. Click **Save**.

> **Caution:** If you configure the **Compute and Networking Isolation** pane and the **Gorouters reject requests for isolation segments** check box in the TAS for VMs tile in a manner other than the use cases described in Options for Configuring Compute and Networking Isolation, the smoke tests fail.

Configure Errands

Errands are scripts that Ops Manager runs automatically when it installs or uninstalls a product, such as a new version of TAS for VMs. There are two types of errands: *post-deploy errands* run after the product is installed, and *pre-delete errands* run before the product in uninstalled.

By default, Ops Manager always runs all errands.

In the **Errands** pane, you can select **On** to always run an errand or **Off** to never run it.

For more information about how Ops Manager manages errands, see Managing Errands in Ops Manager.

> **Caution:** You must deactivate the Smoke Test errand if:
>
> - You configure the **Compute and Networking Isolation** pane and the **Gorouter isolation segment request handling** check box in the TAS for VMs tile in a manner other than the use cases described in Options for Configuring Compute and Networking Isolation. Otherwise, apps fail to schedule in Isolation Segment.
> - Both the V2 and V1 Firehoses are deactivated in TAS for VMs. Otherwise, the deploy fails.

Configure Resources

In the **Resource Config** pane, you must associate load balancers with the VMs in your deployment to enable traffic.

To configure the **Resource Config** pane:

1. Select **Resource Config**.
2. If you are using a dedicated Gorouter for your isolation segment:

   > **Note:** The configuration settings available in **Resource Config** vary depending on your IaaS.

   - **If you have the Load Balancers column in your Resource Config**:
     1. Enter the wildcard DNS entry attached to your load balancer into the **Router** row under **Load Balancers**.

- **If you do not have the Load Balancers column in your Resource Config**:
    1. Go to the **Networking** pane.
    2. Specify **Gorouter IPs**.
    3. Attach the IP addresses to your load balancer manually.

After tile configuration

After you configure the Isolation Segment tile:

1. Register the isolation segment in the CCDB by following the procedure in Register an Isolation Segment in *Managing Isolation Segments*.

    > **Note:** If you run smoke tests as a post-deploy errand in the Isolation Segment tile, the smoke tests check if your isolation segment in registered in the Cloud Controller database (CCDB). If not, the smoke tests register it in the CCDB. This eliminates the need to manually register the isolation segment with the `cf create-isolation-segment` command. For more information about running smoke tests, see Configure Smoke Tests.

2. Return to the Ops Manager Installation Dashboard.
3. Click **Review Pending Changes**, then **Apply Changes** to deploy the tile.

After the tile finishes deploying, see Managing Isolation Segments for more information about managing an isolation segment.

(Optional) Step 4: Copy the Tile

To create multiple isolation segments, you can copy the Isolation Segment tile with the Replicator tool.

To create multiple isolation segments:

1. Download the Replicator tool from the Isolation Segment page of VMware Tanzu Network.
2. Go to the directory where you downloaded the Replicator tool.
3. Replicate the tile by running:

    ```
    ./replicator \
          --name "TILE-NAME" \
          --path /PATH/TO/ORIGINAL.pivotal \
          --output /PATH/TO/COPY.pivotal
    ```

    Where:

    - `TILE-NAME` is a unique name you provide for the new Isolation Segment tile. The name must be ten characters or less and only contain alphanumeric characters, dashes, underscores, and spaces.
    - `/PATH/TO/ORIGINAL.pivotal` is the absolute path to the original Isolation Segment tile you downloaded from VMware Tanzu Network.

  - `/PATH/TO/COPY.pivotal` is the absolute path for the copy that the Replicator tool produces.

4. Follow the procedures in this topic using the new `.pivotal` file, starting with Step 1.

Upgrading Replicated Tiles

For information about upgrading replicated Isolation Segment tiles, see Upgrading Replicated Tiles in *Upgrading TAS for VMs and Other Ops Manager Products*.

Managing Isolation Segments in TAS for VMs

With TAS for VMs, you can isolate deployment workloads into dedicated resource pools called isolation segments. This article explains how.

Isolation segments overview

To enable isolation segments, an operator must install the Isolation Segment tile by performing the procedures in Installing Isolation Segment. Installing the tile creates a single isolation segment.

After an admin creates a new isolation segment, the admin can then create and manage relationships between the orgs and spaces of a VMware Tanzu Application Service for VMs (TAS for VMs) deployment and the new isolation segment.

Requirements

Target the API endpoint of your deployment with `cf api` and log in with `cf login` before performing the procedures in this topic. For more information, see Identifying the API Endpoint for Your TAS for VMs Instance.

Register an isolation segment

After you install the Isolation Segment tile, you must register it with Cloud Controller. If you only install the Isolation Segment tile without registering the isolation segment it creates, Cloud Controller does not detect the isolation segment and cannot place apps on it.

To register an isolation segment in the Cloud Controller database (CCDB), use the cf CLI.

Note: If you run smoke tests as a post-deploy errand in the Isolation Segment tile, the smoke tests check if your isolation segment is registered in the CCDB. If the isolation segment is not registered, the smoke tests register it in the CCDB. This eliminates the need to manually register an isolation segment with `cf create-isolation-segment`. For more information, see `cf login`

-

Run:

```
cf create-isolation-segment SEGMENT-NAME
```

Where `SEGMENT-NAME` is the name you give your isolation segment.

> **Note:** The isolation segment name used in the cf CLI command must match the value specified in the **Segment name** field of the Isolation Segment tile. If the names do not match, Ops Manager fails to place apps in the isolation segment when apps are started or restarted in the space assigned to the isolation segment.

If successful, the command returns an `OK` message:

```
Creating isolation segment SEGMENT-NAME as admin...
OK
```

Retrieve isolation segment information

The `cf isolation-segments`, `cf org`, and `cf space` commands retrieve information about isolation segments. The isolation segments you can see depends on your role:

- **Admins** see all isolation segments in the system.
- **Other users** only see the isolation segments that their orgs are entitled to.

List isolation segments

To see a list of the isolation segments that are available to you:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf isolation-segments
   ```

   The command returns results similar to this example output:

   ```
   Getting isolation segments as admin...
   OK
   name               orgs
   SEGMENT-NAME     org1, org2
   ```

Display isolation segments enabled for an org

An admin can entitle an org to multiple isolation segments.

To view the isolation segments that are available to an org:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

```
cf org ORG-NAME
```

Where `ORG-NAME` is the name of your org.

The command returns results similar to this example output:

```
Getting info for org ORG-NAME as user@example.com...

name:                 ORG-NAME
domains:              example.com, apps.example.com
quota:                paid
spaces:               development, production, sample-apps, staging
isolation segments:   SEGMENT-NAME, OTHER-SEGMENT-NAME
```

Showing the isolation segment assigned to a space

Only one isolation segment can be assigned to a space.

To view the isolation segment assigned to a space:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf space SPACE-NAME
   ```

   Where `SPACE-NAME` is the name of the space to which your isolation segment is assigned.

   The command returns results similar to this example output:

   ```
   name:                staging
   org:                 ORG-NAME
   apps:
   services:
   isolation segment:   SEGMENT-NAME
   space quota:
   security groups:     dns, p-mysql, p.mysql, public_networks, rabbitmq,
   ssh-logging
   ```

Deleting an isolation segment

Note: An isolation segment with deployed apps cannot be deleted.

Only admins can delete isolation segments.

To delete an isolation segment:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf delete-isolation-segment SEGMENT-NAME
   ```

   Where `SEGMENT-NAME` is the name of the isolation segment you want to delete.

   If successful, the command returns an `OK` message:

   ```
   $ cf delete-isolation-segment SEGMENT-NAME
   Deleting isolation segment SEGMENT-NAME as admin...
   OK
   ```

Managing isolation segment relationships

The commands listed in the sections below manage the relationships between isolation segments, orgs, and spaces.

Enabling an org to use isolation segments

Only admins can enable orgs to use isolation segments.

To enable the use of an isolation segment:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf enable-org-isolation ORG-NAME SEGMENT-NAME
   ```

   Where:

   - `ORG-NAME` is the name of your org.
   - `SEGMENT-NAME` is the name of the isolation segment you want your org to use.

If an org is entitled to use only one isolation segment, that isolation segment does not automatically become the default isolation segment for the org. You must explicitly set the default isolation segment of an org. For more information, see Set the Default Isolation Segment for an Org.

Deactivating an org from using isolation segments

> **Note:** You cannot deactivate an org from using an isolation segment if a space within that org is assigned to the isolation segment. Additionally, you cannot deactivate an org from using an isolation segment if the isolation segment is configured as the default for that org.

To disable an org from using an isolation segment:

1. Log in to your deployment by running:

```
cf login
```

2. Run:

   ```
   cf disable-org-isolation ORG-NAME SEGMENT-NAME
   ```

   Where:

   - `ORG-NAME` is the name of your org.
   - `SEGMENT-NAME` is the name of the isolation segment you want to disable the org from using.

   If successful, the command returns an `OK` message:

   ```
   Removing entitlement to isolation segment SEGMENT-NAME from org org1 as
   admin...
   OK
   ```

Setting the default isolation segment for an org

Note: This section requires cf CLI v6.29.0 or later. To download cf CLI v6.29.0 or later, go to the Releases section of the Cloud Foundry CLI repository on GitHub.

Only admins and org managers can set the default isolation segment for an org.

When an org has a default isolation segment, apps in its spaces belong to the default isolation segment unless you assign them to another isolation segment. You must restart running apps to move them into the default isolation segment.

To set the default isolation segment for an org:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf set-org-default-isolation-segment ORG-NAME SEGMENT-NAME
   ```

   Where:

   - `ORG-NAME` is the name of your org.
   - `SEGMENT-NAME` is the name of the isolation segment you want to set as your org's default.

   If successful, the command returns an `OK` message:

   ```
   $ cf set-org-default-isolation-segment org1 SEGMENT-NAME
   Setting isolation segment SEGMENT-NAME to default on org org1 as admi
   n...
   OK
   ```

To display the default isolation segment for an org:

1. Run:

   ```
   cf org
   ```

Assign an Isolation Segment to a Space

Admins and org managers can assign an isolation segment to a space. Apps in that space start in the specified isolation segment.

To assign an isolation segment to a space, you must first enable the space's org to use the isolation segment. For more information, see Enable an Org to Use Isolation Segments.

To assign an isolation segment to a space:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf set-space-isolation-segment SPACE-NAME SEGMENT-NAME
   ```

   Where:

   - `SPACE-NAME` is the name of your space.
   - `SEGMENT-NAME` is the name of the isolation segment you want to assign to your space.

Resetting the isolation segment assignment for a space

Admins can reset the isolation segment assigned to a space to use the org's default isolation segment.

To assign the default isolation segment for an org to a space:

1. Log in to your deployment by running:

   ```
   cf login
   ```

2. Run:

   ```
   cf reset-space-isolation-segment SPACE-NAME
   ```

   Where `SPACE-NAME` is the name of the space to which you want to assign your org's default isolation segment.

Routing for isolation segments in TAS for VMs

You can configure and manage routing for isolation segments in TAS for VMs as described in this topic. You can also deploy a set of Gorouters for each isolation segment to handle requests for apps within the segment.

For more information about how isolation segments work, see Isolation Segments in *TAS for VMs Security*. For more information about creating isolation segments, see Installing Isolation Segment.

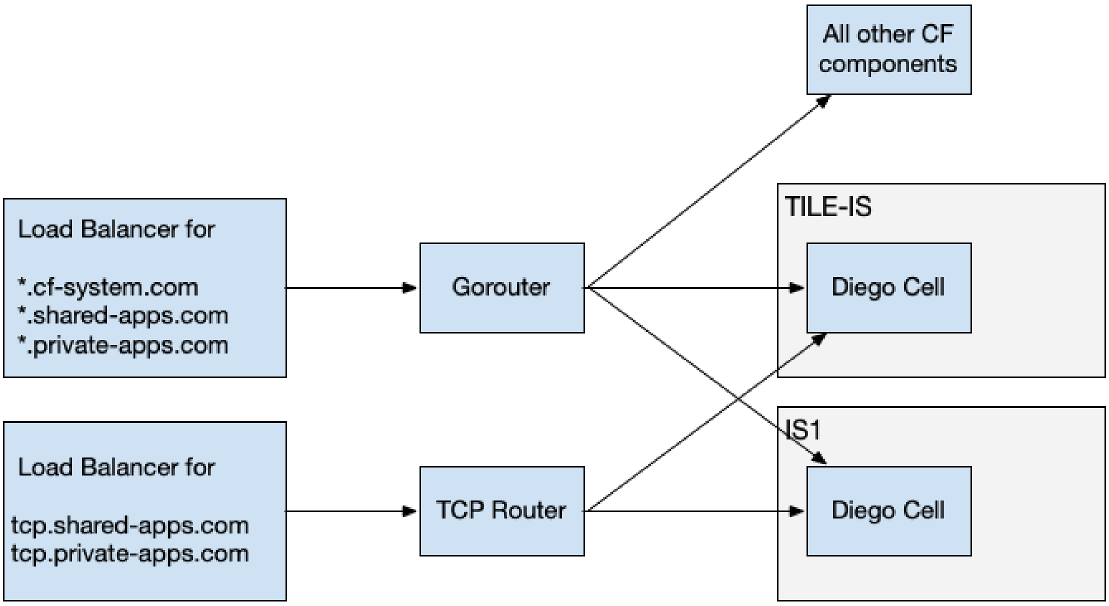
Note: The instructions in this topic assume you are using Google Cloud Platform (GCP). The procedures can differ on other IaaSes, but the concepts must be transferable.

Isolation segments overview

Isolation segments isolate the compute resources for one group of apps from another. However, these apps still share the same network resources. Requests for apps on all isolation segments, as well as for system components, transit the same load balancers, TAS for VMs Gorouters, and TAS for VMs TCP Routers.

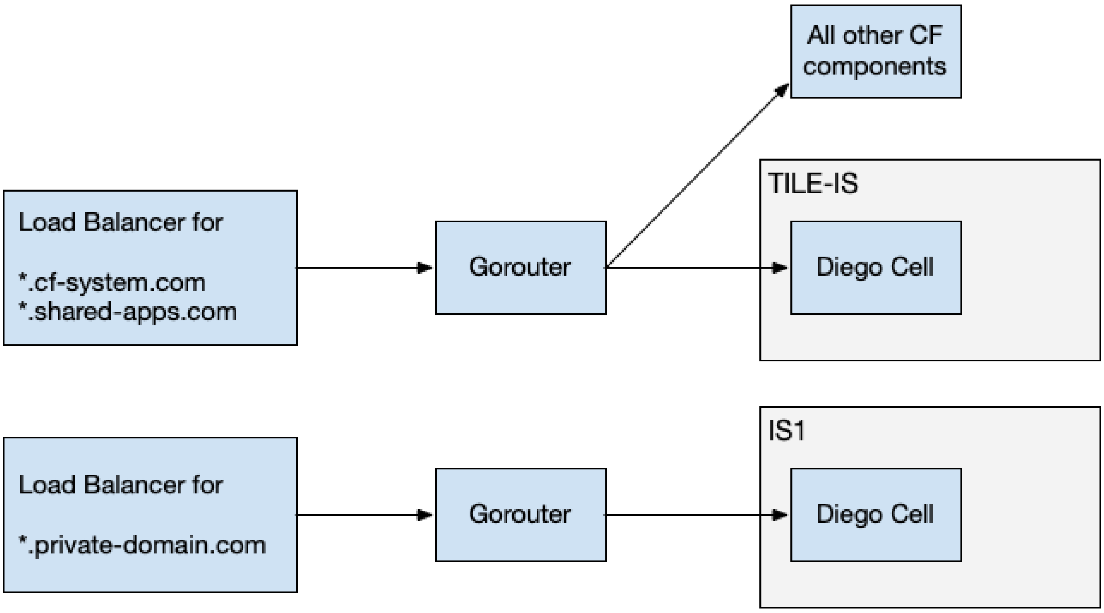
When you use isolation segments, TAS for VMs designates its Diego Cells as belonging to an isolation segment called `shared`. This isolation segment is the default isolation segment assigned to every org and space. This can be overwritten by assigning an explicit default for an organization. For more information about creating isolation segments, see Installing Isolation Segment.

The illustration below shows isolation segments sharing the same network resources:



The two isolation segments each contain a single Diego Cell. These isolation segments use the same Gorouter, TCP Router, and load balancers.

Operators who want to prevent all isolation segments and system components from using the same network resources can deploy an additional set of Gorouters for each isolation segment:



The two isolation segments each use separate Gorouters and Diego Cells. However, the use of an isolated TCP router in this scenario is not supported.

Use cases include:

- Requests for apps in an isolation segment must not share networking resources with requests for other apps.
- The TAS for VMs management plane can only be accessible from a private network. As multiple IaaS load balancers cannot typically share the same pool of back ends, such as TAS for VMs Gorouters, each load balancer requires an additional deployment of Gorouters.

Step 1: Create networks

Create a network or subnet for each isolation segment on your infrastructure. For example, an operator who wants one isolation segment separated from their TAS for VMs Diego Cells could create one network named `sample-network` with two subnets named `sample-subnet-tas` and `sample-subnet-is1`.

The diagram below describes the network topology:

```
IaaS network: sample-network
   |
   |_____ IaaS subnet: sample-subnet-tas
   |
   |_____ IaaS subnet: sample-subnet-is1
```

Subnets do not generally span IaaS availability zones (AZs), so the same operator with two AZs needs four subnets.

```
IaaS network: sample-network
|
|_____ IaaS subnet: sample-subnet-tas-az1
|
|_____ IaaS subnet: sample-subnet-tas-az2
|
|_____ IaaS subnet: sample-subnet-is1-az1
|
|_____ IaaS subnet: sample-subnet-is1-az2
```

For more information about networks and subnets in GCP, see Using Networks and Firewalls in the GCP documentation.

Step 2: Configure networks for Gorouters

To configure the subnets with BOSH, use BOSH cloud config subnets. Each subnet in the IaaS should correspond to a BOSH subnet that is labeled with the correct isolation segment. For more information, see Usage in the BOSH documentation.

Go to the **Assign AZs and Networks** pane of the Isolation Segment tile to assign your isolation segment to the network you created in Step 1. For more information, see Step 3: Configure the Tile in *Installing Isolation Segment*.

Step 3: Configure additional Gorouters

Go to the **Resource Config** pane of the Isolation Segment tile and use the dropdown to set your **Router** instances to a number greater than zero. For more information, see Step 3: Configure the Tile in *Installing Isolation Segment*.

Step 4: Add Gorouters to load balancer

If your IaaS supports it, go to the **Resource Config** pane of the Isolation Segment tile and enter the name of your load balancer under **Load Balancers**. For more information, see Step 3: Configure the Tile in *Installing Isolation Segment*. If your IaaS does not support this configuration, you must create static IP addresses and assign them to your load balancer out of band.

Step 5: Configure DNS and Load Balancers

Create a separate domain name for each Gorouter instance group, and configure DNS to resolve these domain names to a load balancer that routes requests to the matching Gorouters.

Note: You must configure your load balancers to forward requests for a given domain to one Gorouter instance group only.

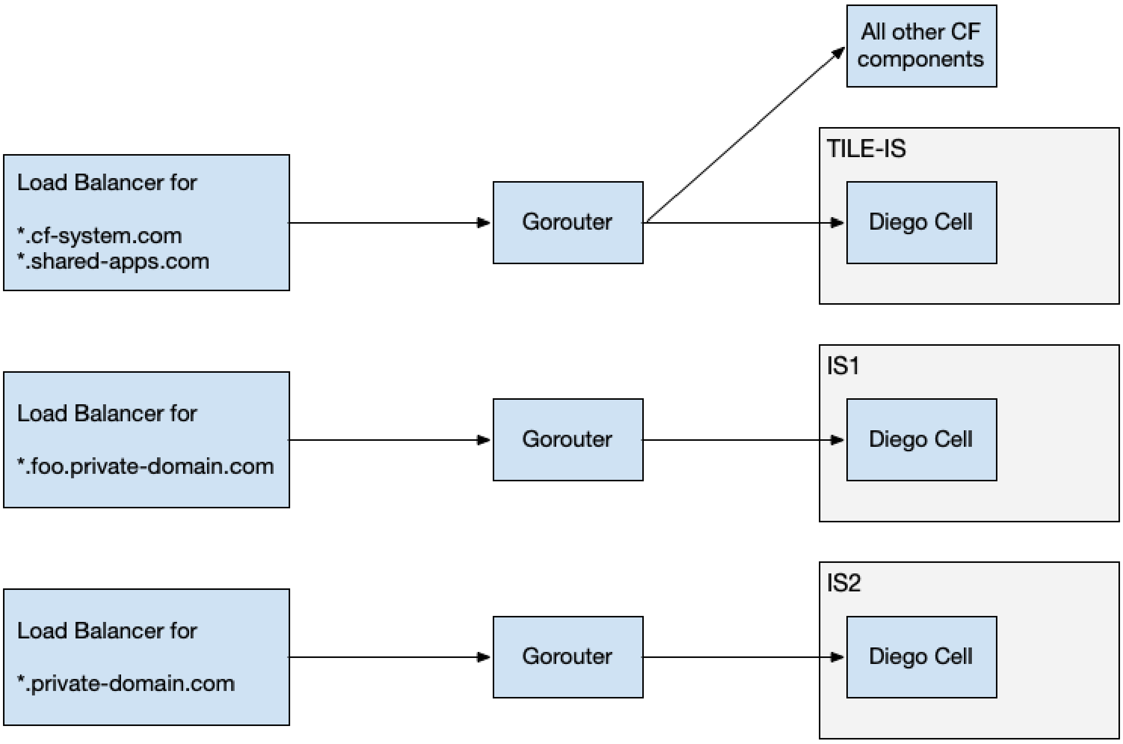
As Gorouter instance groups might be responsible for separate isolation segments, and an app might be deployed to only one isolation segment, requests can only reach a Gorouter that has access to the apps for that domain name. Load balancing requests for a domain across more than Gorouter instance group can result in request failures unless all the Gorouter instance groups have access to the isolation segments where apps for that domain are deployed.

Shared Domain Name

It is a common requirement for apps on separate isolation segments to be accessible at domain names that share a domain, such as `private-domain.com`. To achieve this configuration while also obeying the guideline for forwarding requests for a domain to only one Gorouter instance group, create a new TAS for VMs domain for a needed subdomain, such as `*.foo.private-domain.com.`

The diagrams illustrate a topology with separate load balancers, but you could also use one load balancer with multiple interfaces. In this configuration:

- Requests for system domain `*.cf-system.com` and the shared domain `*.shared-apps.com` are forwarded to the Gorouters for the TAS for VMs Diego Cells.
- Requests for private domain `*.foo.private-domain.com` are forwarded to the Gorouters for IS1. Requests for private domain `*.private-domain.com` are forwarded to the Gorouters for IS2.



The example has three isolation segments and three Gorouters. See the following long description.") %>

The three isolation segments each use separate Gorouters and load balancers.

Step 6: Configure Firewall Rules

Configure firewall rules to allow for necessary ingress and egress traffic for isolation segments and TAS for VMs Diego Cells. Assuming a default deny-all rule, properly configuring firewall rules prevents a request with a spoofed Host header from being forwarded by a Gorouter to an app in a different isolation segment.

To configure firewall rules for isolation segment traffic:

1. Configure the firewall rules in the table below:

   > **Note:** Firewall rules are specific to each IaaS, so the exact definition of `Source` and `Destination` depends on the IaaS. For example, on GCP, a `Source` is a subnet and a `Destination` is a tag. On AWS, both `Source` and `Destination` are security groups.

   For information about the processes that use these ports and their corresponding manifest properties, see Port Reference Table.

Rule Name	Source	Allowed Protocols/Ports	Destination	Reason
`tas-to-bosh`	TAS for VMs Diego Cells	`tcp:4222, 25250, 25777`	BOSH Director	BOSH Agent on VMs in the TAS for VMs Diego Cells to reach BOSH Director
`tas-internal`	TAS for VMs Diego Cells	`tcp:any, udp:any, icmp:any`	TAS for VMs Diego Cells	VMs within the TAS for VMs Diego Cells to reach one another
`tas-to-is1`	TAS for VMs Diego Cells	`tcp:1801, 8853, 9100`	Isolation segment	Diego BBS in TAS for VMs Diego Cells to reach Diego Cells in isolation segment
`is1-to-bosh`	Isolation segment	`tcp:4222, 25250, 25777`	BOSH Director	BOSH Agent on VMs in isolation segment to reach BOSH Director
`is1-internal`	Isolation segment	`tcp:all, udp:all, icmp:all`	Isolation segment	VMs within isolation segment to reach one another
`is1-to-tas`	Isolation segment	`tcp:3000, 3001, 4003, 4103, 4222, 4443, 8080, 8082, 8083, 8443, 8447, 8844, 8853, 8889, 8891, 9000, 9022, 9023, 9090, 9091`	TAS for VMs Diego Cells	Diego Cells in isolation segment to reach Diego BBS, Diego Auctioneer, and CredHub in TAS for VMs Diego Cells. Loggregator Agent to reach Traffic Controller. Gorouters to reach NATS, UAA, and Routing API.

2. (Optional) Configure the firewall rules in the table below:

Rule Name	Source	Allowed Protocols/Ports	Destination	Reason
`jumpbox-to-is1`	Jumpbox VM	`tcp:22`	Isolation segment	Jumpbox VMs to reach isolation segment through SSH or BOSH SSH.
`diego-cell-egress`	Diego Cell VM on isolation segment	`tcp:any, udp:any`	Internet	If Diego Cells must download buildpacks to stage apps, allow egress traffic from all Diego Cell VMs on isolation segments to reach the Internet.

Additional firewall rules might be necessary to allow logs to egress to application syslog drains. Connections for syslog drains are initiated both from Diego Cells and Routers.

For more information about ports used by agents to communicate with BOSH, see the bosh-deployment repository on GitHub.

For more information about networks and firewall rules for GCP, see Using Subnetworks in the GCP documentation.

Port Reference Table

To understand which protocols and ports map to which processes and manifest properties for the preceding rules, see the following table:

Protocol	Port	Process	Manifest Property
`tcp`	`1801`	Diego Rep	`diego.rep.listen_addr_securable`
`tcp`	`3000`	Routing API	`routing_api.port`
`tcp`	`3001`	Routing API	`routing_api.mtls_port`
`tcp`	`4003`	VXLAN Policy Agent	`cf_networking.policy_server.internal_listen_port`
`tcp`	`4103`	Silk Controller	`cf_networking.silk_controller.listen_port`
`tcp`	`4222`	NATS	`nats.nats.port`
`tcp`	`4443`	CAPI Blobstore Port - HTTPS	`capi.blobstore.tls.port`

`tcp`	`8080`	CAPI Blobstore Port - HTTP,
		Diego file server - HTTP
			`capi.blobstore.port,`
			`diego.file_server.listen_addr`

`tcp`	`8082`	Doppler gRPC, Reverse Log Proxy Gateway	`loggregator.doppler.grpc_port,`
			`loggregator.reverse_log_proxy.egress.port`

`tcp`	`8083`	Log Cache cf-auth-proxy	`log-cache.log-cache-cf-auth-proxy.proxy_port`
`tcp`	`8084`	Diego file server - HTTP (Small Footprint TAS)	`diego.file_server.listen_addr`

`tcp`	`8443`	UAA	`uaa.uaa.ssl.port`
`tcp`	`8447`	Diego file server - HTTPS	`diego.file_server.https_listen_addr`
`tcp`	`8844`	CredHub	`credhub.credhub.port`
`tcp`	`8853`	BOSH DNS health	`health.server.port` from `bosh-dns-release`
`tcp`	`8889`	Diego BBS	`diego.rep.bbs.api_location`
`tcp`	`8891`	Diego Database (Locket)	`diego.locket.listen_addr`
`tcp`	`9000`	Loggregator Syslog Binding Cache	`loggr-syslog-binding-cache.external_port`
`tcp`	`9022`	Cloud Controller Stager	`capi.stager.cc.external_port`
`tcp`	`9023`	Cloud Controller TPS	`capi.tps.cc.external_port`
`tcp`	`9090`	Cloud Controller Uploader	`capi.cc_uploader.http_port`
`tcp`	`9091`	Cloud Controller Uploader	`capi.cc_uploader.https_port`
`tcp`	`9100`	System Metrics Scraper	`system-metrics-scraper.loggr-system-metric-scraper.scrape_port`
`tcp`	`25250`	BOSH Blobstore	`bosh.blobstore.port`
`tcp`	`25777`	BOSH Registry	`bosh.registry.port`

Additional GCP information

For more information, see Understanding backend services in the GCP documentation and the BOSH Google CPI Release repository on GitHub.

Sharding Gorouters for isolation segments

You can configure Gorouter sharding for isolation segments depending on your use case:

Use Case	Description	How to Configure
Securing apps that run in an isolation segment	To provide security guarantees in addition to the firewall rules described above, you can configure sharding of the Gorouter's routing table, resulting in a Gorouter dedicated to an isolation segment that only recognizes routes for apps in the same isolation segment.	1. In the **Networking** pane of the TAS for VMs tile, enable the **Gorouters reject requests for isolation segments** checkbox. 2. Set the **Gorouter sharding mode** in the Isolation Segment tile to **Isolation segment only**.

Deploying additional Gorouters for TAS for VMs	The flexibility of the configuration also supports deployment of a Gorouter that excludes all isolation segments.	1. In the **Networking** pane of the TAS for VMs tile, enable the **Gorouters reject requests for isolation segments** checkbox. 2. Set the **Gorouter sharding mode** in the Isolation Segment tile to **No isolation segment**.

> **Note:** For compute isolation only, you can leave the **Gorouters reject requests for isolation segments** checkbox deselected. This is the default setting, which does not require any additional Gorouters for the Isolation Segment tile.

Metrics for Gorouters associated with isolation segments

For metrics emitted by the Gorouter, metrics can be distinguished by the name of the job. For example, this line is a metric emitted on `uptime`:

```
origin:"gorouter" eventType:ValueMetric timestamp:1491338040750977602 deployment:"superman.cf-app.com" job:"router_is1" index:"9a4b639c-8f0e-4b2b-b332-4161ee4646e6" ip:"10.0.16.23" valueMetric:<name:"uptime" value:118 unit:"seconds" >
```

Upgrading TAS for VMs

In this section:

- Configuring TAS for VMs for Upgrades
- What Happens During TAS for VMs Upgrades
- cf push Availability During TAS for VMs Upgrades

Configuring TAS for VMs for Upgrades

This topic describes several configuration options for VMware Tanzu Application Service for VMs (TAS for VMs) that can help ensure successful upgrades. In addition to following the Upgrade Preparation Checklist, review this topic to better understand how to prepare for TAS for VMs upgrades.

> **Breaking Change**: The Service Mesh feature was removed in TAS for VMs v2.11. You must deactivate the Service Mesh feature before upgrading to TAS for VMs v2.11.

Limit Component Instance Restarts

The `max_in_flight` variable limits how many instances of a component can restart simultaneously during updates or upgrades. Increasing the value of `max_in_flight` can make updates run faster, but setting it too high risks overloading VMs and causing failure. For guidance on setting `max_in_flight` values, see Basic Advice.

Values for `max_in_flight` can be any integer between 1 and 100, or a percentage of the total number of instances. For example, a `max_in_flight` value of `20%` in a deployment with 10 Diego Cell instances would make no more than two Diego Cell instances restart at once.

Set max_in_flight

The `max_in_flight` variable is a system-wide value with optional component-specific overrides. You can override the default value for individual jobs using an API endpoint.

Use the max_in_flight API Endpoint

Use the `max_in_flight` API endpoint to configure the maximum value for component instances that can start at a given time. This endpoint overrides product defaults. You can specify values as a percentage or an integer.

Use the string `default` as the `max_in_flight` value to force the component to use the deployment's default value.

Note: The example below lists three `JOB_GUID`s. These three GUIDs are examples of the three different types of values you can use to configure `max_in_flight`. The endpoint only requires one GUID.

```
        curl "https://EXAMPLE.com/api/v0/staged/products/PRODUCT-TYPE1-GUID/m
ax_in_flight" \
    -X PUT \
    -H "Authorization: Bearer UAA_ACCESS_TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
          "max_in_flight": {
            "JOB_1_GUID": 1,
            "JOB_2_GUID": "20%",
            "JOB_3_GUID": "default"
          }
        }'
```

Specific Guidance for Diego Cells

To upgrade TAS for VMs, BOSH must drain all Diego Cell VMs that host app instances. BOSH manages this process by upgrading a batch of Diego Cells at a time.

The number of Diego Cells that undergo upgrade simultaneously (either in a state of shutting down or coming back online) is controlled by the `max_in_flight` value of the Diego Cell job. For example, if `max_in_flight` is set to `10%` and your deployment has 20 Diego Cell job instances, then the maximum number of Diego Cells that BOSH can upgrade at a single time is `2`.

When BOSH triggers an upgrade, each Diego Cell undergoing upgrade enters “evacuation” mode. Evacuation mode means that the Diego Cell stops accepting new work and signals the rest of the

Diego system to schedule replacements for its app instances. This scheduling is managed by the Diego auctioneer process. For more information, see How Diego Balances App Processes.

The evacuating Diego Cells continue to interact with the Diego system as replacements come online. The Diego Cell undergoing upgrade waits until either its app instance replacements run successfully before shutting down the original local instances, or for the evacuation process to time out. This “evacuation timeout” defaults to 10 minutes.

If Diego Cell evacuation exceeds this timeout, then the Diego Cell stops its app instances and shuts down. The Diego system continues to re-emit start requests for the app replacements.

Prevent Overload

A potential issue arises if too many app instance replacements are slow to start or do not start successfully at all.

If too many app instances are starting concurrently, then the load of these starts on the rest of the system can cause other apps that are already running to crash and be rescheduled. These events can result in a cascading failure.

To prevent this issue, TAS for VMs provides two throttle configurations:

- The maximum number of in-flight Diego Cell instances
- The maximum number of starting containers

The values of the above throttle configurations depend on the version of TAS for VMs that you have deployed and whether you have overridden the default values.

The following list describes the existing defaults for determining the override values in your deployment:

- **Starting container count maximum:** 200
- **Starting container count overridable:** Yes
- **Maximum in-flight Diego cell instances:** 4% of total instances
- **Maximum in-flight Diego cell instances overridable:** Yes

Basic Advice

Set the `max_in_flight` variable low enough that the remaining component instances are not overloaded by typical use. If component instances are overloaded during updates, upgrades, or typical use, users may experience downtime.

Some more precise guidelines include:

- Increase the count of Diego Cells before an upgrade, and then set `max_in_flight` to the number of Diego Cells that you added. For example, if you have 200 Diego Cells, adding 20 Diego Cells and then setting `max_in_flight` to 20 enables the Diego Cells to be upgraded in 10 batches of 20 Diego Cells each. The default settings would require 25 batches of 8 Diego Cells each. If each batch takes 10 minutes, using this process would save you over 2.5 hours of upgrade time. Ensure that you scale the number of Diego Cells down after the upgrade.
- Quorum-based components are components with odd-numbered settings in the manifest. For quorum-based components such as etcd and Diego BBS, set `max_in_flight` to 1. This

preserves quorum and prevents a split-brain scenario from occurring as jobs restart. For more information, about split-brain scenarios, see Split-brain (computing) on Wikipedia.

- For other components, set `max_in_flight` to the number of instances that you can afford to have down at any one time. The best values for your deployment vary based on your capacity planning. In a highly redundant deployment, you can set the number to a higher value to allow updates to run faster. However, if your components are at high utilization, you can keep the number low to prevent downtime.
- Setting `max_in_flight` to a value greater than or equal to the number of instances you have running may reduce functionality.

Set a Maximum Number of Starting Containers

This section describes how to use the Diego Auctioneer job to configure the maximum number of app instances starting at a given time. This prevents Diego from scheduling too much new work for your platform to handle concurrently. A lower default can prevent server overload during cold start, which may be important if your infrastructure is not sized for a large number of concurrent cold starts.

The Diego Auctioneer only schedules a fixed number of app instances to start concurrently. This limit applies to both single and multiple Diego Cells. For example, if you set the limit to five starting instances, it does not matter if you have one Diego Cell with ten instances or five Diego Cells with two instances each. The auctioneer does not allow more than five instances to start at the same time.

If you are using a cloud-based IaaS, rather than a smaller on-premise solution, VMware recommends setting a larger default. By default, the maximum number of started instances is 200.

To configure the maximum number of started instances in the **Settings** tab of the TAS for VMs tile:

1. Log in to Ops Manager.
2. Click the TAS for VMs tile.
3. Select **App Containers**.
4. In the **Max-in-flight container starts** field, enter the maximum number of started instances.
5. Click **Save**.

Configure File Storage

This section describes critical factors to consider when evaluating the type of file storage to use in your TAS for VMs deployment. The TAS for VMs blobstore relies on the file storage system to read and write resources, app packages, and droplets. For more information, see Blobstore in *Cloud Controller*.

During an upgrade, file storage with insufficient IOPS numbers can negatively impact the performance and stability of your TAS for VMs deployment.

If disk processing time takes longer than the evacuation timeout for Diego Cells, then Diego Cells and app instances may take too long to start up, resulting in a cascading failure.

However, the minimum required IOPS depends upon a number of deployment-specific factors and configuration choices. Use this section as a guide when deciding on the file storage configuration for your deployment.

Select Internal or External File Storage

When you deploy TAS for VMs, you can select internal file storage or external file storage, either network-accessible or IaaS-provided, as an option in the TAS for VMs tile.

Selecting internal storage causes TAS for VMs to deploy a dedicated VM that uses either NFS or WebDAV for file storage. Selecting external storage allows you to configure file storage provided in network-accessible location or by an IaaS, such as Amazon S3, Google Cloud Storage, or Azure Storage.

Whenever possible, VMware recommends using external file storage.

Calculate Potential Disk Load Requirements

As a best-effort calculation, estimate the total number of bits needed to move during a system upgrade to determine how IOPS-performant your file storage needs to be.

- **Number of Diego Cells**: As a first calculation, determine the number of Diego Cells that your deployment currently uses. To view the number of Diego Cell instances currently running in your deployment, see the **Resource Config** pane of the TAS for VMs tile. If you expect to scale up the number of instances, use the anticipated scaled number.

  > **Note:** If your deployment uses more than 20 Diego Cells, you should avoid using internal file storage. Instead, you should always select external or IaaS-provided file storage.

- **Maximum In-Flight Load and Container Starts for Diego Cells**: Operators can limit the number of containers and Diego Cell instances that Diego starts concurrently. If operators impose no limits, your file storage may experience exceptionally heavy load during an upgrade.

What Happens During TAS for VMs Upgrades

This topic tells you what happens to VMware Tanzu Application Service for VMs (TAS for VMs) components and apps during a TAS for VMs upgrade.

BOSH Drains Diego Cell VMs

During a TAS for VMs upgrade, BOSH drains all Diego Cell VMs that host app instances. BOSH manages this process by upgrading a batch of cells one at a time.

When BOSH triggers an upgrade, each upgrading Diego Cell enters evacuation mode. In evacuation mode, BOSH stops Diego Cells and then schedules replacements for its app instances.

For more information, see Specific Guidance for Diego Cells in *Configuring TAS for VMs for Upgrades*.

cf push Can Become Unavailable

`cf push` is mostly available for the duration of a TAS for VMs upgrade. However, `cf push` can become unavailable when a single VM is in use or during BOSH Backup and Restore (BBR).

For more information, see cf push Availability During TAS for VMs Upgrades.

TAS for VMs Components Upgrade

This section describes the order in which VMware Tanzu Operations Manager upgrades components and runs tasks during a full platform upgrade. It also explains how the scale of different TAS for VMs components affects uptime during upgrades, and which components are scalable.

When performing an upgrade, Ops Manager first upgrades individual components, and then runs one-time tasks.

1. Components describes how Ops Manager upgrades TAS for VMs components and explains how individual component upgrades affect broader TAS for VMs capabilities.
2. One-Time Tasks lists the tasks that Ops Manager runs after it upgrades the TAS for VMs components.

Components

Ops Manager upgrades TAS for VMs components in a fixed order that honors component dependencies and minimizes downtime and other system limitations during the upgrade process.

The type and duration of downtime and other limitations that you can expect during a TAS for VMs upgrade reflect:

- Component instance scaling. For more information, see How Single-Component Scaling Affects Upgrades below.
- Component upgrade order. For more information, see Component Upgrade Order and Behavior below.

How Single-Component Scaling Affects Upgrades

In Ops Manager, the **Resource Config** pane of the TAS for VMs tile shows the components that the BOSH Director installs:

- **Scalable** component fields let you select the instance count from a range of settings or enter a custom value.
- **Unscalable** component fields allow a maximum of one instance.

When a component is scaled at a single instance, it can experience downtime and other limitations while the single VM restarts. This behavior might be acceptable for a test environment. To avoid downtime in a production environment, you must scale any scalable components, such as Router, and Diego Cells, to more than one instance.

For more information about how the scale of each component affects upgrade behavior, see Component Upgrade Order and Behavior below.

Note: A full Ops Manager upgrade might take close to two hours. You have limited ability to deploy an app during this time.

Component Upgrade Order and Behavior

The table below lists components in the order that Ops Manager upgrades each. It also lists which components are scalable and explains how component downtime affects TAS for VMs app and control availability. The table includes these columns:

- **Scalable:** Indicates whether the component is scalable above a single instance.

  Note: For components marked with a checkmark in this column, VMware recommends that you change the preconfigured instance value of `1` to a value that best supports your production environment. For more information about scaling a deployment, see High Availability in TAS for VMs.

- **Extended Downtime:** Indicates that if there is only one instance of the component, that component is unavailable for up to five minutes during an upgrade.
- **Downtime Affects...:** Indicates the plane of the TAS for VMs platform that component downtime affects, if the component is scaled at single instance:
  - **Apps**: Downtime can affect app availability.
  - **Platform**: Apps remain available during component downtime, but you cannot push, stage, or restart apps, or run other Cloud Foundry command-line interface (cf CLI) commands.
- **Other Limitations and Information:** Provides:
  - Component availability, behavior, and usage during an upgrade
  - Guidance on disabling the component before an upgrade

Component	Scalable	Extended Downtime	Downtime Affects...		Other Limitations and Information
			Apps	Platform	
NATS	✓	✓			
File Storage		✓		✓	
MySQL Proxy	✓	✓		✓	The MySQL Proxy is responsible for managing failover of the MySQL Servers. If the Proxy becomes unavailable, then access to the MySQL Server could be broken.
MySQL Server	✓	✓		✓	The MySQL Server is responsible for persisting internal databases for the platform. If the MySQL Server becomes unavailable, then platform services that rely upon a database, such as Cloud Controller and UAA, also become unavailable.

Backup Restore Node					
Diego BBS	✓	✓		✓	
UAA	✓			✓	If a user has an active authorization token prior to performing an upgrade, they can still log in using either a UI or the CLI.
Cloud Controller	✓	✓		✓	
HAProxy	✓	✓	✓		HAProxy is used to load-balance incoming requests to the Router. If HAProxy is unavailable, you might lose the ability to make requests to apps unless there is another routing path from your load balancer to the Gorouter.
Gorouter	✓	✓	✓		The Gorouter is responsible for routing requests to their app containers. If the Gorouter is not available, then apps cannot receive requests.
MySQL Monitor					
Clock Global	✓				
Cloud Controller Worker	✓	✓			
Diego Brain	✓	✓		✓	
Diego Cell	✓	✓	✓	✓	If you only have one Diego Cell, upgrading causes downtime for all apps that run on it. These include apps pushed with `cf push` and apps automatically installed by TAS for VMs, such as Apps Manager and the App Usage Service.
Loggregator Trafficcontroller	✓				Operators experience 2-5 minute gaps in logging.
Doppler Server	✓				Operators experience 2-5 minute gaps in logging.
TCP Router (if enabled)	✓				
CredHub	✓	✓	✓	✓	App downtime for apps that use secure credentials. Platform downtime for cf CLI commands such as `bind-service` and `unbind-service` applied to services configured with CredHub.
Istio Control		✓		✓	
Istio Router	✓	✓	✓		Downtime for this component only affects routes on mesh domains.

Route Syncer	✓	✓		✓	Downtime for this component only affects routes on mesh domains.

One-Time Tasks

After Ops Manager upgrades components, it performs system checks and launches UI apps and other TAS for VMs components as Cloud Foundry apps. These tasks run in this order:

1	Apps Manager Errand - Push Apps Manager
2	Smoke Test Errand - Run smoke tests
3	Usage Service Errand - Push Usage Service app
4	Notifications Errand - Push Notifications app
5	Notifications UI Errand - Push Notifications UI
6	App Autoscaler Errand - Push App Autoscaler
7	App Autoscaler Smoke Test Errand - Run smoke tests against App Autoscaler
8	Register Autoscaling Service Broker
9	Destroy Autoscaling Service Broker
10	Bootstrap Errand - Recover MySQL cluster
11	MySQL Rejoin Unsafe Errand

cf push Availability During TAS for VMs Upgrades

This topic describes what you can expect regarding the availability of `cf push` during upgrades of VMware Tanzu Application Service for VMs (TAS for VMs).

Overview

Availability of `cf push` during TAS for VMs upgrades varies from release to release. There are various considerations, such as Cloud Controller database (CCDB) migrations or the number of VMs in use, that can impact `cf push` availability. However, `cf push` is mostly available for the duration of an upgrade.

Impact on Single VMs vs. High-Availability Infrastructure

Having a single VM in use, such as a WebDAV, a HAProxy, a Gorouter, a UAA, or a Cloud Controller, impacts whether `cf push` is unavailable during an upgrade.

If you have scaled out your app to achieve high availability and are not using WebDAV, `cf push` should be available for the entire duration of the upgrade. However, upgrades to certain versions of TAS for VMs sometimes require a CCDB schema or data migration, which may cause `cf push` to be unavailable while Cloud Controllers are rolling during the upgrade.

Availability During a BBR Backup

The Cloud Controller API is taken down during the `pre-backup-lock` stage of a BBR backup and put back up again during the `post-backup-unlock` stage. As a result, `cf push` becomes unavailable during that time. However, the backup that takes place between those stages is very short, no longer than a few minutes. The bulk of the BBR backup operation happens after the `post-backup-unlock` stage, so the Cloud Controller API and `cf push` are available for most of the duration of a BBR backup.

The Uptimer tool can help you measure `cf push` availability during an upgrade. For more information, see the Uptimer repository on GitHub.

Administering TAS for VMs

In this section:

    - Configuring Load Balancer Health Checks for TAS for VMs Routers
    - Configuring Route Service Lookup
  - **Internal App Security**
    - Container Security
    - Container-to-Container Networking
    - App Security Groups
    - Restricting App Access to Internal TAS for VMs Components
  - **Certificates and Credentials**
    - Rotating Runtime CredHub Encryption Keys
    - Rotating the Cloud Controller Database Encryption Key
    - Securing Service Instance Credentials with Runtime CredHub
    - Providing a Certificate for Your TLS Termination Point
    - Trusted System Certificates
  - **Component Communications**
    - BOSH DNS Network Communications
    - Cloud Controller Network Communications
    - Container-to-Container Networking Communications
    - CredHub Network Communications
    - Diego Network Communications
    - Loggregator Network Communications
    - MySQL Network Communications
    - NATS Network Communications
    - Routing Network Communications
    - UAA Network Communications

Managing the Runtime

In this section:

- Configuring File Storage for TAS for VMs
- Configuring Load Balancing for TAS for VMs
- Configuring SSL Termination for TAS for VMs on vSphere
- Identifying the API Endpoint for Your TAS for VMs Instance
- Creating and Modifying Quota Plans
- Stopping and Starting Virtual Machines
- Scaling TAS for VMs
- Scaling Cloud Controller

- Configuring Diego Cell Disk Cleanup Scheduling
- Examining GrootFS Disk Usage
- Using Metadata
- Custom-Branding Apps Manager
- Planning TAS for VMs Orgs and Spaces
- Orgs, Spaces, Roles, and Permissions

Configuring File Storage for TAS for VMs

This topic provides instructions for configuring file storage for VMware Tanzu Application Service for VMs (TAS for VMs) based on your IaaS and installation method. See the section that applies to your use case.

To minimize system downtime, VMware recommends using highly resilient and redundant *external* filestores for your TAS for VMs file storage. For more factors to consider when selecting file storage, see Configure File Storage in *Configuring TAS for VMs for Upgrades*.

> **Note:** After initial installation, do not change file storage configuration without first migrating existing files to the new provider.

Internal File Storage

Internal file storage is only appropriate for small, non-production deployments.

To use the TAS for VMs internal filestore:

1. Select **Internal WebDAV**.
2. Click **Save**.

AWS

This section describes how to configure file storage for AWS.

For production-level Ops Manager deployments on AWS, VMware recommends selecting **External S3-compatible filestore**. For instructions, see External S3-Compatible Filestore below.

You can also configure Fog blobstores to use AWS IAM instance profiles. For instructions, see Fog with AWS IAM Instance Profiles below.

For more information about production-level Ops Manager deployments on AWS, see AWS Reference Architecture.

External S3-Compatible Filestore

To use an external S3-compatible filestore for TAS for VMs file storage:

1. Select **External S3-compatible filestore**.

2. For **URL endpoint**, enter the `https://` URL endpoint for your region. For example, `https://s3.us-west-2.amazonaws.com/`.
3. Enter the **Access key** and **Secret key** of the `pcf-user` you created when configuring AWS for Ops Manager.
4. (Optional) If your TAS for VMs deployment is on AWS, you can alternatively enter the **Access key** and **Secret key** of the `pcf-user` you created when configuring AWS for Ops Manager. If you enable the **S3 AWS with instance profile** checkbox and also enter an **Access key** and **Secret key**, the instance profile overrules the access key and secret key.
5. From the **S3 signature version** dropdown, select **V4 signature**. For more information about S3 signatures, see Signing AWS API Requests in the AWS documentation.
6. For **Region**, enter the region in which your S3 buckets are located. For example, `us-west-2`.
7. To encrypt the contents of your S3 filestore, select **Server-side encryption**. This option is only available for AWS S3.
8. (Optional) If you selected **Server-side encryption**, you can also specify a **KMS key ID**. TAS for VMs uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, TAS for VMs uses the default AWS key. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS) in the AWS documentation.
9. Deselect **Path-style S3 URLs (deprecated)** checkbox. When this checkbox is deselected, the S3 bucket is accessed using the virtual-hosted model instead of the path-based model. The deprecated path-based model is removed from AWS as of September 30, 2020. For more information about S3 path deprecation, see Amazon S3 Path Deprecation Plan – The Rest of the Story on the AWS News Blog.
10. Enter names for your S3 buckets:

Ops Manager Field	Value	Description
Buildpacks bucket name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.
Droplets bucket name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. VMware recommends that you use a unique bucket name for droplets, but you can also use the same name as above.
Packages bucket name	`pcf-packages-bucket`	This S3 bucket stores app packages. VMware recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources bucket name	`pcf-resources-bucket`	This S3 bucket stores app resources. VMware recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

11. Configure these checkboxes depending on whether your S3 buckets have versioning enabled:

- For versioned S3 buckets, enable the **Use versioning for backup and restore** checkbox to archive each bucket to a version.
  If you are using Dell ECS, VMware recommends against versioned buckets. For more information, see *Step 3: Configure PAS File Storage* in Dell EMC ECS with Pivotal Cloud Foundry. You can use mirroring as an alternative to versioning.
- For unversioned S3 buckets, deselect the **Use versioning for backup and restore** checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up. For more information about setting up external S3 blobstores, see Enable Versioning on Your S3-Compatible Blobstore in *Backup and Restore for External Blobstores* in the Cloud Foundry documentation.

12. Enter the name of the region in which your backup S3 buckets are located. For example, `us-west-2`. These are the buckets used to back up and restore the contents of your S3 filestore.
13. (Optional) Enter names for your backup S3 buckets:

Ops Manager Field	Value	Description
Backup buildpacks bucket name	`buildpacks-backup-bucket`	This S3 bucket is used to back up and restore your buildpacks bucket. This bucket name must be different from the buckets you named above.
Backup droplets bucket name	`droplets-backup-bucket`	This S3 bucket is used to back up and restore your droplets bucket. VMware recommends that you use a unique bucket name for droplet backups, but you can also use the same name as above.
Backup packages bucket name	`packages-backup-bucket`	This S3 bucket is used to back up and restore your packages bucket. VMware recommends that you use a unique bucket name for package backups, but you can also use the same name as above.

14. Click **Save**.

Note: For more information about AWS S3 signatures, see Authenticating Requests in the AWS documentation.

Fog with AWS IAM Instance Profiles

To configure Fog blobstores to use AWS IAM instance profiles:

1. Configure an additional `cloud-controller` IAM role with the following policy to give access to the S3 buckets you plan to use:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [ "s3:*" ],
    "Resource": [
      "arn:aws:s3:::YOUR-AWS-BUILDPACK-BUCKET",
      "arn:aws:s3:::YOUR-AWS-BUILDPACK-BUCKET/*",
      "arn:aws:s3:::YOUR-AWS-DROPLET-BUCKET",
      "arn:aws:s3:::YOUR-AWS-DROPLET-BUCKET/*",
      "arn:aws:s3:::YOUR-AWS-PACKAGE-BUCKET",
      "arn:aws:s3:::YOUR-AWS-PACKAGE-BUCKET/*",
      "arn:aws:s3:::YOUR-AWS-RESOURCE-BUCKET",
      "arn:aws:s3:::YOUR-AWS-RESOURCE-BUCKET/*",
    ]
  }]
}
```

Replace `YOUR-AWS-BUILDPACK-BUCKET`, `YOUR-AWS-DROPLET-BUCKET`, `YOUR-AWS-PACKAGE-BUCKET`, and `YOUR-AWS-RESOURCE-BUCKET` with the names of your AWS buckets. Do not use periods (`.`) in your AWS bucket names.

If you use the AWS console, an IAM role is automatically assigned to an IAM instance profile with the same name, `cloud-controller`. If you do not use the AWS console, you must create an IAM instance profile with a single assigned IAM role. For more information, see Step 4: Create an IAM Instance Profile for Your Amazon EC2 Instances in the AWS documentation.

2. In your BOSH cloud config, create a VM extension to add the IAM instance profile you created to VMs using the extension.

```
vm_extensions:
- cloud_properties:
    iam_instance_profile: cloud-controller
  name: cloud-controller-iam
```

Note: You can also create a VM extension using the Ops Manager API. For more information, see Create or Update a VM Extension in *Managing Custom VM Extensions*.

3. In your TAS for VMs deployment manifest, use the `cloud-controller-iam` VM extension you created for the instance groups containing `cloud_controller`, `cloud_controller_worker`, and `cloud_controller_clock`, as in the example below:

```
instance_groups:
...
- name: api
  ...
  vm_extensions:
  - cloud-controller-iam
...
- name: cc-worker
  ...
  vm_extensions:
  - cloud-controller-iam
...
- name: scheduler
  ...
  vm_extensions:
  - cloud-controller-iam
```

4. Insert the following configuration into your deployment manifest under `properties.cc`:

```
cc:
  buildpacks:
    blobstore_type: fog
    buildpack_directory_key: YOUR-AWS-BUILDPACK-BUCKET
    fog_connection: &fog_connection
      provider: AWS
      region: us-east-1
      use_iam_profile: true
  droplets:
    blobstore_type: fog
    droplet_directory_key: YOUR-AWS-DROPLET-BUCKET
    fog_connection: *fog_connection
  packages:
    blobstore_type: fog
    app_package_directory_key: YOUR-AWS-PACKAGE-BUCKET
    fog_connection: *fog_connection
  resource_pool:
    blobstore_type: fog
    resource_directory_key: YOUR-AWS-RESOURCE-BUCKET
    fog_connection: *fog_connection
```

   Replace `YOUR-AWS-BUILDPACK-BUCKET`, `YOUR-AWS-DROPLET-BUCKET`, `YOUR-AWS-PACKAGE-BUCKET`, and `YOUR-AWS-RESOURCE-BUCKET` with the names of your AWS buckets. Do not use periods (`.`) in your AWS bucket names.

5. (Optional) Provide other configuration with the `fog_connection` hash, which is passed through to the Fog gem.

GCP

This section describes how to configure file storage for GCP. Follow the procedure that corresponds to your installation method:

- Manual
- Terraform

For production-level Ops Manager deployments on GCP, VMware recommends selecting **External Google Cloud Storage**. For more information about production-level Ops Manager deployments on GCP, see GCP Reference Architecture.

Manual

This section describes how to configure file storage for GCP if you installed Ops Manager manually.

TAS for VMs can use Google Cloud Storage (GCS) as its external filestore by using either a GCP interoperable storage access key or your GCS Service Account.

To configure file storage for GCP, follow one of these procedures:

- Use an access key and secret key. For more information, see External Google Cloud Storage with Access Key and Secret Key below.

- Use a service account. For more information, see External Google Cloud Storage with Service Account below.

External Google Cloud Storage with Access Key and Secret Key

To configure file storage for GCP using an access key and secret key:

1. Select **External Google Cloud Storage with access key and secret key**.
2. Enter values for **Access key** and **Secret key**. To obtain the values for these fields:
    1. In the GCP Console, navigate to the **Storage** tab.
    2. Click **Settings**.
    3. Click **Interoperability**.
    4. If necessary, click **Enable interoperability access**. If interoperability access is already enabled, confirm that the default project matches the project where you are installing TAS for VMs.
    5. Click **Create a new key**.
    6. Copy and paste the generated values into the corresponding TAS for VMshttps://docs.vmware.com/en/VMware-Tanzu-Operations-Manager/2.10/vmware-tanzu-ops-manager/gcp-prepare-env-manual.html#dbs fields. TAS for VMs uses these values for authentication when connecting to Google Cloud Storage.
3. Enter the names of the storage buckets you created in Step 6: Create Database Instance and Databases in *Preparing to Deploy Ops Manager on GCP*:
    - **Buildpacks bucket name:** `PREFIX-PCF-buildpacks`
    - **Droplets bucket name:** `PREFIX-PCF-droplets`
    - **Packages bucket name:** `PREFIX-PCF-packages`
    - **Resources bucket name:** `PREFIX-PCF-resources`

      Where `PREFIX` is a prefix of your choice, required to make the bucket name unique.
4. Click **Save**.

External Google Cloud Storage with Service Account

To configure file storage for GCP using a service account:

> **Note:** You can either use the same service account that you created for Ops Manager, or create a separate service account for TAS for VMs file storage. To create a separate service account for TAS for VMs file storage, follow the procedure in Step 1: Set Up IAM Service Accounts in *Preparing to Deploy Ops Manager on GCP*, but only select the **Storage > Storage Admin** role.

1. Select **External Google Cloud Storage with service account**.

2. For **GCP project ID**, enter the Project ID on your GCP Console that you want to use for your TAS for VMs file storage.
3. For **GCP service account email**, enter the email address associated with your GCP account.
4. For **GCP service account key JSON**, enter the account key that you use to access the specified GCP project, in JSON format.
5. Enter the names of the storage buckets you created in Step 7: Create Storage Buckets in *Preparing to Deploy Ops Manager on GCP*:
   - **Buildpacks bucket name:** `PREFIX-PCF-buildpacks`
   - **Droplets bucket name:** `PREFIX-PCF-droplets`
   - **Packages bucket name:** `PREFIX-PCF-packages`
   - **Resources bucket name:** `PREFIX-PCF-resources`
   - **Backup bucket name:** `PREFIX-PCF-backup`

     Where `PREFIX` is a prefix of your choice, required to make the bucket name unique.
6. Click **Save**.

Terraform

This section describes how to configure file storage for GCP if you installed Ops Manager with Terraform.

TAS for VMs can use Google Cloud Storage (GCS) as its external filestore by using either a GCP interoperable storage access key or your GCS Service Account.

To configure file storage for GCP, follow one of these procedures:

- Use an access key and secret key. For more information, see External Google Cloud Storage with Access Key and Secret Key below.
- Use a service account. For more information, see External Google Cloud Storage with Service Account below.

External Google Cloud Storage with Access Key and Secret Key

To configure file storage for GCP using an access key and secret key:

1. Select **External Google Cloud Storage with access key and secret key**.
2. Enter values for **Access key** and **Secret key**. To obtain the values for these fields:
   1. In the GCP Console, navigate to the **Storage** tab.
   2. Click **Settings**.
   3. Click **Interoperability**.
   4. If necessary, click **Enable interoperability access**. If interoperability access is already enabled, confirm that the default project matches the project where you are installing TAS for VMs.

5. Click **Create a new key**.
6. Copy and paste the generated values into the corresponding TAS for VMs fields. TAS for VMs uses these values for authentication when connecting to Google Cloud Storage.

3. Enter the names of the storage buckets you created in GCP Service Account Key for Blobstore in *Deploying Ops Manager on GCP Using Terraform*:
   - **Buildpacks bucket name:** Enter the value of `buildpacks_bucket` from your Terraform output.
   - **Droplets bucket name:** Enter the value of `droplets_bucket` from your Terraform output.
   - **Packages bucket name:** Enter the value of `resources_bucket` from your Terraform output.
   - **Resources bucket name:** Enter the value of `packages_bucket` from your Terraform output.
4. Click **Save**.

External Google Cloud Storage with Service Account

To configure file storage for GCP using a service account:

> **Note:** You can either use the same service account that you created for Ops Manager, or create a separate service account for TAS for VMs file storage. To create a separate service account for TAS for VMs file storage, follow the procedure in Step 1: Set Up IAM Service Accounts in *Preparing to Deploy Ops Manager on GCP Manually*, but only select the **Storage > Storage Admin** role.

1. Select **External Google Cloud Storage with service account**.
2. For **GCP project ID**, enter the Project ID on your GCP Console that you want to use for your TAS for VMs file storage.
3. For **GCP service account email**, enter the email address associated with your GCP account.
4. For **GCP service account key JSON**, enter the account key that you use to access the specified GCP project, in JSON format.
5. Enter the names of the storage buckets you created in GCP Service Account Key for Blobstore in *Deploying Ops Manager on GCP Using Terraform*:
   - **Buildpacks bucket name:** Enter the value of `buildpacks_bucket` from your Terraform output.
   - **Droplets bucket name:** Enter the value of `droplets_bucket` from your Terraform output.
   - **Packages bucket name:** Enter the value of `resources_bucket` from your Terraform output.

- **Resources bucket name:** Enter the value of `packages_bucket` from your Terraform output.
- **Backup bucket name:** Enter the value of `backup_bucket` from your Terraform output.

6. Click **Save**.

Azure

This section describes how to configure file storage for Azure.

For production-level Ops Manager deployments on Azure, VMware recommends selecting **External Azure Storage**. For more information about production-level Ops Manager deployments on Azure, see Azure Reference Architecture.

For more factors to consider when selecting file storage, see Configure File Storage in *Configuring TAS for VMs for Upgrades*.

To use external Azure file storage for your TAS for VMs filestore:

1. Select **External Azure storage**.
2. To create a new storage account for the TAS for VMs filestore:
    1. Navigate to the Azure Portal.
    2. Select the **Storage accounts** tab.
    3. Click the **+** icon to add a new storage account.
    4. In the **Name** field, enter a unique name for the storage account. This name must be all lowercase and contain 3 to 24 alphanumeric characters.
    5. For **Deployment model**, select **Resource manager**.
    6. From the **Account kind** dropdown, select **General purpose**.
    7. For **Performance**, select **Standard**.
    8. From the **Replication** dropdown, select **Locally-redundant storage (LRS)**.
    9. For **Storage service encryption**, select **Disabled**.
    10. From the **Subscription** dropdown, select the subscription where you want to deploy TAS for VMs resources.
    11. For **Resource group**, select **Use existing** and enter the name of the resource group where you deployed TAS for VMs.
    12. From the **Location** dropdown, select the location where you are deploying TAS for VMs.
    13. Click **Create**.
3. To create new storage containers in the storage account you created in the previous step:
    1. In the Azure Portal, select the new storage account from the dashboard.
    2. Under **Blob Service**, select **Containers** to create one or more containers in this storage account for buildpacks, droplets, resources, and packages.

   3. Select **Soft Delete**.
   4. Select **Enabled** to enable soft delete in your Azure storage account.

      > **Note:** BBR requires that you enable soft delete in your Azure storage account before you enable backup and restore for your Azure blobstores in Ops Manager. You should set a reasonable retention policy to minimize storage costs. For more information on enabling soft delete in your Azure storage account, see Soft delete for blobs in the Azure documentation.

   5. For each container that you create, set the **Access type** to **Private**.

4. In TAS for VMs, enter the name of the storage account you created for **Account name**.
5. In the **Access key** field, enter one of the access keys provided for the storage account. To obtain a value for this field:
   1. Navigate to the Azure Portal.
   2. Select the **Storage accounts** tab.
   3. Click **Access keys**.
6. For **Environment**, enter the name of the Azure Cloud environment that contains your storage. This value defaults to AzureCloud, but other options include AzureChinaCloud, AzureUSGovernment, and AzureGermanCloud.
7. For **Buildpacks container name**, enter the container name for storing your app buildpacks.
8. For **Droplets container name**, enter the container name for your app droplet storage. VMware recommends that you use a unique container name, but you can use the same container name as the previous step.
9. For **Packages container name**, enter the container name for packages. VMware recommends that you use a unique container name, but you can use the same container name as the previous step.
10. For **Resources container name**, enter the container name for resources. VMware recommends that you use a unique container name, but you can use the same container name as the previous step.
11. (Optional) To enable backup and restore for your Azure blobstores in TAS for VMs, select the **Enable backup and restore** checkbox.

    > **Note:** Soft deletes must be enabled for all storage containers listed.

12. (Optional) To enable TAS for VMs to restore your containers to a different Azure storage account than the account where you take backups:
    1. Under **Restore from storage account**, enter the name of the Azure storage account you want to restore your containers from. Leave this field blank if you want to restore to the same storage account where you take backups.

2. Under **Restore using access key**, enter the access key for the Azure storage account you specified in **Restore from storage account**. Leave this field blank if you want to restore to the same storage account where you take backups.

13. Click **Save**.

> **Note:** To enable backup and restore of your TAS for VMs tile that uses an S3-compatible blobstore, see Enabling External Blobstore Backups.

OpenStack

For production-level Ops Manager deployments on OpenStack, VMware recommends selecting **External S3-compatible filestore**. For more information about production-level Ops Manager deployments on OpenStack, see OpenStack Reference Architecture.

For more factors to consider when selecting file storage, see Configure File Storage in *Configuring TAS for VMs for Upgrades*.

To use an external S3-compatible filestore for TAS for VMs file storage:

1. Select **External S3-compatible filestore**.
2. For **URL endpoint**, enter the `https://` URL endpoint for your region. For example, `https://s3.us-west-2.amazonaws.com/`.
3. Enter the **Access key** and **Secret key** of the `pcf-user` you created when configuring AWS for Ops Manager.
4. From the **S3 signature version** dropdown, select **V4 signature**. For more information about S3 signatures, see Signing AWS API Requests in the AWS documentation.
5. For **Region**, enter the region in which your S3 buckets are located. For example, `us-west-2`.
6. To encrypt the contents of your S3 filestore, select **Server-side encryption**. This option is only available for AWS S3.
7. (Optional) If you selected **Server-side encryption**, you can also specify a **KMS key ID**. TAS for VMs uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, TAS for VMs uses the default AWS key. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS) in the AWS documentation.
8. Deselect **Path-style S3 URLs (deprecated)** checkbox. When this checkbox is deselected, the S3 bucket is accessed using the virtual-hosted model instead of the path-based model. The deprecated path-based model is removed from AWS as of September 30, 2020. For more information about S3 path deprecation, see Amazon S3 Path Deprecation Plan – The Rest of the Story on the AWS News Blog.
9. Enter names for your S3 buckets:

Ops Manager Field	Value	Description
Buildpacks bucket name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.

Droplets bucket name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. VMware recommends that you use a unique bucket name for droplets, but you can also use the same name as above.
Packages bucket name	`pcf-packages-bucket`	This S3 bucket stores app packages. VMware recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources bucket name	`pcf-resources-bucket`	This S3 bucket stores app resources. VMware recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

10. Configure these checkboxes depending on whether your S3 buckets have versioning enabled:

    - For versioned S3 buckets, enable the **Use versioning for backup and restore** checkbox to archive each bucket to a version.
      If you are using Dell ECS, VMware recommends against versioned buckets. For more information, see *Step 3: Configure PAS File Storage* in Dell EMC ECS with Pivotal Cloud Foundry. You can use mirroring as an alternative to versioning.

    - For unversioned S3 buckets, deselect the **Use versioning for backup and restore** checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up. For more information about setting up external S3 blobstores, see Enable Versioning on Your S3-Compatible Blobstore in *Backup and Restore for External Blobstores* in the Cloud Foundry documentation.

11. Enter the name of the region in which your backup S3 buckets are located. For example, `us-west-2`. These are the buckets used to back up and restore the contents of your S3 filestore.

12. (Optional) Enter names for your backup S3 buckets:

Ops Manager Field	Value	Description
Backup buildpacks bucket name	`buildpacks-backup-bucket`	This S3 bucket is used to back up and restore your buildpacks bucket. This bucket name must be different from the buckets you named above.
Backup droplets bucket name	`droplets-backup-bucket`	This S3 bucket is used to back up and restore your droplets bucket. VMware recommends that you use a unique bucket name for droplet backups, but you can also use the same name as above.
Backup packages bucket name	`packages-backup-bucket`	This S3 bucket is used to back up and restore your packages bucket. VMware recommends that you use a unique bucket name for package backups, but you can also use the same name as above.

13. Click **Save**.

> **Note:** For more information about AWS S3 signatures, see Authenticating Requests in the AWS documentation.

vSphere

For production-level Ops Manager deployments on vSphere, VMware recommends selecting **External S3-compatible filestore** and using an external filestore. For more information about production-level Ops Manager deployments on vSphere, see vSphere Reference Architecture.

For more factors to consider when selecting file storage, see Configure File Storage in *Configuring TAS for VMs for Upgrades*.

To use an external S3-compatible filestore for TAS for VMs file storage:

1. Select **External S3-compatible filestore**.
2. For **URL endpoint**, enter the `https://` URL endpoint for your region. For example, `https://s3.us-west-2.amazonaws.com/`.
3. Enter the **Access key** and **Secret key** of the `pcf-user` you created when configuring AWS for Ops Manager.
4. From the **S3 signature version** dropdown, select **V4 signature**. For more information about S3 signatures, see Signing AWS API Requests in the AWS documentation.
5. For **Region**, enter the region in which your S3 buckets are located. For example, `us-west-2`.
6. To encrypt the contents of your S3 filestore, select **Server-side encryption**. This option is only available for AWS S3.
7. (Optional) If you selected **Server-side encryption**, you can also specify a **KMS key ID**. TAS for VMs uses the KMS key to encrypt files uploaded to the blobstore. If you do not provide a KMS Key ID, TAS for VMs uses the default AWS key. For more information, see Protecting Data Using Server-Side Encryption with AWS KMS–Managed Keys (SSE-KMS) in the AWS documentation.
8. Deselect **Path-style S3 URLs (deprecated)** checkbox. When this checkbox is deselected, the S3 bucket is accessed using the virtual-hosted model instead of the path-based model. The deprecated path-based model is removed from AWS as of September 30, 2020. For more information about S3 path deprecation, see Amazon S3 Path Deprecation Plan – The Rest of the Story on the AWS News Blog.
9. Enter names for your S3 buckets:

Ops Manager Field	Value	Description
Buildpacks bucket name	`pcf-buildpacks-bucket`	This S3 bucket stores app buildpacks.
Droplets bucket name	`pcf-droplets-bucket`	This S3 bucket stores app droplets. VMware recommends that you use a unique bucket name for droplets, but you can also use the same name as above.

Packages bucket name	`pcf-packages-bucket`	This S3 bucket stores app packages. VMware recommends that you use a unique bucket name for packages, but you can also use the same name as above.
Resources bucket name	`pcf-resources-bucket`	This S3 bucket stores app resources. VMware recommends that you use a unique bucket name for app resources, but you can also use the same name as above.

10. Configure these checkboxes depending on whether your S3 buckets have versioning enabled:

    - For versioned S3 buckets, enable the **Use versioning for backup and restore** checkbox to archive each bucket to a version.
      If you are using Dell ECS, VMware recommends against versioned buckets. For more information, see *Step 3: Configure PAS File Storage* in Dell EMC ECS with Pivotal Cloud Foundry. You can use mirroring as an alternative to versioning.

    - For unversioned S3 buckets, deselect the **Use versioning for backup and restore** checkbox, and enter a backup bucket name for each active bucket. The backup bucket name must be different from the name of the active bucket it backs up. For more information about setting up external S3 blobstores, see Enable Versioning on Your S3-Compatible Blobstore in *Backup and Restore for External Blobstores* in the Cloud Foundry documentation.

11. Enter the name of the region in which your backup S3 buckets are located. For example, `us-west-2`. These are the buckets used to back up and restore the contents of your S3 filestore.

12. (Optional) Enter names for your backup S3 buckets:

Ops Manager Field	Value	Description
Backup buildpacks bucket name	`buildpacks-backup-bucket`	This S3 bucket is used to back up and restore your buildpacks bucket. This bucket name must be different from the buckets you named above.
Backup droplets bucket name	`droplets-backup-bucket`	This S3 bucket is used to back up and restore your droplets bucket. VMware recommends that you use a unique bucket name for droplet backups, but you can also use the same name as above.
Backup packages bucket name	`packages-backup-bucket`	This S3 bucket is used to back up and restore your packages bucket. VMware recommends that you use a unique bucket name for package backups, but you can also use the same name as above.

13. Click **Save**.

> **Note:** For more information about AWS S3 signatures, see Authenticating Requests in the AWS documentation.

Configuring Load Balancing for TAS for VMs

This topic tells you how to configure load balancing for VMware Tanzu Application Service for VMs (TAS for VMs) by entering the names of your load balancers in the **Resource Config** pane of the TAS for VMs tile. This procedure varies by your IaaS and how you prepared (“paved”) it for installing TAS for VMs. See the section below that corresponds to your use case.

AWS

AWS with Terraform

To set up load balancing for TAS for VMs on AWS using Terraform:

1. Create a file named `vm_extensions_config.yml` with the following content, depending on which release you are using:
    - **VMware Tanzu Application Service for VMs (TAS for VMs)**:

      ```
      ---
      product-name: cf
      resource-config:
        diego_brain:
          elb_names:
          - alb:SSH_TARGET_GROUP_1
          - alb:SSH_TARGET_GROUP_2
          additional_vm_extensions:
          - ssh-lb-security-groups
        router:
          elb_names:
          - alb:WEB_TARGET_GROUPS_1
          - alb:WEB_TARGET_GROUPS_2
          additional_vm_extensions:
          - web-lb-security-groups
        tcp_router:
          elb_names:
          - alb:TCP_TARGET_GROUP_1
          - alb:TCP_TARGET_GROUP_2
          additional_vm_extensions:
          - tcp-lb-security-groups
      ```

    - **Small Footprint TAS for VMs**:

      ```
      ---
      product-name: cf
      resource-config:
        control:
          elb_names:
          - alb:SSH_TARGET_GROUP_1
          - alb:SSH_TARGET_GROUP_2
          additional_vm_extensions:
          - ssh-lb-security-groups
        router:
          elb_names:
          - alb:WEB_TARGET_GROUPS_1
          - alb:WEB_TARGET_GROUPS_2
          additional_vm_extensions:
      ```

```
    - web-lb-security-groups
  tcp_router:
    elb_names:
    - alb:TCP_TARGET_GROUP_1
    - alb:TCP_TARGET_GROUP_2
    additional_vm_extensions:
    - tcp-lb-security-groups
```

2. Replace values in the file as follows:

    - `SSH_TARGET_GROUP_X`: Enter your SSH target groups. You can find these values by running:

      ```
      terraform output ssh_target_groups
      ```

    - `WEB_TARGET_GROUPS_X`: Enter your web target groups. You can find these values by running:

      ```
      terraform output web_target_groups
      ```

    - `TCP_TARGET_GROUP_X`: Enter your TCP target groups. You can find these values by running:

      ```
      terraform output tcp_target_groups
      ```

3. Apply the VM extension configuration using the `om` CLI. For more information about `om`, see the Om repository on GitHub.

   ```
   om -k \
     -t "OPS-MANAGER-FQDN" \
     -u "USERNAME" \
     -p "PASSWORD" \
     configure-product \
     -c vm_extensions_config.yml
   ```

   Where:

    - `OPS-MANAGER-FQDN` is the URL at which you access your Ops Manager instance. This corresponds to `ops_manager_dns` in the Terraform output.
    - `USERNAME` is the user name you entered when configuring internal authentication.
    - `PASSWORD` is the password you entered when configuring internal authentication.

      Note: If you did not configure internal authentication, you must modify this command to use a client ID and secret instead of user name and password. For more information, see Authentication in the Om repository on GitHub.

AWS Paved Manually

To configure the Gorouter or HAProxy to use AWS Elastic Load Balancers:

1. Record the names of your ELBs. If you followed the procedures in Preparing to Deploy Ops Manager on AWS, you created:
   - `-ssh-elb`: An SSH load balancer. This is a Classic Load Balancer.
   - `-tcp-elb`: A TCP load balancer. This is a Classic Load Balancer.
   - `-web-elb`: A web load balancer. This is an Application Load Balancer.
   - `-web-elb-target-group`: A target group for the web load balancer.
2. In the TAS for VMs tile, select **Resource Config**.
3. Enter the name of your SSH load balancer.
   1. Show the **LOAD BALANCERS** field underneath the job that handles SSH requests. This depends on the TAS for VMs release you are using:
      - **TAS for VMs**: Click the icon next to the **Diego Brain** job name to expand the row.
      - **Small Footprint TAS for VMs**: Click the icon next to the **Control** job name to expand the row.
   2. In the **LOAD BALANCERS** field, enter the name of your SSH load balancer: `-ssh-elb`.
4. Click to expand the **Router** row, and fill in the **LOAD BALANCERS** field with a value determined by the type of load balancer you are using:
   - **Application Load Balancer**: Enter a comma-separated list of the names of the target group of your web load balancer, prefixed with `alb:`: `alb:-web-elb-target-group`. The prefix indicates to Ops Manager that you entered the name of a target group. The prefix is required for AWS Application Load Balancers and Network Load Balancers.
   - **Classic Load Balancer**: Enter the name of the load balancer: `-web-elb`.

   Note: If you are using HAProxy in your deployment, then put the name of the load balancers in the **LOAD BALANCERS** field of the **HAProxy** row instead of the **Router** row. For a high-availability configuration, scale up the HAProxy job to more than one instance.
5. If you enabled TCP routing, expand the **TCP Router** row and enter the name of your TCP load balancer: `-tcp-elb`.

Azure

Azure with Terraform

To configure the Gorouter to use Azure load balancers:

1. Select **Resource Config**.
2. Ensure a `Standard` VM type is selected for the **Router** VM. The TAS for VMs deployment fails if you select a `Basic` VM type.

3. Click the icon next to the **Router** job name to expand the row, showing a **LOAD BALANCERS** field and **INTERNET CONNECTED** checkbox underneath.
4. Enter the value of `web_lb_name` from your Terraform output in the **Resource Config** pane under **LOAD BALANCERS** for the **Router** job.
5. Click to expand the row for the job that handles SSH requests. This depends on the TAS for VMs release you are using:
   - **TAS for VMs**: Click the icon next to **Diego Brain**.
   - **Small Footprint TAS for VMs**: Click the icon next to **Control**.
6. For the SSH load balancer, enter the value of `diego_ssh_lb_name` from your Terraform output.
7. Ensure that the **INTERNET CONNECTED** checkboxes are deselected for all jobs.
8. Scale the number of instances as appropriate for your deployment.

> **Note:** For a high-availability deployment of Ops Manager on Azure, VMware recommends scaling the number of each TAS for VMs job to a minimum of three instances. Using three or more instances for each job creates a sufficient number of availability sets and fault domains for your deployment. For more information, see Azure Reference Architecture.

Azure Paved Manually

To configure the Gorouter to use Azure load balancers:

1. Select **Resource Config**.
2. Ensure a `Standard` VM type is selected for the **Router** VM. The TAS for VMs deployment fails if you select a `Basic` VM type.
3. Retrieve the name(s) of your external Azure Load Balancer (ALB) by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer** resource.

> **Note:** The Azure portal sometimes displays the names of resources with incorrect capitalization. Always use the Azure CLI to retrieve the correctly capitalized name of a resource. To see the list of resources, run `az network lb list`.

4. In the **Resource Config** pane of the TAS for VMs tile, click the icon next to the **Router** job name to expand the row, showing a **LOAD BALANCERS** field and **INTERNET CONNECTED** checkbox underneath.
5. Enter the name of your external ALB in the field under **LOAD BALANCERS**. If you have multiple, external ALBs, this field accepts a comma-separated list.
6. Retrieve the name of your Diego SSH Load Balancer by navigating to the Azure portal, clicking **All resources**, and locating your **Load balancer** resource.

7. In the **Resource Config** pane of the TAS for VMs tile, expand the row for the job that handles SSH requests. This depends on the TAS for VMs release you are using:
   - **TAS for VMs**: Click the icon next to **Diego Brain**.
   - **Small Footprint TAS for VMs**: Click the icon next to **Control**.
8. Enter the name of the Diego SSH Load Balancer in the field under **LOAD BALANCERS**.
9. Ensure that the **INTERNET CONNECTED** checkboxes are deselected for all jobs.
10. Scale the number of instances as appropriate for your deployment.

    > **Note:** For a high availability deployment of Ops Manager on Azure, VMware recommends scaling the number of each TAS for VMs job to a minimum of three instances. Using three or more instances for each job creates a sufficient number of availability sets and fault domains for your deployment. For more information, see Azure Reference Architecture.

GCP

GCP with Terraform

To configure the Gorouter to use GCP load balancers:

1. Select **Resource Config**.
2. Click the icon next to the **Router** job name to expand the row, showing a **LOAD BALANCERS** field and **INTERNET CONNECTED** checkbox underneath.
3. In the **LOAD BALANCERS** field, enter a comma-separated list consisting of the values of `ws_router_pool` and `http_lb_backend_name` from your Terraform output. For example, `tcp:-cf-ws,http:-httpslb`. These are the names of the TCP WebSockets and HTTP(S) load balancers for your deployment.

    > **Note:** Do not add a space between key and value pairs in the `LOAD BALANCER` field, or it fails.

    > **Note:** If you are using HAProxy in your deployment, then enter the above load balancer values in the `LOAD BALANCERS` field of the **HAProxy** row instead of the **Router** row. For a high-availability configuration, scale up the HAProxy job to more than one instance.

4. If you enabled TCP routing in the **Networking** pane of the TAS for VMs tile, add the value of `tcp_router_pool` from your Terraform output, prepended with `tcp:`, to the **LOAD BALANCERS** column of the **TCP Router** row. For example, `tcp:-cf-tcp`.
5. Expand the row for the job that handles SSH requests. This depends on the TAS for VMs release you are using:
   - **TAS for VMs**: Click the icon next to **Diego Brain**.

   - **Small Footprint TAS for VMs**: Click the icon next to **Control**.

6. Under **LOAD BALANCERS** for the SSH job, enter the value of `ssh_router_pool` from your Terraform output, prepended with `tcp:`. For example, `tcp:PCF-ssh-proxy`.

7. Verify that the **Internet Connected** checkbox for every job is enabled. The Terraform templates do not provision a Network Address Translation (NAT) box for Internet connectivity to your VMs, so they are provided with ephemeral public IP addresses to allow the jobs to reach the Internet.

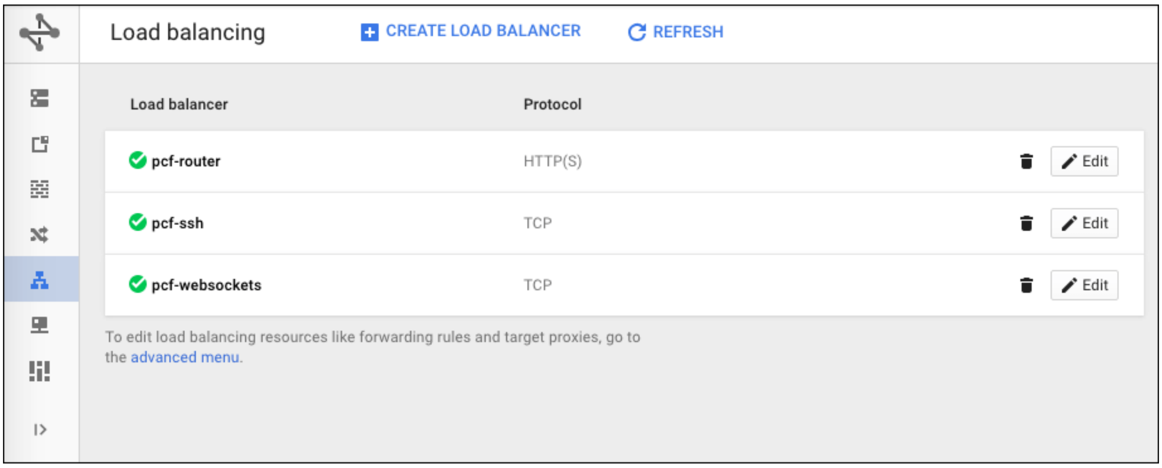
   > **Note:** If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deactivate the **Internet Connected** checkboxes. For more information about using NAT in GCP, see VPC network overview in the GCP documentation.

8. Click **Save**.

GCP Paved Manually

To configure the Gorouter to use GCP load balancers:

1. Go to the GCP Console and select **Load balancing**.



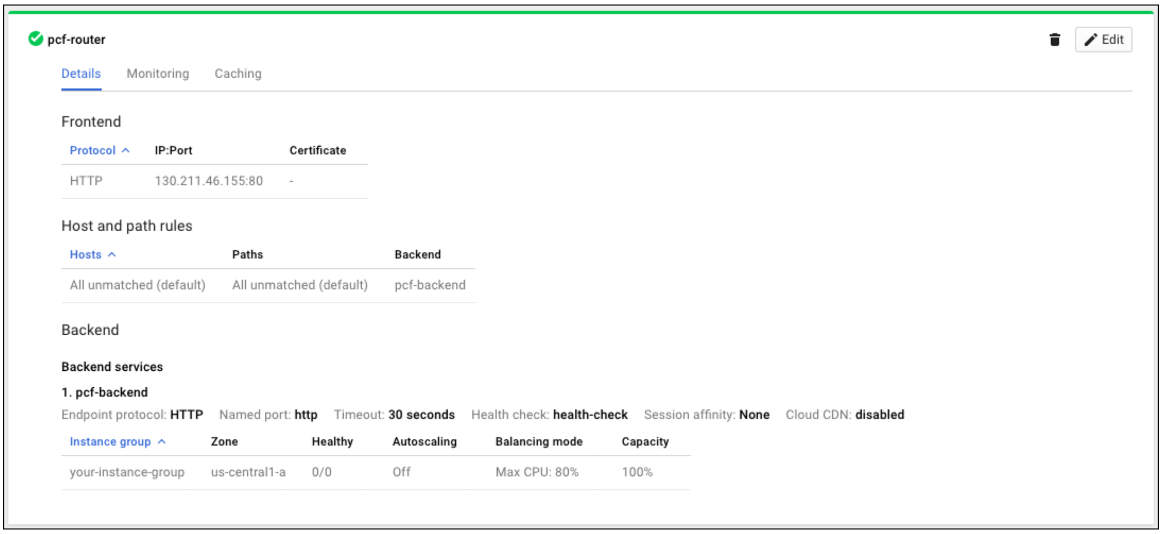
Load balancer	Protocol
pcf-router	HTTP(S)
pcf-ssh	TCP
pcf-websockets	TCP

To edit load balancing resources like forwarding rules and target proxies, go to the advanced menu.

   You can see the SSH load balancer, the HTTP(S) load balancer, the TCP WebSockets load balancer, and the TCP router that you created in Preparing to Deploy Ops Manager on GCP.

2. Record the name of your SSH load balancer and your TCP WebSockets load balancer, `PCF-wss-logs` and `PCF-ssh-proxy`.

3. Click your HTTP(S) load balancer, `PCF-global-`.



Frontend

Protocol ^	IP:Port	Certificate
HTTP	130.211.46.155:80	-

Host and path rules

Hosts ^	Paths	Backend
All unmatched (default)	All unmatched (default)	pcf-backend

Backend

Backend services

1. pcf-backend

Endpoint protocol: **HTTP** Named port: **http** Timeout: **30 seconds** Health check: **health-check** Session affinity: **None** Cloud CDN: **disabled**

Instance group ^	Zone	Healthy	Autoscaling	Balancing mode	Capacity
your-instance-group	us-central1-a	0/0	Off	Max CPU: 80%	100%

4. Under **Backend services**, record the name of the back end service of the HTTP(S) load balancer, `PCF-http-lb-backend`.

5. In the TAS for VMs tile, select **Resource Config**.

6. Click the icon next to the **Router** job name to expand the row, showing a **LOAD BALANCERS** field and **INTERNET CONNECTED** checkbox underneath.

7. In the **LOAD BALANCERS** field, enter a comma-separated list consisting of the name of your TCP WebSockets load balancer and the name of your HTTP(S) load balancer back end with the protocol prepended. For example, `tcp:PCF-wss-logs,http:PCF-http-lb-backend`.

   Note: Do not add a space between key and value pairs in the `LOAD BALANCER` field, or it fails.

   Note: If you are using HAProxy in your deployment, enter the above load balancer values in the `LOAD BALANCERS` field of the **HAProxy** row instead of the **Router** row. For a high-availability configuration, scale up the HAProxy job to more than one instance.

8. If you enabled TCP routing in the **Networking** pane of the TAS for VMs tile and set up the TCP load balancer in GCP, add the name of your TCP load balancer, prepended with `tcp:`, to the **LOAD BALANCERS** column of the **TCP Router** row. For example, `tcp:-tcp-router`.

9. Expand the row for the job that handles SSH requests. This depends on the TAS for VMs release you are using:
   - **TAS for VMs**: Click the icon next to **Diego Brain**.
   - **Small Footprint TAS for VMs**: Click the icon next to **Control**.

10. Under **LOAD BALANCERS** for the SSH job, enter the value of `ssh_router_pool` from your Terraform output, prepended with `tcp:`. For example, `tcp:PCF-ssh-proxy`.

11. Verify that the **Internet Connected** checkbox for every job is deactivated. When preparing your GCP environment, you provisioned a Network Address Translation (NAT) box to

provide Internet connectivity to your VMs instead of providing them with public IP addresses to allow the jobs to reach the Internet.

> **Note:** If you want to provision a Network Address Translation (NAT) box to provide Internet connectivity to your VMs instead of providing them with public IP addresses, deactivate the **Internet Connected** checkboxes. For more information about using NAT in GCP, see VPC network overview in the GCP documentation.

12. Click **Save**.

OpenStack

Unless you are using your own load balancer, you must provide HAProxy with public IP addresses to use as floating IP addresses. This allows the HAProxy route traffic into the OpenStack private subnet.

To provide HAProxy with public IP addresses:

1. Select **Resource Config**.
2. Click the icon next to the **HAProxy** job name to expand the row, showing the **FLOATING IPS** field underneath.
3. Enter one or more IP addresses under **FLOATING IPS**.
4. (Optional) If you enabled TCP routing, expand the **TCP Router** row and enter the IP addresses of your TCP routers under **FLOATING IPS**.
5. Click **Save**.

vSphere with NSX-T or NSX-V

In the **Resource Config** pane of the TAS for VMs tile, you configure vSphere NSX-T or NSX-V load balancing and security group membership for TAS for VMs jobs.

To configure load balancing and security group membership for TAS for VMs jobs:

1. Select **Resource Config**.
2. For each TAS for VMs job that you want to load-balance, click the down-arrow to reveal its configuration fields. The external-facing job instance groups in TAS for VMs are:
    - **Diego Brain** in TAS for VMs, or **Control** in Small Footprint TAS for VMs, for SSH traffic
    - **Router** for web traffic
    - **TCP Router** for TCP traffic
3. Enter a load balancer configuration for each job:
    - **NSX-T**:
        1. In the vSphere NSX Manager > **Advanced Networks & Security** > **Groups** pane, define an NSGroup to include VMs running each load-balanced job,

        such as `pas-ssh-group`, `pas-tcp-group`, and `pas-web-group`.

    2. In the **NS Groups** field in **Resource Config**, enter a comma-separated list of the NSGroups you defined.
    3. In the **Logical Load Balancer**: Enter a JSON-formatted structure defining a list of `server_pools` as pairs of `name` and `port` definitions.

  - **NSX-V**:
    1. **Security Groups**: Enter a comma-separated list of Security Groups defined in NSX-V to include each load-balanced job, such as `pas-ssh-group`, `pas-tcp-group`, and `pas-web-group`.
    2. **Edge Load Balancers**: Enter a JSON-formatted structure listing edge load balancers, each defined by `edge_name`, `pool_name`, `security_group`, `port`, and `monitor_port` definitions.

4. Click **Save**.

Configuring SSL Termination for Ops Manager on vSphere

To use SSL termination in Ops Manager, you must configure the HAProxy load balancer or your own load balancer.

VMware recommends that you use HAProxy in lab and test environments only. Production environments should instead use a highly-available customer-provided load balancing solution.

Choose an SSL termination method to determine the steps you must take to configure VMware Tanzu Application Service for VMs (TAS for VMs).

Use the HAProxy Load Balancer

Ops Manager deploys with a single instance of HAProxy for use in lab and test environments. You can use this HAProxy instance for SSL termination and load balancing to the TAS for VMs Gorouters. HAProxy can generate a self-signed certificate if you do not want to obtain a signed certificate from a well-known certificate authority (CA).

> **Note:** Certificates generated in TAS for VMs are signed by the Ops Manager Certificate Authority. They are not technically self-signed, but they are referred to as "self-signed certificates" in the Ops Manager UI and throughout this documentation.

To use the HAProxy load balancer:

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the **System domain** and **Apps domain** that you configure in the **Domains** pane of the TAS for VMs tile with the HAProxy IP address.

   For example, with `.example.com` as the main subdomain for your Ops Manager installation and an HAProxy IP address `203.0.113.1`, you must create an A record in your DNS that serves `example.com` and points `*.platform_name_lc` to `203.0.113.1`.

Name	Type	Data	Domain
*.	A	203.0.113.1	example.com

2. Test your DNS entry by running:

   ```
   host
   ```

   This command should return your HAProxy IP address.

3. Navigate to the Ops Manager Installation Dashboard.
4. Click the TAS for VMs tile.
5. Select **Networking**.
6. Leave the **Gorouter IPs** field blank. HAProxy assigns the Gorouter IPs internally.
7. Enter the IP address for HAProxy in the **HAProxy IPs** field.
8. Provide your SSL certificate in the **Certificates and private keys for the Gorouter and HAProxy** fields. For more information, see Providing a Certificate for Your TLS Termination Point.

Use Another Load Balancer

Production environments must use a highly-available customer-provided load balancing solution that:

- Provides SSL termination with wildcard DNS location
- Provides load balancing to each of the TAS for VMs Gorouter IPs
- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers

To use your own load balancer:

1. Register one or more static IP address for Ops Manager with your load balancer.
2. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System domain** and **Apps domain** that you configure in the **Domains** pane of the TAS for VMs tile with the IP address of your load balancer.

   For example, with `.example.com` as the main subdomain for your Ops Manager installation and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.` to `198.51.100.1`.

Name	Type	Data	Domain
*.	A	198.51.100.1	example.com

3. Go to the Ops Manager Installation Dashboard.
4. Click the TAS for VMs tile.
5. Select **Networking**.
6. In the **Gorouter IPs** field, enter the static IP address for Ops Manager that you have registered with your load balancer.

7. Leave the **HAProxy IPs** field blank.
8. Provide your SSL certificate in the **Certificates and private keys for the Gorouter and HAProxy** fields. For more information, see Providing a Certificate for Your TLS Termination Point.

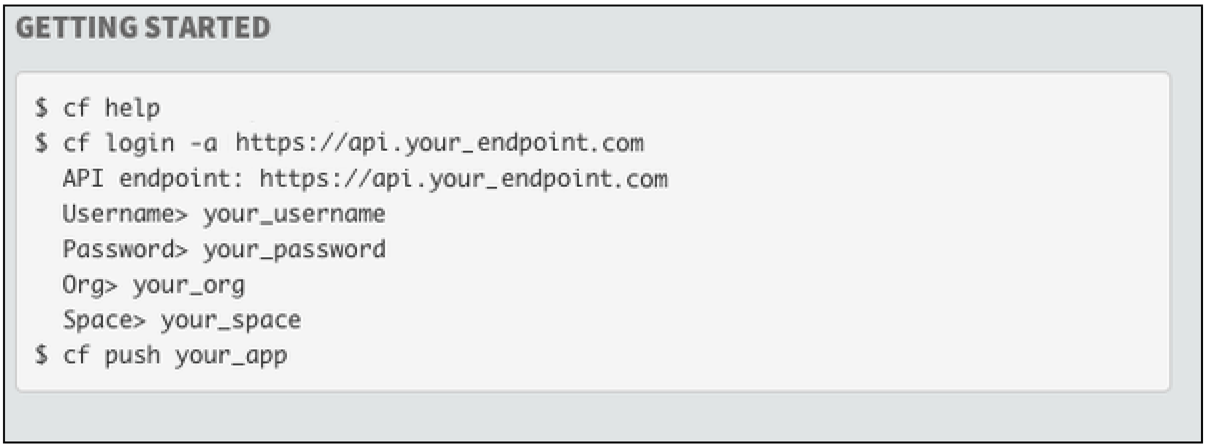
> **Note:** When adding or removing TAS for VMs Gorouters, you must update your load balancing solution configuration with the appropriate IP addresses.

Identifying the API Endpoint for Your TAS for VMs Instance

The API endpoint for your VMware Tanzu Application Service for VMs (TAS for VMs) deployment, its target URL, is the API endpoint of the deployment's Cloud Controller. Find your Cloud Controller API endpoint by consulting your cloud operator, from the Apps Manager, or from the command line.

From the Apps Manager

Log in to the Apps Manager for your TAS for VMs instance, then click **Tools** in the left navigation panel. The **Getting Started** section of the Tools page shows your API endpoint.



GETTING STARTED

```
$ cf help
$ cf login -a https://api.your_endpoint.com
  API endpoint: https://api.your_endpoint.com
  Username> your_username
  Password> your_password
  Org> your_org
  Space> your_space
$ cf push your_app
```

From the Command Line

From a command line, use the `cf api` command to view your API endpoint.

Example:

```
$ cf api
API endpoint: https://api.example.com (API version: 2.2.0)
```

Creating and modifying quota plans in TAS for VMs

This article discusses how you can use quota plans in TAS for VMs, including creating and modifying quota plans for orgs and spaces.

Quota plans overview

Quota plans are named sets of memory, service, log rate, and instance usage quotas. For example, one quota plan might allow up to 10 services, 10 routes, 2 GB of RAM, and 2 KB of generated logs, while another might offer 100 services, 100 routes, 10 GB of RAM, and 16 KB of generated logs. Quota plans have user-friendly names, but are referenced in VMware Tanzu Application Service for VMs (TAS for VMs) internal systems by unique GUIDs.

Quota plans are not directly associated with user accounts. Instead, every org has a list of available quota plans, and the account admin assigns a specific quota plan from the list to the org. Everyone in the org shares the quotas described by the plan. There is no limit to the number of defined quota plans an account can have, but only one plan can be assigned at a time.

You must set a quota plan for an org, but you can choose whether to set a space quota. For more information, see the Orgs and Spaces sections of the *Orgs, Spaces, Roles, and Permissions* topic.

For information about managing network policy quotas, see Manage Network Policy Quotas in *Configuring Container-to-Container Networking*.

Org quota plan attributes

Name	Description	Valid Values	Example Value
name	The name you use to identify the plan	A sequence of letters, digits, and underscore characters. Quota plan names within an account must be unique.	silver_quota
memory_limit	Maximum memory usage allowed	An integer and a unit of measurement like M, MB, G, or GB	2048M
app_instance_limit	Maximum app instances allowed. Stopped apps do not count toward this instance limit. Crashed apps count toward the limit because their desired state is `starting`.	An integer	25
non_basic_services_allowed	Determines whether users can provision instances of non-free service plans. Does not control plan visibility. When false, non-free service plans may be visible in the marketplace but instances can not be provisioned.	`true` or `false`	true
total_routes	Maximum routes allowed	An integer	500
total_reserved_route_ports	Maximum routes with reserved ports	An integer not greater than total_routes	60
total_services	Maximum services allowed	An integer	25
trial_db_allowed	Legacy Field. Value can be ignored.	`true` or `false`	true

Default quota plan for an org

TAS for VMs installs with a quota plan named `default` with the following values:

- Memory Limit: `10240 MB`
- Total Routes: `1000`
- Total Services: `100`
- Non-basic Services Allowed: `True`
- Trial DB Allowed: `True`

Create a new quota plan for an org

Note: The org manager sets and manages quotas. For more information, see Orgs, Spaces, Roles, and Permissions.

You must set an org quota. You can create a new quota plan for org with `cf create-quota`.

Use cf create-quota

In a terminal window, run the following command. Replace the placeholder attributes with the values for this quota plan:

```
cf create-quota QUOTA [-m TOTAL-MEMORY] [-i INSTANCE-MEMORY] [-r ROUTES] [-s SERVICE-I
NSTANCES] [--allow-paid-service-plans]
```

This command accepts the following flags:

- `-m`: Total amount of memory
- `-i`: Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-r`: Total number of routes
- `-s`: Total number of service instances
- `-allow-paid-service-plans`: Can provision instances of paid service plans

Example:

```
$ cf create-quota small -m 2048M -i 1024M -r 10 -s 10 --allow-paid-service-pl
ans
```

Modify an existing quota plan for an org

Use cf update-quota

1. Find the names of all quota definitions available to your org by running the following command, and record the name of the quota plan to be modified:

```
cf quotas
```

For example:

```
$ cf quotas
Getting quotas as admin@example.com...
OK

name           total memory limit   instance memory limit   routes
service instances   paid service plans
free           0                    0                       1000
0                   disallowed
paid           10G                  0                       1000
-1                   allowed
small          2G                   0                         10
10                   allowed
trial          2G                   0                       1000
10                   disallowed
```

2. Run the following command, replacing `QUOTA` with the name of your quota:

```
cf update-quota QUOTA [-i INSTANCE-MEMORY] [-m MEMORY] [-n NEW-NAME] [-r ROUTE
S] [-s SERVICE-INSTANCES] [--allow-paid-service-plans | --disallow-paid-service
-plans]
```

This command accepts the following flags:

- `-i`: Maximum amount of memory an application instance can have (`-1` represents an unlimited amount)
- `-m`: Total amount of memory a space can have
- `-n`: New name
- `-r`: Total number of routes
- `-s`: Total number of service instances
- `--allow-paid-service-plans`: Can provision instances of paid service plans
- `--disallow-paid-service-plans`: Can not provision instances of paid service plans

For example:

```
 $ cf update-quota small -i 2048M -m 4096M -n medium -r 20 -s 20 --allo
w-paid-service-plans
```

Create and modify quota plans for a space

For each org, Org Managers create and modify quota plans for spaces in the org. If an Org Manager allocates a space quota, TAS for VMs verifies that resources do not exceed the allocated space limit. For example, when a Space Developer deploys an app, TAS for VMs first checks the memory allocation at the space level, then at the org level.

Perform the following procedures to create and modify quota plans for individual spaces within an org.

Create a new quota plan for a space

In a terminal window, run the following command to create a quota for a space. Replace the placeholder attributes with the values for this quota plan:

```
cf create-space-quota QUOTA [-i INSTANCE-MEMORY] [-m MEMORY] [-r ROUTES] [-s SERVICE-I
NSTANCES] [--allow-paid-service-plans]
```

For example:

```
$ cf create-space-quota big -i 1024M -m 4096M -r 20 -s 20 --allow-paid-servic
e-plans
```

Modify a quota plan for a space

To find the names of all space quota available to your org, run the following command and record the name of the quota plan to be modified:

```
cf space-quotas
```

```
$ cf space-quotas
Getting quotas as admin@example.com...
OK

name            total memory limit   instance memory limit   routes       serv
ice instances   paid service plans
big             2G                   unlimited               0            10
allowed
trial           2G                   0                       0            10
allowed
```

To modify that quota, run the following command. Replace the placeholder attributes with the values for this quota plan.

```
cf update-space-quota SPACE-QUOTA-NAME [-i MAX-INSTANCE-MEMORY] [-m MEMORY] [-n NEW-NA
ME] [-r ROUTES] [-s SERVICES] [--allow-paid-service-plans | --disallow-paid-service-pl
ans]
```

For example:

```
$ cf update-space-quota big -i 20 -m 4096M -n bigger -r 20 -s 20 --allow-paid
-service-plans
```

Run cf help

For more information regarding quotas, run `cf help` to view a list and brief description of all cf CLI commands. Scroll to view org and space quotas usage and information.

```
$ cf help
...
ORG ADMIN:
   quotas                                  List available usage quotas
   quota                                   Show quota info
   set-quota                               Assign a quota to an org

   create-quota                            Define a new resource quota
   delete-quota                            Delete a quota
   update-quota                            Update an existing resource quota

   share-private-domain                    Share a private domain with an org
   unshare-private-domain                  Unshare a private domain with an or
g

SPACE ADMIN:
   space-quotas                            List available space resource quota
s
   space-quota                             Show space quota info
   create-space-quota                      Define a new space resource quota
   update-space-quota                      update an existing space quota
   delete-space-quota                      Delete a space quota definition and
unassign the space quota from all spaces
   set-space-quota                         Assign a space quota definition to
a space
   unset-space-quota                       Unassign a quota from a space
```

Stopping and starting virtual machines in TAS for VMs

You can stop and start the component TAS for VMs virtual machines that make up your deployment. This article walks you through the steps.

This article assumes you are using BOSH CLI version 2.

In some cases, you may want to stop all your VMs (for example, power down your deployment) or start all of your VMware Tanzu Application Service for VMs (TAS for VMs) VMs (for example, recover from a power outage). You can stop or start all TAS for VMs VMs with a single `bosh` command.

If you want to shut down or start up a single VM in your deployment, you can use the manual process described in Stopping and Starting Individual TAS for VMs VMs.

This procedure uses the BOSH Command Line Interface (BOSH CLI). For more information about using this tool, see Advanced Troubleshooting with the BOSH CLI.

Stopping and Starting All TAS for VMs VMs

This section describes how to stop and start all the VMs in your deployment.

Stopping All TAS for VMs VMs

To shut down all the VMs in your deployment:

1. Scale down the following jobs to one instance:
    - `consul_server`
    - `mysql`
2. Run the following command for each of the deployments listed in the previous step:

    ```
    bosh -e MY-ENV -d MY-DEPLOYMENT stop --hard
    ```

    Where:

    - `MY-ENV` is the alias you set for the BOSH Director.
    - `MY-DEPLOYMENT` is the name of your deployment.

    For example:

    ```
    $ bosh -e prod -d mysql stop --hard
    ```

    This command stops all VMs in the specified deployment. The `--hard` flag instructs BOSH to delete the VMs but retain any persistent disks.

Starting All TAS for VMs VMs

To start all the VMs in your deployment:

1. Select the product deployment for the VMs you want to shut down. You can run the following command to locate CF deployment manifests:

    ```
    $ find /var/tempest/workspaces/default/deployments -name cf-*.yml
    ```

2. Run the following command:

    ```
    bosh -e MY-ENV -d MY-DEPLOYMENT start
    ```

    Where:

    - `MY-ENV` is the alias you set for the BOSH Director.
    - `MY-DEPLOYMENT` is the name of your deployment.

    For example:

    ```
    $ bosh -e prod -d mysql start
    ```

    This command starts all VMs in the specified deployment.

3. If you require high availability in your deployment, scale up all instance groups to the original or desired counts.

Stopping and Starting Individual TAS for VMs VMs

This section describes how to stop and start individual VMs in your deployment.

Find the Names of Your TAS for VMs VMs

You need the full names of the VMs to stop and start them using the BOSH CLI. To find full names for the VMs running each component, run `bosh -e MY-ENV instances`, replacing `MY-ENV` with the alias you set for your BOSH Director. To filter the list of instances by deployment, run `bosh -e MY-ENV -d MY-DEPLOYMENT instances`.

For example:

```
$ bosh -e prod -d mysql instances
...
Deployment 'mysql'

Instance                    Process State  AZ  IPs
mysql/0123-abcd-4567ef89    running        -   10.244.0.6
mysql/abcd-0123-ef4567ab    running        -   10.244.0.2

2 instances
...
```

You can see the full name of each VM in the `Instance` column of the terminal output. Each full name includes:

- A prefix indicating the component function of the VM.
- An identifier string specific to the VM.

For any component, you can look for its prefix in the `bosh instances` output to find the full name of the VM or VMs that run it.

Stopping an Individual TAS for VMs VM

To stop a job, run the following command for the component in your TAS for VMs deployment, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment:

```
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME
```

To delete the instance that contains the job, run the following command for the component in your TAS for VMs deployment:

```
bosh -e MY-ENV -d MY-DEPLOYMENT stop VM-NAME --hard
```

Use the full name of the component VM as listed in your `bosh instances` terminal output without the unique identifier string.

For example, the following command stops the Loggregator Traffic Controller job:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller
```

To stop a specific instance of a job, include the identifier string at the end of its full name.

For example, the following command stops the Loggregator Traffic Controller job on only one Diego Cell instance:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller/0123-abcd-45
67ef89
```

To delete the VM, include `--hard` at the end of the command. This command does not delete persistent disks.

For example, the following command deletes a specific Loggregator Traffic Controller instance:

```
$ bosh -e prod -d loggregator stop loggregator_trafficcontroller/0123-abcd-45
67ef89 --hard
```

Starting an Individual TAS for VMs VM

Run the following command for the component in your TAS for VMs deployment you wish to start, replacing `MY-ENV` with the alias you set for your BOSH Director and `MY-DEPLOYMENT` with the name of the deployment. Use the full name of the component VM as listed in your `bosh vms` terminal output without the unique identifier string.

```
bosh -e MY-ENV -d MY-DEPLOYMENT start VM-NAME
```

The following example command starts the Loggregator Traffic Controller VM:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller
```

To start a specific instance of a VM, include the identifier string at the end of its full name.

For example, the following command starts the Loggregator Traffic Controller job on one Diego Cell instance:

```
$ bosh -e prod -d loggregator start loggregator_trafficcontroller/0123-abcd-4
567ef89
```

Scaling TAS for VMs

This topic describes how to scale VMware Tanzu Application Service for VMs (TAS for VMs) for different deployment scenarios.

For information about capacity scaling indicators for TAS for VMs, see Key Capacity Scaling Indicators.

TAS for VMs defaults to a highly available resource configuration. For more information, see High Availability in TAS for VMs.

Overview

To increase the capacity and availability of the TAS for VMs platform, and to decrease the chances of downtime, you can scale a deployment up using the instructions below.

Scaling Recommendations

The table below provides the instance counts VMware recommends for a high-availability deployment and the minimum instances for a functional deployment:

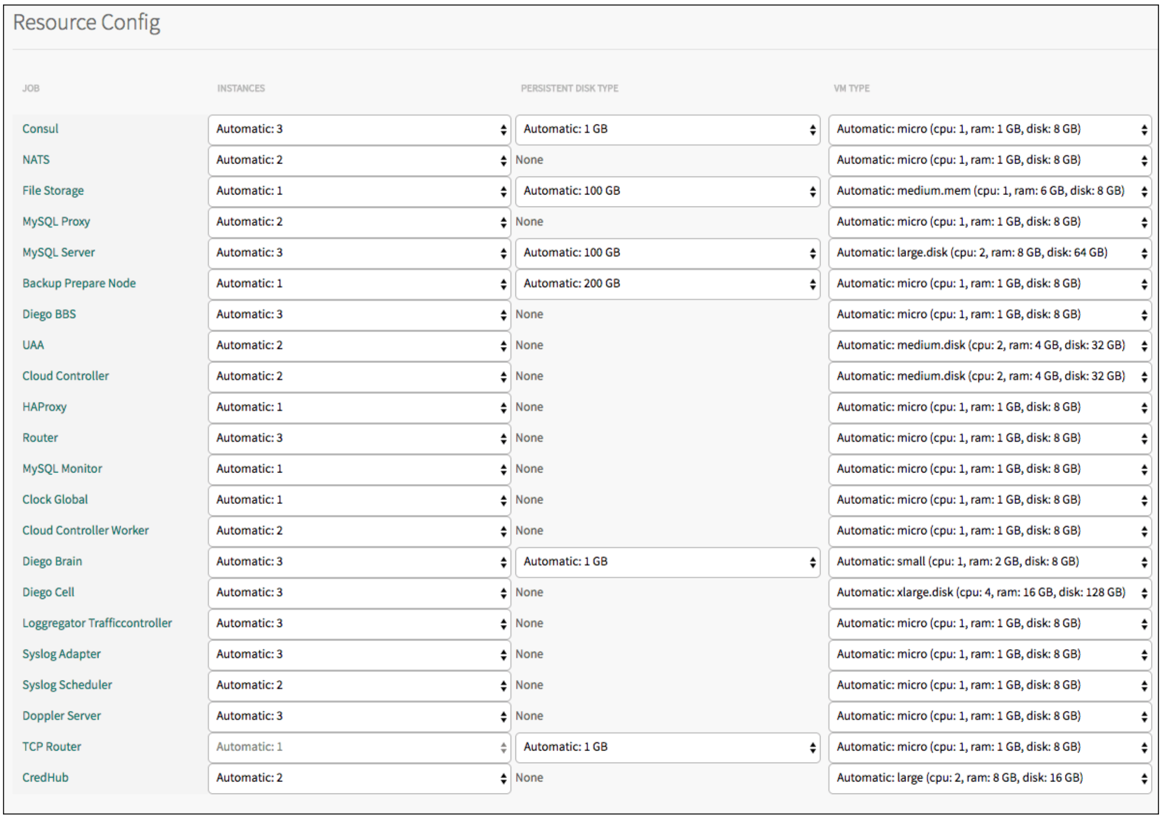
VMware Tanzu Application Service for VMs (TAS for VMs) Job	Recommended Instance Number for HA	Minimum Instance Number	Notes
Diego Cell	≥ 3	1	The optimal balance between CPU and memory sizing and instance count depends on the performance characteristics of the apps that run on Diego Cells. Scaling vertically with larger Diego Cells makes for larger points of failure, and more apps go down when a Diego Cell fails. On the other hand, scaling horizontally decreases the speed at which the system re-balances apps. Re-balancing 100 Diego Cells takes longer and demands more processing overhead than re-balancing 20 Diego Cells.
Diego Brain	≥ 2	1	For high availability, use at least one per AZ, or at least two if only one AZ.
Diego BBS	≥ 2	1	For high availability in a multi-AZ deployment, use at least one instance per AZ. Scale Diego BBS to at least two instances for high availability in a single-AZ deployment.
MySQL Server	3	1	If you use an external database in your deployment, then you can set the MySQL Server instance count to `0`. For instructions about scaling down an internal MySQL cluster, see Scaling Down Your MySQL Cluster.
MySQL Proxy	2	1	If you use an external database in your deployment, then you can set the MySQL Proxy instance count to `0`.
NATS Server	≥ 2	1	In a high-availability deployment, you might run a single NATS instance if your deployment lacks the resources to deploy two stable NATS servers. Components using NATS are resilient to message failures and the BOSH Resurrector recovers the NATS VM quickly if it becomes non-responsive.
Cloud Controller	≥ 2	1	Scale the Cloud Controller to accommodate the number of requests to the API and the number of apps in the system.
Clock Global	≥ 2	1	For a high-availability deployment, scale the Clock Global job to a value greater than 1 or to the number of AZs you have.
Router	≥ 2	1	Scale the Gorouter to accommodate the number of incoming requests. Additional instances increase available bandwidth. In general, this load is much less than the load on Diego Cells.
UAA	≥ 2	1	
Doppler Server	≥ 2	1	Deploying additional Doppler servers splits traffic across them. For a high-availability deployment, VMware recommends at least two per AZ.
Loggregator Traffic Controller	≥ 2	1	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner. For a high-availability deployment, VMware recommends at least two per AZ.
Syslog Scheduler	≥ 2	1	The Syslog Scheduler is a scalable component. For high availability, use at least one instance per AZ, or at least two instances if only one AZ is present.
CredHub	≥ 3	2	CredHub is a scalable component. For high availability, use at least one instance per AZ, or at least three instances if only one AZ is present.

Scaling Up TAS for VMs

You can determine when to scale up TAS for VMs components by reviewing the capacity scaling indicators. For more information, see Key Capacity Scaling Indicators.

To scale up TAS for VMs instances:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 3	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
NATS	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)
MySQL Proxy	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 64 GB)
Backup Prepare Node	Automatic: 1	Automatic: 200 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego BBS	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
UAA	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
Cloud Controller	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Router	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Monitor	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Cloud Controller Worker	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 128 GB)
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Adapter	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Scheduler	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Doppler Server	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
CredHub	Automatic: 2	None	Automatic: large (cpu: 2, ram: 8 GB, disk: 16 GB)

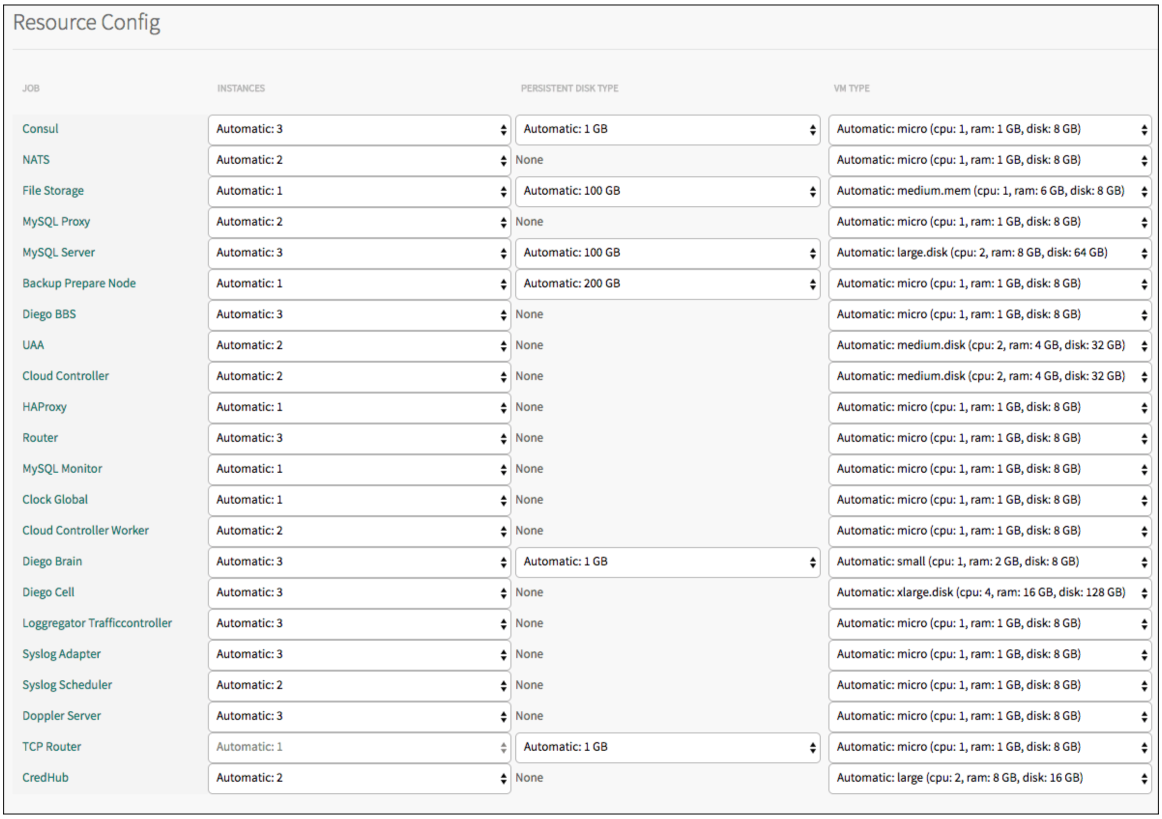
4. To scale your deployment horizontally, increase the number of **Instances** of a job. For guidance on the number of job instances required to ensure high availability, see Scaling Recommendations.
5. To scale your deployment vertically, adjust the **Persistent Disk Type** and **VM Type** of a job to allocate more disk space and memory. If you choose **Automatic** from the dropdown, TAS for VMs uses the recommended amount of resources for the job.
6. Click **Save**.
7. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

Scaling Down TAS for VMs

If you are deploying a TAS for VMs configuration that does not need to be highly available, VMware recommends scaling down job instances to the minimum number required for a functional deployment.

To scale down your deployment:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Resource Config**.



Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 3	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
NATS	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)
MySQL Proxy	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Server	Automatic: 3	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 64 GB)
Backup Prepare Node	Automatic: 1	Automatic: 200 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego BBS	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
UAA	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
Cloud Controller	Automatic: 2	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 32 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Router	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
MySQL Monitor	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Cloud Controller Worker	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Diego Brain	Automatic: 3	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 8 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 128 GB)
Loggregator Trafficcontroller	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Adapter	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Syslog Scheduler	Automatic: 2	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
Doppler Server	Automatic: 3	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
TCP Router	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
CredHub	Automatic: 2	None	Automatic: large (cpu: 2, ram: 8 GB, disk: 16 GB)

4. In the **Resource Config** screen, decrease the number of **Instances** for each job. Choose the suggested values outlined in Scaling Recommendations or in the Scaling Recommendations for Specific Deployment Configurations.
5. Click **Save**.
6. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

Scaling Down Jobs with Persistent Disk

If you scale down or delete a job that uses persistent disk, TAS for VMs marks the disk as orphaned. Orphaned disks are not attached to any job, and TAS for VMs deletes them after five days.

Use the BOSH CLI to list and recover orphaned disks. Follow the instructions in Advanced Troubleshooting with the BOSH CLI to log in to the BOSH Director, and then follow the procedures in Orphaned Disks in the BOSH documentation.

Scaling Recommendations for Specific Deployment Configurations

If you use one of the following configurations, choose the values in the corresponding table to scale instances for your particular deployment:

- Deployments Using External Databases
- Deployments Using Internal MySQL
- Deployments Using an External Blobstore
- Deployments Using External Load Balancers

Deployments Using External Databases

If you use an external database, you can scale down the instance counts for internal MySQL jobs.

Select the following values in the **Resource Config** pane:

Job	Instance Count
MySQL Server	0
MySQL Proxy	0

Deployments Using Internal MySQL

If you use the internal MySQL database on a clean install, or on an upgrade from a configuration that previously used internal MySQL databases, you do not need to change the default values shown in the table below.

To revert back to this configuration, choose the values shown in the **Resource Config** pane.

> **Note:** Changing back to this configuration deletes any data written to your other database option.

Job	Instance Count
MySQL Server	3
MySQL Proxy	2

> **Note:** Apps that do not use MySQL for VMware Tanzu are not affected by the scaling process when you redeploy TAS for VMs. In addition, redeploying TAS for VMs with the MySQL cluster means that the TAS for VMs API is unavailable for a brief period of time. For example, you are not able to push apps or query their state during this time.

For more information, see the MySQL for VMware Tanzu documentation.

Deployments Using an External Blobstore

If you use an external blobstore, select the following value in the **Resource Config** pane:

Job	Instance Count
File Storage	0

Deployments Using External Load Balancers

If you use an external load balancer, select the following values in the **Resource Config** pane:

Job	Instance Count
HAProxy	0
Router	≥ 1
Diego Brain	≥ 1

For more information about configuring load balancers in the **Resource Config** pane of the TAS for VMs tile, see Configuring Load Balancing for TAS for VMs.

Scaling Cloud Controller

This topic describes how and when to scale BOSH jobs in CAPI, and includes details about some key metrics, heuristics, and logs.

cloud_controller_ng

The `cloud_controller_ng` Ruby process is the primary job in CAPI. It, along with `nginx_cc`, powers the Cloud Controller API that all users of VMware Tanzu Application Service for VMs (TAS for VMs) interact with. In addition to serving external clients, `cloud_controller_ng` also provides APIs for internal components within TAS for VMs, such as Loggregator and Networking subsystems.

Note: Running `bosh instances --vitals` returns CPU values. The **CPU User** value corresponds with the `system.cpu.user` metric and is scaled by the number of CPUs. For example, on a 4-core `api` VM, a `cloud_controller_ng` process that is using 100% of a core is listed as using 25% in the `system.cpu.user` metric.

When to Scale

When determining whether to scale `cloud_controller_ng`, look for the following:

Key Metrics

Cloud Controller emits the following metrics:

- `cc.requests.outstanding` is at or consistently near 20.
- `system.cpu.user` is above 0.85 utilization of a single core on the API VM.
- `cc.vitals.cpu_load_avg` is 1 or higher.
- `cc.vitals.uptime` is consistently low, indicating frequent restarts, possibly due to memory pressure.

Heuristic Failures

The following behaviors may occur:

- There is a latency in average response.
- Web UI responsiveness or timeouts are degraded.
- `bosh is --ps --vitals` has elevated CPU usage for the `cloud_controller_ng` job in the API instance group.

Relevant Log Files

You can find the above heuristic failures in the following log files:

- `/var/vcap/sys/log/cloud_controller_ng/cloud_controller_ng.log`
- `/var/vcap/sys/log/cloud_controller_ng/nginx-access.log`

How to Scale

Before and after scaling Cloud Controller API VMs, verify that the Cloud Controller database is not overloaded. All Cloud Controller processes are backed by the same database, so heavy load on the database impacts API performance regardless of the number of Cloud Controllers deployed. Cloud Controller supports both PostgreSQL and MySQL, so there is no specific scaling guidance for the database.

In TAS for VMs deployments with internal MySQL clusters, a single MySQL database VM with CPU usage over 80% can be considered overloaded. When this happens, the MySQL VMs must be scaled up to prevent the added load of additional Cloud Controllers exacerbating the issue.

Cloud Controller API VMs can primarily be scaled horizontally. Scaling up the number of cores on a single VM is not effective. This is because Ruby’s Global Interpreter Lock (GIL) limits the `cloud_controller_ng` process so that it can only effectively use a single CPU core on a multi-core machine.

> **Note:** Since Cloud Controller supports both PostgreSQL and MySQL external databases, there is no absolute guidance on what a healthy database looks like. In general, high database CPU utilization is a good indicator of scaling issues, but always defer to the documentation specific to your database.

cloud_controller_worker_local

This job, also called “local workers”, is primarily responsible for handling files uploaded to the API VMs during `cf push`, such as `packages`, `droplets`, and resource matching.

When to Scale

When determining whether to scale `cloud_controller_worker_local`, look for the following:

Key Metrics

Cloud Controller emits the following metrics:

- `cc.job_queue_length.cc-VM_NAME-VM_INDEX` is continuously growing.
- `cc.job_queue_length.total` is continuously growing.

Heuristic Failures

The following behaviors may occur:

- `cf push` is intermittently failing.
- `cf push` average time is elevated.

Relevant Log Files

You can find the above heuristic failures in the following log files:

- `/var/vcap/sys/log/cloud_controller_ng/cloud_controller_ng.log`

How to Scale

Because local workers are located with the Cloud Controller API job, they are scaled horizontally along with the API.

cloud_controller_worker

Colloquially known as “generic workers” or just “workers”, this job and VM are responsible for handling asynchronous work, batch deletes, and other periodic tasks scheduled by the `cloud_controller_clock`.

When to Scale

When determining whether to scale `cloud_controller_worker`, look for the following:

Key Metrics

Cloud Controller emits the following metrics:

- `cc.job_queue_length.cc-VM_TYPE-VM_INDEX` is continuously growing. For example, `cc.job_queue_length.cc-cc-worker-0`.
- `cc.job_queue_length.total` is continuously growing.

Heuristic Failures

The following behaviors may occur:

- `cf delete-org ORG_NAME` appears to leave its contained resources around for a long time.
- Users report slow deletes for other resources.
- cf-acceptance-tests succeed generally, but fail during cleanup.

Relevant Log Files

You can find the above heuristic failures in the following log files:

- `/var/vcap/sys/log/cloud_controller_worker/cloud_controller_worker.log`

How to Scale

The cc-worker VM can safely scale horizontally in all deployments, but if your worker VMs have CPU/memory headroom, you can also use the `cc.jobs.generic.number_of_workers` BOSH property to increase the number of worker processes on each VM.

cloud_controller_clock and cc_deployment_updater

The `cloud_controller_clock` job runs Diego sync process and schedules periodic background jobs. The `cc_deployment_updater` job is responsible for handling v3 rolling app deployments. For more information, see Rolling App Deployments.

> **Note:** Running `bosh instances --vitals` returns CPU values. The **CPU User** value corresponds with the `system.cpu.user` metric and is scaled by the number of CPUs. For example, on a 4-core `api` VM, a `cloud_controller_ng` process that is using 100% of a core is listed as using 25% in the `system.cpu.user` metric.

When to Scale

When determining whether to scale `cloud_controller_clock` and `cc_deployment_updater`, look for the following:

Key Metrics

Cloud Controller emits the following metrics:

- `cc.Diego_sync.duration` is continuously increasing over time.
- `system.cpu.user` is high on the scheduler VM.

Heuristic Failures

The following behaviors may occur:

- Diego domains are frequently unfresh. For more information, see Domain Freshness in *Overview of Domains* in the BBS Server repository on GitHub.
- The Diego Desired LRP count is larger than the total process instance count reported through the Cloud Controller APIs.
- Deployments are slow to increase and decrease instance count.

Relevant Log Files

You can find the above heuristic failures in the following log files:

- `/var/vcap/sys/log/cloud_controller_clock/cloud_controller_clock.log`
- `/var/vcap/sys/log/cc_deployment_updater/cc_deployment_updater.log`

How to Scale

Both of these jobs are singletons, so extra instances are for failover HA rather than scalability. Performance issues are likely due to database overloading or greedy neighbors on the scheduler

VM.

blobstore_nginx

The internal WebDAV blobstore that comes included with TAS for VMs by default. It is used by the platform to store `packages`, staged `droplets`, `buildpacks`, and cached app resources. Files are typically uploaded to the internal blobstore through the Cloud Controller local workers and downloaded by Diego when app instances are started.

When to Scale

When determining whether to scale `blobstore_nginx`, look for the following:

Key Metrics

Cloud Controller emits the following metrics:

- `system.cpu.user` is consistently high on the `singleton-blobstore` VM.
- `system.disk.persistent.percent` is high, indicating that the blobstore is running out of room for additional files.

Heuristic Failures

The following behaviors may occur:

- `cf push` is intermittently failing.
- `cf push` average time is elevated.
- App droplet downloads are timing out or failing on Diego.

Relevant Log Files

You can find the above heuristic failures in the following log files:

- `/var/vcap/sys/log/blobstore/internal_access.log`

How to Scale

The internal WebDAV blobstore cannot be scaled horizontally, not even for availability purposes, because of its reliance on the `singleton-blobstore` VM's persistent disk for file storage. For this reason, it is not recommended for environments that require high availability. For these environments, you must use an external blobstore. For more information, see Cloud Controller Blobstore Configuration in the open source Cloud Foundry documentation and Blob Storage in *High Availability in TAS for VMs* topic.

The internal WebDAV blobstore can be scaled vertically, so scaling up the number of CPUs or adding faster disk storage can improve the performance of the internal WebDAV blobstore if it is under high load.

High numbers of concurrent app container starts on Diego can cause stress on the blobstore. This typically happens during upgrades in environments with a large number of apps and Diego cells. If vertically scaling the blobstore or improving its disk performance is not an option, limiting the max number of concurrent app container starts can mitigate the issue. For more information, see starting_container_count_maximum in *auctioneer job* in the BOSH documentation.

Configuring Diego Cell Disk Cleanup Scheduling

Disk cleanup is the process of removing unused layers from the Diego Cell disk. The disk cleanup process removes all unused Docker image layers and old Ops Manager stacks, regardless of their size or age.

To perform a detailed analysis of disk usage in your VMware Tanzu Application Service for VMs deployment, see Examining GrootFS Disk Usage.

This topic describes how to configure disk cleanup scheduling on Diego Cells in Ops Manager.

Overview

Ops Manager isolates app instances (AIs) from each other using containers that run inside Diego Cells. Containers enforce a set of isolation layers, including file system isolation. A Ops Manager container file system can either be a Ops Manager stack or the result of pulling a Docker image.

For performance reasons, the Diego Cells cache the Docker image layers and the Ops Manager stacks used by running AIs. When Ops Manager destroys an AI or reschedules an AI to a different Diego Cell, a chance exist that certain Docker image layers or an old Ops Manager stack becomes unused. If Ops Manager does not clean these unused layers, the cell ephemeral disk slowly fills.

Disk cleanup is the process of removing unused layers from the Diego Cell disk. The disk cleanup process removes all unused Docker image layers and old Ops Manager Stacks, regardless of their size or age.

To perform a detailed analysis of disk usage in your VMware Tanzu Application Service for VMs deployment, see Examining GrootFS Disk Usage.

Options for Disk Cleanup

Ops Manager provides the following options for scheduling the disk cleanup process on Diego Cells in the **App Containers** pane of the TAS for VMs tile:

- **Never clean up Diego Cell disk space**: VMware does not recommend selecting this option for production environments.
- **Routinely clean up Diego Cell disk space**: This option makes the Diego Cell schedule a disk cleanup whenever a container is created. Running the disk cleanup process this frequently can result in a negative impact on the Diego Cell performance.
- **Clean up disk space once usage fills disk**: This option makes the Diego Cell schedule the disk cleanup process only when disk usage encroaches on the space reserved for other jobs.

For more information about these options, see Configure Disk Cleanup Scheduling.

Recommendations

To select the best option for disk cleanup, you must consider the workload that the Diego Cells run.

For Ops Manager installations that primarily run buildpack-based apps, VMware recommends selecting the **Routinely clean up Diego Cell disk space** option. The **Routinely clean up Diego Cell disk space** option ensures that when a new stack becomes available on a cell, the old stack is dropped immediately from the cache.

For Ops Manager installations that primarily run Docker images, or both Docker images and buildpack-based apps, VMware recommends selecting the **Clean up disk space once usage fills disk** option.

Calculating a Reserve

By default, the space reserved for other jobs is set to a reasonable value for the majority of Ops Manager deployments. The reserve default value accounts for the most typical deployment case where the only other component that writes to `/var/vcap/data` is the Diego executor cache.

However, if you are deploying any BOSH add-ons that require space in the ephemeral disk, you may need to increase the reserve.

For more information on disk usage and garbage collection in GrootFS, see GrootFS Disk Usage and Examining GrootFS Disk Usage.

⚠ **Caution:** Setting the reserve to an excessive value can cause garbage collection to occur more frequently. An increase in frequency can result in regular slowdowns in performance of apps on particular Diego Cells. VMware recommends leaving the default value for the reserve unless you are advised to modify it.

Configure Disk Cleanup Scheduling

1. Go to the Ops Manager Ops Manager **Installation Dashboard**.
2. Click the VMware Tanzu Application Service for VMs tile, and navigate to **App Containers**.
3. Under **Docker images disk cleanup scheduling on Diego Cell VMs**, select an option.
4. If you select **Clean up disk space once usage fills disk**, you can modify the **Reserved disk space for other jobs** field. VMware recommends that you leave the default value of `15360` MB, or 15 GB, in **Reserved disk space for other jobs** unless you are instructed to change it. For more information, see Calculating a Reserve.
5. Click **Save**.

Next Steps

If you are configuring TAS for VMs for the first time, return to your specific IaaS configuration to continue the installation process.

If you are modifying an existing TAS for VMs installation, return to the **Ops Manager Installation Dashboard**, click **Review Pending Changes**, and click **Apply Changes**.

Examining GrootFS disk usage in TAS for VMs

You can analyze GrootFS disk space usage in TAS for VMs. This article gives you important background and the steps to analyze GrootFS disk space usage.

You run the commands in this topic as root on any BOSH-deployed VM that hosts the Garden job in a VMware Tanzu Application Service for VMs (TAS for VMs) deployment.

Note: This article provides different commands depending on whether you are using privileged or unprivileged containers in your deployment. By default, all deployments use unprivileged containers. For more information about these container types, see Container Security.

For more information about the GrootFS concepts, see GrootFS Disk Usage.

Reconcile Container Disk Usage and Host Disk Usage

To reconcile disk allocations for containers with the actual disk usage on the host VM, you need to understand how GrootFS uses its disk.

Command line tools such as `du` and `df` can provide misleading information because of the way container file systems work.

For example, the following situations can occur:

If...	Then...
the Diego cell rep appears to be out of disk capacity, but the actual disk usage on the Garden host is low	Diego does not schedule containers on the cell.
the Diego cell rep appears to have available disk capacity, but the combined space used by containers and system components prevents Diego from allocating the remaining disk space	Diego continues to place containers on the cell, but they fail to start.

About Container Disk Usage

On disk, the read-write layer for each container can be found at `/var/vcap/data/grootfs/store/unprivileged/images/CONTAINER-ID/diff`.

When GrootFS calls on the built-in XFS quota tooling to get disk usage for a container, it takes into account data written to that directory and not the data in the read-only volumes.

Running `grootfs stats` returns the following values:

- `total_bytes_used`: This is the disk usage of the container **including** the rootfs image volumes.
- `exclusive_bytes_used`: This is the disk usage of the container **not including** the rootfs image volumes.

Retrieve Disk Usage Stats for a Single Container

To obtain the disk usage stats of a single container, perform the following steps:

1. In your TAS for VMs deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.

2. On the VM, run the following command to look up the container ID:

   ```
   ls /var/vcap/data/garden/depot/
   ```

   The command above returns a container ID in the following format:

   ```
   55afbf65-5cbf-49c6-4461-f803
   ```

3. Based on the container type used in your deployment, run one of the following commands on the VM:

   - For unprivileged containers, run the following command:

     ```
     /var/vcap/packages/grootfs/bin/grootfs --config \
     /var/vcap/jobs/garden/config/grootfs_config.yml stats CONTAINER-ID
     ```

     Where `CONTAINER-ID` corresponds to the container ID you obtained in step 2.
     For example:

     ```
     $ /var/vcap/packages/grootfs/bin/grootfs --config \
     /var/vcap/jobs/garden/config/grootfs_config.yml stats 55afbf65-5cbf-49c6-4461-f803
     ```

   - For privileged containers, run the following command:

     ```
     /var/vcap/packages/grootfs/bin/grootfs --config \
     /var/vcap/jobs/garden/config/privileged_grootfs_config.yml stats CONTAINER-ID
     ```

     Where `CONTAINER-ID` corresponds to the container ID you obtained in step 2.

     For example:

     ```
     $ /var/vcap/packages/grootfs/bin/grootfs --config \
       /var/vcap/jobs/garden/config/privileged_grootfs_config.yml stats 55afbf65-5cbf-49c6-4461-f803
     ```

   The commands above return output in the following format:

   ```
   {"disk_usage":{"total_bytes_used":23448093,"exclusive_bytes_used":8192}}
   ```

Retrieve exclusive disk usage stats for all running containers

To check the disk usage of all running containers, perform the following steps:

1. In your TAS for VMs deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.

2. Based on the container type used in your deployment, run one of the following commands on the VM:

   - For unprivileged containers, run the following command:

```
ls /var/vcap/data/garden/depot/ \
| xargs -I{} /var/vcap/packages/grootfs/bin/grootfs --config \
/var/vcap/jobs/garden/config/grootfs_config.yml stats {} \
| cut -d: -f4 | cut -d} -f1 | awk '{sum += $1} END {print sum}'
```

- For privileged containers, run the following command:

```
ls /var/vcap/data/garden/depot/ \
| xargs -I{} /var/vcap/packages/grootfs/bin/grootfs --config \
/var/vcap/jobs/garden/config/privileged_grootfs_config.yml stats {} \
| cut -d: -f4 | cut -d} -f1 | awk '{sum += $1} END {print sum}'
```

The commands above return the total disk usage in bytes for all running containers.

About Volumes in GrootFS

Underlying layers are known as `volumes` in GrootFS.

They are read-only and their changesets are layered together through an **OverlayFS** mount to create the rootfs for containers. When GrootFS writes each file system volume to disk, it also stores the number of bytes written to a file in the `meta` directory.

Check volume disk size

To find out the size of an individual volume, you can read the corresponding metadata file or run `du` on the volume itself. Perform the following steps:

1. In your TAS for VMs deployment, use `bosh ssh` to connect to the BOSH-deployed VM running the Garden job.
2. Based on the container type used in your deployment, run one of the following commands on the VM:
   - For unprivileged containers, run the following command on the VM:

     ```
     cat /var/vcap/data/grootfs/store/unprivileged/meta/volume-VOLUME-SHA
     ```

     Where `VOLUME-SHA` corresponds to the SHA value of the volume.
   - For privileged containers, run the following command:

     ```
     cat /var/vcap/data/grootfs/store/privileged/meta/volume-VOLUME-SHA
     ```

     Where `VOLUME-SHA` corresponds to the SHA value of the volume.

   The `cat` commands above return the volume size in bytes in the following format:

   ```
   {"Size":5607885}
   ```
3. Alternatively, use `du` and pass the absolute path to the volume. Run one of the following commands on the VM:
   - For unprivileged containers, run the following command:

```
    du -sch /var/vcap/data/grootfs/store/unprivileged/volumes/VOLUME-SHA/
    ```

    Where `VOLUME-SHA` corresponds to the SHA value of the volume.

  - For privileged containers, run the following command:

    ```
    du -sch /var/vcap/data/grootfs/store/privileged/volumes/VOLUME-SHA/
    ```

    Where `VOLUME-SHA` corresponds to the SHA value of the volume.

  The `du` commands above return the volume size in the following format:

  ```
  5.4M    /var/vcap/data/grootfs/store/unprivileged/volumes/VOLUME-SHA/
  ```

Determine disk usage and reclaimable disk space

This section describes how to calculate the amount of disk space is in use and estimate how much space is reclaimable.

Calculate disk use by all active volumes

For each container, GrootFS mounts the underlying volumes using overlay to a point in the `images` directory. This point is the rootfs for the container and is read-write.

GrootFS also stores the SHA of each underlying volume used by an image in the `meta` folder.

You can determine the bytes of all active volumes on disk by running one of the following commands:

> If you do not have `python3` is not installed, replace `python3` with `python` in the commands below.

- For unprivileged containers, run the following command:

  ```
  for image in $(ls /var/vcap/data/grootfs/store/unprivileged/meta/dependencies/image\:*.json); \
  do cat $image | python3 -c 'import json,sys;obj=json.load(sys.stdin); \
  print("\n".join(obj))' ; done | sort -u \
  | xargs -I{} cat /var/vcap/data/grootfs/store/unprivileged/meta/volume-{} \
  | cut -d : -f 2 | cut -d} -f1 \
  | awk '{sum += $1} END {print sum}'
  ```

- For privileged containers, run the following command:

  ```
  for image in $(ls /var/vcap/data/grootfs/store/privileged/meta/dependencies/image\:*.json); \
  do cat $image | python3 -c 'import json,sys;obj=json.load(sys.stdin); \
  print("\n".join(obj))' ; done | sort -u \
  | xargs -I{} cat /var/vcap/data/grootfs/store/privileged/meta/volume-{} \
  | cut -d : -f 2 | cut -d} -f1 \
  | awk '{sum += $1} END {print sum}'
  ```

The commands above return the total number of bytes used by all active volumes on disk.

Calculate GrootFS store disk usage

To determine how much total disk space the store is using, run the following command:

```
df | grep -E  "/var/vcap/data/grootfs/store/(privileged|unprivileged)$" \
| awk '{sum += $3} END {print sum}'
```

The command above returns the total number of bytes used by the store.

Calculate reclaimable disk space

You can use values gathered from the commands above to calculate how much space can be cleared in GrootFS. Garbage collection reclaims disk space by pruning unused volumes.

The overall formula to calculate reclaimable disk space is to subtract the total disk in use by the store from the total disk used by active volumes.

For example, perform the following steps:

1. Calculate how much disk space the store is using by following the instructions in Calculate GrootFS Store Disk Usage. For example, your result might be:

   ```
   Total disk store = 5607885 bytes
   ```

2. Calculate how much disk space active volumes are using by following the instructions in Calculate Disk Use by All Active Volumes. For example, your result might be:

   ```
   Active volumes = 3212435 bytes
   ```

3. Subtract the amount of space used by active volumes from the space used by the store. For example, your result might be:

   ```
   5607885 - 3212435 = 2395450 bytes
   ```

In this example, you can reclaim 2395450 bytes through garbage collection.

How GrootFS reclaims disk space

The thresholder component calculates and sets a value so that GrootFS’s garbage collector can attempt to ensure that a small reserved space is kept free for other jobs. GrootFS only tries to garbage collect or reclaim space when that threshold is reached. However, if all the rootfs layers are actively in use by images, then garbage collection cannot occur and that space is used up.

If you determine that there is not enough reclaimable disk and more space is needed on disk, you should scale up your VMs to a larger size or add more VMs to provide more disk space.

Alternatively, you can configure a lower threshold for Diego Cell disk cleanup in TAS for VMs. For more information, see Configuring Diego Cell Disk Cleanup Scheduling.

Other categories of GrootFS disk usage

There may be categories of GrootFS disk usage other than those listed in the above sections. However, the bulk of disk usage is stored in the `images/CONTAINER-ID/diff` and `volumes` directories, so these are rarely taken into consideration when calculating store usage.

You can find these directories under `/var/vcap/data/grootfs/store/unprivileged` for unprivileged containers and `/var/vcap/data/grootfs/store/privileged` for privileged containers.

GrootFS also stores information in the following directories:

- `l`: link directories. Shorter directory names are symlinked to volume directories to allow Groot to union mount more file paths.
- `locks`: file system lock directory to ensure safety during concurrent cleans and creates.
- `meta`: per image and volume metadata.
- `projectids`: empty numbered directories used to track image quotas.
- `tmp`: normal temporary directory contents.

These directories typically use less than 2 MB disk in total.

Using metadata in TAS for VMs

This topic explains using metadata in TAS for VMs and gives you instructions for adding, updating, removing, and viewing metadata.

About metadata

VMware Tanzu Application Service for VMs (TAS for VMs) allows you to add metadata to resources such as spaces and apps. You can use metadata to provide additional information about the resources in your TAS for VMs deployment. This can help with operating, monitoring, and auditing.

For example, you can tag resources with metadata that describes the type of environment they belong to. You could also use metadata to describe app characteristics, such as front end or back end. Other examples include billing codes, points of contact, resource consumption, and information about security or risk.

Methods of adding metadata

You can add metadata to resources using any of the following methods:

- **Cloud Foundry Command Line Interface (cf CLI) v7:** For procedures using this method of adding metadata, see cf CLI Procedures. For more information about cf CLI v7, see Upgrading to cf CLI v7.
- **Cloud Foundry API (CAPI):** For procedures using this method of adding metadata, see API Procedures. For more information about adding metadata with CAPI, see Metadata in the CAPI documentation.

You can add metadata to apps and spaces using Apps Manager. For more information, see Managing Apps and Service Instances Using Apps Manager. For more information about adding metadata to spaces using Apps Manager, see Managing Orgs and Spaces Using Apps Manager.

Types of metadata

You can add two types of metadata to resources in TAS for VMs:

- **Labels:** Labels allow you to identify and select TAS for VMs resources. For example, if you have labeled all apps running in production, or all spaces that contain Internet-facing apps, you can then search for them.
- **Annotations:** Annotations allow you to add non-identifying metadata to TAS for VMs resources. You cannot query based on annotations. Also, there are fewer restrictions for key-value pairs of annotations than there are for labels. For example, you can include contact information of persons responsible for the resource, or tool information for debugging purposes.

Annotations sent to service brokers

For installations using CAPI v1.108.0 and later, TAS for VMs sends annotations with key prefixes to service brokers when service instances and service bindings are created.

When a service instance is created, TAS for VMs sends the following annotations to service brokers:

- `organization_annotations`
- `space_annotations`
- `instance_annotations`

When a service instance is bound to an app, TAS for VMs also sends `app_annotations` to service brokers.

For more information about the annotations listed above, see Cloud Foundry Context Object in the Open Service Broker API Profile on GitHub. For more general information about annotations, see Annotations in the CAPI documentation.

Metadata Requirements

The following tables describe requirements for creating metadata.

Requirements for labels

The following table describes the requirements for creating labels:

Label Requirements

Part of Label	Length in characters	Allowed characters	Other Requirements
(Optional) Key Prefix	0-253	• Alphanumeric (\[a-z0-9A-Z\]) • `-` • `.`	• DNS subdomain format, with at least one `.` • Must end with `/`

Key Name	1-63	• Alphanumeric (\[a-z0-9A-Z\]) • `-` • `_` • `.`	Must begin and end with an alphanumeric character
Value	0-63	• Alphanumeric • `-` • `_` • `.`	• Must begin and end with an alphanumeric character • Empty values allowed

Requirements for annotations

The following table describes the requirements for creating annotations:

Annotation Requirements

Part of Annotation	Length in characters	Allowed characters	Other Requirements
(Optional) Key Prefix	0-253	• Alphanumeric (\[a-z0-9A-Z\]) • `-` • `.`	• DNS subdomain format, with at least one `.` • Must end with `/`
Key Name	1-63	• Alphanumeric (\[a-z0-9A-Z\]) • `-` • `_` • `.`	Must begin and end with an alphanumeric character
Value	0-5000	Any unicode character	N/A

Metadata key prefixes

You can ensure a label or annotation key is easily differentiated from other keys by using a prefix. A prefix is a namespacing pattern that helps you more clearly identify resources. Prefixes are in DNS subdomain format. For example, `prefix.example.com`.

Consider an example in which you have two scanner tools: one for security and one for compliance. Both tools use a `scanned` label or annotation. You can disambiguate between the two tools using a prefix. The security tool can prefix a label or annotation with `security.example.com/scanned` and the compliance tool can prefix a label or annotation with `compliance.example.com/scanned`.

cf CLI procedures

The following sections describe how to add, update, view, and list metadata using the cf CLI.

Note: To see which resources are supported for this feature, run `cf labels -h`. cf CLI v7 supports adding labels to apps, orgs, spaces, buildpacks, stacks, routes, domains, and various service resources.

Add metadata to a resource

This section describes how to add metadata using the cf CLI.

Add a label

To add a label to a resource:

1. Run:

   ```
   cf set-label RESOURCE RESOURCE-NAME KEY=VALUE
   ```

   Where:

   - `RESOURCE` is the type of resource you want to label, such as `app` or `space`.
   - `RESOURCE-NAME` is the name of the resource you want to label, such as `example-app`.
   - `KEY` is the key for the label.
   - `VALUE` is the corresponding value for the label key. You can enter multiple key-value pairs in the same command.

Update metadata for a resource

To update metadata for a resource, follow the procedure for adding metadata and provide a new value for an existing key. For more information, see Add Metadata to a Resource above.

Remove metadata from a resource

This section describes how to remove metadata using the cf CLI.

Remove a label

To remove a label from a resource:

1. Run:

   ```
   cf unset-label RESOURCE RESOURCE-NAME KEY
   ```

   Where:

   - `RESOURCE` is the type of resource you want to remove the label from, such as `app` or `space`.
   - `RESOURCE-NAME` is the name of the resource you want to remove the label from , such as `example-app`.
   - `KEY` is the key for the label.

View Metadata for a resource

This section describes how to view metadata with the cf CLI.

View labels

To view labels for a resource:

1. Run:

   ```
   cf labels RESOURCE RESOURCE-NAME
   ```

   Where:

   - `RESOURCE` is the type of resource you want to remove the label from, such as `app` or `space`.
   - `RESOURCE-NAME` is the name of the resource you want to remove the label from , such as `example-app`.

Select resources by labels

To select resources by labels:

1. Run:

   ```
   cf apps --labels 'environment in (production,staging),tier in (backend)'
   ```

API procedures

The following sections describe how to add, update, remove, list, and query metadata using CAPI.

Add metadata to a resource

The sections below describe how to add labels and annotations to resources.

Add a label

To add a label to a resource using CAPI:

1. Run:

   ```
   cf curl v3/RESOURCE-ENDPOINT/GUID \
     -X PATCH \
     -d '{
       "metadata": {
         "labels": {
           "LABEL-KEY": "LABEL-VALUE"
         }
       }
     }'
   ```

   Where:

   - `RESOURCE-ENDPOINT` is the CAPI endpoint for the type of resource you want to label, such as `apps` or `organizations`.
   - `GUID` is the GUID of the resource you want to label.
   - `LABEL-KEY` is the key for the label.
   - `LABEL-VALUE` is the corresponding value for the label key.

Add an annotation

To add an annotation:

1. Run:

   ```
   cf curl v3/RESOURCE-ENDPOINT/GUID \
     -X PATCH \
     -d '{
       "metadata": {
         "annotations": {
           "ANNOTATION-KEY": "ANNOTATION-VALUE"
         }
       }
     }'
   ```

   Where:

   - `RESOURCE-ENDPOINT` is the CAPI endpoint for the type of resource you want to label, such as `apps` or `organizations`.
   - `GUID` is the GUID of the resource you want to label.
   - `ANNOTATION-KEY` is the key for the label.
   - `ANNOTATION-VALUE` is the corresponding value for the annotation key.

Update metadata for a resource

To update metadata for a resource, follow the procedure for adding metadata and provide a new value for an existing key. For more information, see Add Metadata to a Resource above.

Remove metadata from a resource

To remove metadata from a resource, follow the procedure for adding metadata and provide a `null` value for an existing key. For more information, see Add Metadata to a Resource above.

View metadata for a resource

To view metadata using the list endpoint of a resource:

1. Run:

   ```
   cf curl /v3/RESOURCE-ENDPOINT/GUID
   ```

   Where:

- `RESOURCE-ENDPOINT` is the CAPI endpoint for the type of resource you want to view, such as `apps` or `organizations`.
- `GUID` is the GUID of the resource you want to view.

List resources by querying labels

To list resources by querying label metadata:

1. To query a resource by using the `label_selector` parameter on its list endpoint, run:

   ```
   cf curl /v3/RESOURCE-ENDPOINT/?label_selector=SELECTOR-REQUIREMENTS
   ```

   Where:

   - `RESOURCE-ENDPOINT` is the CAPI endpoint for the type of resource you want to view, such as `apps` or `organizations`.
   - `SELECTOR-REQUIREMENTS` is one of requirement types specified in Selector Requirement Reference below. You can add multiple selector requirements using a comma-separated list.

     > **Note:** Ensure that this part of the URL is appropriately escaped.

Selector requirement reference

The following table describes how to form selector requirements:

Requirement	Format	Description
existence	`KEY`	Returns all resources labeled with the given key
inexistence	`!KEY`	Returns all resources not labeled with the given key
equality	`KEY==VALUE` or `KEY=VALUE`	Returns all resources labeled with the given key and value
inequality	`KEY!=VALUE`	Returns all resources not labeled with the given key and value
set inclusion	`KEY in (VALUE1,VALUE2...)`	Returns all resources labeled with the given key and one of the specified values
set exclusion	`KEY notin (VALUE1,VALUE2...)`	Returns all resources not labeled with the given key and one of the specified values

Example: Label resources with a git commit

This section provides the following:

- A procedure for labeling an app, package, and droplet with a Git commit SHA. For more information, see Manually Label Resources below.
- A script that automates the procedure. For more information, see Automate Labeling Resources below.

Labeling your app and related resources with a Git commit SHA allows you to track which version of your code is running on TAS for VMs.

For more information about app packages and droplets, see the CAPI documentation.

Manually label resources

To label an app, droplet, and package with a Git commit SHA:

1. Run:

   ```
   cf app APP-NAME --guid
   ```

   Where `APP-NAME` is the name of the app.

2. Record the app GUID you retrieved in the previous step,

3. Return the GUID of the droplet and package associated with the app by running:

   ```
   cf curl /v3/apps/APP-GUID/droplets/current
   ```

   Where `APP-GUID` is the GUID of the app.

4. Record the GUID of the droplet and package:

   - The droplet GUID is the value for the `"guid"` key.
   - The package GUID is the end of the `"href"` URL for the `"package"` key.

   For example, the droplet and package GUIDs are highlighted in blue in the following output:

   ```
   {
     "guid": "fd35633f-5c5c-4e4e-a5a9-0722c970a9d2",
     ...
     "links": {
       "package": {
         "href": "https://api.run.pivotal.io/v3/packages/fd35633f-5c5c-4e4
   e-a5a9-0722c970a9d2"
       }
     }
   ```

5. Label the app with a Git commit SHA by running:

   ```
   cf curl /v3/apps/APP-GUID -X PATCH -d '{"metadata": { "labels": { "commit": COM
   MIT-SHA } } }'
   ```

   Where:

   - `APP-GUID` is the GUID of the app.
   - `COMMIT-SHA` is the SHA of the Git commit.

6. Label the app droplet with the same Git commit SHA by running:

   ```
   cf curl /v3/droplets/DROPLET-GUID -X PATCH -d '{"metadata": { "labels": { "comm
   it": COMMIT-SHA } } }'
   ```

   Where:

   - `DROPLET-GUID` is the GUID of the droplet.
   - `COMMIT-SHA` is the SHA of the Git commit.

7. Label the app package with the same Git commit SHA by running:

   ```
   cf curl /v3/packages/PACKAGE-GUID -X PATCH -d '{"metadata": { "labels": { "commit": COMMIT-SHA } } }'
   ```

   Where:

   - `PACKAGE-GUID` is the GUID of the package.
   - `COMMIT-SHA` is the SHA of the Git commit.

Automate labeling resources

You can automate labeling resources by running a script either programmatically or manually in the app repository.

Prerequisite

To run the following example script, you must install `jq`. To download `jq`, see jq.

Example script

The following script retrieves the GUID of the app, droplet, and package. It then `curls` CAPI to label each resource with the current Git commit SHA.

Replace `APP-NAME` with the name of your app.

```
#!/usr/bin/env bash

set -ex

APP_GUID="$(cf app APP-NAME --guid)"
APP_URI="/v3/apps/${APP_GUID}"

DROPLET_GUID="$(cf curl "/v3/apps/${APP_GUID}/droplets/current" | jq -r .guid)"
DROPLET_URI="/v3/droplets/${DROPLET_GUID}"

PACKAGE_GUID="$(cf curl "/v3/droplets/${DROPLET_GUID}" | jq -r .links.package.href | xargs basename)"
PACKAGE_URI="/v3/packages/${PACKAGE_GUID}"

COMMIT_SHA="$(git rev-parse --short HEAD)"
REQUEST_BODY="$(jq -nc --arg commit "${COMMIT_SHA}" '{"metadata": { "labels": { "commit": $commit } } }')"

cf curl "${APP_URI}" -X PATCH -d "${REQUEST_BODY}"
cf curl "${PACKAGE_URI}" -X PATCH -d "${REQUEST_BODY}"
cf curl "${DROPLET_URI}" -X PATCH -d "${REQUEST_BODY}"
```

Example: Add custom tags to log and metric envelopes

Log and metric envelopes emitted by applications are tagged with information about the application such as the application name.

It is possible to define additional custom log and metric tags by adding a label with a specific prefix. This prefix defaults to `metric.tag.cloudfoundry.org`. Following a restart of the application the custom metric tag will then be visible in the logs and metrics emitted for processes associated with that application.

The following commands add a tag named `custom_tag` with the value `some_value` for logs and metrics emitted for the application `sample-app`:

```
$ cf set-label app sample-app metric.tag.cloudfoundry.org/custom_tag=some_value
$ cf restart sample-app
```

You can observe that the custom tag has been applied by querying Log Cache with the log-cache cf CLI plugin. The commands below assume that you have the `jq` command line utility:

```
$ cf install-plugin -r CF-Community 'log-cache'

$ cf tail sample-app --json --follow | jq -r '.tags.custom_tag'
some_value
some_value
some_value
```

Custom-Branding Apps Manager

This topic tells you how Ops Manager operators can visually brand Apps Manager by changing certain text, colors, and images of the interface. Developers view the customized interface when logging in, creating an account, resetting a password, or using Apps Manager.

Operators customize Apps Manager by configuring the **Custom Branding** and **Apps Manager Config** pages of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.

Custom Branding Pane

To customize your Apps Manager interface:

1. In a browser, go to the fully-qualified domain name (FQDN) of your Ops Manager deployment and log in.
2. Click the TAS for VMs tile.
3. Select **Custom Branding**.
4. For **Company name**, enter the name of your organization. If left blank, the name defaults to **VMware**.
5. For **Accent color**, enter the hexadecimal code for the color used to accent various visual elements, such as the currently selected space in the sidebar. For example, `#71ffda`.
6. For **Main logo**, enter a Base64-encoded URL string for a PNG image to use as your main logo. The image can be square or wide. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the VMware logo.

7. For **Square logo**, enter a Base64-encoded URL string for a PNG image to use in the Apps Manager header and in places that require a smaller logo. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the VMware logo.
8. For **Favicon**, enter a Base64-encoded URL string for a PNG image to use as your favicon. For example, `data:image/png;base64,iVBORw0...`. If left blank, the image defaults to the VMware logo.
9. For **Footer text**, enter a string to be displayed as the footer. If left blank, the footer text defaults to **VMware Software, Inc; All rights reserved.**
10. To add up to three footer links that appear to the right of the footer text:
    - Click **Add**.
    - For **Link text**, enter a label for the link.
    - For **URL**, enter an external or relative URL. For example, `http://docs.vmware.com` or `/tools.html`.
11. For special notification purposes such as governmental or restricted usage, use the **Classification** fields to create a special header and footer:
    - For **Classification header and footer background color**, enter the hexadecimal code for the desired background color of the header and footer.
    - For **Classification header and footer text color**, enter the hexadecimal code for the desired color of header and footer text.
    - For **Classification header content**, enter content for the header in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom header does not appear.
    - For **Classification footer content**, enter content for the footer in either plain text or HTML. If you enter HTML content, eliminate white spaces and new lines. If you do not provide any content, the custom footer does not appear. The Classification footer appears below the normal footer, which you can customize in the **Footer text** and **Footer links** fields.

Note: The header and footer do not appear on the UAA login page.

Apps Manager Pane

To continue customizing your Apps Manager interface:

1. In a browser, go to the FQDN of your Ops Manager and log in.
2. Click the TAS for VMs tile.
3. Select **Apps Manager**.
4. For **Product name**, enter text to replace **Apps Manager** in the header and the title of Apps Manager. This text defaults to **Apps Manager** if left blank.
5. For **Marketplace name**, enter text to replace the header in the Marketplace pages. This text defaults to **Marketplace** if left blank.

6. By default, Apps Manager includes three secondary navigation links: **Marketplace**, **Docs**, and **Tools**. You can edit existing secondary navigation links by clicking the name of the link and editing the **Link text** and **URL** fields. You can also remove the link by clicking the trash icon next to its name. If you want to add a new secondary navigation link, click **Add** and complete the **Link text** and **URL** fields.

   > **Note:** Removing any of the default links will remove them from the secondary navigation for all users.

7. For **Apps Manager Buildpack**, enter a static content-compatible buildpack you want TAS for VMs to use when deploying the Apps Manager app. By default, this is `staticfile_buildpack`.
8. For **Search Server Buildpack**, enter a Node.js-compatible buildpack you want TAS for VMs to use when deploying the Search Server app. By default, this is `nodejs_buildpack`.
9. For **Invitations Buildpack**, enter a Node.js-compatible buildpack you want TAS for VMs to use when deploying the Invitations app. By default, this is `nodejs_buildpack`.
10. (Optional) Enter in MB your desired **Apps Manager memory usage**. The minimum number you can enter is `128`. Leave this field blank to use the default value of 128 MB.
11. (Optional) Enter in MB your desired **Search Server memory usage**. This is the memory limit used to deploy the Search Server app. The minimum number you can enter is `256`. Leave this field blank to use the default value of 256 MB.
12. (Optional) Enter in MB your desired **Invitations memory usage**. This is the memory limit used to deploy the Invitations app. The minimum number you can enter is `256`. Leave this field blank to use the default value of 256 MB.
13. The **Apps Manager polling interval** field provides a temporary fix if Apps Manager usage degrades Cloud Controller response times. In this case, you can use this field to reduce the load on the Cloud Controller and ensure Apps Manager remains available while you troubleshoot the Cloud Controller. VMware recommends that you do not keep this field modified as a long-term fix because it can degrade Apps Manager performance. Optionally, you can:
    - Increase the polling interval above the default of 30 seconds.

      > **Note:** If you enter a value between `0` and `30`, the field is automatically set to `30`.

    - Disable polling by entering `0`. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.
14. The **App details polling interval** field provides an additional way to reduce the load on the Cloud Controller when the **Apps Manager polling interval** field is not sufficient. This field controls the rate at which Apps Manager polls for data when a user views the **Overview** page of an app. VMware recommends that you do not keep this field modified as a long-term fix because it can degrade Apps Manager performance. Optionally, you can:

- Increase the polling interval above the default of 10 seconds.

  > **Note:** If you enter a value between `0` and `30`, the field is automatically set to `10`.

- Disable polling by entering `0`. This stops Apps Manager from refreshing data automatically, but users can update displayed data by reloading Apps Manager manually.

Planning TAS for VMs Orgs and Spaces

This topic tells you about the considerations for effectively planning foundations, orgs, and spaces. You can plan your orgs and spaces to make the best use of the authorization features in VMware Tanzu Application Service for VMs (TAS for VMs).

An installation of TAS for VMs is referred to as a *foundation*. Each foundation has *orgs* and *spaces*. For more information, see Orgs, Spaces, Roles, and Permissions.

The TAS for VMs roles described in *Orgs, Spaces, Roles, and Permissions* use the principle of least privilege. Each role exists for a purpose and features in TAS for VMs enable these purposes.

Consider these roles when planning your foundations, orgs, and spaces. This allows for full use of the features and assumptions of TAS for VMs.

How TAS for VMs layers relate to your company

The following sections describe what TAS for VMs layers are and how they relate to your company structure.

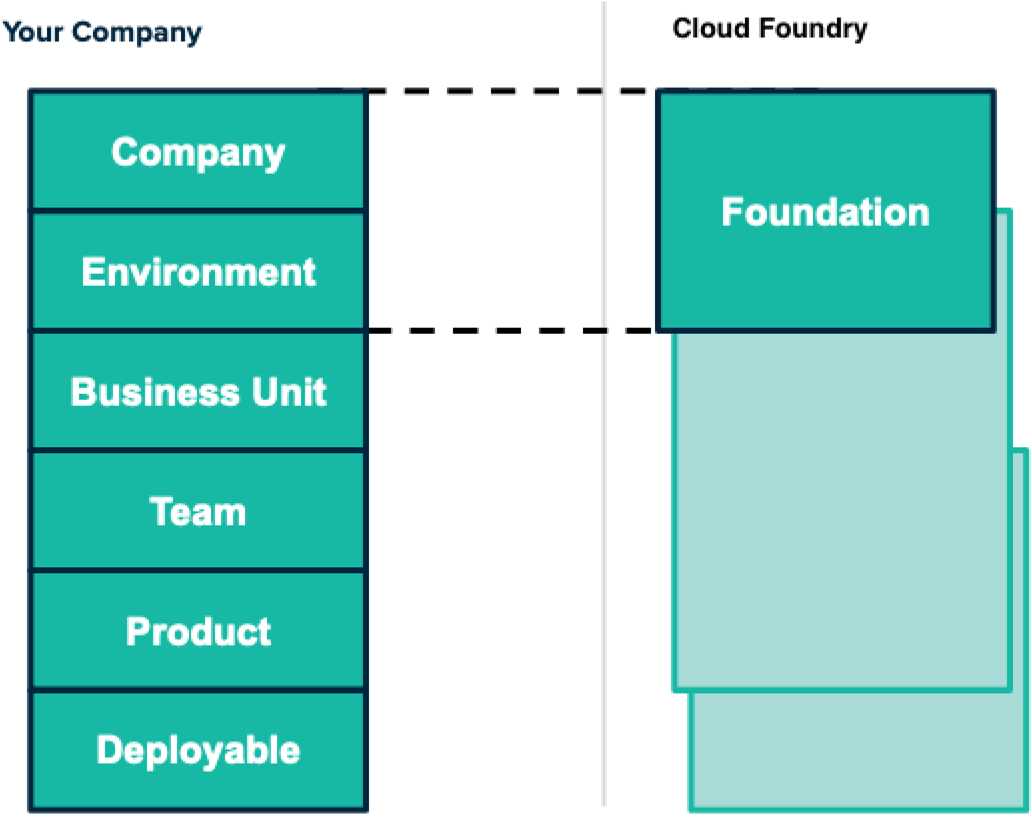
Overview of TAS for VMs Layers

For an overview of each of the structural TAS for VMs layers, see the following table:

TAS for VMs Layer	Challenge to Maintain	Contains	Description	Roles
Foundations	Hardest	Orgs	For shared components: domains, service tiles, and the physical infrastructure	Admin, Admin Read-Only, Global Auditor
Orgs	Average	Spaces	A group of users who share a resource quota plan, apps, services availability, and custom domains	Org Manager, Org Auditor, Org Billing Manager
Spaces	Easiest	Apps	A shared location for app development, deployment, and maintenance	Space Manager, Space Developer, Space Auditor

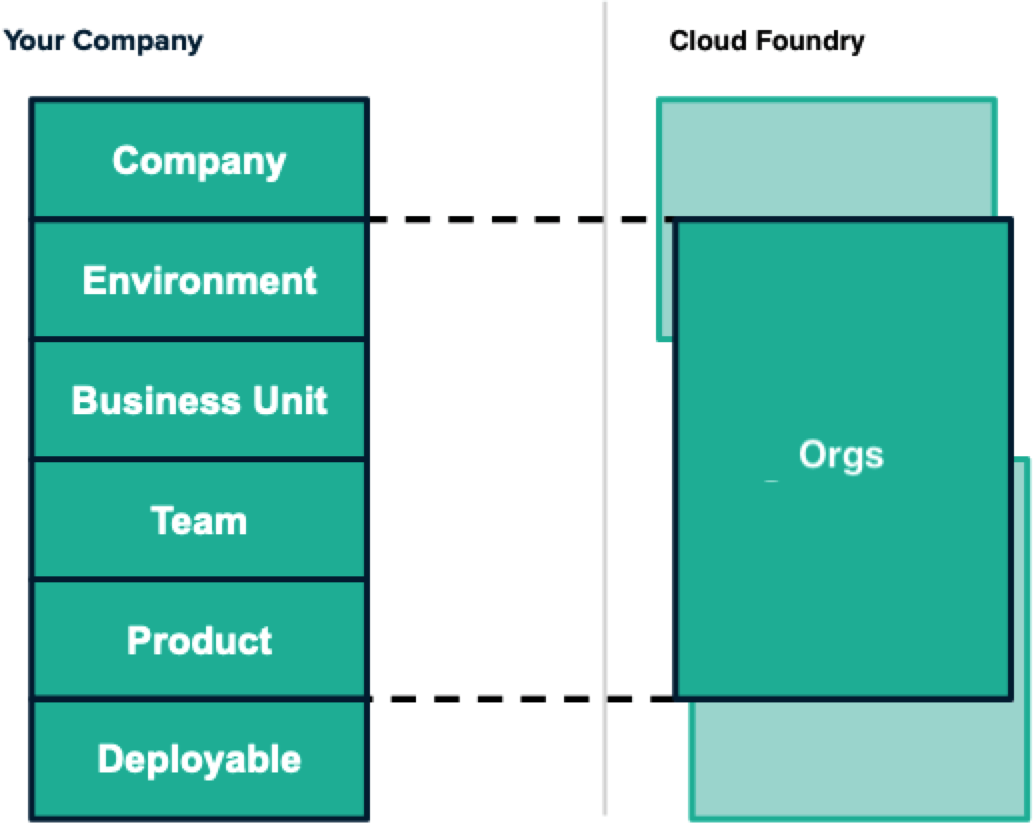
Foundations

Foundations roughly map to a company and environments. For an illustration, see the diagram below:



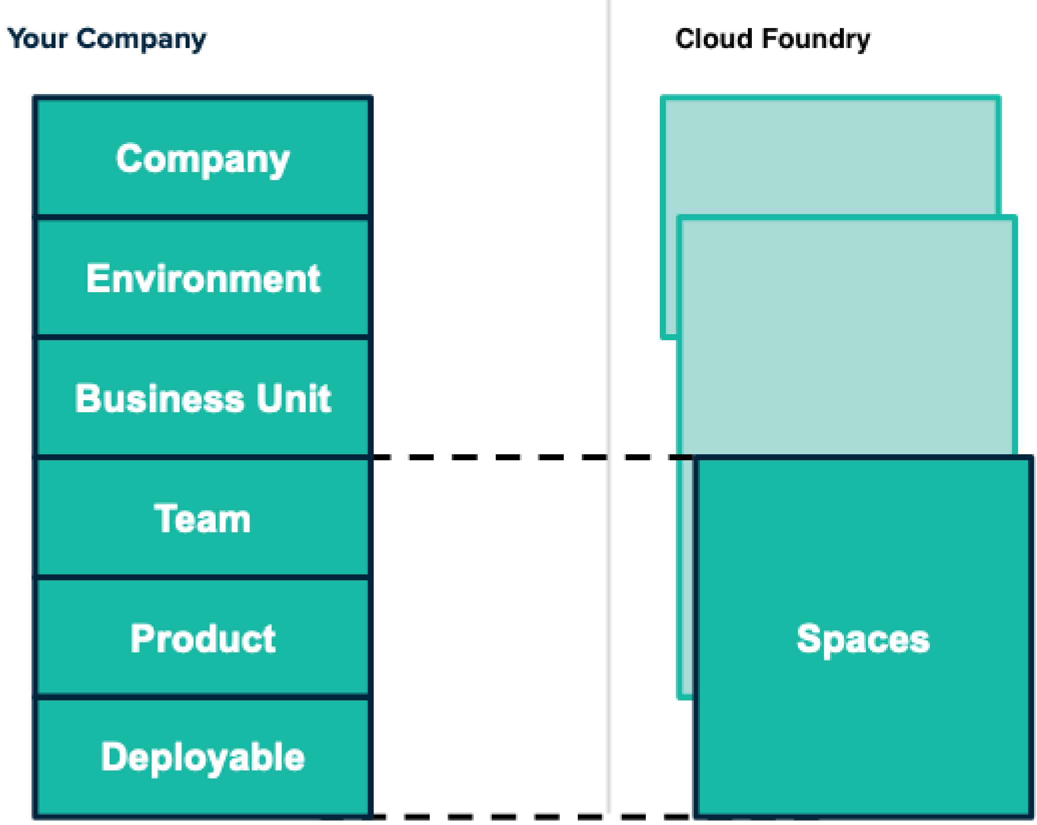
Orgs

Orgs most often map to a business unit in a particular foundation. To understand how you can map your company structure to a TAS for VMs org, see the diagram below:



Spaces

Spaces can encompass teams, products and specific deployables. To understand how you can map your company structure to a TAS for VMs space, see the diagram below:



Mapping Considerations

The sections below describe considerations you can make when mapping foundations, orgs, and spaces.

Planning for your environment

To plan your environments effectively, you must decide at what TAS for VMs layer they belong.

Broad environments, such as production environments, are commonly mapped to a foundation. More specific environments are mapped to an org or space.

Because of the large human cost to maintaining a foundation, you may see foundations mapped to production and staging environments separately.

For examples of environments and how they map to TAS for VMs layers, see the following table:

TAS for VMs Layer	Examples of Environments
Foundations	Production, Non-production, Sandbox
Orgs and Spaces	Development, UAT, QA

Questions to Consider About Each TAS for VMs Layer

For guiding questions to help you make decisions about planning your TAS for VMs structure, see the following table:

TAS for VMs Layer	Questions to Consider
Foundation	• How many foundations can your platform team create, update, and monitor? • How much isolation does your organization require? • Do you need foundations local to a particular cloud or IaaS environment?
Org	• How do you plan for capacity needs and changes? • What groups need to self-organize together? • How do you measure cost and perform billing and chargeback?
Space	• Are teams building single apps or constellations of microservices? • Are teams building a portfolio of apps or standalone apps? • When should a new space be created or destroyed? • What developer processes require the sandboxed isolation? • Do all apps need public routes? • What apps need to share the same service instance?

Mapping larger and smaller subsets

Subsets are the company divisions you decide to map to TAS for VMs. When creating your subsets, consider that the lower the TAS for VMs layer, the more specific you want to map your subsets. Conversely, the higher the TAS for VMs layer, the broader you want to make your subsets.

For more information about mapping larger subsets for each TAS for VMs layer, see the following table:

TAS for VMs Layer	The impact of mapping larger subsets of your company
Foundations	• Less maintenance • Less isolation • Better use of shared platform components
Orgs	• Less quota micromanagement • Ability to delegate user onboarding • More likely there are people with divergent needs • BU must be platform trained and manage potentially many spaces
Spaces	• More likelihood of accidental changes to someone else’s app or service • Easier integration between apps • More apps can use non-public routes

For more information about mapping smaller subsets for each TAS for VMs layer, see the following table:

TAS for VMs Layer	The impact of mapping smaller subsets of your company
Foundations	• More maintenance, which could be offset with platform automation • Higher likelihood of foundations being different • More isolation
Orgs	• More quota management from platform team • More freedom to create spaces as needed
Spaces	• More app isolation • Security with more specific ASGs • More reliant on external services or service instance sharing

Orgs, spaces, roles, and permissions

This topic tells you about orgs and spaces in VMware Tanzu Application Service for VMs (TAS for VMs) foundations. It also describes the default permissions for user roles in TAS for VMs.

TAS for VMs uses a role-based access control (RBAC) system to grant appropriate permissions to TAS for VMs users.

Admins, Org Managers, and Space Managers can assign user roles using the Cloud Foundry Command Line Interface (cf CLI). For more information, see Users and Roles in *Getting Started with the cf CLI* or Apps Manager.

Orgs

An org is a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts, which have roles such as Org Manager, Org Auditor, and Org Billing Manager. Collaborators in an org share a resource quota plan, apps, services availability, and custom domains.

By default, an org has the status of *active*. An admin can set the status of an org to *suspended* for various reasons such as failure to provide payment or misuse. When an org is suspended, users cannot perform certain activities within the org, such as push apps, modify spaces, or bind services.

For more information about the actions that each role can perform, see User Roles and User Role Permissions.

For details on what activities are allowed for suspended orgs, see Roles and Permissions for Suspended Orgs.

Spaces

A space provides users with access to a shared location for app development, deployment, and maintenance. An org can contain multiple spaces. Every app, service, and route is scoped to a space. Roles provide access control for these resources and each space role applies only to a particular space.

Org managers can set quotas on the following for a space:

- Usage of paid services
- Number of app instances
- Number of service keys
- Number of routes
- Number of reserved route ports
- Memory used across the space
- Memory used by a single app instance

User roles

A user account represents an individual person within the context of a TAS for VMs foundation. A user can have one or more roles. These roles define the user's permissions in orgs and spaces.

Roles can be assigned different scopes of User Account and Authentication (UAA) privileges. For more information about UAA scopes, see Scopes in *User Account and Authentication (UAA) Server*.

The following describes each type of user role in TAS for VMs:

- **Admin**: Perform operational actions on all orgs and spaces using the Cloud Controller API. Assigned the `cloud_controller.admin` scope in UAA.
- **Admin Read-Only**: Read-only access to all Cloud Controller API resources. Assigned the `cloud_controller.admin_read_only` scope in UAA.
- **Global Auditor**: Read-only access to all Cloud Controller API resources except for secrets, such as environment variables. The Global Auditor role cannot access those values. Assigned the `cloud_controller.global_auditor` scope in UAA.
- **Org Managers**: Administer the org.
- **Org Auditors**: Read-only access to user information and org quota usage information.
- **Org Users**: Read-only access to the list of other org users and their roles. In the v2 Cloud Controller API, when an Org Manager gives a person an Org or Space role, that person automatically receives Org User status in that org. This is no longer the case in the V3 Cloud Controller API.
- **Space Managers**: Administer a space within an org.
- **Space Developers**: Manage apps, services, and space-scoped service brokers in a space.
- **Space Auditors**: Read-only access to a space.
- **Space Supporters**: Troubleshoot and debug apps and service bindings in a space.

> **Note:** The Space Supporter role is only available for the Cloud Controller V3 API. If a user with this role tries to access a V2 endpoint, the API returns a 403.

For non-admin users, the `cloud_controller.read` scope is required to view resources, and the `cloud_controller.write` scope is required to create, update, and delete resources.

Before you assign a space role to a user, you must assign an org role to the user. The error message `Server error, error code: 1002, message: cannot set space role because user is not part of the org` occurs when you try to set a space role before setting an org role for the user.

User role permissions

Each user role includes different permissions in a TAS for VMs foundation. The following sections describe the permissions associated with each user role in both active and suspended orgs in TAS for VMs.

Roles and permissions for active orgs

The following table describes the default permissions for various TAS for VMs roles in active orgs.

Note: You can use feature flags to edit some of the default permissions in the following table. For more information, see Using Feature Flags.

Activity	Admin	Admin Read-Only	Global Auditor	Org Manager	Org Auditor	Space Manager	Space Developer	Space Auditor	Space Supporter
Scope of operation	Org	Org	Org	Org	Org	Space	Space	Space	Space
Add and edit users and roles	Yes			[1]		[1]			
View users and roles	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create and assign org quota plans	Yes								
View org quota plans	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create orgs	Yes			[2]	[2]	[2]	[2]	[2]	[2]
View all orgs	Yes	Yes	Yes						
View orgs where user is a member	Yes[3]	Yes[3]	Yes[3]	Yes	Yes	Yes	Yes	Yes	Yes
Edit, rename, and delete orgs	Yes			Yes[4]					
Suspend or activate an org	Yes								

Create and assign space quota plans	Yes			Yes					
Create spaces	Yes			Yes					
View spaces	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
Edit spaces	Yes			Yes		Yes			
Delete spaces	Yes			Yes					
Rename spaces	Yes			Yes		Yes			
View the status, number of instances, service bindings, and resource use of apps	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
Add private domains[5]	Yes			Yes					
Deploy, run, and manage apps	Yes						Yes		Yes[8]
Use app SSH[6]	Yes						Yes		
Instantiate services	Yes						Yes		
Bind services to apps	Yes						Yes		Yes
Manage global service brokers	Yes								
Manage space-scoped service brokers	Yes						Yes		Yes
Associate routes[5], instance counts, memory allocation, and disk limit of apps	Yes						Yes		Yes
Rename apps	Yes						Yes		
Create and manage Application Security Groups	Yes								
Create, update, and delete an Isolation Segment	Yes								
List all Isolation Segments for an org	Yes	Yes	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]

Entitle or revoke an Isolation Segment	Yes								
List all orgs entitled to an Isolation Segment	Yes	Yes	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]	Yes[7]
Assign a default Isolation Segment to an org	Yes			Yes					
List and manage Isolation Segments for spaces	Yes			Yes					
List entitled Isolation Segment for a space	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
List the Isolation Segment on which an app runs	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes

[1] Not by default, unless feature flag `set_roles_by_username` is set to `true`.

[2] Not by default, unless feature flag `user_org_creation` is set to `true`.

[3] Admin, admin read-only, and global auditor roles do not need to be added as members of orgs or spaces to view resources.

[4] Org Managers can rename their orgs and edit some fields; they cannot delete orgs.

[5] Unless disabled by feature flags.

[6] This assumes that SSH is enabled for the platform, space, and app. For more information, see SSH Access Control Hierarchy.

[7] Applies only to orgs to which the user account belongs.

[8] Cannot create packages or delete resources. For more information, see the Cloud Controller API V3 Documentation.

Roles and permissions for suspended orgs

The following table describes roles and permissions applied after an operator sets the status of an org to *suspended*.

Activity	Admin	Admin Read-Only	Global Auditor	Org Manager	Org Auditor	Space Manager	Space Developer	Space Auditor
Scope of operation	Org	Org	Org	Org	Org	Space	Space	Space
Add and edit users and roles	Yes							
View users and roles	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Create and assign org quota plans	Yes							
View org quota plans	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Create orgs	Yes							
View all orgs	Yes	Yes	Yes					
View orgs where user is a member	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Edit, rename, and delete orgs	Yes							
Suspend or activate an org	Yes							
Create and assign space quota plans	Yes							
Create spaces	Yes							
View spaces	Yes	Yes	Yes	Yes		Yes	Yes	Yes
Edit spaces	Yes							
Delete spaces	Yes							
Rename spaces	Yes							
View the status, number of instances, service bindings, and resource use of apps	Yes	Yes	Yes	Yes		Yes	Yes	Yes
Add private domains†	Yes							
Deploy, run, and manage apps	Yes							
Instantiate and bind services to apps	Yes							
Associate routes†, instance counts, memory allocation, and disk limit of apps	Yes							
Rename apps	Yes							
Create and manage Application Security Groups	Yes							

Enabling Developers

In this section:

- Using Docker in TAS for VMs
- Using Docker Registries
- Enabling Volume Services
- Managing Service Brokers
- Managing Access to Service Plans
- Dashboard Single Sign-On
- Using Feature Flags
- Managing Custom Buildpacks
- Supporting WebSockets

- Enabling Zipkin Tracing

Using Docker in TAS for VMs

You can activate Docker support so TAS for VMs can deploy and manage apps running in Docker containers. This page tells you how.

For information about Diego, the VMware Tanzu Application Service for VMs (TAS for VMs) component that manages app containers, see Diego Components and Architecture. For information about how TAS for VMs developers push apps with Docker images, see Deploying an App with Docker.

Enable Docker

By default, apps deployed with the `cf push` command run in standard TAS for VMs Linux containers. With Docker support enabled, TAS for VMs can also deploy and manage apps running in Docker containers.

To deploy apps to Docker, developers run `cf push` with the `--docker-image` option and the location of a Docker image to create the containers from. For information about how TAS for VMs developers push apps with Docker images, see Push a Docker Image.

To enable Docker support on a TAS for VMs deployment, an operator must:

- Enable the `diego_docker` feature flag.
- Configure access to any Docker registries that developers want to use images from.

Enable and deactivate the diego_docker feature flag

The `diego_docker` feature flag governs whether a TAS for VMs deployment supports Docker containers.

To enable Docker support, run:

```
cf enable-feature-flag diego_docker
```

To deactivate Docker support, run:

```
cf disable-feature-flag diego_docker
```

Note: Deactivating the `diego_docker` feature flag stops all Docker-based apps in your deployment within a few convergence cycles, on the order of a minute.

Configure Docker registry access

To support Docker, TAS for VMs needs the ability to access Docker registries using either a Certificate Authority or an IP address allow list. For information about configuring this access, see Installing Certificates on VMs in the BOSH documentation.

Docker image contents

A Docker image consists of a collection of layers. Each layer consists of one or both of the following:

- Raw bits to download and mount. These bits form the file system.
- Metadata that describes commands, users, and environment for the layer. This metadata includes the `ENTRYPOINT` and `CMD` directives, and is specified in the Dockerfile.

How Garden-runC creates containers

Diego currently uses Garden-runC to construct Linux containers.

Both Docker and Garden-runC use libraries from the Open Container Initiative (OCI) to build Linux containers. After creation, these containers use name space isolation, or *namespaces*, and control groups, or *cgroups*, to isolate processes in containers and limit resource usage. These are common kernel resource isolation features used by all Linux containers.

Before Garden-runC creates a Linux container, it creates a file system that is mounted as the root file system of the container. Garden-runC supports mounting Docker images as the root file systems for the containers it creates.

When creating a container, both Docker and Garden-runC:

- Fetch and cache the individual layers associated with a Docker image
- Combine and mount the layers as the root file system

These actions produce a container whose contents exactly match the contents of the associated Docker image.

Earlier versions of Diego used Garden-Linux. For more information, see Garden.

How Diego runs and monitors processes

After Garden-runC creates a container, Diego runs and monitors the processes inside of it.

To determine which processes to run, the Cloud Controller fetches and stores the metadata associated with the Docker image. The Cloud Controller uses this metadata to:

- Run the start command as the user specified in the Docker image.
- Instruct Diego and the Gorouter to route traffic to the lowest-numbered port exposed in the Docker image, or port 8080 if the Dockerfile does not explicitly expose a listen port.

For more information about Cloud Controller, see Cloud Controller. For more information about Gorouter and the routing tier, see TAS for VMs Routing Architecture. For more information about exposed ports in Docker images, see the Expose section of the *Dockerfile reference* topic in the Docker documentation.

> **Note:** When launching an app on Diego, the Cloud Controller honors any user-specified overrides such as a custom start command or custom environment variables.

Docker security concerns in a multi-tenant environment

The attack surface area for a Docker-based container running on Diego remains somewhat higher than that of a buildpack app because Docker allows users to fully specify the contents of their root file systems. A buildpack app runs on a trusted root filesystem.

Garden-runC provides features that allow the platform to run Docker images more securely in a multi-tenant context. In particular, TAS for VMs uses the `user-namespacing` feature found on modern Linux kernels to ensure that users cannot gain escalated privileges on the host even if they escalate privileges within a container.

The Cloud Controller always runs Docker containers on Diego with user namespaces enabled. This security restriction prevents certain features, such as the ability to mount FuseFS devices, from working in Docker containers. Docker apps can use fuse mounts through volume services, but they cannot directly mount fuse devices from within the container. For more information about volume services, see Using an External File System (Volume Services),

To mitigate security concerns, VMware recommends that you run only trusted Docker containers on the platform. By default, the Cloud Controller does not allow Docker-based apps to run on the platform.

Using Docker Registries

This topic describes how to configure Ops Manager to access Docker registries such as Docker Hub, by using either a root certificate authority (CA) certificate or by adding its IP address to an allowlist. It also explains how to configure Ops Manager to access Docker registries through a proxy.

Docker registries store Docker container images. Ops Manager uses these images to create the Docker containers that it runs apps in.

Prerequisite: Enable Docker Support

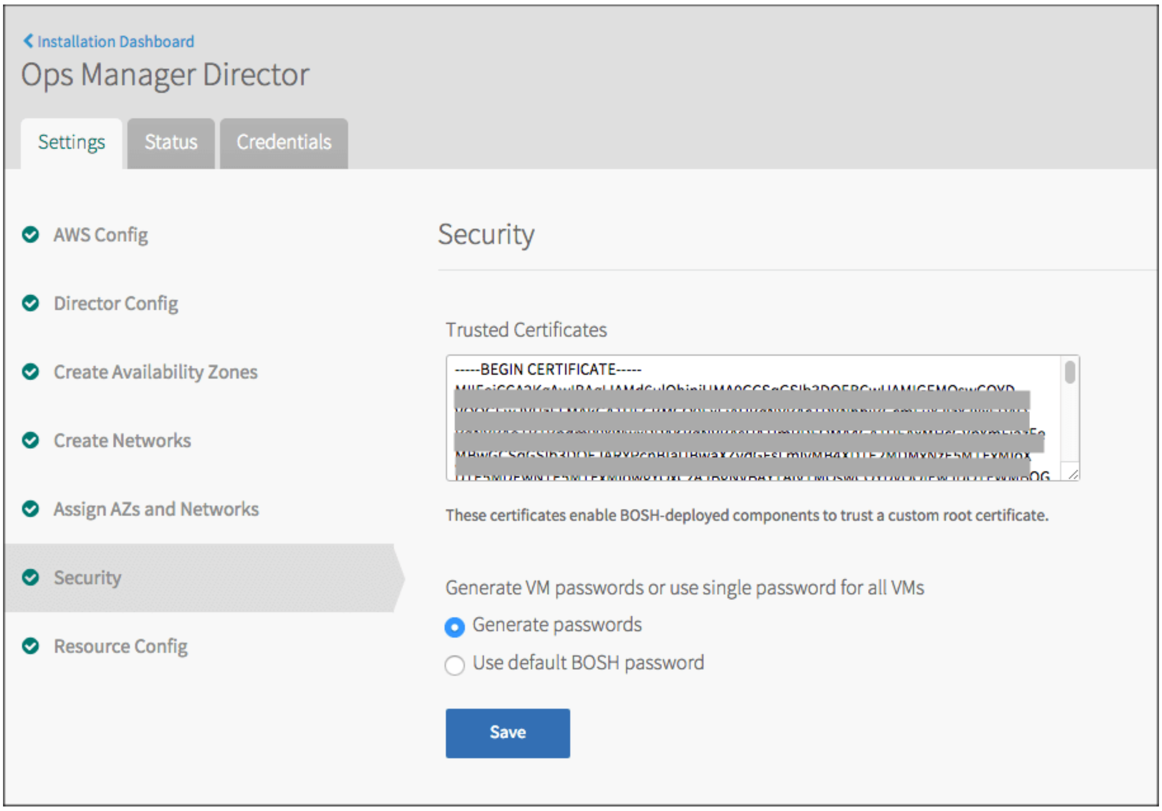
Ops Manager can only access Docker registries if an operator has enabled Docker support with the `cf enable-feature-flag diego_docker` command, as described in the Enable Docker section of the *Using Docker in TAS for VMs* topic.

With Docker enabled, developers can push an app with a Docker image using the Cloud Foundry Command Line Interface (cf CLI). For more information, see Deploying an App with Docker.

Use a CA Certificate

If you provide your root CA certificate in the Ops Manager configuration:

1. In the Ops Manager Installation Dashboard, click the **BOSH Director** tile.
2. Click **Security**.



Security

Trusted Certificates

-----BEGIN CERTIFICATE-----
[redacted]

These certificates enable BOSH-deployed components to trust a custom root certificate.

Generate VM passwords or use single password for all VMs

● Generate passwords

○ Use default BOSH password

3. In the **Trusted Certificates** field, paste one or more root CA certificates. The Docker registry does not use the CA certificate itself but uses a certificate that is signed by the CA certificate.
4. Click **Save**.
5. Select one of the following:
   - If you are configuring Ops Manager for the first time, return to your specific IaaS installation instructions (AWS, Azure, GCP, OpenStack, vSphere) to continue the installation process.
   - If you are editing an existing Ops Manager installation, return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

After configuration, BOSH propagates your CA certificate to all application containers in your deployment. You can then push and pull images from your Docker registries.

Use an IP Address Allow List

If you choose not to provide a CA certificate, you must provide the IP address of your Docker registry.

> **Note:** Using an allow list skips SSL validation. If you want to enforce SSL validation, enter the IP address of the Docker registry in the **No proxy** field described in Configure Ops Manager to Access Proxies for Docker Registries.

To configure an IP address allow list with the IP address of your Docker registry:

1. Go to the Ops Manager Installation Dashboard.

2. Click the VMware Tanzu Application Service for VMs (TAS for VMs) tile.
3. Select **App Containers**.
4. Select **Allow SSH access to app containers** to enable app containers to accept SSH connections. If you use a load balancer instead of HAProxy, you must open port 2222 on your load balancer to enable SSH traffic. To open an SSH connection to an app, a user must have Space Developer privileges for the space where the app is deployed. Operators can grant those privileges in Apps Manager or using the cf CLI.
5. For **Private Docker insecure registry allow list**, provide the hostname or IP address and port that point to your private Docker registry. For example, enter `198.51.100.1:80` or `mydockerregistry.com:80`. Enter multiple entries in a comma-delimited sequence. SSL validation is ignored for private Docker image registries secured with self-signed certificates at these locations.
6. Under **Docker images disk cleanup scheduling on Diego Cell VMs**, choose one of the options listed below. For more information about these options, see Configuring Docker Images Disk-Cleanup Scheduling.
    - **Never clean up Diego Cell disk space**
    - **Routinely clean up Diego Cell disk space**
    - **Clean up disk space once threshold is reached**. If you choose this option, enter the amount of disk space limit the Diego Cell must reach before disk cleanup initiates under **Reserved disk space for other jobs**.
7. Click **Save**.
8. Do one of the following:
    - If you are configuring Ops Manager for the first time, return to your specific IaaS installation instructions (AWS, Azure, GCP, OpenStack, vSphere) to continue the installation process.
    - If you are editing an existing TAS for VMs installation, return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes**.

After configuration, TAS for VMs allows Docker images to pass through the specified IP address without checking certificates.

Configure Ops Manager to Access Proxies for Docker Registries

If you have proxies already set up for Docker registries, you should configure Ops Manager to access your Docker registries through a proxy.

To configure Ops Manager to access a Docker registry through a proxy:

1. On the Installation Dashboard, go to **Username**; then click **Settings**, followed by **Proxy Settings**.
2. On the **Update Proxy Settings** pane, complete one of the following fields:

   - **HTTP proxy**: If you have an HTTP proxy server for your Docker registry, enter its IP address.
   - **HTTPS proxy**: If you have an HTTPS proxy server for your Docker registry, enter its IP address.
   - **No proxy**: If you do not use a proxy server, enter the IP address for the Docker registry. This field may already contain proxy settings for the BOSH Director.

   Enter multiple IP addresses as a comma-separated list.

3. Click **Update**.
4. Return to the Ops Manager dashboard, click **Review Pending Changes**, and click **Apply Changes**.

Enabling Volume Services

This topic describes how VMware Tanzu Application Service for VMs (TAS for VMs) operators can deploy NFS or SMB volume services.

Overview

A volume service gives apps access to a persistent filesystem, such as NFS or SMB. By performing the procedures in this topic, operators can add a volume service to the Marketplace that provides an NFS filesystem or an SMB share. For more information about adding a volume service to the Marketplace, see Services.

Developers can then use the Cloud Foundry Command Line Interface (cf CLI) to create service instances of the volume service and bind them to their apps. For more information, see Using an External File System (Volume Services).

> **Note:** You must have a running NFS or SMB server to test NFS or SMB volume services. TAS for VMs packages the necessary software to provide app connectivity to remote network-attached storage (NAS), but does not supply the NAS itself.

Enable NFS Volume Services

To enable NFS volume services in TAS for VMs:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **App Containers**.
4. Under **NFSv3 volume services**, select **Enable**.

   > **Note:** In a clean install, NFS volume services are enabled by default. In an upgrade, NFS volume services match the setting of the previous deployment.

5. (Optional) To configure LDAP for NFS volume services:

   > **Note:** If you already use an LDAP server with your network-attached storage (NAS) file server, enter its information below. This ensures that the identities known to the file server match those checked by the NFS driver.

   - For **LDAP service account user**, enter either the full domain name for the service account or the username of the service account that manages volume services. This value must be what you enter when binding the account to your LDAP server.
   - For **LDAP service account password**, enter the password for the service account.
   - For **LDAP server host**, enter the hostname or IP address of the LDAP server.
   - For **LDAP server port**, enter the LDAP server port number. If you do not specify a port number, Ops Manager uses 389.
   - For **LDAP user search base**, enter the location in the LDAP directory tree from which any LDAP user search begins. The typical LDAP search base matches your domain name. For example, a domain named `cloud.example.com` typically uses the following LDAP user search base: `ou=Users,dc=example,dc=com`.

     > **Note:** Additionally, your LDAP user records must be configured with an attribute `objectClass: User`, which is used by the NFS driver to identify records as user records. They must also have `uidNumber` and `gidNumber` fields, which are used to establish the correct UID for a named user.

   - (Optional) For **LDAP server CA certificate**, you can enter a Certificate Authority (CA) certificate if your LDAP server supports TLS. Enter a certificate if you want to enable TLS connections from the NFS driver to your LDAP server. Paste in the root certificate from your CA certificate or your self-signed certificate.

6. Click **Save**.
7. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes** to redeploy.
8. Using the cf CLI, enable access to the service by running:

   ```
   cf enable-service-access nfs
   ```

   To limit access to a specific org, use the `-o` flag, followed by the name of the org where you want to enable access. For more information, see Access Control.

9. (Optional) Enable access to the `nfs-legacy` service by running:

   ```
   cf enable-service-access nfs-legacy
   ```

   For details about the differences between the two `nfs` services, see the NFS Volume Service section of the *Using an External File System (Volume Services)* topic.

After completing these steps, developers can use the cf CLI to create service instances of the `nfs` service and bind them to their apps.

NFS Security

You can use ASGs and LDAP to secure your NFS server against traffic apps running on TAS for VMs:

- **App Security Groups (ASGs):** Prevent apps from sending traffic directly to your NFS ports. Apps must never need to use NFS ports directly. VMware recommends defining an ASG that blocks direct access to your NFS server IP, especially ports 111 and 2049. For more information on setting up ASGs, see App Security Groups.
- **LDAP:** In addition to ASGs, secures the NFS volume service so that app developers cannot bind to the service using an arbitrary UID. App developers also cannot gain access to sensitive data. With LDAP support enabled, app developers must provide credentials for any user they want to bind as.

The Diego Cells running on TAS for VMs must be able to reach your LDAP server on the port you use for connections, which are typically 389 or 636. You cannot limit which Diego Cells have access to your NFS or LDAP servers.

Enable SMB Volume Services

Note: In a clean install, SMB volume services are enabled by default. In an upgrade, SMB volume services match the setting of the previous deployment.

To enable SMB volume services in TAS for VMs:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **App Containers**.
4. Select the **Enable SMB volume services** checkbox.
5. Click **Save**.
6. Select **Errands**.
7. Set the **SMB Broker Errand** to **On**.
8. Click **Save**.
9. Return to the Ops Manager Installation Dashboard, click **Review Pending Changes**, and click **Apply Changes** to redeploy.
10. Using the cf CLI, enable access to the service by running:

    ```
    cf enable-service-access smb
    ```

    To limit access to a specific org, use the `-o` flag, followed by the name of the org where you want to enable access. For more information, see Access Control.

After you complete these steps, developers can use the cf CLI to create service instances of the `smb` service and bind them to their apps.

Managing Service Brokers

This page assumes you are using cf CLI v6.16 or later.

In order to run many of the commands below, you must be authenticated with VMware Tanzu Application Service for VMs as an admin user or as a space developer.

Quick Start

Given a service broker that has implemented the Service Broker API, two steps are required to make its services available to end users in all orgs or a limited number of orgs by service plan.

1. Register a Broker
2. Make Plans Public

Register a Broker

Registering a broker causes Cloud Controller to fetch and validate the catalog from your broker, and save the catalog to the Cloud Controller database. The basic auth username and password which are provided when adding a broker are encrypted in Cloud Controller database, and used by the Cloud Controller to authenticate with the broker when making all API calls. Your service broker should validate the username and password sent in every request; otherwise, anyone could curl your broker to delete service instances. When the broker is registered with an URL having the scheme `https`, Cloud Controller will make all calls to the broker over HTTPS.

As of cf-release 229, CC API 2.47.0, Ops Manager supports two types of brokers: *standard brokers* and *space-scoped brokers*. A list of their differences follows:

Standard Brokers

- Publish service plans to specific orgs or all orgs in the deployment. Can also keep plans unavailable, or *private*.
- Created by admins, with the command `cf create-service-broker`

  ```
  $ cf create-service-broker mybrokername someuser somethingsecure http
  s://mybroker.example.com
  ```

- Managed by admins
- Service plans are created private. Before anyone can use them, an admin must explicitly make them available within an org or across all orgs.

Space-Scoped Brokers

- Publish service plans only to users within the space they are created. Plans are unavailable outside of this space.
- Created by space developers using the command `cf create-service-broker` with the `--space-scoped` flag

```
$ cf create-service-broker mybrokername someuser somethingsecure http
s://mybroker.example.com --space-scoped
```

Note: If a space developer runs `cf create-service-broker` without the `--space-scoped` flag, they receive an error.

- Managed by space developers
- Newly-created plans automatically publish to all users in their space.

Make Plans Public

After an admin creates a new service plan from a standard broker, no one can use it until the admin explicitly makes it available to users within a specific org or all orgs in the deployment.

See the Access Control topic for how to make standard broker service plans available to users.

Multiple Brokers, Services, Plans

Many service brokers may be added to a Ops Manager instance, each offering many services and plans. The following constraints should be kept in mind:

- It is not possible to have multiple brokers with the same name
- Prior to Cloud Foundry API (CAPI) v1.71, the service ID and plan IDs of each service advertised by the broker must be unique across Ops Manager.
- With CAPI v1.71 or later, the service ID and plan IDs of each service advertised by the broker must be unique only within the broker and can overlap ids defined in other brokers
- GUIDs are recommended for the service ID and plan IDs of each service.

Note: If your deployment uses CAPI v1.71 or later, you can add multiple brokers with the same URL. In this case, the brokers must have different names. CAPI v1.70 and earlier do not support this feature.

See Possible Errors below for error messages and what do to when you see them.

List Service Brokers

```
$ cf service-brokers
Getting service brokers as admin...
OK

Name              URL
my-service-broker https://mybroker.example.com
```

Update a Broker

Updating a broker is how to ingest changes a broker author has made into Cloud Foundry. Similar to adding a broker, update causes Cloud Controller to fetch the catalog from a broker, validate it, and update the Cloud Controller database with any changes found in the catalog.

Update also provides a means to change the basic auth credentials cloud controller uses to authenticate with a broker, as well as the base URL of the broker's API endpoints.

```
$ cf update-service-broker mybrokername someuser somethingsecure https://mybr
oker.example.com
```

Rename a Broker

A service broker can be renamed with the `rename-service-broker` command. This name is used only by the Ops Manager operator to identify brokers, and has no relation to configuration of the broker itself.

```
$ cf rename-service-broker mybrokername mynewbrokername
```

Remove a Broker

Removing a service broker will remove all services and plans in the broker's catalog from the Ops Manager Marketplace.

```
$ cf delete-service-broker mybrokername
```

Note: Attempting to remove a service broker will fail if there are service instances for any service plan in its catalog. When planning to shut down or delete a broker, make sure to remove all service instances first. Failure to do so will leave orphaned service instances in the Ops Manager database. If a service broker has been shut down without first deleting service instances, you can remove the instances with the CLI; see Purge a Service.

Purge a Service

If a service broker has been shut down or removed without first deleting service instances from Ops Manager, you will be unable to remove the service broker or its services and plans from the Marketplace. In development environments, broker authors often destroy their broker deployments and need a way to clean up the Cloud Controller database.

The following command will delete a service offering, all of its plans, as well as all associated service instances and bindings from the Cloud Controller database, without making any API calls to a service broker. Once all services for a broker have been purged, the broker can be removed normally.

```
$ cf purge-service-offering service-test
Warning: This operation assumes that the service broker responsible for this
service offering is no longer available, and all service instances have been
deleted, leaving orphan records in Ops Manager's database. All knowledge of
```

```
the service will be removed from Ops Manager, including service instances and
service bindings. No attempt will be made to contact the service broker; runn
ing
this command without destroying the service broker will cause orphan service
instances. After running this command you may want to run either
delete-service-auth-token or delete-service-broker to complete the cleanup.

Really purge service offering service-test from Ops Manager? y
OK
```

Purge a Service Instance

The following command will delete a single service instance, its service bindings and its service keys from the Cloud Controller database, without making any API calls to a service broker. This can be helpful in instances a Service Broker is not conforming to the Service Broker API and not returning a 200 or 410 to requests to delete the service instance.

```
$ cf purge-service-instance mysql-dev
WARNING: This operation assumes that the service broker responsible for this
service instance is no longer available or is not responding with a 200 or 41
0,
and the service instance has been deleted, leaving orphan records in Cloud
Foundry's database. All knowledge of the service instance will be removed fro
m
Ops Manager, including service bindings and service keys.

Really purge service instance mysql-dev from Ops Manager?> y
Purging service mysql-dev...
OK
```

`purge-service-instance` requires cf-release v218 and cf CLI 6.14.0.

Note: When multiple brokers provide two or more service instances with the same name, you must specify the broker by including the `-b BROKER` flag in the `cf purge-service-instance` command.

Catalog Validation Behaviors

When Ops Manager fetches a catalog from a broker, it will compare the broker’s id for services and plans with the `unique_id` values for services and plans in the Cloud Controller database.

Event	Action
The catalog fails to load or validate.	Ops Manager will return a meaningful error that the broker could not be reached or the catalog was not valid.
A service or plan in the broker catalog has an ID that is not present among the `unique_id` values in the marketplace database.	A new record must be added to the marketplace database.

Event	Action
A service or plan in the marketplace database are found with a `unique_id` that matches the broker catalog's ID.	The marketplace must update the records to match the broker's catalog.
The database has plans that are not found in the broker catalog, and there are no associated service instances.	The marketplace must remove these plans from the database, and then delete services that do not have associated plans from the database.
The database has plans that are not found in the broker catalog, but there are provisioned instances.	The marketplace must mark the plan inactive and no longer allow it to be provisioned.

Possible Errors

If incorrect basic auth credentials are provided:

```
Server error, status code: 500, error code: 10001, message: Authentication
failed for the service broker API.
Double-check that the username and password are correct:
    https://github-broker.a1-app.example.com/v2/catalog
```

If you receive the following errors, check your broker logs. You may have an internal error.

```
Server error, status code: 500, error code: 10001, message:
    The service broker response was not understood

Server error, status code: 500, error code: 10001, message:
    The service broker API returned an error from
    https://github-broker.a1-app.example.com/v2/catalog: 404 Not Found

Server error, status code: 500, error code: 10001, message:
    The service broker API returned an error from
    https://github-broker.primo.example.com/v2/catalog: 500 Internal Server E
rror
```

If your broker’s catalog of services and plans violates validation of presence, uniqueness, and type, you will receive meaningful errors.

```
Server error, status code: 502, error code: 270012, message: Service broker c
atalog is invalid:
Service service-name-1
  service id must be unique
  service description is required
  service "bindable" field must be a boolean, but has value "true"
  Plan plan-name-1
    plan metadata must be a hash, but has value [{"bullets"=>["bullet1", "bul
let2"]}]
```

Managing Access to Service Plans

All new service plans from standard brokers are private by default. This means that when adding a new broker, or when adding a new plan to an existing broker's catalog, service plans won't immediately be available to end users. This lets an admin control which service plans are available to end users, and manage limited service availability.

Space-scoped brokers are registered to a specific space, and all users within that space can automatically access the broker's service plans. With space-scoped brokers, service visibility is not managed separately.

Prerequisites

- CLI v6.4.0
- Cloud Controller API v2.9.0 (cf-release v179)
- Admin user access; the following commands can be run only by an admin user

Display Access to Service Plans

The `service-access` CLI command enables an admin to see the current access control setting for every service plan in the marketplace, across all service brokers.

```
$ cf service-access
getting service access as admin...
broker: elasticsearch-broker
   service        plan     access    orgs
   elasticsearch  standard limited

broker: p-mysql
   service   plan        access   orgs
   p-mysql   100mb-dev   all
```

The `access` column shows values `all`, `limited`, or `none`, defined as follows:

- `all` - The service plan is available to all users, or *public*.
- `none` - No one can use the service plan; it is *private*.
- `limited` - The plan is available only to users within the orgs listed.

The `-b`, `-e`, and `-o` flags let you filter by broker, service, and org.

```
$ cf help service-access
NAME:
   service-access - List service access settings

USAGE:
   cf service-access [-b BROKER] [-e SERVICE] [-o ORG]

OPTIONS:
   -b     access for plans of a particular broker
   -e     access for plans of a particular service offering
   -o     plans accessible by a particular org
```

Enable Access to Service Plans

Admins use the `cf enable-service-access` command to give users access to service plans. The command grants access at the org level or across all orgs.

When an org has access to a plan, its users see the plan in the services marketplace (`cf marketplace`) and its Space Developer users can provision instances of the plan in their spaces.

Enable Access to a Subset of Users

The `-p` and `-o` flags to `cf enable-service-access` let the admin limit user access to specific service plans or orgs as follows:

- `-p PLAN` grants all users access to one service plan (access:`all`)
- `-o ORG` grants users in a specified org access to all plans (access: `limited`)
- `-p PLAN -o ORG` grants users in one org access to one plan (access: `limited`)

For example, the following command grants the org dev-user-org access to the p-mysql service.

```
$ cf enable-service-access p-mysql -o dev-user-org
Enabling access to all plans of service p-mysql for the org dev-user-org as a
dmin...
OK

$ cf service-access
getting service access as admin...
broker: p-mysql
   service   plan        access   orgs
   p-mysql                        dev-user-org
```

Run `cf help enable-service-access` to review these options from the command line.

Running `cf enable-service-access SERVICE-NAME` without any flags lets all users access every plan carried by the service. For example, the following command grants all-user access to all `p-mysql` service plans:

```
$ cf enable-service-access p-mysql
Enabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-mysql
   service   plan        access   orgs
   p-mysql   100mb-dev   all
```

Note: When multiple brokers provide two or more services with the same name, you must specify the broker by including the `-b BROKER` flag in the `cf enable-service-access` command.

Disable Access to Service Plans

Admins use the `cf disable-service-access` command to disable user access to service plans. The command denies access at the org level or across all orgs.

Disable Access to All Plans for All Users

Running `cf disable-service-access SERVICE-NAME` without any flags disables all user access to all plans carried by the service. For example, the following command denies any user access to all `p-mysql` service plans:

```
$ cf disable-service-access p-mysql
Disabling access to all plans of service p-mysql for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-mysql
   service    plan        access   orgs
   p-mysql    100mb-dev   none
```

Disable Access for Specific Orgs or Plans

The `-p` and `-o` flags to `cf disable-service-access` let the admin deny access to specific service plans or orgs as follows:

- `-p PLAN` disables user access to one service plan
- `-o ORG` disables access to all plans for users in a specified org
- `-p PLAN -o ORG` prevents users in one org from accessing one plan

Run `cf help disable-service-access` to review these options from the command line.

> **Note**: When multiple brokers provide two or more services with the same name, you must specify the broker by including the `-b BROKER` flag in the `cf disable-service-access` command.

Limitations

- You cannot disable access to a service plan for an org if the plan is currently available to all orgs. You must first disable access for all orgs; then you can enable access for a particular org.

Dashboard Single Sign-On

Introduction

Single Sign-On (SSO) enables VMware Tanzu Application Service for VMs (TAS for VMs) users to authenticate with third-party service dashboards using their TAS for VMs credentials. Service dashboards are web interfaces which enable users to interact with some or all of the features the service offers. SSO provides a streamlined experience for users, limiting repeated logins and multiple accounts across their managed services. The user's credentials are never directly transmitted to the service since the OAuth protocol handles authentication.

Service Broker Responsibilities

Registering the Dashboard Client

1. A service broker must include the `dashboard_client` field in the JSON response from its catalog endpoint for each service implementing this feature. A valid response would appear as follows:

   ```
   {
     "services": [
       {
         "id": "44b26033-1f54-4087-b7bc-da9652c2a539",
         ...
         "dashboard_client": {
           "id": "p-mysql-client",
           "secret": "p-mysql-secret",
           "redirect_uri": "http://p-mysql.example.com"
         }
       }
     ]
   }
   ```

   The `dashboard_client` field is a hash containing three fields:

   - `id` is the unique identifier for the OAuth client that will be created for your service dashboard on the token server (UAA), and will be used by your dashboard to authenticate with the token server (UAA). If the client id is already taken, TAS for VMs will return an error when registering or updating the broker.
   - `secret` is the shared secret your dashboard will use to authenticate with the token server (UAA).
   - `redirect_uri` is used by the token server as an additional security precaution. UAA will not provide a token if the callback URL declared by the service dashboard doesn't match the domain name in `redirect_uri`. The token server matches on the domain name, so any paths will also match; e.g. a service dashboard requesting a token and declaring a callback URL of `http://p-mysql.example.com/manage/auth` would be approved if `redirect_uri` for its client is `http://p-mysql.example.com/`.

2. When a service broker which advertises the `dashboard_client` property for any of its services is added or updated, Cloud Controller will create or update UAA clients as necessary. This client will be used by the service dashboard to authenticate users.

Dashboard URL

A service broker should return a URL for the `dashboard_url` field in response to a provision request. Cloud Controller clients should expose this URL to users. `dashboard_url` can be found in the response from Cloud Controller to create a service instance, enumerate service instances, space summary, and other endpoints.

Users can then navigate to the service dashboard at the URL provided by `dashboard_url`, initiating the OAuth login flow.

Service Dashboard Responsibilities

OAuth Flow

When a user navigates to the URL from `dashboard_url`, the service dashboard should initiate the OAuth login flow. A summary of the flow can be found in section 1.2 of the OAuth RFC. OAuth expects the presence of an Authorization Endpoint and a Token Endpoint. In TAS for VMs these endpoints are provided by the UAA. Clients can discover the location of UAA from Cloud Controller's info endpoint; in the response the location can be found in the `token_endpoint` field.

```
$ curl api.example.com/info
{"name":"vcap","build":"2222","support":"http://support.example.com","version
":2,"description":"Cloud Foundry sponsored by Example Company","authorization_endpoin
t":
"https://login.example.com","token_endpoint":"https://uaa.example.com",
"allow_debug":true}
```

> To enable service dashboards to support SSO for service instances created from different TAS for VMs instances, the /v2/info url is sent to service brokers in the `X-Api-Info-Location` header of every API call. A service dashboard should be able to discover this URL from the broker, and enabling the dashboard to contact the appropriate UAA for a particular service instance.

A service dashboard should implement the OAuth Authorization Code Grant type (UAA documentation, RFC documentation).

1. When a user visits the service dashboard at the value of `dashboard_url`, the dashboard should redirect the user's browser to the Authorization Endpoint and include its `client_id`, a `redirect_uri` (callback URL with domain matching the value of `dashboard_client.redirect_uri`), and list of requested scopes.

   Scopes are permissions included in the token a dashboard client will receive from UAA, and which Cloud Controller uses to enforce access. A client should request the minimum scopes it requires. The minimum scopes required for this workflow are `cloud_controller_service_permissions.read` and `openid`. For an explanation of the scopes available to dashboard clients, see On Scopes.

2. UAA authenticates the user by redirecting the user to the Login Server, where the user then approves or denies the scopes requested by the service dashboard. The user is presented with human readable descriptions for permissions representing each scope. After authentication, the user's browser is redirected back to the Authorization endpoint on UAA with an authentication cookie for the UAA.

3. Assuming the user grants access, UAA redirects the user's browser back to the value of `redirect_uri` the dashboard provided in its request to the Authorization Endpoint. The

   `Location` header in the response includes an authorization code.

   ```
   HTTP/1.1 302 Found
   Location: https://p-mysql.example.com/manage/auth?code=F45jH
   ```

4. The dashboard UI should then request an access token from the Token Endpoint by including the authorization code received in the previous step. When making the request the dashboard must authenticate with UAA by passing the client `id` and `secret` in a basic auth header. UAA will verify that the client id matches the client it issued the code to. The dashboard should also include the `redirect_uri` used to obtain the authorization code for verification.
5. UAA authenticates the dashboard client, validates the authorization code, and ensures that the redirect URI received matches the URI used to redirect the client when the authorization code was issues. If valid, UAA responds back with an access token and a refresh token.

Checking User Permissions

UAA is responsible for authenticating a user and providing the service with an access token with the requested permissions. However, after the user has been logged in, it is the responsibility of the service dashboard to verify that the user making the request to manage an instance currently has access to that service instance.

The service can accomplish this with a GET to the `/v2/service_instances/:guid/permissions` endpoint on the Cloud Controller. The request must include a token for an authenticated user and the service instance guid. The token is the same one obtained from the UAA in response to a request to the Token Endpoint, described above.

Example Request

```
curl -H 'Content-Type: application/json' \
       -H 'Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoid' \
       http://api.cloudfoundry.com/v2/service_instances/44b26033-1f54-4087-b7bc-da9652c2a539/permissions
```

Response

```
{
  "manage": true,
  "read": true
}
```

The response includes the following fields which indicate the various user permissions for the given service instance:

- `manage` – a `true` value indicates that the user has sufficient permissions to make changes to and update the service instance; `false` indicates insufficient permissions.

- read – a true value indicates that the user has permission to access read-only diagnostic and monitoring information for the given service instance (e.g. permission to view a read-only dashboard); false indicates insufficient permissions.

Since administrators may change the permissions of users at any time, the service should check this endpoint whenever a user uses the SSO flow to access the service's UI.

On Scopes

Scopes let you specify exactly what type of access you need. Scopes limit access for OAuth tokens. They do not grant any additional permission beyond that which the user already has.

Minimum Scopes

The following two scopes are necessary to implement the integration. Most dashboard shouldn't need more permissions than these scopes enabled.

Scope	Permissions
openid	Allows access to basic data about the user, such as email addresses
cloud_controller_service_permissions.read	Allows access to the CC endpoint that specifies whether the user can manage a given service instance

Additional Scopes

Dashboards with extended capabilities may need to request these additional scopes:

Scope	Permissions
cloud_controller.read	Allows read access to all resources the user is authorized to read
cloud_controller.write	Allows write access to all resources the user is authorized to update / create / delete

Reference Implementation

The MySQL Service Broker is an example of a broker that also implements a SSO dashboard. The login flow is implemented using the OmniAuth library and a custom UAA OmniAuth Strategy. See this OmniAuth wiki page for instructions on how to create your own strategy.

The UAA OmniAuth strategy is used to first get an authorization code, as documented in this section of the UAA documentation. The user is redirected back to the service (as specified by the callback_path option or the default auth/cloudfoundry/callback path) with the authorization code. Before the application / action is dispatched, the OmniAuth strategy uses the authorization code to get a token and uses the token to request information from UAA to fill the omniauth.auth environment variable. When OmniAuth returns control to the application, the omniauth.auth environment variable hash will be filled with the token and user information obtained from UAA as seen in the Auth Controller.

Restrictions

- UAA clients are scoped to services. There must be a `dashboard_client` entry for each service that uses SSO integration.
- Each `dashboard_client id` must be unique across the TAS for VMs deployment.

Resources

- OAuth
- Example broker with SSO implementation
- Cloud Controller API Docs
- User Account and Authentication (UAA) Service APIs

Using feature flags with TAS for VMs

With TAS for VMs, you can set feature flags by using the cf CLI to activate or deactivate the features available to users. This article tells you how.

View and edit feature flags

To perform the following procedures, you must be logged in to your deployment as an admin using the Cloud Foundry Command Line Interface (cf CLI).

To view and edit feature flags:

1. Use the `cf feature-flags` command to list the feature flags:

```
$ cf feature-flags

Features                                     State
user_org_creation                            disabled
private_domain_creation                      enabled
app_bits_upload                              enabled
app_scaling                                  enabled
route_creation                               enabled
service_instance_creation                    enabled
diego_docker                                 disabled
set_roles_by_username                        enabled
unset_roles_by_username                      enabled
task_creation                                enabled
env_var_visibility                           enabled
space_scoped_private_broker_creation         enabled
space_developer_env_var_visibility           enabled
service_instance_sharing                     disabled
hide_marketplace_from_unauthenticated_users  disabled
resource_matching                            enabled
```

   For descriptions of the features enabled by each feature flag, see [Feature Flags](#flags).

2. To view the status of a feature flag, run:

```
cf feature-flag FEATURE-FLAG-NAME
```

Where `FEATURE-FLAG-NAME` is the name of the feature flag you want to view.

3. To enable a feature flag, run:

   ```
   cf enable-feature-flag FEATURE-FLAG-NAME
   ```

   Where `FEATURE-FLAG-NAME` is the name of the feature flag you want to enable.

4. To disable a feature flag, run:

   ```
   cf disable-feature-flag FEATURE-FLAG-NAME
   ```

   Where `FEATURE-FLAG-NAME` is the name of the feature flag you want to disable.

Feature flags

Only admins can set feature flags. All flags are enabled by default except `user_org_creation` and `diego_docker`. When disabled, these features are only available to admins.

The following list provides descriptions of the features enabled or disabled by each flag, and the minimum Cloud Controller API (CC API) version necessary to use the feature. To determine your Cloud Controller API version, see Identifying the API Endpoint for Your TAS for VMs Instance.:

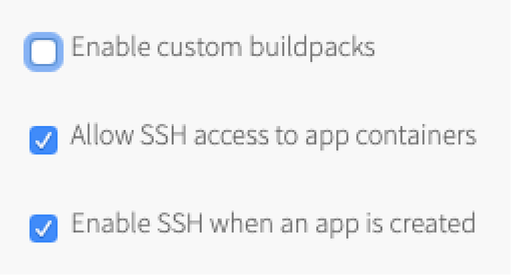
- `user_org_creation`: Any user can create an org. If enabled, this flag activates the **Create a New Org** link in the dropdown of the left navigation menu in Apps Manager. Minimum CC API version: 2.12.
- `private_domain_creation`: An Org Manager can create private domains for that org. Minimum CC API version: 2.12.
- `app_bits_upload`: Space Developers can upload app bits. Minimum CC API version: 2.12.
- `app_scaling`: Space Developers can perform scaling operations such as changing memory, disk, or instances. Minimum CC API version: 2.12.
- `route_creation`: Space Developers can create routes in a space. Minimum CC API version: 2.12.
- `route_sharing`: Space Developers can share routes between two spaces (including across orgs) in which they have the Space Developer role. When deactivated, Space Developers cannot share routes between two spaces. Minimum CC API version: 3.117.
- `service_instance_creation`: Space Developers can create service instances in a space. Minimum CC API version: 2.12.
- `diego_docker`: Space Developers can push Docker apps. Minimum CC API version 2.33.
- `set_roles_by_username`: Org Managers and Space Managers can add roles by username. Minimum CC API version: 2.37.
- `unset_roles_by_username`: Org Managers and Space Managers can remove roles by username. Minimum CC API version: 2.37.

- `task_creation`: Space Developers can create tasks on their app. Minimum CC API version: 2.47.
- `env_var_visibility`: All users can view environment variables. Minimum CC API version: 2.58.
- `space_scoped_private_broker_creation`: Space Developers can create space-scoped private service brokers. Minimum CC API version: 2.58.
- `space_developer_env_var_visibility`: Space Developers can view their v2 environment variables. Org Managers and Space Managers can view their v3 environment variables. Minimum CC API version: 2.58.
- `service_instance_sharing`: Space Developers can share service instances between two spaces (across orgs) in which they have the Space Developer role.
- `hide_marketplace_from_unauthenticated_users`: Do not allow unauthenticated users to see the service offerings available in the marketplace.
- `resource_matching`: Allow the package upload endpoint to cache uploaded packages for resource matching. When disabled, the resource match endpoint always returns an empty array of matches. Minimum CC API version: 3.77.

For more information about feature flag commands, see Feature Flags in the Cloud Foundry API documentation.

Deactivate custom buildpacks

You can disable custom buildpacks using the TAS for VMs tile. In the **App Containers** pane, deselect the **Enable custom buildpacks** check box, as shown in the image below.



□ Enable custom buildpacks

☑ Allow SSH access to app containers

☑ Enable SSH when an app is created

By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to TAS for VMs. To do so, they use the `-b` option to provide a custom buildpack URL with the `cf push` command. Deselecting the **Enable custom buildpacks** check box prevents the `-b` option from being used with external buildpack URLs.

For more information about custom buildpacks, see Buildpacks.

Managing custom buildpacks in TAS for VMs

You can manage additional buildpacks in TAS for VMs by using the Cloud Foundry Command Line Interface tool (cf CLI). This page tells you how.

If your app uses a language or framework that the VMware Tanzu Application Service for VMs (TAS for VMs) system buildpacks do not support, you can take one of the following actions:

- Write your own buildpack. For more information, see Creating Custom Buildpacks.
- Customize an existing buildpack.
- Use a Cloud Foundry Community Buildpack.
- Use a Heroku Third-Party Buildpack.

Add a buildpack

> **Note:** You must be a TAS for VMs admin user to run the commands discussed in this section.

To add a buildpack:

1. Run:

   ```
   cf create-buildpack BUILDPACK PATH POSITION
   ```

   Where:

   - `BUILDPACK` specifies the buildpack name.
   - `PATH` specifies the location of the buildpack. `PATH` can point to a ZIP file, the URL of a ZIP file, or a local directory.
   - `POSITION` specifies where to place the buildpack in the detection priority list.

   For more information, see create-buildpack in the *Cloud Foundry CLI Reference Guide*.

2. To confirm that you have successfully added a buildpack, run:

   ```
   cf buildpacks
   ```

Rename a buildpack

To rename a buildpack:

- If you use **cf CLI v7**, run:

  ```
  cf update-buildpack --rename BUILDPACK-NAME NEW-BUILDPACK-NAME
  ```

  Where:

  - `BUILDPACK-NAME` is the original buildpack name.
  - `NEW-BUILDPACK-NAME` is the new buildpack name.

- If you use **cf CLI v6**, run:

  ```
  cf rename-buildpack BUILDPACK-NAME NEW-BUILDPACK-NAME
  ```

  Where:

  - BUILDPACK-NAME is the original buildpack name.
  - NEW-BUILDPACK-NAME is the new buildpack name.

For more information about renaming buildpacks, see rename-buildpack in the *Cloud Foundry CLI Reference Guide*.

Update a buildpack

To update a buildpack, run:

```
cf update-buildpack BUILDPACK [-p PATH] [-i POSITION] [-s STACK][--enable|--disable]
[--lock|--unlock] [--rename NEW_NAME]
```

Where:

- BUILDPACK is the buildpack name.
- PATH is the location of the buildpack.
- POSITION is where the buildpack is in the detection priority list.
- STACK is the stack that the buildpack uses.

Note: If you are using cf CLI v6, use the cf rename-buildpack command instead of the --rename flag.

For more information about updating buildpacks, see update buildpack in the *Cloud Foundry CLI Reference Guide*.

Delete a buildpack

To delete a buildpack, run:

```
cf delete-buildpack BUILDPACK [-s STACK] [-f]
```

Where:

- BUILDPACK is the buildpack name.
- STACK is the stack that the buildpack uses.

For more information about deleting buildpacks, see delete-buildpack in the *Cloud Foundry CLI Reference Guide*.

Lock and unlock a buildpack

Every new version of Ops Manager includes an updated buildpack. By default, your deployment applies the most recent buildpack when you upgrade. In some cases, however, you may want to preserve an existing buildpack, rather than upgrade to the latest version. For example, if an app you deploy depends on a specific component in Buildpack A that is not available in Buildpack B, you may want to continue using Buildpack A.

The `-lock` flag lets you continue to use your existing buildpack even after you upgrade. Locked buildpacks are not updated when Ops Manager updates. You must manually unlock them to update them.

If you elect to use the `-unlock` flag, your deployment will apply the most recent buildpack when you upgrade Ops Manager.

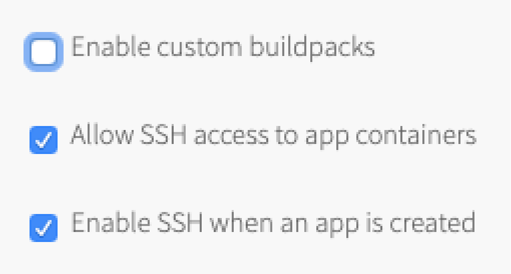
```
cf update-buildpack BUILDPACK [-p PATH] [-i POSITION] [-s STACK] [--enable|--disable]
[--lock|--unlock] [--rename NEW_NAME]
```

Note: If you are using cf CLI v6, the `--rename` flag is not supported. Use the `cf rename-buildpack` instead.

This feature is also available through the API. For more information, see Lock or unlock a Buildpack in the Cloud Foundry API documentation.

Deactivate custom buildpacks

You can disable custom buildpacks using the TAS for VMs tile. In the **App Containers** pane, deselect the **Enable custom buildpacks** check box, as shown in the image below.



□ Enable custom buildpacks

☑ Allow SSH access to app containers

☑ Enable SSH when an app is created

By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to TAS for VMs. To do so, they use the `-b` option to provide a custom buildpack URL with the `cf push` command. Deselecting the **Enable custom buildpacks** check box prevents the `-b` option from being used with external buildpack URLs.

For more information about custom buildpacks, see Buildpacks.

Supporting WebSockets in TAS for VMs

In this article, learn how TAS for VMs uses WebSockets, why use WebSockets in your apps, and how to configure your load balancer to support WebSockets.

Operators who use a load balancer to distribute incoming traffic across VMware Tanzu Application Service for VMs (TAS for VMs) router instances must configure their load balancer for WebSockets. Otherwise, the Loggregator system cannot stream app logs to developers, or app event data and component metrics to third-party aggregation services. Additionally, developers cannot use WebSockets in their apps.

WebSocket overview

Overview

The WebSocket protocol provides full-duplex communication over a single TCP connection. Apps can use WebSockets to perform real-time data exchange between a client and a server more efficiently than HTTP.

TAS for VMs uses WebSockets for the following metrics and logging purposes:

1. To stream all app event data and component metrics from the Doppler server instances to the Traffic Controller.
2. To stream app logs from the Traffic Controller to developers using the Cloud Foundry Command Line Interface (cf CLI) or Apps Manager.
3. To stream all app event data and component metrics from the Traffic Controller over the Firehose endpoint to external apps or services.

For more information about these Loggregator components, see Loggregator Architecture.

Configuring your load balancer for WebSockets

To form a WebSocket connection, the client sends an HTTP request that contains an `Upgrade` header and other headers required to complete the WebSocket handshake. You must configure your load balancer to not upgrade the HTTP request, but rather to pass the `Upgrade` header through to the TAS for VMs router. The procedures required to configure your load balancer depends on your IaaS and load balancer. The following list includes several possible approaches:

- Some load balancers can recognize the `Upgrade` header and pass these requests through to the TAS for VMs router without returning the WebSocket handshake response. This may or may not be default behavior, and may require additional configuration.
- Some load balancers do not support passing WebSocket handshake requests containing the `Upgrade` header to the TAS for VMs router. For instance, the Amazon Web Services (AWS) Elastic Load Balancer (ELB) does not support this behavior. In this scenario, you must configure your load balancer to forward TCP traffic to your TAS for VMs router to support WebSockets. If your load balancer does not support TCP pass-through of WebSocket requests on the same port as other HTTP requests, you can do one of the following:
    - Configure your load balancer to listen on a non-standard port (the built-in TAS for VMs load balancer listens on 8443 by default for this purpose), and forward requests for this port in TCP mode. App clients must make WebSockets upgrade requests to this port. If you choose this strategy, you must follow the steps in Set Your Loggregator Port.
    - If a non-standard port is not acceptable, add a load balancer that handles WebSocket traffic (or another IP on an existing load balancer) and configure it to listen on standard ports 80 and 443 in TCP mode. Configure DNS with a new hostname, such as `ws.cf.example.com`, to be used for WebSockets. This hostname should resolve to the new load balancer interface.

Note: Regardless of your IaaS and configuration, you must configure your load balancer to send the X-Forwarded-For and X-Forwarded-Proto headers for non-

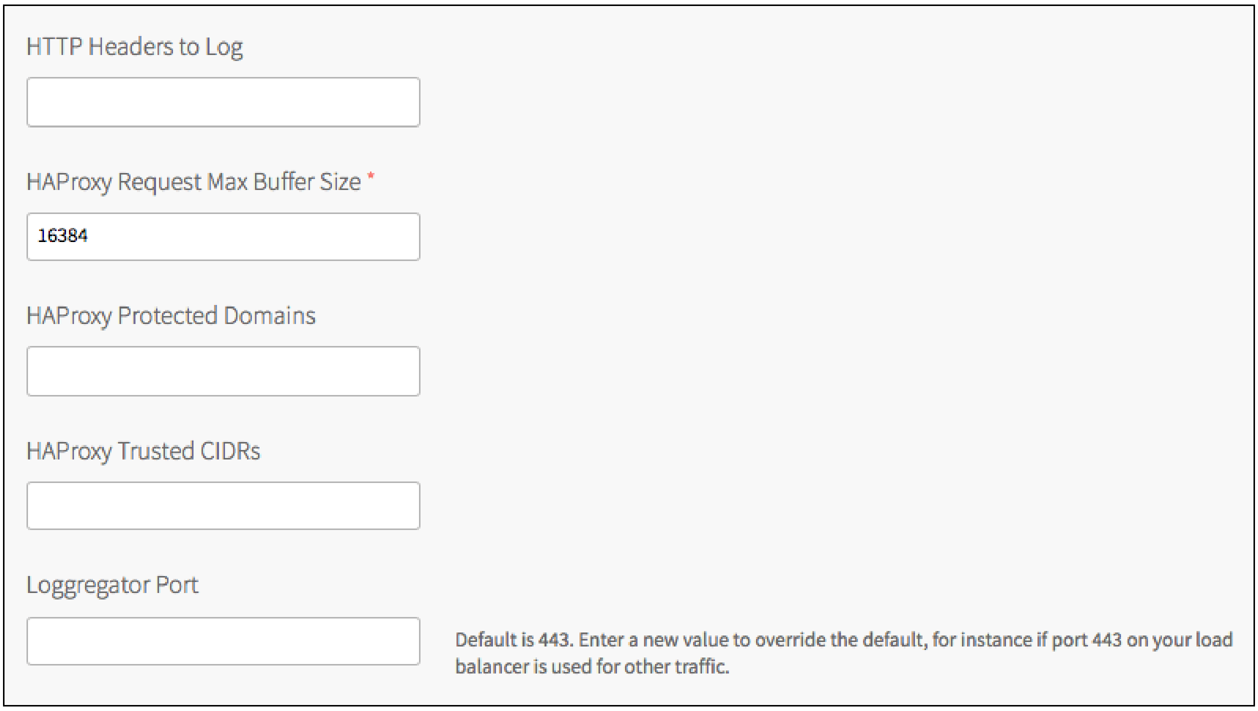
WebSocket HTTP requests on ports 80 and 443. For more information, see Securing Traffic into TAS for VMs.

Note: Gorouter rejects WebSockets requests for routes that are bound to route services. These requests return a 503 error and a `X-Cf-Routererror route_service_unsupported` header.

Note: Gorouter does not support WebSockets over HTTP/2. For more information, see RFC 8441. Configure your load balancer to always send WebSocket requests to Gorouter over HTTP/1.1. For more information, see Configuring HTTP/2 Support.

Setting your Loggregator port

By default, TAS for VMs assigns port 443 for TCP/WebSocket communications. If you have configured your load balancer to use a port other than 443 for TCP/WebSocket traffic, you must edit the **Loggregator port** field in the **Networking** pane of the TAS for VMs tile.



HTTP Headers to Log

HAProxy Request Max Buffer Size *

16384

HAProxy Protected Domains

HAProxy Trusted CIDRs

Loggregator Port

Default is 443. Enter a new value to override the default, for instance if port 443 on your load balancer is used for other traffic.

Enabling Zipkin tracing in TAS for VMs

Zipkin tracing allows you to troubleshoot failures or latency issues in your apps. You can trace requests and responses across distributed systems. For more information, see Zipkin.io.

Zipkin tracing is enabled by default in VMware Tanzu Application Service for VMs (TAS for VMs). To deactivate this feature:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.

4. Deselect the **Enable Zipkin tracing headers on the Gorouter** checkbox.
5. Click **Save**.

For more information about how the Gorouter works with HTTP headers and Zipkin tracing, see the HTTP Headers section of the *HTTP Routing* topic.

To trace app requests and responses in VMware Tanzu Application Service for VMs (TAS for VMs), apps must also log Zipkin headers.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs myapp` to correlate the trace and span IDs logged by the Gorouter with the trace IDs logged by their app. To correlate trace IDs for a request through multiple apps, each app must forward appropriate values for the headers with requests to other apps.

Managing Internal MySQL

In this section:

- Scaling Internal MySQL
- Using the MySQL Proxy
- Running mysql-diag
- Recovering From MySQL Cluster Downtime

Scaling Internal MySQL

This topic describes scaling down your VMware Tanzu Application Service for VMs (TAS for VMs) MySQL cluster and provides example sizing data from two environments that have significant load on their MySQL clusters.

For additional resources about scaling Internal MySQL, see the Deployments Using Internal MySQL section of the *Scaling TAS for VMs* topic and the Scalable Components section of the *High Availability in TAS for VMs* topic.

> **Note:** The procedure does not apply to databases configured as external in the **Databases** pane of the TAS for VMs tile.

TAS for VMs components that use system databases include the Cloud Controller, Diego Brain, Gorouter, and User Authorization and Authentication (UAA) server. For more information, see TAS for VMs Components.

Scale Down Your MySQL Cluster

This procedure explains how to safely scale your MySQL cluster down to a single node. If you are already running the MySQL cluster with a single node, you do not need to do this procedure.

By default, internal MySQL deploys as a single node. To take advantage of the high availability features of MySQL, you may have scaled the configuration up to three or more server nodes.

Check the Health of Your Cluster

Before scaling down your MySQL cluster, you must ensure the cluster is healthy.

To check the health of your cluster:

1. Use the Cloud Foundry Command Line Interface (cf CLI) to target the API endpoint of your Ops Manager deployment. Run:

   ```
   cf api api.YOUR-SYSTEM-DOMAIN
   ```

   Where `YOUR-SYSTEM-DOMAIN` is the system domain defined in the **Domains** pane of the TAS for VMs tile.

2. To obtain your UAA admin user credentials:
   1. Click the **Credentials** tab of the TAS for VMs tile.
   2. Locate the **Admin Credentials** entry in the **UAA** section.
   3. Click **Link to Credential**.

3. Log in with the UAA admin credentials you obtained in the previous step by running:

   ```
   cf login -u admin
   ```

4. Create a test organization to verify the database across all nodes by running:

   ```
   cf create-org data-integrity-test-organization
   ```

5. To obtain the IP addresses of your MySQL server:
   1. From the Ops Manager Installation Dashboard, click the TAS for VMs tile.
   2. Click the **Status** tab.
   3. Record the IP addresses for all instances of the **MySQL Server** job.

6. To retrieve Cloud Controller database credentials from CredHub using the Ops Manager API:
   1. Follow the procedures in Using the Ops Manager API to authenticate and access the Ops Manager API.
   2. Retrieve a list of deployed products by running:

      ```
      curl "https://OPS-MANAGER-FQDN/api/v0/deployed/products" \
      -X GET \
      -H "Authorization: Bearer UAA-ACCESS-TOKEN"
      ```

      Where:

      - `OPS-MANAGER-FQDN` is the fully-qualified domain name (FQDN) of your Ops Manager deployment.
      - `UAA-ACCESS-TOKEN` is the access token you recorded in the previous step.
   3. In the response to the above request, locate the product with an `installation_name` starting with `cf-` and copy its `guid`.
   4. Run:

```
    curl "https://OPS-MANAGER-FQDN/api/v0/deployed/products/PRODUCT-GUID/vari
    ables?name=cc-db-credentials" \
    -X GET \
    -H "Authorization: Bearer UAA-ACCESS-TOKEN"
    ```

    Where:

    - `OPS-MANAGER-FQDN` is the fully-qualified domain name (FQDN) of your Ops Manager deployment.
    - `PRODUCT-GUID` is the value of `guid` you recorded in the previous step.
    - `UAA-ACCESS-TOKEN` is the access token you recorded in the previous step.

5. Record the Cloud Controller database `username` and `password` from the response to the above request.

7. SSH into the Ops Manager VM. Because the procedures vary by IaaS, see the SSH into Ops Manager section of the *Advanced Troubleshooting with the BOSH CLI* topic for instructions specific to your IaaS.

8. For each of the MySQL server IP addresses recorded above:

    1. Query the new organization by running:

        ```
        mysql -h YOUR-IP -u YOUR-IDENTITY -D ccdb -p -e "select created_at, name
        from organizations where name = 'data-integrity-test-organization'"
        ```

        Where:

        - `YOUR-IP` is the IP address of the MySQL server.
        - `YOUR-IDENTITY` is the `identity` value of the CCDB credentials you obtained above.

    2. When prompted, provide the `password` value of the CCDB credentials you obtained above.

    3. Examine the output of the `mysql` command and verify the `created_at` date is recent. For example:

        ```
        +---------------------+----------------------------------+
        | created_at          | name                             |
        +---------------------+----------------------------------+
        | 2016-05-28 01:11:42 | data-integrity-test-organization |
        +---------------------+----------------------------------+
        ```

9. If each MySQL server instance does not return the same `created_at` result, contact VMware Tanzu Support before proceeding further or making any changes to your deployment. If each MySQL server instance does return the same result, then you can safely proceed to scaling down your cluster to a single node by performing the steps in the following section.

Set Server Instance Count to 1

To scale your server instance count to 1:

1. Select **Resource Config** in the TAS for VMs tile.
2. Use the dropdown to change the **Instances** count for **MySQL Server** to `1`.
3. Click **Save**, then **Review Pending Changes** and **Apply Changes** to apply the changes.
4. Delete your test organization by running:

```
cf delete-org data-integrity-test-organization
```

MySQL Cluster Sizing Example

This topic describes a sizing example for internal MySQL in TAS for VMs.

Use this data as guidance to ensure your MySQL Clusters are scaled to handle the number of app instances running on your deployment.

Example: Cloud Foundry Diego Test Environment

The information in this section comes from an environment used by the Diego to test the MySQL cluster with a high load similar to one generated large deployment.

- **IAAS**: GCP
- **App Instances**: 250,000
- **Average SQL Queries Per Minute**: ~5,100,000
- **Average IOPS**:
  - **Reads**: 3.63
  - **Writes**: 363.98

VM Sizing

The following table displays MySQL VM settings for this environment:

Setting	Value
VM Type	n1-standard-16
Storage Volume Type	pd-ssd
Storage Volume Size	1 TB

Using the MySQL Proxy

- MySQL Proxy
- Node Health
  - Healthy
  - Unhealthy
  - Unresponsive
- Monitor Node Health

This topic describes how to use the MySQL proxy to monitor the health of your MySQL instance.

If you cannot write or connect to your app, you should check the health status of your MySQL VMs. You can observe the health status of MySQL VMs through a proxy using the Switchboard dashboard or API endpoint. You can also view the number of client connections routed through a proxy to each VM.

Ops Manager uses a proxy to send client connections to the healthy MySQL database cluster nodes in a highly available cluster plan. Using a proxy gracefully handles failure of nodes, enabling fast, failover to other nodes within the cluster. When a node becomes unhealthy, the proxy closes all connections to the unhealthy node and re-routes all subsequent connections to a healthy node.

The proxy used in TAS for VMs is Switchboard. Switchboard was developed to replace HAProxy as the proxy tier for the high availability cluster for MySQL databases in TAS for VMs.

Switchboard offers the following features:

- **MySQL Server Access**

  MySQL clients communicate with nodes through this network port. These connections are automatically passed through to the nodes.

- **Switchboard and API**

  Operators can connect to Switchboard to view the state of the nodes.

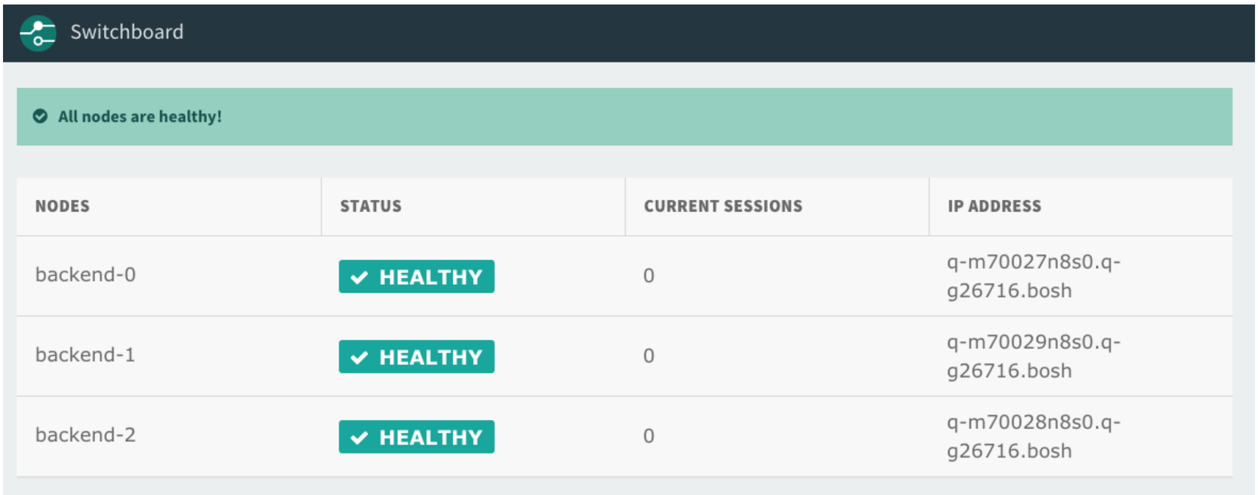
Node Health

When determining where to route traffic, the proxy queries an HTTP health check process running on the node. This health check can return as either healthy or unhealthy, or the node can be unresponsive.

Healthy

If the health check process returns HTTP status code `200`, the proxy includes the node in its pool of healthy nodes.

When a new or resurrected nodes rejoin the cluster, the proxy continues to route all connections to the currently active node. In the case of failover, the proxy considers all healthy nodes as candidates for new connections.



Switchboard

All nodes are healthy!

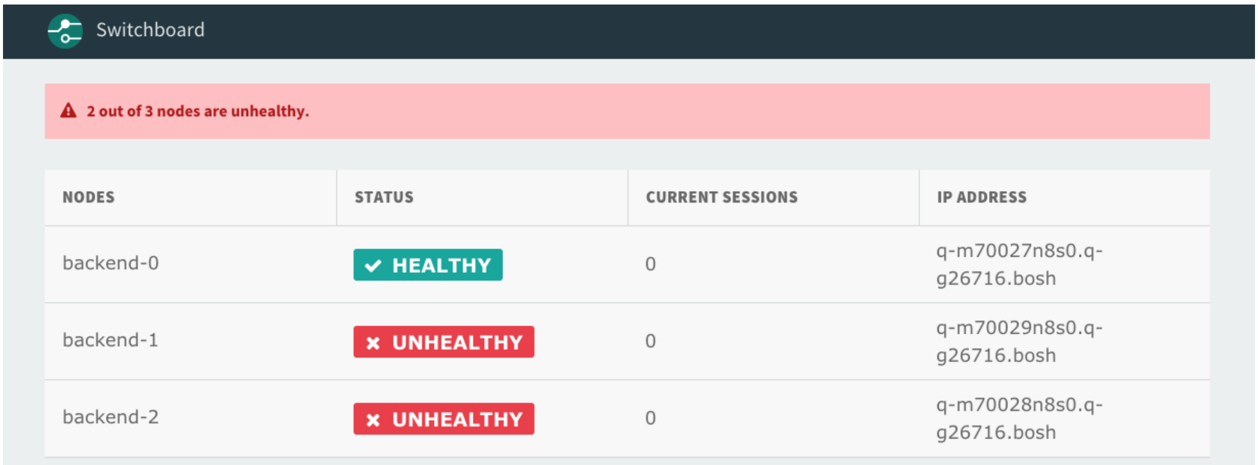
NODES	STATUS	CURRENT SESSIONS	IP ADDRESS
backend-0	✓ HEALTHY	0	q-m70027n8s0.q-g26716.bosh
backend-1	✓ HEALTHY	0	q-m70029n8s0.q-g26716.bosh
backend-2	✓ HEALTHY	0	q-m70028n8s0.q-g26716.bosh

Unhealthy

If the health check returns HTTP status code `503`, the proxy considers the node unhealthy.

This happens when a node becomes non-primary. For more information, see About Multi-Site Replication.

The proxy severs existing connections to newly unhealthy node. The proxy routes new connections to a healthy node, assuming such a node exists. Clients are expected to handle reconnecting on connection failure should the entire cluster become inaccessible.



Switchboard

⚠ 2 out of 3 nodes are unhealthy.

NODES	STATUS	CURRENT SESSIONS	IP ADDRESS
backend-0	✓ HEALTHY	0	q-m70027n8s0.q-g26716.bosh
backend-1	✗ UNHEALTHY	0	q-m70029n8s0.q-g26716.bosh
backend-2	✗ UNHEALTHY	0	q-m70028n8s0.q-g26716.bosh

Unresponsive

If node health cannot be determined due to an unreachable or unresponsive health check endpoint, the proxy considers the node unhealthy. This may happen if there is a network partition or if the VM running the node and healthcheck died.

Monitor Node Health

You can monitor the health status of your nodes by doing one of the following:

- Monitor Node Health Using the Dashboard
- Monitor Node Health Using the API

Monitor Node Health Using the Dashboard

Construct the URL for the Dashboard

Each MySQL Proxy instance has its own Switchboard dashboard. To access the dashboard, you must construct the dashboard URL.

To construct the URL for the Switchboard dashboard:

1. Find and record the system domain on the **Domains** pane in the TAS for VMs tile.
2. Find and record the number of MySQL Proxy VM instances in your deployment in the **Resource Config** pane.
3. Use the above information to build the Switchboard dashboard URL as follows: `https://[PROXY-VM-INDEX]-proxy-p-mysql-ert.[SYSTEM-DOMAIN]`

Log In to the Dashboard

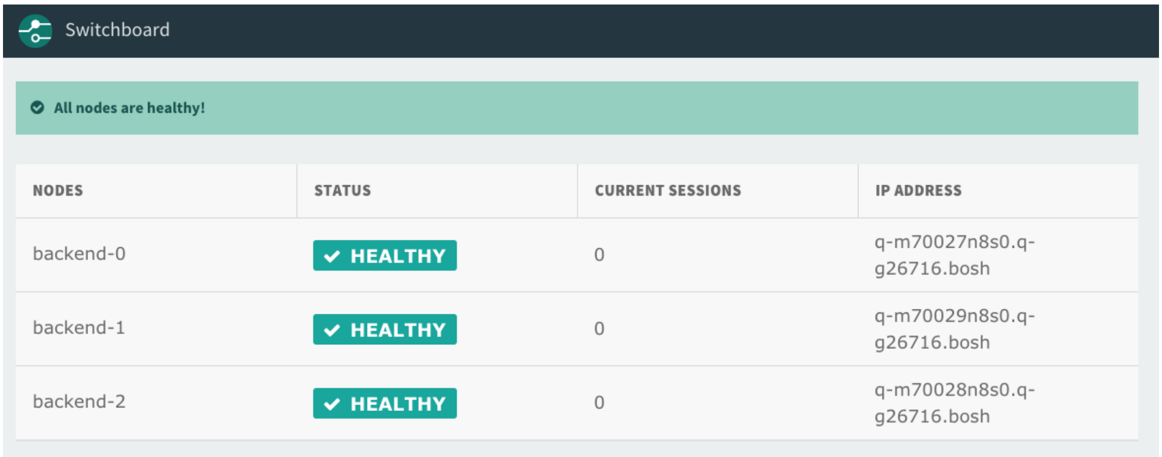
To log into the Switchboard dashboard:

1. In the TAS for VMs tile, the URL in your browser contains the Ops Manager fully qualified domain name (FQDN) and the TAS for VMs product GUID, as in the following example: `https://OPS-MANAGER-FQDN/products/APPLICTION-SERVICE-PRODUCT-GUID/az_and_network_assignments/edit`
   To log into the dashboard, you need the username and password for the MySQL Proxy VM that are stored in the BOSH Director CredHub.
   To obtain them, enter `https://[OPS-MANAGER-FQDN]/api/v0/deployed/products/[APPLICTION-SERVICE-PRODUCT-GUID]/variables?name=mysql-proxy-dashboard-credentials` into your browser and record the username and password.
2. Enter the dashboard URL you constructed into your browser and log in using the credentials you obtained in Construct the URL for the Dashboard.

Use the Dashboard

To monitor node health using the Switchboard dashboard:

1. To view a list of proxies in your browser, navigate to the URL that you created in Construct the URL for the Dashboard.
2. When prompted, enter the username and password that you recorded in Log In to the Dashboard.
3. Click the link for the proxy that you want to use to monitor node health.
4. If you are prompted, enter the username and password that you recorded in Log In to the Dashboard.



All nodes are healthy!

NODES	STATUS	CURRENT SESSIONS	IP ADDRESS
backend-0	✓ HEALTHY	0	q-m70027n8s0.q-g26716.bosh
backend-1	✓ HEALTHY	0	q-m70029n8s0.q-g26716.bosh
backend-2	✓ HEALTHY	0	q-m70028n8s0.q-g26716.bosh

Monitor Node Health Using the API

You can also use the Switchboard API to obtain the information that is shown on the Switchboard dashboard. For example, you might want to use the API to write your own app to monitor the cluster.

To monitor node health using the Switchboard API:

1. To monitor node health, run:

   ```
   curl https://USERNAME:PASSWORD@N-HOSTNAME/v0/backends
   ```

   Where:

   - `USERNAME` is the `username` you recorded in Log In to the Dashboard.
   - `PASSWORD` is the `password` you recorded in Log In to the Dashboard.
   - `N` is either 0, 1, or 2 depending on the proxy you want to connect to.
   - `HOSTNAME` is the IP address of the **Database** found in the **Status** tab of your TAS for VMs tile.

   The above command outputs a JSON object similar to:

   ```
   $ curl https://abcdefghijklmno:012345678912345@0-proxy.123abc45-67d8-91
   2e-34f5-g34612c10dba.org.dedicated-mysql.cf-app.com/v0/backends
   [
     {
       "host": "a-b1234c5d6.e-f891.bosh",
       "port": 6033,
       "healthy": true,
       "name": "backend-0",
       "currentSessionCount": 0,
       "active": true,
       "trafficEnabled": true
     },
     {
       "host": "a-b1234c5d6.e-f891.bosh",
       "port": 6033,
       "healthy": true,
   ```

```
    "name": "backend-1",
    "currentSessionCount": 0,
    "active": false,
    "trafficEnabled": true
  },
  {
    "host": "a-b1234c5d6.e-f891.bosh",
    "port": 6033,
    "healthy": true,
    "name": "backend-2",
    "currentSessionCount": 0,
    "active": false,
    "trafficEnabled": true
  }
]
```

Running mysql-diag

- Run mysql-diag the Using BOSH Command Line Interface (CLI)
- Example Healthy Output
- Example Unhealthy Output

This topic discusses how to use the `mysql-diag` tool in Ops Manager. `mysql-diag` prints the state of your MySQL highly available (HA) cluster and suggests solutions if your node fails. VMware recommends running this tool against your HA cluster before all deployments.

`mysql-diag` checks the following information about the status of your HA cluster:

- Membership status of all nodes
- Size as it appears to all nodes
- If it needs to be bootstrapped
- If replication is working
- Used disk space and inodes per server

Run mysql-diag Using the BOSH Command Line Interface (CLI)

To use the BOSH CLI to run `mysql-diag`, do the following:

1. Obtain the information needed to use the BOSH CLI by doing the procedure in Gather Credential and IP Address Information.
2. SSH into your Ops Manager VM by doing the procedure in Log in to the Ops Manager VM with SSH for your IaaS.
3. Log in to your BOSH Director by doing the procedure in Authenticate with the BOSH Director VM.
4. Identify the VM to log in to with SSH by running the following command:

```
bosh -e MY-ENV -d MY-DEPLOYMENT vms
```

Where:

- `MY-ENV` is the name of your environment.
- `MY-DEPLOYMENT` is the name of your deployment.

5. Record the GUID associated with the `mysql-monitor` VM, also known as the jumpbox VM.
6. SSH into your `mysql-monitor` VM by running the following command:

   ```
   bosh -e MY-ENV -d MY-DEP ssh mysql-monitor/GUID
   ```

   Where:

   - `MY-ENV` is the name of your environment.
   - `MY-DEPLOYMENT` is the name of your deployment.
   - `GUID` is the GUID you recorded in the previous step.
7. View the status of your HA cluster by running the following command:

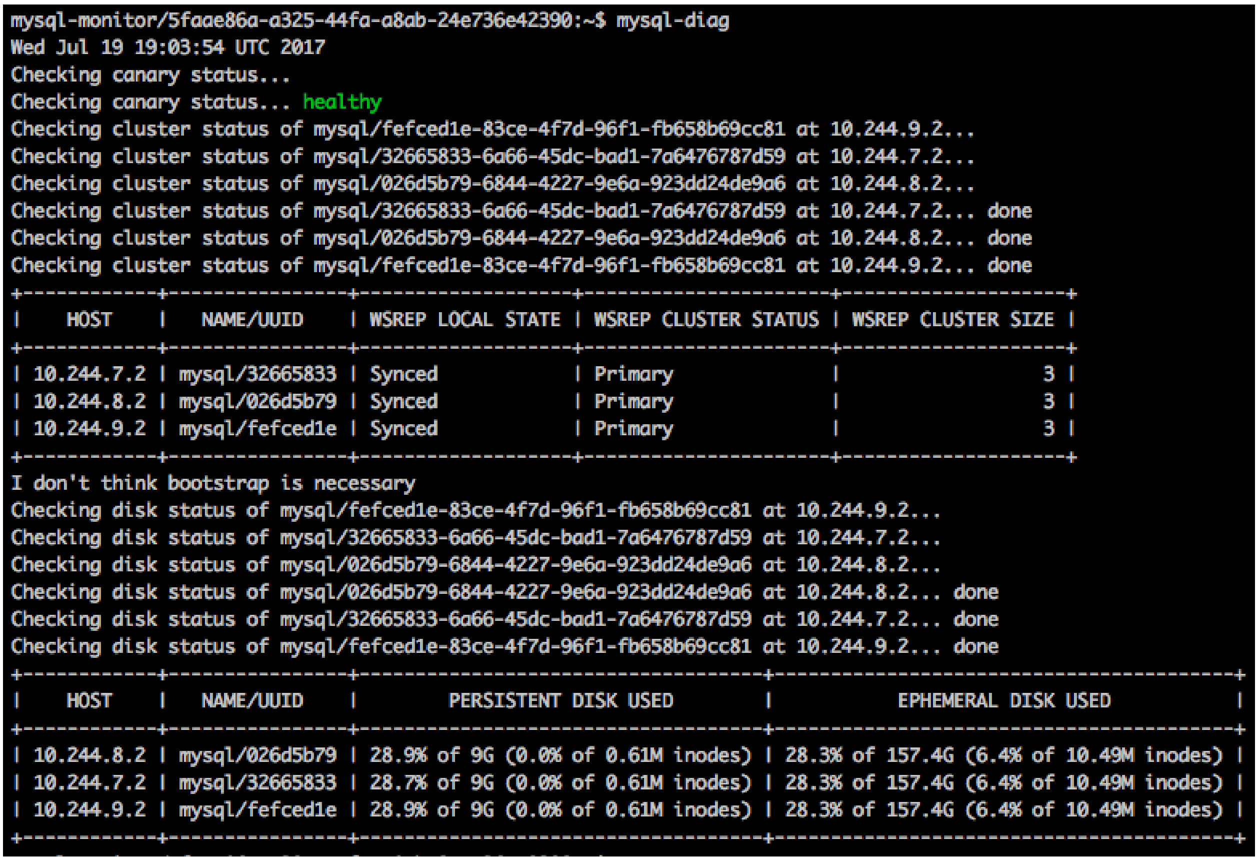
   ```
   mysql-diag
   ```

Example Healthy Output

The `mysql-diag` command returns the following message if your canary status is healthy:

```
Checking canary status...healthy
```

Here is a sample `mysql-diag` output after the tool identified a healthy HA cluster:



```
mysql-monitor/5faae86a-a325-44fa-a8ab-24e736e42390:~$ mysql-diag
Wed Jul 19 19:03:54 UTC 2017
Checking canary status...
Checking canary status... healthy
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+----------------+-------------------+----------------------+--------------------+
|    HOST    |   NAME/UUID    | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+------------+----------------+-------------------+----------------------+--------------------+
| 10.244.7.2 | mysql/32665833 | Synced            | Primary              |                  3 |
| 10.244.8.2 | mysql/026d5b79 | Synced            | Primary              |                  3 |
| 10.244.9.2 | mysql/fefced1e | Synced            | Primary              |                  3 |
+------------+----------------+-------------------+----------------------+--------------------+
I don't think bootstrap is necessary
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+----------------+------------------------------------+-----------------------------------------+
|    HOST    |   NAME/UUID    |        PERSISTENT DISK USED        |           EPHEMERAL DISK USED           |
+------------+----------------+------------------------------------+-----------------------------------------+
| 10.244.8.2 | mysql/026d5b79 | 28.9% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
| 10.244.7.2 | mysql/32665833 | 28.7% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
| 10.244.9.2 | mysql/fefced1e | 28.9% of 9G (0.0% of 0.61M inodes) | 28.3% of 157.4G (6.4% of 10.49M inodes) |
+------------+----------------+------------------------------------+-----------------------------------------+
```

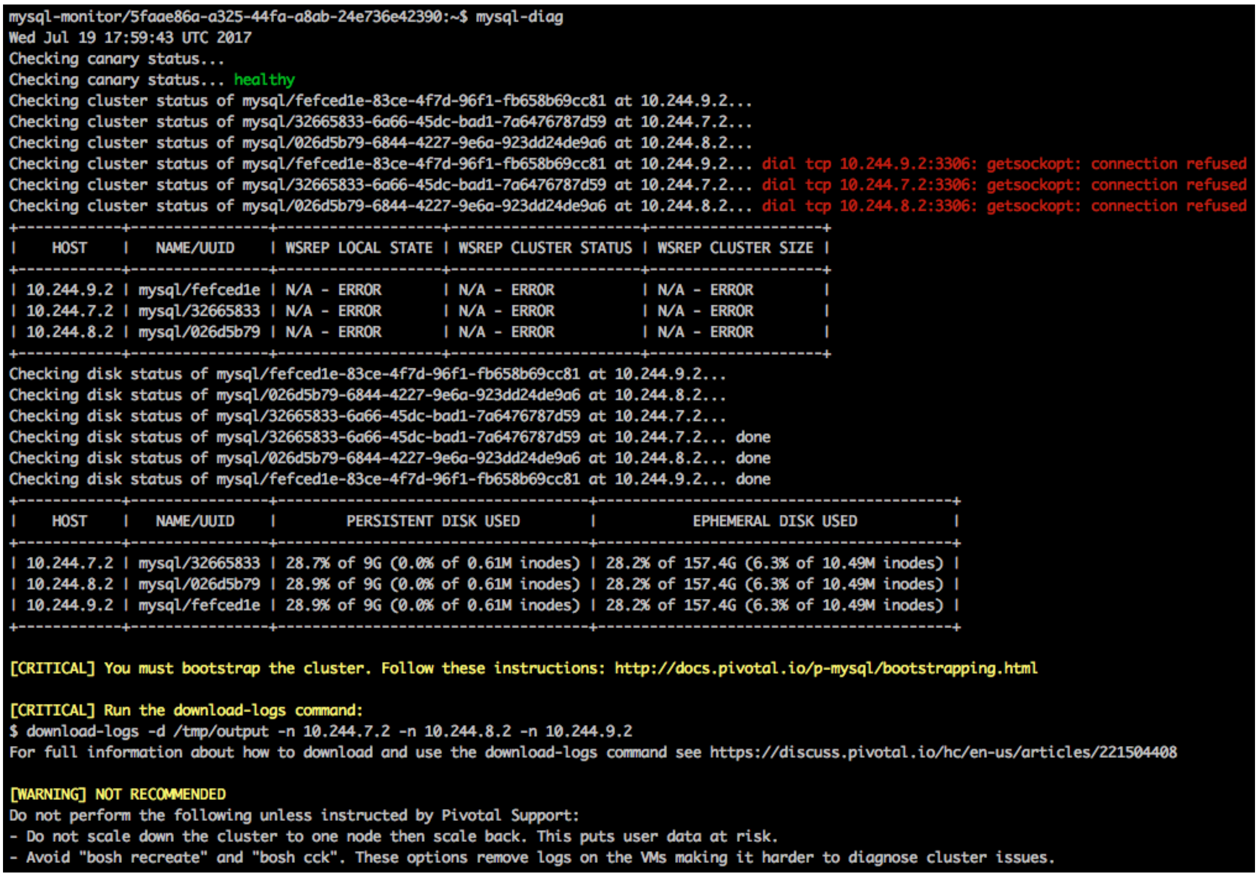
View a larger version of this image.

Example Unhealthy Output

The `mysql-diag` command returns the following message if your canary status is unhealthy:

```
Checking canary status...unhealthy
```

In the event of a broken HA cluster, running `mysql-diag` outputs actionable steps meant to expedite the recovery of that HA cluster. Below is a sample `mysql-diag` output after the tool identified an unhealthy HA cluster:



```
mysql-monitor/5faae86a-a325-44fa-a8ab-24e736e42390:~$ mysql-diag
Wed Jul 19 17:59:43 UTC 2017
Checking canary status...
Checking canary status... healthy
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking cluster status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... dial tcp 10.244.9.2:3306: getsockopt: connection refused
Checking cluster status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... dial tcp 10.244.7.2:3306: getsockopt: connection refused
Checking cluster status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... dial tcp 10.244.8.2:3306: getsockopt: connection refused
+------------+----------------+-------------------+----------------------+--------------------+
|    HOST    |   NAME/UUID    | WSREP LOCAL STATE | WSREP CLUSTER STATUS | WSREP CLUSTER SIZE |
+------------+----------------+-------------------+----------------------+--------------------+
| 10.244.9.2 | mysql/fefced1e | N/A - ERROR       | N/A - ERROR          | N/A - ERROR        |
| 10.244.7.2 | mysql/32665833 | N/A - ERROR       | N/A - ERROR          | N/A - ERROR        |
| 10.244.8.2 | mysql/026d5b79 | N/A - ERROR       | N/A - ERROR          | N/A - ERROR        |
+------------+----------------+-------------------+----------------------+--------------------+
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2...
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2...
Checking disk status of mysql/32665833-6a66-45dc-bad1-7a6476787d59 at 10.244.7.2... done
Checking disk status of mysql/026d5b79-6844-4227-9e6a-923dd24de9a6 at 10.244.8.2... done
Checking disk status of mysql/fefced1e-83ce-4f7d-96f1-fb658b69cc81 at 10.244.9.2... done
+------------+----------------+------------------------------------+-----------------------------------------+
|    HOST    |   NAME/UUID    |        PERSISTENT DISK USED        |           EPHEMERAL DISK USED           |
+------------+----------------+------------------------------------+-----------------------------------------+
| 10.244.7.2 | mysql/32665833 | 28.7% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
| 10.244.8.2 | mysql/026d5b79 | 28.9% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
| 10.244.9.2 | mysql/fefced1e | 28.9% of 9G (0.0% of 0.61M inodes) | 28.2% of 157.4G (6.3% of 10.49M inodes) |
+------------+----------------+------------------------------------+-----------------------------------------+

[CRITICAL] You must bootstrap the cluster. Follow these instructions: http://docs.pivotal.io/p-mysql/bootstrapping.html

[CRITICAL] Run the download-logs command:
$ download-logs -d /tmp/output -n 10.244.7.2 -n 10.244.8.2 -n 10.244.9.2
For full information about how to download and use the download-logs command see https://discuss.pivotal.io/hc/en-us/articles/221504408

[WARNING] NOT RECOMMENDED
Do not perform the following unless instructed by Pivotal Support:
- Do not scale down the cluster to one node then scale back. This puts user data at risk.
- Avoid "bosh recreate" and "bosh cck". These options remove logs on the VMs making it harder to diagnose cluster issues.
```

View a larger version of this image.

Recovering From MySQL Cluster Downtime

This topic describes how to bootstrap your MySQL cluster in the event of a cluster failure.

You can bootstrap your cluster by using one of two methods:

- Run the bootstrap errand. See Run the Bootstrap Errand below.

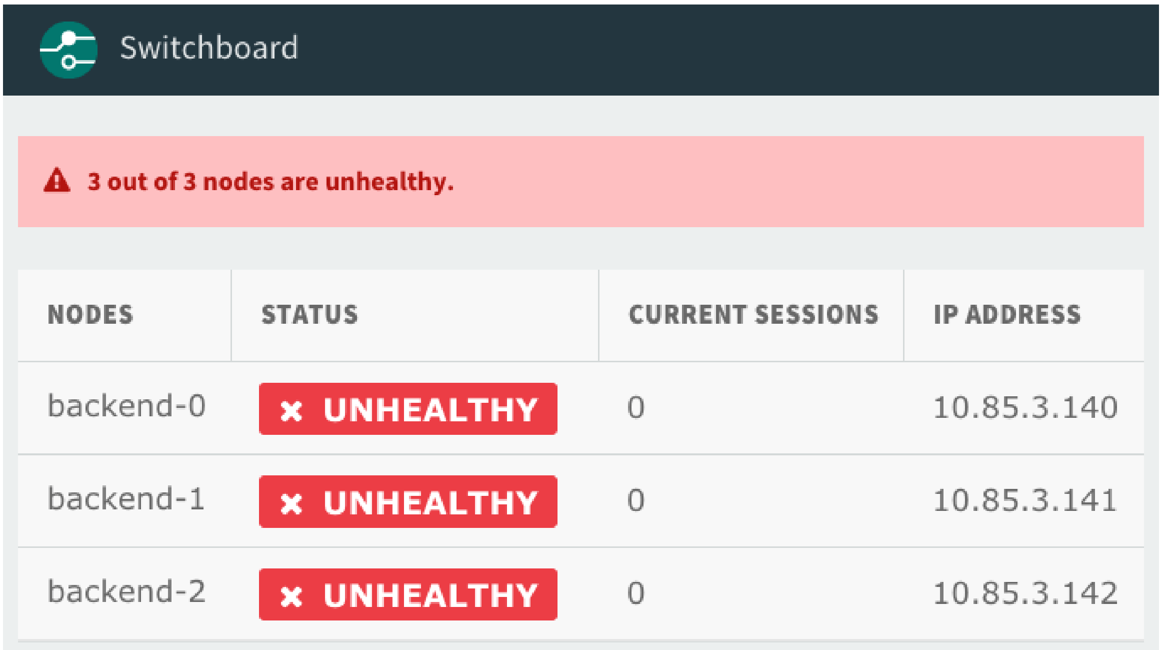
- Bootstrap manually. See Bootstrap Manually below.

When to Bootstrap

You must bootstrap a cluster that loses quorum. A cluster loses quorum when fewer than half the nodes can communicate with each other for longer than the configured grace period. If a cluster does not lose quorum, individual unhealthy nodes automatically rejoin the cluster after resolving the error, restarting the node, or restoring connectivity.

To check whether your cluster has lost quorum, look for the following symptoms:

- All nodes appear “Unhealthy” on the proxy dashboard. To log in to the proxy dashboard, see Log In to Proxy Dashboard below.



Switchboard

⚠ 3 out of 3 nodes are unhealthy.

NODES	STATUS	CURRENT SESSIONS	IP ADDRESS
backend-0	✖ UNHEALTHY	0	10.85.3.140
backend-1	✖ UNHEALTHY	0	10.85.3.141
backend-2	✖ UNHEALTHY	0	10.85.3.142

- All responsive nodes report the value of `wsrep_cluster_status` as `non-Primary`:

```
mysql> SHOW STATUS LIKE 'wsrep_cluster_status';
+----------------------+-------------+
| Variable_name        | Value       |
+----------------------+-------------+
| wsrep_cluster_status | non-Primary |
+----------------------+-------------+
```

- All unresponsive nodes respond with `ERROR 1047` when using most statement types in the MySQL client:

```
mysql> select * from mysql.user;
ERROR 1047 (08S01) at line 1: WSREP has not yet prepared node for appli
cation use
```

Log In to Proxy Dashboard

To log in to the proxy dashboard:

1. Get the proxy dashboard credentials by entering the following URL into your web browser and recording the output:

   ```
   https://OPS-MAN-FQDN/api/v0/deployed/products/TAS-FOR-VMS-GUID/variables?name=m
   ysql-proxy-dashboard-credentials
   ```

   Where:

   - `OPS-MAN-FQDN` is the fully-qualified domain name (FQDN) for your Ops Manager deployment.
   - `TAS-FOR-VMS-GUID` is the GUID of your TAS for VMs tile. For more information about finding your GUID, see the Ops Manager API documentation

2. Log in to the proxy dashboard by entering the following URL into your web browser:

   ```
   https://BOSH-JOB-INDEX-proxy-p-mysql-ert.YOUR-SYSTEM-DOMAIN/
   ```

   Where:

   - `BOSH-JOB-INDEX` is `0`, `1`, or `2`, representing the three proxies deployed on each node.
   - `SYSTEM-DOMAIN` is the **System domain** you configured in the **Domains** pane of the BOSH Director tile.

Run the Bootstrap Errand

Ops Manager includes a BOSH errand that automates the manual bootstrapping procedure in the Bootstrap Manually section below.

It finds the node with the highest transaction sequence number and asks it to start up by itself (i.e. in bootstrap mode) and then asks the remaining nodes to join the cluster.

In most cases, running the errand recovers your cluster. But, certain scenarios require additional steps.

Determine Type of Cluster Failure

To determine which set of instructions to follow:

1. List your MySQL instances by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:

   - `YOUR-ENV` is the environment where you deployed the cluster.
   - `YOUR-DEPLOYMENT` is the deployment cluster name.

   For example:

   ```
   $ bosh -e prod -d mysql instances
   ```

2. Find and record the `Process State` for your MySQL instances. In the following example output, the MySQL instances are in the `failing` process state.

```
Instance                                                             Pr
ocess State  AZ             IPs
backup-restore/c635410e-917d-46aa-b054-86d222b6d1c0                  ru
nning        us-central1-b  10.0.4.47
bootstrap/a31af4ff-e1df-4ff1-a781-abc3c6320ed4                       -
us-central1-b  -
broker-registrar/1a93e53d-af7c-4308-85d4-3b2b80d504e4                -
us-central1-b  10.0.4.58
cf-mysql-broker/137d52b8-a1b0-41f3-847f-c44f51f87728                 ru
nning        us-central1-c  10.0.4.57
cf-mysql-broker/28b463b1-cc12-42bf-b34b-82ca7c417c41                 ru
nning        us-central1-b  10.0.4.56
deregister-and-purge-instances/4cb93432-4d90-4f1d-8152-d0c238fa5aab  -
us-central1-b  -
monitoring/f7117dcb-1c22-495e-a99e-cf2add90dea9                      ru
nning        us-central1-b  10.0.4.48
mysql/220fe72a-9026-4e2e-9fe3-1f5c0b6bf09b                           fa
iling        us-central1-b  10.0.4.44
mysql/28a210ac-cb98-4ab4-9672-9f4c661c57b8                           fa
iling        us-central1-f  10.0.4.46
mysql/c1639373-26a2-44ce-85db-c9fe5a42964b                           fa
iling        us-central1-c  10.0.4.45
proxy/87c5683d-12f5-426c-b925-62521529f64a                           ru
nning        us-central1-b  10.0.4.60
proxy/b0115ccd-7973-42d3-b6de-edb5ae53c63e                           ru
nning        us-central1-c  10.0.4.61
rejoin-unsafe/8ce9370a-e86b-4638-bf76-e103f858413f                   -
us-central1-b  -
smoke-tests/e026aaef-efd9-4644-8d14-0811cb1ba733                     -
us-central1-b  10.0.4.59
```

3. Choose your scenario:

   - If your MySQL instances are in the `failing` state, continue to Scenario 1.
   - If your MySQL instances are in the `-` state, continue to Scenario 2.
   - If your MySQL instances are in the `stopped` state, continue to Scenario 3.

Scenario 1: VMs Running, Cluster Disrupted

In this scenario, the VMs are running, but the cluster has been disrupted.

To bootstrap in this scenario:

1. Run the bootstrap errand on the VM where the bootstrap errand is co-located, by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT run-errand --instance=VM bootstrap
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

- `VM` is `clock_global/0` if you use TAS for VMs and `control/0` if you use Small Footprint TAS for VMs.

Note: The errand runs for a long time, during which no output is returned.

The command returns many lines of output, eventually followed by:

```
Bootstrap errand completed
[stderr]
+ echo 'Started bootstrap errand ...'
+ JOB\_DIR=/var/vcap/jobs/bootstrap
+ CONFIG\_PATH=/var/vcap/jobs/bootstrap/config/config.yml
+ /var/vcap/packages/bootstrap/bin/cf-mysql-bootstrap -configPath=/var/
vcap/jobs/bootstrap/config/config.yml
+ echo 'Bootstrap errand completed'
+ exit 0
Errand 'bootstrap' completed successfully (exit code 0)
```

2. If the errand fails, run the bootstrap errand command again after a few minutes. The bootstrap errand might not work the first time.
3. If the errand fails after several tries, bootstrap your cluster manually. See Bootstrap Manually below.

Scenario 2: VMs Terminated or Lost

In severe circumstances, such as a power failure, it is possible to lose all your VMs. You must re-create them before you can begin recovering the cluster.

When MySQL instances are in the `-` state, the VMs are lost. The procedures in this scenario bring the instances from a `-` state to a `failing` state. Then you run the bootstrap errand similar to Scenario 1 above and restore configuration.

To recover terminated or lost VMs, follow the procedures in the sections below:

1. Re-create the Missing VMs: Bring MySQL instances from a `-` state to a `failing` state.
2. Run the Bootstrap Errand: Because your instances are now in the `failing` state, continue similarly to Scenario 1 above.
3. Restore the BOSH Configuration: Go back to unignoring all instances and redeploy. This is a critical and mandatory step.

Caution: If you do not set each of your ignored instances to `unignore`, your instances are not updated in future deploys. You must follow the procedure in the final section of *Scenario 2*, Restore the BOSH Configuration.

Re-create the Missing VMs

The procedure in this section uses BOSH to re-create the VMs, install software on them, and try to start the jobs.

The procedure below allows you to:

- Redeploy your cluster while expecting the jobs to fail.
- Instruct BOSH to ignore the state of each instance in your cluster. This allows BOSH to deploy the software to each instance even if the instance is failing.

To re-create your missing VMs:

1. If BOSH Resurrector is activated, deactivate it by running:

   ```
   bosh -e YOUR-ENV update-resurrection off
   ```

   Where `YOUR-ENV` is the name of your environment.

2. Download the current manifest by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT manifest > /tmp/manifest.yml
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

3. Redeploy deployment by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy /tmp/manifest.yml
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

   > **Note:** Expect one of the MySQL VMs to fail. Deploying causes BOSH to create new VMs and install the software. Forming a cluster is in a subsequent step.

4. View the instance GUID of the VM that attempted to start by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

   Record the instance GUID, which is the string after `mysql/` in your BOSH instances output.

5. Instruct BOSH to ignore each MySQL VM by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/INSTANCE-GUID
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.

   - `YOUR-DEPLOYMENT` is the name of your deployment.
   - `INSTANCE-GUID` is the GUID of your instance you recorded in the above step.
6. Repeat steps 4 and 5 until all instances have attempted to start.
7. If you deactivated the BOSH Resurrector in step 2, re-enable it by running:

   ```
   bosh -e YOUR-ENV update-resurrection on
   ```

   Where `YOUR-ENV` is the name of your environment.
8. Confirm that your MySQL instances have gone from the `-` state to the `failing` state by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

Run the Bootstrap Errand

After you re-create the VMs, all instances now have a `failing` process state and have the MySQL code. You must run the bootstrap errand to recover the cluster.

To bootstrap:

1. Run the bootstrap errand by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT run-errand --instance=VM bootstrap
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.
   - `VM` is `clock_global/0` if you use TAS for VMs and `control/0` if you use Small Footprint TAS for VMs.

   Note: The errand runs for a long time, during which no output is returned.

   The command returns many lines of output, eventually with the following successful output:

   ```
     Bootstrap errand completed
     [stderr]
     echo 'Started bootstrap errand ...'
     JOB\_DIR=/var/vcap/jobs/bootstrap
     CONFIG\_PATH=/var/vcap/jobs/bootstrap/config/config.yml
     /var/vcap/packages/bootstrap/bin/cf-mysql-bootstrap -configPath=/var/
   vcap/jobs/bootstrap/config/config.yml
     echo 'Bootstrap errand completed'
     exit 0
   ```

```
Errand 'bootstrap' completed successfully (exit code 0)
```

2. If the errand fails, run the bootstrap errand command again after a few minutes. The bootstrap errand might not work immediately.
3. See that the errand completes successfully in the shell output and continue to Restore the BOSH Configuration below.

   Note: After you complete the bootstrap errand, you might still see instances in the `failing` state. Continue to the next section anyway.

Restore the BOSH Configuration

Caution: If you do not set each of your ignored instances to `unignore`, your instances are never updated in future deploys.

To restore your BOSH configuration to its previous state, this procedure unignores each instance that was previously ignored:

1. For each ignored instance, run:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/INSTANCE-GUID
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.
   - `INSTANCE-GUID` is the GUID of your instance.
2. Redeploy your deployment by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.
3. Verify that all `mysql` instances are in a `running` state by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT instances
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

Bootstrap Manually

If the bootstrap errand is not able to automatically recover the cluster, you need to do the steps manually.

Follow the procedures in the sections below to manually bootstrap your cluster.

> ⚠ **Caution**: The following procedures are prone to user-error and can result in lost data if followed incorrectly. Follow the procedure in Bootstrap with the BOSH Errand above first, and only resort to the manual process if the errand fails to repair the cluster.

Shut Down MySQL

Follow these steps to stop the `galera-init` process for each node in the cluster. For each node, record if the `monit stop` command was successful:

1. SSH into the node using the procedure in Advanced Troubleshooting with the BOSH CLI.
2. To shut down the `mysqld` process on the node, run:

   ```
   monit stop galera-init
   ```

3. Record if the monit command succeeds or exits with an error:
   - **If monit succeeds in stopping `galera-init`,** then you can use monit to restart this node. Follow all the steps below including the steps marked **Monit Restart** but omitting the steps marked **Manual Redeploy**.

   - **If monit exits with the following error,** then you must manually deploy this node:

     ```
     Warning: include files not found '/var/vcap/monit/job/*.monitrc'
     monit: action failed -- There is no service by that name
     ```

     Follow all the steps below including the steps marked **Manual Redeploy** but omitting the steps marked **Monit Restart**.
4. Repeat the steps above for each node in the cluster.

   You cannot bootstrap the cluster unless you have shut down the `mysqld` process on all nodes in the cluster.

Determine which Node to Bootstrap

To avoid losing data, you must bootstrap from a node in the cluster that has the highest transaction sequence number (`seqno`).

For each node in the cluster, to find its `seqno`:

1. SSH into the node using the procedure in Advanced Troubleshooting with the BOSH CLI.
2. Find `seqno` values in the node's Galera state file, `grastate.dat` by running:

   ```
   cat /var/vcap/store/pxc-mysql/grastate.dat | grep 'seqno:'
   ```

3. Do one of the following, based on the last and highest `seqno` value in the `grep` output above:

    - **If `seqno` is a positive integer**, then the node shut down gracefully. Record this number for comparison with the latest `seqno` of other nodes in the cluster.
    - **If `seqno` is `-1`**, then the node crashed or was stopped. Proceed as follows to attempt to recover the `seqno` from the database:

        1. Temporarily start the database and append the last sequence number to its error log by running:

            ```
            $ /var/vcap/jobs/pxc-mysql/bin/get-sequence-number
            ```

        2. The output of the `get-sequence-number` utility will look similar to this:

            ```
            { "cluster_uuid": "1f594c30-a709-11ed-a00e-5330bbda96d3", "seqno":
            4237, "instance_id": "4213a73e-069f-4ac7-b01b-43068ab312b6" }
            ```

        The `seqno` in the above output is `4237`.

4. After you retrieve the `seqno` for all nodes in your cluster, identify the one with the highest `seqno`. If multiple nodes share the same highest `seqno`, and it is not `-1`, you can bootstrap from any of them.

Bootstrap the First Node

After determining the node with the highest `seqno`, do the following to bootstrap the node:

> **Note:** Only run these bootstrap commands on the node with the highest `seqno`. Otherwise the node with the highest `seqno` is unable to join the new cluster unless its data is abandoned. Its `mysqld` process exits with an error.

1. SSH into the node using the procedure in Advanced Troubleshooting with BOSH CLI.
2. Update the node state to trigger its initialization of the cluster by running:

    ```
    echo -n "NEEDS_BOOTSTRAP" > /var/vcap/store/pxc-mysql/state.txt
    ```

3. **Monit Restart:** If in Shut Down MySQL above you successfully used `monit` to shut down your `galera-init` process, then re-launch the `mysqld` process on the new bootstrap node.

    1. Start the `mysqld` process by running:

        ```
        monit start galera-init
        ```

    2. It can take up to ten minutes for `monit` to start the `mysqld` process. To confirm if the `mysqld` process has started successfully, run:

        ```
        watch monit summary
        ```

        If monit succeeds in starting the galera-init process, then the output includes the line `Process 'galera-init' running`.

4. **Manual Redeploy:** If in Shut Down MySQL above you encountered `monit` errors, then redeploy the `mysqld` software to your bootstrap node as follows:

    1. Leave the MySQL SSH login shell and return to your local environment.

    2. Target BOSH on your bootstrap node by instructing it to ignore the other nodes in your cluster. For nodes all nodes except the bootstrap node you identified above, run:

        ```
        bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/M
        bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/N
        ```

        Where `N` and `M` are the numbers of the non-bootstrapped nodes. For example, if you bootstrap node 0, then `M`=1 and `N`=2.

    3. Turn off the BOSH Resurrector by running:

        ```
        bosh update-resurrection off
        ```

    4. Use the BOSH manifest to bootstrap your bootstrap machine by running:

        ```
        bosh -e YOUR-ENV -d YOUR-DEPLOYMENT manifest > /tmp/manifest.yml
        bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy /tmp/manifest.yml
        ```

Restart Remaining Nodes

After the bootstrapped node is running, restart the remaining nodes.

The procedure that you follow for restarting a node, depends on the output you got for that node during Shut Down MySQL above. Do one of the following procedures:

Monit Restart

If in Shut Down MySQL above you successfully used `monit` to shut down your `galera-init` process, then restart the nodes as follows:

1. SSH into the node using the procedure in Advanced Troubleshooting with BOSH CLI.

2. Start the `mysqld` process with `monit` by running:

    ```
    monit start galera-init
    ```

    If the interruptor prevents the node from starting, follow the manual procedure to force the node to rejoin the cluster. See How to manually force a MySQL node to rejoin if a node cannot rejoin the HA (Galera) cluster in the Support knowledge base.

    > **Caution:** Forcing a node to rejoin the cluster is a destructive procedure. Only follow the procedure with the assistance of Support.

3. If the `monit start` command fails, it might be because the node with the highest `seqno` is `mysql/0`.

    In this case:

1. Ensure BOSH ignores updating `mysql/0` by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT ignore mysql/0
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

2. Navigate to Ops Manager in a browser, log in, and click **Apply Changes**.
3. When the deploy finishes, run the following command from the Ops Manager VM:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/0
   ```

   Where:

   - `YOUR-ENV` is the name of your environment.
   - `YOUR-DEPLOYMENT` is the name of your deployment.

Manual Redeploy

If in Shut Down MySQL above you encountered `monit` errors, then restart the nodes as follows:

1. Instruct BOSH to no longer ignore the non-bootstrap nodes in your cluster by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/M
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT unignore mysql/N
   ```

   Where `N` and `M` are the numbers of the non-bootstrapped nodes. For example, if you bootstrap node 0, then `M`=1 and `N`=2.

2. Redeploy software to the other two nodes and have them rejoin the cluster, bootstrapped from the node above by running:

   ```
   bosh -e YOUR-ENV -d YOUR-DEPLOYMENT deploy /tmp/manifest.yml
   ```

   You only need to run this command once to deploy both the nodes that you unignored in the step above.

3. With your redeploys completed, turn the BOSH Resurrector back on by running:

   ```
   bosh -e YOUR-ENV update-resurrection on
   ```

Verify that the Nodes Have Joined the Cluster

The final task is to verify that all the nodes have joined the cluster.

1. SSH into the bootstrap node then run the following command to output the total number of nodes in the cluster:

   ```
   mysql> SHOW STATUS LIKE 'wsrep_cluster_size';
   ```

User Accounts and Communications

In this section:

TAS for VMs User Types

This topic describes the types of VMware Tanzu Application Service for VMs (TAS for VMs) users. It also describes the roles and permissions for TAS for VMs users and who creates and manages their user accounts.

Overview

TAS for VMs users are app developers, managers, and auditors who work within orgs and spaces, the virtual compartments within a deployment where TAS for VMs users can run apps and locally manage their roles and permissions.

A Role-Based Access Control (RBAC) system defines and maintains the different TAS for VMs user roles:

- Org Manager, Org Auditor, Org Billing Manager
- Space Manager, Space Developer, Space Auditor

For more information about TAS for VMs user roles and what actions they can take within the orgs and spaces they belong to, see Orgs, Roles, Spaces, Permissions.

TAS for VMs User Tools

All TAS for VMs users use system tools such as the Cloud Foundry Command Line Interface (cf CLI), Ops Manager Metrics, and Apps Manager, a dashboard for managing TAS for VMs users, orgs, spaces, and apps. Space Developer TAS for VMs users work with their software development tools and the apps deployed on host VMs.

For more information about Apps Manager, see Using Apps Manager.

TAS for VMs User Accounts

When an operator configures TAS for VMs for the first time, they specify one of the following authentication systems for TAS for VMs user accounts:

- Internal authentication, using a new UAA database created for TAS for VMs. This system-wide UAA differs from the Ops Manager internal UAA, which only stores Ops Manager Admin accounts.
- External authentication, through an existing identity provider accessed through SAML or LDAP protocol.

In either case, TAS for VMs user role settings are saved internally in the Cloud Controller Database, separate from the internal or external user store.

Org and Space Managers then use Apps Manager to invite and manage additional TAS for VMs users within their orgs and spaces. TAS for VMs users with proper permissions can also use the cf CLI to assign user roles. For more information, see Managing User Roles with Apps Manager.

User Types Summary

The table below summarizes TAS for VMs user types, their roles, the tools they use, the System of Record (SOR) that stores their accounts, and what accounts they can provision.

User Type	Available Roles	Tools They Use	Account SOR	Accounts They Can Provision
TAS for VMs Users	• UAA Administrator • Org Manager • Org Auditor • Org Billing Manager • Space Manager • Space Developer • Space Auditor	• cf CLI • CAPI • Apps Manager • Ops Manager Metrics • Marketplace	TAS for VMs user store through UAA *or* External store through SAML or LDAP	TAS for VMs users within permitted orgs and spaces, and end users of the app

Creating and managing users with the cf CLI

You can manage users with the cf CLI. On this page, learn how to view users by role, assign roles, and remove roles from a user.

Ops Manager uses role-based access control, with each role granting permissions in either an organization or an application space.

For more information, see Organizations, spaces, roles, and permissions.

About roles

To manage all users, organizations, and roles with the Cloud Foundry Command Line Interface (cf CLI), log in with your admin credentials. In Ops Manager, see the **Credentials** tab in the TAS for VMs tile for the admin name and password.

If the feature flag `set_roles_by_username` is enabled, Org Managers can assign org roles to existing users in their org and Space Managers can assign space roles to existing users in their space. For more information about using feature flags, see the Feature Flags topic.

Creating and deleting users

FUNCTION	COMMAND	EXAMPLE
Create a new user	cf create-user USERNAME PASSWORD	`cf create-user Alice pa55w0rd`
Create a new user, and prompt for password for better security	cf create-user USERNAME --password-prompt	`cf create-user Alice`
Create a new user, specifying LDAP as an external identity provider	cf create-user USERNAME --origin ORIGIN	`cf create-user Aayah ldap`
Create a new user, specifying SAML or OpenID Connect as an external identity provider	cf create-user USERNAME --origin ORIGIN	`cf create-user Aiko provider-alias`
Delete a user	cf delete-user USERNAME	`cf delete-user Alice`

Creating administrator accounts

To create a new administrator account, use the UAA CLI.

Note: The cf CLI cannot create new administrator accounts.

Org and app space roles

A user can have one or more roles. The combination of these roles defines the user’s overall permissions in the org and within specific app spaces in that org.

Org roles

Valid org roles are OrgManager, BillingManager, and OrgAuditor.

FUNCTION	COMMAND	EXAMPLE
View the organizations belonging to an account	cf orgs	`cf orgs`
View all users in an organization by role	cf org-users ORGANIZATION-NAME	`cf org-users my-example-org`
Assign an org role to a user	cf set-org-role USERNAME ORGANIZATION-NAME ROLE	`cf set-org-role Alice my-example-org OrgManager`
Remove an org role from a user	cf unset-org-role USERNAME ORGANIZATION-NAME ROLE	`cf unset-org-role Alice my-example-org OrgManager`

If multiple accounts share a username, `set-org-role` and `unset-org-role` return an error. See Identical Usernames in Multiple Origins for details.

App space roles

Each app space role applies to a specific app space.

Valid app space roles are SpaceManager, SpaceDeveloper, and SpaceAuditor.

FUNCTION	COMMAND	EXAMPLE
View the spaces in an org	cf spaces	`cf spaces`
View all users in a space by role	cf space-users ORGANIZATION-NAME SPACE-NAME	`cf space-users my-example-org development`
Assign a space role to a user	cf set-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	`cf set-space-role Alice my-example-org development SpaceAuditor`
Remove a space role from a user	cf unset-space-role USERNAME ORGANIZATION-NAME SPACE-NAME ROLE	`cf unset-space-role Alice my-example-org development SpaceAuditor`

If multiple accounts share a username, `set-space-role` and `unset-space-role` return an error. See Identical Usernames in Multiple Origins for details.

Creating and managing users with the UAA CLI (UAAC)

Using the Cloud Foundry User Account and Authentication Command Line Interface (UAAC), you can create users in the UAA server.

The UAAC only creates users in UAA, and does not assign roles in the Cloud Controller database (CCDB). In general, admins create users using the Cloud Foundry Command Line Interface (cf CLI). The cf CLI both creates user records in the UAA and associates them with org and space roles in the CCDB. Before admins can assign roles to the user, the user must log in through Apps Manager or the cf CLI for the user record to populate the CCDB. For more information on creating and managing users, see Creating and Managing Users with the cf CLI.

For more information, see UAA Overview, UAA Sysadmin Guide in the UAA repository on GitHub, and Docs in the UAA repository on GitHub.

UAAC requires Ruby v2.3.1 or later. If you have an earlier version of Ruby installed, install Ruby v2.3.1 or later before using the UAAC.

For more information about which roles can perform various operations in Ops Manager, see User roles in *Orgs, Spaces, Roles, and Permissions*.

Create an admin user

To create an admin user for UAA:

1. Install the UAAC by running:

   ```
   gem install cf-uaac
   ```

2. Target your UAA server by running:

   ```
   uaac target uaa.UAA-DOMAIN
   ```

   Where `UAA-DOMAIN` is the domain of your UAA server.

3. Record the `uaa:admin:client_secret` from your deployment manifest.
4. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin secret you recorded in the previous step.

   UAAC stores the token in `~/.uaac.yml`.

5. Display the users and apps authorized by the UAA server, as well as the permissions granted to each user and app, by running:

   ```
   uaac contexts
   ```

6. In the output from `uaac contexts`, check the `scope` section of the `client_id: admin` user for `scim.write`. The value `scim.write` represents sufficient permissions to create accounts.
7. If the admin user lacks permissions to create accounts, add the permissions:
   1. Add the necessary permissions to the admin user account on the UAA server by running:

      ```
      uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"
      ```

      Where `EXISTING-PERMISSIONS` is the current contents of the `scope` section from the output from `uaac contexts`.

   2. Delete the local token by running:

```
uaac token delete
```

3. Obtain an updated access token from the UAA server by running:

   ```
   uaac token client get admin
   ```

8. Create an admin user by running:

   ```
   uaac user add NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD --emails NEW-ADMIN-EMAIL
   ```

   Where:

   - `NEW-ADMIN-USERNAME` is the username you want to give the admin user.
   - `NEW-ADMIN-PASSWORD` is the password you want to give the admin user.
   - `NEW-ADMIN-EMAIL` is the email address of the admin user.

9. Add the new admin user to the groups `cloud_controller.admin`, `uaa.admin`, `scim.read`, and `scim.write` by running:

   ```
   uaac member add GROUP NEW-ADMIN-USERNAME
   ```

   Where:

   - `GROUP` is the name of the group to which you want to add the new admin user.
   - `NEW-ADMIN-USERNAME`is the username of the new admin user.

Create an admin read-only user

The admin read-only account can view but not modify almost all Cloud Controller API resources. The admin read-only account cannot view process `stats` or `logs`.

To create an admin read-only user account:

1. Obtain the credentials of the admin client you created in Create an Admin User, or see the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin secret you recorded in the previous step.

   UAAC stores the token in `~/.uaac.yml`.
3. Create an admin read-only user by running:

   ```
   uaac user add NEW-ADMIN-RO-USERNAME -p NEW-ADMIN-RO-PASSWORD --emails NEW-ADMIN
   -RO-EMAIL
   ```

   Where:

  - NEW-ADMIN-RO-USERNAME is the username you want to give the admin read-only user.
  - NEW-ADMIN-RO-PASSWORD is the password you want to give the admin read-only user.
  - NEW-ADMIN-RO-EMAIL is the email address of the admin read-only user.

4. Add the new admin user to the groups `cloud_controller.admin_read_only` and `scim.read` by running:

   ```
   uaac member add GROUP NEW-ADMIN-RO-USERNAME
   ```

   Where:

   - GROUP is the name of the group to which you want to add the new admin read-only user.
   - NEW-ADMIN-RO-USERNAMEis the username of the new admin read-only user.

Create a global auditor

The global auditor account has read-only access to almost all Cloud Controller API resources but cannot access secret data such as environment variables. The global auditor account cannot view process `stats` or `logs`.

To create a global auditor account:

1. Obtain the credentials of the admin client you created in Create an admin user, or see the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where ADMIN-CLIENT-SECRET is the admin secret you recorded in the previous step.

   UAAC stores the token in `~/.uaac.yml`.
3. Create a global auditor user by running:

   ```
   uaac user add NEW-GLOBAL-AUDITOR-USERNAME -p NEW-GLOBAL-AUDITOR-PASSWORD --emails NEW-GLOBAL-AUDITOR-EMAIL
   ```

   Where:

   - NEW-GLOBAL-AUDITOR-USERNAME is the username you want to give the admin read-only user.
   - NEW-GLOBAL-AUDITOR-PASSWORD is the password you want to give the admin read-only user.
   - NEW-GLOBAL-AUDITOR-EMAIL is the email address of the admin read-only user.
4. To ensure that the `cloud_controller.global_auditor` group exists, run:

   ```
   uaac group add cloud_controller.global_auditor
   ```

5. Add the new global auditor user to the `cloud_controller.global_auditor` group by running:

   ```
   uaac member add GROUP NEW-GLOBAL-AUDITOR-USERNAME
   ```

   Where:

   - `GROUP` is the name of the group to which you want to add the new global auditor user.
   - `NEW-GLOBAL-AUDITOR-USERNAME` is the username of the new global auditor user.

Grant admin permissions to an external group (SAML, LDAP, or OIDC)

To grant all users under an external group admin permissions:

1. Obtain the credentials of the admin client you created in Create an Admin User, or see the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin secret you recorded in the previous step.

   UAAC stores the token in `~/.uaac.yml`.
3. Follow the procedure that corresponds to your use case:
   - Grant Admin Permissions for LDAP
   - Grant Admin Permissions for SAML and OIDC

> **Note:** The UAA does not grant scopes for users in external groups until the next time the user logs in. This means that users granted scopes from external group mappings must log out from Ops Manager and log back in before their new scope takes effect.

Grant admin permissions for LDAP

To grant admin permissions to all users under the mapped LDAP group:

1. Run:

   ```
   uaac group map --name scim.read "GROUP-DISTINGUISHED-NAME"
   ```

   Where `GROUP-DISTINGUISHED-NAME` is the name of the LDAP group.
2. Run:

```
uaac group map --name scim.write "GROUP-DISTINGUISHED-NAME"
```

Where `GROUP-DISTINGUISHED-NAME` is the name of the LDAP group.

3. Run:

```
uaac group map --name cloud_controller.admin "GROUP-DISTINGUISHED-NAME"
```

Where `GROUP-DISTINGUISHED-NAME` is the name of the LDAP group.

Grant admin permissions for SAML and OIDC

To grant admin permissions to all users under the mapped SAML or OIDC group:

1. Retrieve the name of your SAML provider by navigating to the TAS for VMs tile on the Ops Manager Installation Dashboard, selecting **Authentication and Enterprise SSO**, and recording the value under **Provider name**. For more information about configuring TAS for VMs for a SAML identity provider, see the Configure TAS for VMs to Use a SAML Identity Provider section of the *Configuring Authentication and Enterprise SSO for TAS for VMs* topic.
2. Grant all users under the mapped SAML or OIDC group admin permissions by running:

```
uaac group map --name scim.read "GROUP-NAME" --origin PROVIDER-NAME
uaac group map --name scim.write "GROUP-NAME" --origin PROVIDER-NAME
uaac group map --name cloud_controller.admin "GROUP-NAME" --origin PROVIDER-NAM
E
```

Where:

- `GROUP-NAME` is the name of the SAML or OIDC group.
- `PROVIDER-NAME` is the name of your SAML or OIDC IDP.

Note: For OIDC, make sure you configure the IDP's attribute mappings and map `external_groups` to the groups field in the OIDC ID Token issued by the IDP.

Create users

To create new users:

1. Obtain the credentials of the admin client you created in Create an Admin User, or see the `uaa: scim` section of your deployment manifest for the username and password of an admin user.
2. Log in to your UAA API by running:

```
cf login -u ADMIN-USERNAME -p ADMIN-PASSWORD
```

Where:

- `ADMIN-USERNAME` is the username of the admin user.
- `ADMIN-PASSWORD` is the password of the admin user.

3. Create a new user by running:

   ```
   cf create-user NEW-USERNAME NEW-USER-PASSWORD
   ```

   Where:

   - `NEW-USERNAME` is the username you give the new user.
   - `NEW-USER-PASSWORD` is the password you give the new user.

   As of cf CLI v7, you can use the `--password-prompt` option to prompt for the password. This enhances security by removing the requirement to type the password on the command line.

Change passwords

To change the password of a user:

1. Obtain the credentials of the admin client you created in Create an Admin User, or see the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin secret you recorded in the previous step.

   UAAC stores the token in `~/.uaac.yml`.
3. Display the users and apps authorized by the UAA server, as well as the permissions granted to each user and app, by running:

   ```
   uaac contexts
   ```
4. In the output from `uaac contexts`, check the `scope` section of the `client_id: admin` user for `password.write`. The value `password.write` represents sufficient permissions to change passwords.
5. If the admin user lacks permissions to change passwords, add the permissions:
   1. Add the necessary permissions to the admin user account on the UAA server by running:

      ```
      uaac client update admin --authorities "EXISTING-PERMISSIONS password.write"
      ```

      Where `EXISTING-PERMISSIONS` is the current contents of the `scope` section from the output from `uaac contexts`.
   2. Delete the local token by running:

      ```
      uaac token delete
      ```

    3. Obtain an updated access token from the UAA server by running:

       ```
       uaac token client get admin
       ```

6. Change an existing user password to a temporary password by running:

   ```
   uaac password set USERNAME -p TEMP-PASSWORD
   ```

   Where:

   - `USERNAME` is the username of the user whose password you want to change.
   - `TEMP-PASSWORD` is the temporary password you set.

7. Provide the temporary password to the user and instruct the user to run:

   ```
   cf target api.UAA-DOMAIN
   cf login -u USERNAME -p TEMP-PASSWORD
   cf passwd
   ```

   Where:

   - `UAA-DOMAIN` is the domain of your UAA server.
   - `USERNAME` is the username of the user.
   - `TEMP-PASSWORD` is the temporary password you provided the user.

   To configure the password policy, see [Configuring UAA Password Policy](https://docs.vmware.com/en/VMware-Tanzu-Application-Service/2.12/tas-for-vms/pw-policy.html).

Retrieve user email addresses

Some Ops Manager components, like Cloud Controller, only use GUIDs for user identification. You can use UAA to retrieve the emails of your Ops Manager instance users either as a list or, for a specific user, with that user's GUID.

1. Target your UAA server by running:

   ```
   uaac target uaa.UAA-DOMAIN
   ```

   Where `UAA-DOMAIN` is the domain of your UAA server.

2. Record the `uaa:admin:client_secret` from your deployment manifest.

3. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin secret you recorded in the previous step.

UAAC stores the token in `~/.uaac.yml`.

4. Display the users and apps authorized by the UAA server, as well as the permissions granted to each user and app, by running:

   ```
   uaac contexts
   ```

5. In the output from `uaac contexts`, check the `scope` section of the `client_id: admin` user for `scim.write`. The value `scim.write` represents sufficient permissions to query the UAA server for user information.

6. If the admin user lacks permissions to change passwords, add the permissions:

   1. Add the necessary permissions to the admin user account on the UAA server by running:

      ```
      uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"
      ```

      Where `EXISTING-PERMISSIONS` is the current contents of the `scope` section from the output from `uaac contexts`.

   2. Delete the local token by running:

      ```
      uaac token delete
      ```

   3. Obtain an updated access token from the UAA server by running:

      ```
      uaac token client get admin
      ```

7. To list your Ops Manager instance users, run:

   ```
   uaac users
   ```

   By default, the `uaac users` command returns information about each user account, including GUID, name, permission groups, activity status, and metadata. To limit the output of `uaac users` to email addresses, run:

   ```
   uaac users --attributes emails
   ```

8. To retrieve a specific user's email address, run:

   ```
   uaac users "id eq GUID" --attributes emails
   ```

   Where `GUID` is the GUID of a specific user.

Creating New TAS for VMs User Accounts

When you first deploy VMware Tanzu Application Service for VMs (TAS for VMs), there is only one user: an administrator. At this point, you can add accounts for new users who can then push apps using the Cloud Foundry Command Line Interface (cf CLI).

How to add users depends on whether or not you have SMTP enabled, as described in the options below.

Option 1: Adding New Users when SMTP Is Enabled

If you have enabled SMTP, your users can sign up for accounts and create their own orgs. They do this using the Ops Manager Apps Manager, a self-service tool for managing organizations, users, app, and app spaces.

Instruct users to complete the following steps to log in and get started using Apps Manager.

1. Browse to `apps.YOUR-SYSTEM-DOMAIN`. See the **Domains** pane in the TAS for VMs tile to locate your system domain.
2. Select **Create an Account**.
3. Enter your email address and click **Create an Account**. When your account is ready, Apps Manager sends you a new account email.
4. When you receive the new account email, follow the link in the email to complete your registration.
5. Select your organization name.

You now have access to the Apps Manager. For more information about using Apps Manager, see the Apps Manager documentation.

Option 2: Adding New Users when SMTP Is Not Enabled

If you have not enabled SMTP, only an administrator can create new users, and there is no self-service facility for users to sign up for accounts or create orgs.

The administrator creates users with the cf CLI. For more information, see Creating and Managing Users with the cf CLI.

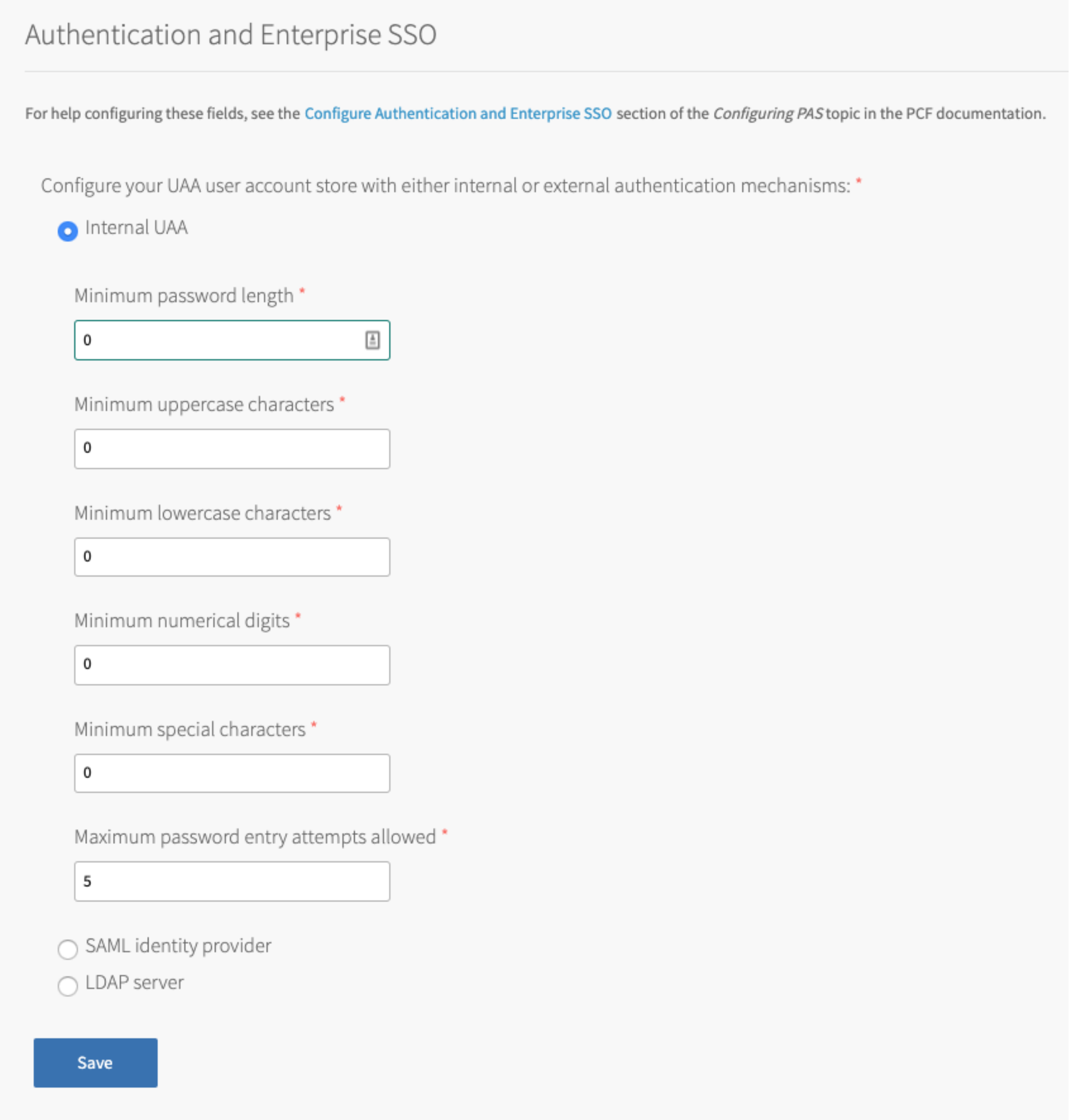
Configuring UAA Password Policy

If your VMware Tanzu Application Service for VMs (TAS for VMs) deployment uses the internal user store for authentication, you can configure its password policy within the TAS for VMs tile.

Open the Internal UAA Configuration

To open the internal UAA configuration for your deployment:

1. In a browser, go to the fully qualified domain name (FQDN) of your Ops Manager and log in.
2. Click the **TAS for VMs** tile.
3. Select **Authentication and Enterprise SSO**.



Authentication and Enterprise SSO

For help configuring these fields, see the Configure Authentication and Enterprise SSO section of the *Configuring PAS* topic in the PCF documentation.

Configure your UAA user account store with either internal or external authentication mechanisms: *

● Internal UAA

Minimum password length *
0

Minimum uppercase characters *
0

Minimum lowercase characters *
0

Minimum numerical digits *
0

Minimum special characters *
0

Maximum password entry attempts allowed *
5

○ SAML identity provider

○ LDAP server

4. Confirm that the **Internal UAA** option is selected.

Set Password Requirements and Entry Attempts

To set password requirements and the maximum password entry attempts allowed:

1. For **Minimum password length**, enter the minimum number of characters for a valid password.
2. For **Minimum uppercase characters**, enter the minimum number of uppercase characters required for a valid password.
3. For **Minimum lowercase characters**, enter the minimum number of lowercase characters required for a valid password.
4. For **Minimum numerical digits**, enter the minimum number of digits required for a valid password.
5. For **Minimum special characters**, enter the minimum number of special characters required for a valid password.

6. For **Maximum password entry attempts allowed**, enter the maximum number of failures allowed to enter a password within a five-minute timespan before the account is locked.

Adding Existing SAML or LDAP Users to a TAS for VMs Deployment

This topic describes the procedure for adding existing SAML or LDAP users to a VMware Tanzu Application Service for VMs (TAS for VMs) deployment enabled with SAML or LDAP.

The following ways exist to add existing SAML or LDAP users to your TAS for VMs deployment:

- Option 1: Import Users in Bulk
- Option 2: Add Users Manually

Prerequisites

To perform the procedures in this topic, you must have:

- Admin access to the Ops Manager Installation Dashboard for your TAS for VMs deployment
- The Cloud Foundry Command Line Interface (cf CLI) v6.23.0 or later

Option 1: Import Users in Bulk

You can import SAML or LDAP users in bulk by using the CF-MGMT Tool. For instructions about installing and using the tool, see the CF-MGMT README.

Option 2: Add Users Manually

The following sections describe how to add existing SAML or LDAP users to your TAS for VMs deployment manually.

Step 1: Create User

To add a SAML or LDAP user:

1. Target the API endpoint for your TAS for VMs deployment by running:

   ```
   cf target https://api.SYSTEM-DOMAIN
   ```

   Where `SYSTEM-DOMAIN` is your system domain.

2. Run:

   ```
   cf login
   ```

3. Provide credentials for an account with the Admin user role. For more information about the Admin user role, see Roles and Permissions in *Orgs, Spaces, Roles, and Permissions*.

4. Create the user in UAA by running:

   ```
   cf create-user USERNAME --origin PROVIDER-NAME
   ```

   Where:

   - USERNAME is the username of the SAML or LDAP user you want to add.
   - PROVIDER-NAME is ldap for an LDAP user, or the name of the SAML provider you provided when configuring Ops Manager for a SAML user.

Step 2: Associate User with Org or Space Role

After creating the SAML or LDAP user, you must associate the user with either an Org or Space role.

For more information about roles, see Roles and Permissions in *Orgs, Spaces, Roles, and Permissions*.

Associate User with Org Role

To associate the SAML or LDAP user with an Org role:

1. Run:

   ```
   cf set-org-role USERNAME ORG ROLE
   ```

   Where:

   - USERNAME is the name of the SAML or LDAP user.
   - ORG is the name of your org.
   - For ROLE, enter one of the following:
     - OrgManager: Org Managers can invite and manage users, select and change plans, and set spending limits.
     - BillingManager: Billing Managers can create and manage the billing account and payment information.
     - OrgAuditor: Org Auditors have read-only access to Org information and reports.

   For example:

   ```
   cf set-org-role j.smith@example.com example-org OrgManager
   ```

Associate User with Space Role

To associate the SAML or LDAP user with a Space role:

1. Run:

   ```
   cf set-space-role USERNAME ORG SPACE ROLE
   ```

   Where:

   - USERNAME is the name of the SAML or LDAP user.
   - ORG is the name of your org.

- SPACE is the name of a space in your org.
- For ROLE, enter one of the following:
  - SpaceManager: Space Managers can invite and manage users, and enable features for a given Space.
  - SpaceDeveloper: Space Developers can create and manage apps and services, and see logs and reports.
  - SpaceAuditor: Space Auditors can view logs, reports, and settings on this Space.

For example:

```
cf set-space-role j.smith@example.com example-org example-space SpaceDe
veloper
```

Configuring App Security Groups for Email Notifications

This topic describes configuring App Security Groups (ASGs) to give network access to the Notifications Service.

Overview

To allow the Notifications Service to have network access, you must create ASGs. Without ASGS, you cannot use the Notifications Service.

For more information, see App Security Groups.

Prerequisite

Before configuring ASGs for the Notifications Service, you must first set up the Notifications Service. To set up the Notifications Service, see Getting Started with the Notifications Service.

Configure Network Connections

The Notifications Service is deployed as a suite of apps to the notifications-with-ui space in the system org. It requires the following outbound network connections:

Destination	Ports	Protocol	Reason
SMTP_SERVER	587 (default)	tcp (default)	This service is used to send out email notifications.
LOAD_BALANCER_IP	80, 443	tcp	This service accesses the load balancer.
ASSIGNED_NETWORK	3306	tcp	This service requires access to internal services. ASSIGNED_NETWORK is the CIDR of the network assigned to this service.

Note: The SMTP server port and protocol are dependent on how you configure your server.

Create a SMTP Server ASG

To create an ASG for your SMTP server:

1. Go to the Ops Manager Installation Dashboard.
2. Click the VMware Tanzu Application Service for VMs (TAS for VMs) tile.
3. Select **Email Notifications**.
4. Record the values in the **SMTP server address** and **SMTP server port** fields.
5. Using the **SMTP server address** you obtained in the previous step, find the IP addresses and protocol of your SMTP server from the service you are using. You might need to contact your service provider for this information.
6. Create a `smtp-server.json` file. For `destination`, you must enter the IP address of your SMTP Server.

```
[
    {
        "protocol": "tcp",
        "destination": SMTP_SERVER_IPS,
        "ports": "587"
    }
]
```

7. Create an ASG called `smtp-server` by running:

```
cf create-security-group smtp-server smtp-server.json
```

Create a Load Balancer ASG

Note: If you already have an ASG set up for a load balancer, you do not need to do this step. To check which groups you have set up, see App Security Groups.

If you are using the internal HAProxy as your load balancer, follow this procedure. If you are using an external load balancer, you must obtain your HAProxy IPs from the service you are using.

To create an ASG for a load balancer:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Record the values in the **HAProxy IPs** field.
5. Create a `load-balancer-https.json` file. For `destination`, use the **HAProxy IPs** you recorded above.

```
[
    {
        "protocol": "tcp",
        "destination": "10.68.196.250",
        "ports": "80,443"
    }
]
```

6. Create an ASG called `load-balancer-https` by running:

   ```
   cf create-security-group load-balancer-https load-balancer-https.json
   ```

Create an Assigned Network ASG

> **Note:** If you use external services, the IP addresses, ports, and protocols depend on the service.

To create an ASG for an assigned network:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Assign AZs and Networks**.
4. Record the network selected in the **Network** dropdown.
5. Return to the Ops Manager Installation Dashboard.
6. Click the BOSH Director tile.
7. Select **Create Networks**.
8. Record the **CIDR** for the network you identified in **Assign AZs and Networks**. Ensure the subnet mask allows the space to access `p-mysql`, `p-rabbitmq`, and `p-redis`.
9. Create a file `assigned-network.json`. For the `destination`, enter the **CIDR** you recorded above.

   ```
   [
       {
           "protocol": "tcp",
           "destination": "10.68.0.0/20",
           "ports": "3306,5672,6379"
       }
   ]
   ```

10. Create an ASG called `assigned-network` by running:

    ```
    cf create-security-group assigned-network assigned-network.json
    ```

Bind the ASGs

After creating your ASGs, you must bind them to the Notifications Service.

To bind your ASGs to the Notifications Service:

1. Target the `system` org by running:

   ```
   cf target -o system
   ```

2. Create a `notifications-with-ui` space by running:

   ```
   cf create-space notifications-with-ui
   ```

3. Bind the ASGs you created in this topic to the `notifications-with-ui` space by running:

   ```
   cf bind-security-group smtp-server system notifications-with-ui
   cf bind-security-group load-balancer-https system notifications-with-ui
   cf bind-security-group assigned-network system notifications-with-ui
   ```

Get started with the Notifications Service in TAS for VMs

You can use the Notifications Service in TAS for VMs to create a client, obtain a token, register notifications, create a custom template, and send notifications.

Prerequisites

Before you can use the Notifications Service you must:

- Install VMware Tanzu Application Service for VMs (TAS for VMs).
- Have `admin` permissions on your TAS for VMs instance. You also must configure Configuring App Security Groups for Email Notifications.
- Install the Cloud Foundry Command Line Interface (cf CLI) and User Account and Authorization Server (UAAC) command line tools.

Create a client and get a token

To interact with the Notifications Service, you must create UAA scopes.

To create UAA scopes:

1. Target your UAA server by running:

   ```
   uaac target uaa.YOUR-DOMAIN
   ```

   Where `YOUR-DOMAIN` is the domain of your UAA server URL.

2. Record the **Admin Client Credentials** from the **UAA** row in the TAS for VMs **Credentials** tab.

3. Authenticate and obtain an access token for the admin client from the UAA server by running:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is the admin client secret.

   UAAC stores the token in `~/.uaac.yml`.

4. Create a `notifications-admin` client with the required scopes by running:

```
uaac client add notifications-admin --authorized_grant_types client_credentials
--authorities \
   notifications.manage,notifications.write,notification_templates.write,notifi
cation_templates.read,critical_notifications.write
```

  - `notifications.write`: Send a notification. For example, you can send notifications to a user, space, or everyone in the system.
  - `notifications.manage`: Update notifications and assign templates for that notification.
  - (Optional) `notification_templates.write`: Create a custom template for a notification.
  - (Optional) `notification_templates.read`: Check which templates are saved in the database.

5. Log in using your newly created client by running:

```
uaac token client get notifications-admin
```

> **Note:** Stay logged in to this client to follow the examples in this topic.

For more information about UAA scopes, see User Account and Authentication (UAA) Server.

Register notifications

> **Note:** To register notifications, you must have the `notifications.manage` scope on the client. To set critical notifications, you must have the `critical_notifications.write` scope.

You must register a notification before sending it. Using the token `notifications-admin` from the previous step, the following example registers two notifications with the following properties:

```
uaac curl https://notifications.user.example.com/notifications -X PUT --data '{  "sour
ce_name": "Cloud Ops Team",
  "notifications": {
     "system-going-down": {"critical": true, "description": "Cloud going down" },
     "system-up": { "critical": true, "description": "Cloud back up" }
     }
 }'
```

- `source_name` has “Cloud Ops Team” set as the description.
- `system-going-down` and `system-up` are the notifications set.
- `system-going-down` and `system-up` are made `critical`, so no users can unsubscribe from that notification.

Create a custom template

> **Note:** To view a list of templates, you must have the `notifications_templates.read` scope. To create a custom template, you must have the `notification_templates.write` scope.

A template is made up of a name, a subject, a text representation of the template you are sending for mail clients that do not support HTML, and an HTML version of the template.

The system provides a default template for all notifications, but you can create a custom template by running:

```
uaac curl https://notifications.user.example.com/templates -X POST --data \
'{"name":"site-maintenance","subject":"Maintenance: {{.Subject}}","text":"The site has
gone down for maintenance. More information to follow {{.Text}}","html":"<p>The site h
as gone down for maintenance. More information to follow {{.HTML}}"}'
```

Variables that take the form `{{.}}` interpolate data provided in the send step before a notification is sent. Data that you can insert into a template during the send step include `{{.Text}}`, `{{.HTML}}`, and `{{.Subject}}`.

This `curl` command returns a unique template ID that can be used in subsequent calls to refer to your custom template. The result looks similar to:

```
{"template-id": "E3710280-954B-4147-B7E2-AF5BF62772B5"}
```

Check all of your saved templates by running:

```
uaac curl https://notifications.user.example.com/templates -X GET
```

Associate a custom template with a notification

In this example, the `system-going-down` notification belonging to the `notifications-admin` client is associated with the template ID `E3710280-954B-4147-B7E2-AF5BF62772B5`. This is the template ID of the template we created in the previous section.

Associating a template with a notification requires the `notifications.manage` scope.

```
uaac curl https://notifications.user.example.com/clients/notifications-admin/notificat
ions/system-going-down/template \
-X PUT --data '{"template": "E3710280-954B-4147-B7E2-AF5BF62772B5"}'
```

Any notification that does not have a custom template applied, such as `system-up`, defaults to a system-provided template.

Send a notification

> **Note:** To send a critical notification, you must have the `critical_notifications.write` scope. To send a non-critical notification, you must have the `notifications_write` scope.

You can send a notification to the following recipients:

- A user
- A space
- An org
- All users in the system
- A UAA scope
- An email address

For more information, see Notifications V1 Documentation in the Notifications repository on GitHub.

The following example command sends the `system-going-down` notification described above to all users in the system:

```
uaac curl https://notifications.user.example.com/everyone -X POST --data \
'{"kind_id":"system-going-down","text":"The system is going down while we upgrade our
storage","html":"<strong>THE SYSTEM IS DOWN</strong><p>The system is going down while
we upgrade our storage</p>","subject":"Upgrade to Storage","reply_to":"no-reply@exampl
e.com"}'
```

UAA

In this section:

- User Account and Authentication (UAA) Server
- UAA Concepts
- Identity Providers in UAA
- UAA Performance Metrics

User Account and Authentication server

This topic tells you about the User Account and Authentication (UAA) Server, the identity management service for VMware Tanzu Application Service for VMs (TAS for VMs).

The primary role of UAA is as an OAuth2 provider, that issues tokens for client apps to use when they act on behalf of TAS for VMs users. In collaboration with the login server, UAA can authenticate users with their TAS for VMs credentials, and can act as an SSO service using those, or other, credentials.

UAA has endpoints for managing user accounts, registering OAuth2 clients, and various other management functions.

Different runtimes and services use separate UAA instances. TAS for VMs has two UAA instances by default: one for BOSH Director, used to bootstrap the rest of the TAS for VMs deployment; and one for the BOSH deployment, used as a shared resource by all apps that require user authentication. This is the minimum number of UAA instances TAS for VMs must have. Other runtimes and services also have UAA instances. These instances are separate from each other. If you log into one runtime or service, you are not also logged into other runtimes and services that authenticate using UAA. You must log in to each runtime or service separately.

You can deploy UAA locally or to TAS for VMs.

Deploying UAA Locally

To deploy and run UAA locally:

1. In a terminal window, clone the UAA GitHub repository by running:

   ```
   git clone git://github.com/cloudfoundry/uaa.git
   ```

2. Go to the directory where you cloned the UAA GitHub repository.
3. To build and run all the components that comprise UAA and the example programs `uaa`, `samples/api`, and `samples/app`, run:

   ```
   ./gradlew run
   ```

4. If successful, the three apps run together on a single instance of Tomcat listening on port 8080, with endpoints `/uaa`, `/app`, and `/api`. Tomcat is a library of Java code used by UAA to create its features. For more information, see the Apache Tomcat website.

Using Local UAA

To access and use a locally deployed UAA server:

1. Run the UAA server as described in Deploy Locally.
2. Open another terminal window. From the project base directory, run `curl localhost:8080/uaa/info -H "Accept: application/json"` to confirm the UAA is running. You should see basic information about the system. For example:

   ```
   $ curl localhost:8080/uaa/info -H "Accept: application/json"
   {
     "app": {
       "version": "4.19.0"
     },
     "links": {
       "uaa": "http://localhost:8080/uaa",
       "passwd": "/forgot_password",
       "login": "http://localhost:8080/uaa",
       "register": "/create_account"
     },
     "zone_name": "uaa",
     "entityID": "cloudfoundry-saml-login",
     "commit_id": "af93628",
     "idpDefinitions": {},
     "prompts": {
       "username": [
         "text",
         "Email"
       ],
   ```

```
    "password": [
      "password",
      "Password"
    ]
  },
  "timestamp": "2018-05-25T15:34:31-0700"
}
```

3. To install the TAS for VMs UAA Command Line Client (UAAC) Ruby gem, run:

```
gem install cf-uaac
```

4. To target the local UAA server endpoint, run:

```
uaac target http://localhost:8080/uaa
```

5. Run `uaac token client get CLIENT_NAME -s CLIENT_SECRET` to obtain an access token. Replace `CLIENT_NAME` and `CLIENT_SECRET` with actual values. For example, when starting up the UAA locally for development, there should be a predefined admin client you can use:

```
uaac token client get admin -s adminsecret
```

If you run the command without `-s CLIENT_SECRET`, UAAC shows an interactive prompt where you must create the client secret value. The `uaac token client get` command requests an access token from the server using the OAuth2 client credentials grant type. For more information about the OAuth2 client credentials, see Client Credentials in the OAuth 2.0 Authorization Framework.

6. View your UAAC token context. When UAAC obtains a token, the token and other metadata is stored in the `~/.uaac.yml` file on your local machine. To view the token you have obtained, run `uaac context`. For example:

```
$ uaac context
[0]*[http://localhost:8080/uaa]

[0]*[admin]
  client_id: admin
  access_token: eyJhbGciOiJIUzI1NiIsImtpZCI6ImxlZ2FjeS10b2tlbi1rZXkiLCJ
0eXAiOiJKV1QifQ.eyJqdGkiOiIxOWYyNWU2Y2E5Y2M0ZWIyYTdmNTAxNmU0NDFjZThkNCI
sInN1YiI6ImFkbWluIiwiYXV0aG9yaXRpZXMiOlsiY2xpZW50cy5yZWFkIiwiY2xpZW50cy
5zZWNyZXQiLCJjbGllbnRzLndyaXRlIiwidWFhLmFkbWluIiwiY2xpZW50cy5hZG1pbiIsI
nNjaW0ud3JpdGUiLCJzY2ltLnJlYWQiXSwic2NvcGUiOlsiY2xpZW50cy5yZWFkIiwiY2xp
ZW50cy5zZWNyZXQiLCJjbGllbnRzLndyaXRlIiwidWFhLmFkbWluIiwiY2xpZW50cy5hZG1
pbiIsInNjaW0ud3JpdGUiLCJzY2ltLnJlYWQiXSwiY2xpZW50X2lkIjoiYWRtaW4iLCJjaW
QiOiJhZG1pbiIsImF6cCI6ImFkbWluIiwiZ3JhbnRfdHlwZSI6ImNsaWVudF9jcmVkZW50a
WFscyIsInJldl9zaWciOiIyNjcxOTlkMSIsImlhdCI6MTUyODIzMjAxNywiZXhwIjoxNTI4
Mjc1MjE3LCJpc3MiOiJodHRwOi8vbG9jYWxob3N0OjgwODAvdWFhL29hdXRoL3Rva2VuIiw
iemlkIjoidWFhIiwiYXVkIjpbInNjaW0iLCJjbGllbnRzIiwidWFhIiwiYWRtaW4iXX0.L2
cn6HqLQAEyqTrYYkL9Al_8JyfwB330er7DshUb9wg
  token_type: bearer
  expires_in: 43199
  scope: clients.read clients.secret clients.write uaa.admin clients.ad
min scim.write scim.read
  jti: 19f25e6ca9cc4eb2a7f5016e441ce8d4
```

Copy the access token from this output for the next step.

7. Run `uaac token decode ACCESS-TOKEN-VALUE` to view information in the token, which is encoded using the JSON Web Token (JWT) format. Replace `ACCESS-TOKEN-VALUE` with your access token, copied from the `uaac context` output. The UAAC should display all the claims inside the token body. For example:

```
$ uaac token decode eyJhbGciOiJIUzI1NiIsImtpZCI6ImxlZ2FjeS10b2tlbi1rZXk
iLCJ0eXAiOiJKV1QifQ.eyJqdGkiOiIxOWYyNWU2Y2E5Y2M0ZWIyYTdmNTAxNmU0NDFjZTh
kNCIsInN1YiI6ImFkbWluIiwiYXV0aG9yaXRpZXMiOlsiY2xpZW50cy5yZWFkIiwiY2xpZW
50cy5zZWNyZXQiLCJjbGllbnRzLndyaXRlIiwidWFhLmFkbWluIiwiY2xpZW50cy5hZG1pb
iIsInNjaW0ud3JpdGUiLCJzY2ltLnJlYWQiXSwic2NvcGUiOlsiY2xpZW50cy5yZWFkIiwi
Y2xpZW50cy5zZWNyZXQiLCJjbGllbnRzLndyaXRlIiwidWFhLmFkbWluIiwiY2xpZW50cy5
hZG1pbiIsInNjaW0ud3JpdGUiLCJzY2ltLnJlYWQiXSwiY2xpZW50X2lkIjoiYWRtaW4iLC
JjaWQiOiJhZG1pbiIsImF6cCI6ImFkbWluIiwiZ3JhbnRfdHlwZSI6ImNsaWVudF9jcmVkZ
W50aWFscyIsInJldl9zaWciOiIyNjcxOTlkMSIsImlhdCI6MTUyODIzMjAxNywiZXhwIjox
NTI4Mjc1MjE3LCJpc3MiOiJodHRwOi8vbG9jYWxob3N0OjgwODAvdWFhL29hdXRoL3Rva2V
uIiwiemlkIjoidWFhIiwiYXVkIjpbInNjaW0iLCJjbGllbnRzIiwidWFhIiwiYWRtaW4iXX
0.L2cn6HqLQAEyqTrYYkL9Al_8JyfwB330er7DshUb9wg

Note: no key given to validate token signature

  jti: 19f25e6ca9cc4eb2a7f5016e441ce8d4
  sub: admin
  authorities: clients.read clients.secret clients.write uaa.admin clie
nts.admin scim.write scim.read
  scope: clients.read clients.secret clients.write uaa.admin clients.ad
min scim.write scim.read
  client_id: admin
  cid: admin
  azp: admin
  grant_type: client_credentials
  rev_sig: 267199d1
  iat: 1528232017
  exp: 1528275217
  iss: http://localhost:8080/uaa/oauth/token
  zid: uaa
  aud: scim clients uaa admin
```

Deploying UAA to TAS for VMs

To build the UAA as an app and push it to TAS for VMs using the Cloud Foundry Command Line Interface (cf CLI):

1. Clone the UAA GitHub repository by running:

```
git clone git://github.com/cloudfoundry/uaa.git
```

2. Go to the directory where you cloned the UAA GitHub repository.
3. Build the UAA as a WAR file by running:

   ```
   ./gradlew :cloudfoundry-identity-uaa:war
   ```

4. Run the cf CLI `cf push APP-NAME -m 512M -p PATH-TO-WAR-FILE --no-start` command to push the app to TAS for VMs. Replace `APP-NAME` with a name for your UAA app, and `PATH-TO-WAR-FILE` with the path to the WAR file you created in the previous step. For example:

   ```
   cf push MYUAA -m 512M -p uaa/build/libs/cloudfoundry-identity-uaa-1.8.0.war --n
   o-start
   ```

5. Run `cf set-env APP-NAME SPRING_PROFILES_ACTIVE default` to set the `SPRING_PROFILES_ACTIVE` environment variable with the value `default`. Replace `APP-NAME` with the name of your app that you used in the previous step. For example:

   ```
   cf set-env MYUAA SPRING_PROFILES_ACTIVE default
   ```

6. Run `cf start APP-NAME` to start your app. Replace `APP-NAME` with the name of your app. For example:

   ```
   cf start MYUAA
   ```

Using Remote UAA

You use a UAA server that you pushed as an app to TAS for VMs in a similar way to one you run locally. You do not need app token encoding because you do not have the client secret.

To access and use a UAA server that you pushed as an app to TAS for VMs:

1. Follow the procedure in Deploy UAA to TAS for VMs.
2. From the project base directory, run `curl -H "Accept: application/json" APP-FQDN/login` to query the external login endpoint about the system. Replace `APP-FQDN` with the fully qualified domain name (FQDN) of your app. For example:

   ```
   $ curl -H "Accept: application/json" uaa.example.org/login
   {
     "timestamp":"2014-09-15T18:25:04+0000",
     "app":{"version":"1.8.3"},
     "commit_id":"git-metadata-not-found",
     "prompts":{"username":["text","Email"],
         "password":["password","Password"]
     }
   }
   ```

3. Install the UAA Command Line Client (UAAC) Ruby gem by running:

   ```
   gem install cf-uaac
   ```

4. Target the remote UAA Server endpoint by running:

```
uaac target APP-FQDN
```

Where `APP-FQDN` is the FQDN of your app.

5. Log in by running:

```
uaac token get USERNAME PASSWORD
```

Where:

- `USERNAME` is your username.
- `PASSWORD` is your password. If you do not specify a username and password, the UAAC prompts you to supply them.

The `uaac token client get` command authenticates and obtains an access token from the server using the OAuth2 implicit grant, similar to the approach intended for a standalone client like the cf CLI.

Running integration tests

When you run integration tests, it makes sure that the UAA instance is running properly.

To run integrations tests, run:

```
./gradlew integrationTest
```

This command starts a UAA server running in a local Apache Tomcat instance. By default, the service URL is set to `http://localhost:8080/uaa`.

You can set the environment variable `CLOUD_FOUNDRY_CONFIG_PATH` to a directory containing a `uaa.yml` file where you change the URLs used in the tests, and where you can set the UAA server context root.

Customizing your YAML configuration

A custom YAML configuration gives integration tests access to your UAA instance by sharing credentials in YAML file.

To create a custom YAML configuration:

1. Create a `uaa.yml` file in the following format:

```
uaa:
  host: UAA-HOSTNAME
  test:
    username: USERNAME
    password: PASSWORD
    email: EMAIL-ADDRESS
```

Where:

- `UAA-HOSTNAME` is the FQDN of UAA app. For example, `uaa.example.org`.
- `USERNAME` is a valid username. For example, `dev@example.org`.

- `PASSWORD` is the password for the above username.
- `EMAIL-ADDRESS` is the email address for the above user. Example: `dev@example.org`.

2. From the `uaa/uaa` directory, run:

   ```
   CLOUD_FOUNDRY_CONFIG_PATH=/tmp ./gradlew test
   ```

The web app looks for a YAML file in the following locations when it starts, with later entries overriding earlier ones:

```
classpath:uaa.yml
file:${CLOUD_FOUNDRY_CONFIG_PATH}/uaa.yml
file:${UAA_CONFIG_FILE}
${UAA_CONFIG_URL}
```

Testing with PostgreSQL or MySQL

The default UAA unit tests, `./gradlew test`, use a HyperSQL database.

To use a different database management system:

1. Create a `uaa.yml` file containing `spring_profiles: default,OTHER-DBMS` in the `src/main/resources/` directory. Replace `OTHER-DBMS` with the name of the other database management system to use.
2. Run the unit tests using your specified database management system instead of a HyperSQL database:

   ```
   echo "spring_profiles: default,OTHER-DBMS" > src/main/resources/uaa.yml
   ./gradlew test integrationTest
   ```

   Where `OTHER-DBMS` is the name of the database management system you specified in your `uaa.yml` file.

You can find the database configuration for the common and scim modules at `common/src/test/resources/(mysql|postgresql).properties` and `scim/src/test/resources/(mysql|postgresql).properties`.

UAA projects

The following UAA projects exist:

- `common`: A module containing a JAR with all the business logic. `common` is used in the web apps listed below.
- `uaa`: The UAA server. `uaa` provides an authentication service and authorized delegation for back-end services and apps by issuing OAuth2 access tokens.
- `api`: A sample OAuth2 resource service that returns a mock list of deployed apps. `api` provides resources that other apps might want to access on behalf of the resource owner.
- `app`: A sample user app that uses both `api` and `uaa`. `app` is a web app that requires single sign-on and access to the `api` service on behalf of users.

- `scim`: The System for Cross-domain Identity Management (SCIM) user management module used by UAA. For more information about SCIM, see Managing Users & Groups with SCIM.

UAA server

The authentication service, `uaa`, is a Spring MVC web app. You can deploy it in Tomcat or your container of choice, or execute `./gradlew run` to run it directly from the `uaa` directory in the source tree. When run with Gradle, `uaa` listens on port 8080 and has the URL `http://localhost:8080/uaa`.

The UAA server supports the APIs defined in the UAA-APIs document which include:

- The OAuth2 `/authorize` and `/token` endpoints.
- A `/login_info` endpoint to allow querying for required login prompts.
- An `/introspect` endpoint to allow resource servers to obtain information about an access token submitted by an OAuth2 client.
- A SCIM user-provisioning endpoint.
- OpenID Connect endpoints `/userinfo` and `/check_id` to support authentication.

Command-line clients can perform authentication by submitting credentials directly to the `/authorize` endpoint.

An `ImplicitAccessTokenProvider` exists in Spring Security OAuth to use if your client is Java.

By default, `uaa` launches with a context root `/uaa`.

Configuration

A `uaa.yml` file exists in the app. This file provides defaults to the placeholders in the Spring XML.

You can override any occurrences of `${placeholder.name}` in the XML by adding it to your `uaa.yml` file, or by providing a System property with the same name, `-D`, to your Java virtual machine.

All passwords and client secrets in the configuration files are in plain text, but are inserted into the UAA database encrypted with BCrypt.

User account data

The default uses an in-memory relational database management system (RDBMS) user store, pre-populated with a single test user `marissa` with the password `koala`.

To use PostgreSQL for user data, you must activate the Spring profile `postgresql`.

You can configure the active profiles in your `uaa.yml` file using `spring_profiles: postgresql,default`.

To use PostgreSQL instead of HyperSQL, run:

```
echo "spring_profiles: postgresql,default" > src/main/resources/uaa.yml
./gradlew run
```

To bootstrap a microcloud-type environment, you need an admin client. During the bootstrapping process, the database creates a database initializer component that inserts an admin client.

If the default profile is active, the bootstrapping process creates a cf CLI client so that the Gem login works with no additional configuration required. You can override the default settings and add additional clients in the `uaa.yml` file, as in the example below:

```
oauth:
  clients:
    admin:
      authorized-grant-types: client_credentials
      scope: read,write,password
      authorities: ROLE_CLIENT,ROLE_ADIN
      id: admin
      secret: adminclientsecret
      resource-ids: clients
```

You can use the admin client to create additional clients. You must have a client with read/write access to the `scim` resource to create user accounts. The integration tests handle this automatically by inserting client and user accounts as necessary for the tests.

Sample apps

Two sample apps are included with UAA: `/api` and `/app`.

You can run `/api` and `/app` with `./gradlew run` from the `uaa` root directory. All three apps, `/uaa`, `/api`, and `/app`, are simultaneously deployed.

API sample app

The `api` sample app is an example resource server. It hosts a service that returns a list of mock apps under `/apps`.

App sample app

The `app` sample app is a user interface app, primarily aimed at browsers, that uses OpenID Connect for authentication and OAuth2 for access grants. `app` authenticates with the Auth service, then accesses resources in the API service. You can run `app` with `./gradlew run` from the `uaa` root directory.

The app can operate in multiple different profiles according to the location and presence of the UAA server and the login app. By default, the app looks for a UAA on `localhost:8080/uaa`, but you can change this by setting the `UAA_PROFILE` environment variable or System Property.

The app source code, `samples/app/src/main/resources`, contains multiple properties files pre-configured with different likely locations for those servers. The names of these properties files follow the format `app-UAA_PROFILE.properties`.

The naming convention for the `UAA_PROFILE` is:

- `local`: a localhost deployment
- `vcap`: a `vcap.me` deployment
- `staging`: a staging deployment

Profile names can be hyphenated to indicate multiple contexts. For example, `local-vcap` can be used when the login server is in a different location than the UAA server.

To see all apps, run:

```
GET /app/apps
```

The browser is redirected through a series of authentication and access grant steps.

To see the currently logged-in user details and a selection of attributes from the OpenID provider, run:

```
GET /app
```

Login app

The `login` app is a user interface for authentication. UAA can also authenticate user accounts, but only if it manages them itself and it only provides a basic UI. You can brand and customize the login app for non-native authentication and for more complicated UI flows, like user registration and password reset.

The log in app is itself an OAuth2 endpoint provider, but delegates those features to the UAA server. Therefore, configuration for the log in app consists of locating the UAA through its OAuth2 endpoint URLs and registering the log in app itself as a client of the UAA. The UAA locations have a `login.yml`file. For example, a local `vcap` instance:

```
uaa:
  url: http://uaa.vcap.example.net
  token:
    url: http://uaa.vcap.example.net/oauth/token
  login:
    url: http://uaa.vcap.example.net/login.do
```

You can define the environment variable or Java System property `login_secret` to use a client secret that the app uses when it authenticates itself with UAA. The `login` app is registered by default in UAA only if there are no active Spring profiles. In UAA, the registration is located in the `oauth-clients.xml` config file, as in the example below:

```
id: login
secret: loginsecret
authorized-grant-types: client_credentials
authorities: ROLE_LOGIN
resource-ids: oauth
```

To authenticate with the `login` app, run:

```
GET /login
```

The sample app presents a form login interface for the back end UAA, and an OpenID widget where a user can authenticate using Google or other credentials.

To approve an OAuth2 token grant, run:

```
GET /oauth/authorize?client_id=app&response_type=code...
```

This is the standard OAuth2 authorization endpoint. UAA handles client credentials and all other features in the back end, and the `login` app is used to render the UI.

To obtain the access token, run:

```
POST /oauth/token
```

This is the standard OAuth2 authorization endpoint passed through to UAA.

Scopes

UAA covers multiple scopes of privilege, including access to UAA, access to Cloud Controller, and access to the router. For more information, see OAuth Scopes.

UAA scopes

The following table describes the scopes of privilege in UAA.

Scope	Description
`uaa.user`	This scope indicates that this is a user. It is required in the token if submitting a GET request to the OAuth 2 `/authorize` endpoint.
`uaa.none`	This scope indicates that this client will not be performing actions on behalf of a user.
`uaa.admin`	This scope indicates that this is the superuser.
`scim.write`	This scope gives admin write access to all SCIM endpoints, `/Users`, and `/Groups`.
`scim.read`	This scope gives admin read access to all SCIM endpoints, `/Users`, and `/Groups`.
`scim.create`	This scope gives the ability to create a user with a POST request to the `/Users` endpoint, but not to modify, read, or delete users.
`scim.userids`	This scope is required to convert a username and origin into a user ID and vice versa.
`scim.invite`	This scope is required to participate in invitations using the `/invite_users` endpoint.
`groups.update`	This scope gives the ability to update a group. This ability can also be provided by the broader `scim.write` scope.
`password.write`	This admin scope gives the ability to change a user's password.
`openid`	This scope is required to access the `/userinfo` endpoint. It is intended for OpenID clients.
`idps.read`	This scope gives read access to retrieve identity providers from the `/identity-providers` endpoint.
`idps.write`	This scope gives the ability to create and update identity providers from the `/identity-providers` endpoint.
`clients.admin`	This scope gives the ability to create, modify, and delete clients.
`clients.write`	This scope is required to create and modify clients. The scopes are prefixed with the scope holder's client ID. For example, `id:testclient authorities:client.write` gives the ability to create a client that has scopes with the `testclient.` prefix. Authorities are limited to `uaa.resource`.
`clients.read`	This scope gives the ability to read information about clients.
`clients.secret`	This admin scope is required to change the password of a client.

Scope	Description
`zones.read`	This scope is required to invoke the `/identity-zones` endpoint to read identity zones.
`zones.write`	This scope is required to invoke the `/identity-zones` endpoint to create and update identity zones.
`scim.zones`	This is a limited scope that only allows adding a user to, or removing a user from, zone management groups under the path `/Groups/zones`.
`oauth.approval`	`/approvals endpoint`. This scope is required to approve or reject clients to act on a user's behalf. This is a default scope defined in the `uaa.yml` file.
`oauth.login`	This scope is used to indicate a login app, such as external login servers, can perform trusted operations, such as creating users not authenticated in the UAA.
`approvals.me`	This scope is not currently used.
`uaa.resource`	This scope indicates that this is a resource server, used for the `/introspect` endpoint.
`zones.ZONE-ID.admin`	This scope permits operations in a designated zone, such as creating identity providers or clients in another zone, by authenticating against the default zone. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.read`	This scope permits reading the given identity zone. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.admin`	This scope translates into `clients.admin` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.read`	This scope translates into `clients.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.write`	his scope translates into `clients.write` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.read`	This scope translates into `scim.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.create`	This scope translates into `scim.create` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.clients.scim.write`	This scope translates into `scim.write` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.
`zones.ZONE-ID.idps.read`	This scope translates into `idps.read` after zone switch completes. This scope is used with the `X-Identity-Zone-Id header`.

Cloud Controller scopes

The following table describes the scopes of privilege in Cloud Controller.

Scope	Description
`cloud_controller.read`	This scope gives the ability to read from any Cloud Controller route the token has access to.

Scope	Description
`cloud_controller.write`	This scope gives the ability to post to Cloud Controller routes the token has access to.
`cloud_controller.admin`	This admin scope gives full permissions to Cloud Controller.
`cloud_controller.admin_read_only`	This admin scope gives read permissions to Cloud Controller.
`cloud_controller.global_auditor`	This scope gives read-only access to all Cloud Controller API resources except for secrets such as environment variables.

Routing scopes

The following table describes the scopes of privilege in the router.

Scope	Description
`routing.routes.read`	This scope gives the ability to read the full routing table from the router.
`routing.routes.write`	This scope gives the ability to write the full routing table from the router.
`routing.router_groups.read`	This scope gives the ability to read the full list of routing groups.
`routing.router_groups.write`	This scopes gives the ability to write the full list of routing groups.

Scopes of privilege

The following table describes the scopes of privilege in other TAS for VMs services.

Scope	Description
`doppler.firehose`	This scope gives the ability to read logs from the How the Loggregator Firehose forwards Logs and Metrics endpoint.
`notifications.write`	This scope gives the ability to send notifications through the Getting Started with the Notifications Service.

User Accounts and Authentication concepts

There are six main components of the Cloud Foundry User Accounts and Authentication (UAA) architecture:

- Identity zones
- Subdomains
- Users
- User groups
- Client
- Choosing scopes and authorities

Identity zones

UAA is built to support a multi-tenant architecture. Each tenant is referred to as an *identity zone*.

Identity zones are configured with JSON payloads using a REST API. For examples, see Creating an identity zone in the UAA API documentation.

An identity zone is a logical boundary around the entities UAA manages. The entities in a zone include, but are not limited to:

- Client registrations
- Users
- Groups
- Group mappings
- User approvals
- Identity providers (IDPs) and access
- ID and refresh tokens

Having two identity zones is equivalent to standing up two separate UAA deployments, but using fewer resources. This type of resource management can reduce operational and maintenance overhead.

Subdomains

An identity zone is uniquely identified by a subdomain identifier in UAA. If a UAA deployment is hosted on the URL `https://login.EXAMPLE-CF-DOMAIN.com`, identity zones are hosted as *subdomains* of that same deployment.

For example:

- `zone1`: `https://ZONE1.login.EXAMPLE-CF-DOMAIN.com`
- `zone2`: `https://ZONE2.login.EXAMPLE-CF-DOMAIN.com`

Default zones

A UAA deployment always has one zone referred to as the *default zone*. You configure and bootstrap a default zone using a YAML configuration file.

Users

A *user* is the central domain object of the UAA server. Because UAA acts as both an account store and an authorization server, many different types of information are linked to users and can be accessed through user-centric API calls.

In its capacity as a user account store, UAA can provide unique attributes that describe an individual user, such as email, name, phone number, and group memberships. In addition to these attributes, UAA also tracks some dynamic user metadata, such as the last successful logon time and last updated time.

You can make additional attributes available if UAA is configured to use custom attribute mappings from an external IDP, such as an existing LDAP or SAML provider. For more information about IDP options, see Identity Providers in UAA.

External IDPs are read-only, as are attributes from those providers. Any change to the external user account can be performed on the external IDP directly. These read-only attributes are refreshed each time the user authenticates with the external IDP.

UAA can be used as an authorization server, which allows client apps to interact with resources on a user's behalf using the four standard OAuth2 authorization grant flows for obtaining access tokens:

- Authorization code
- Implicit
- Resource owner password credentials
- Client credentials

A UAA user is the *resource owner* of the OAuth2 protocol. Access tokens issued to the user contain scopes at the intersection of the requesting client's allowed scopes and a user's group memberships.

user.id

A `user.id` is a string used to identify a user in APIs. This universally unique identifier is randomly generated at the time of user creation and does not change. It is guaranteed to be unique across all identity zones in the UAA deployment. The `user.id` is a 128-bit number formatted as a UUID. This is also expressed as a "sub" claim in the tokens generated by UAA.

user.origin

A user in UAA always belongs to a user store with an alias called an `origin`. For example, users that are authenticated by the UAA itself with a username and password have their origin set to the value `uaa`.

The fixed origin values are:

- `uaa` for users that are authenticated by the UAA deployment
- `ldap` for users that are authenticated by the LDAP provider
- {OIDC provider alias} for users authenticated through an OIDC provider
- {SAML provider alias} for users authenticated through a SAML IDP

Users that are authenticated with an external IDP are often referred to as shadow users in UAA. For more information, see Shadow Users.

user.userName

A `user.userName` is a user-readable string that refers to the user, typically an email address. The user enters their username when authenticating against UAA.

If the user authenticates against an external IDP, the username is transferred from that IDP to the shadow user in UAA. An individual user can be uniquely identified by the combination of the username and the origin values.

The username alone is not a unique value. Because usernames can change, UAA provides user IDs as unchanging references to a single user. For more information, see user.id.

Users that create accounts through the UAA UI use their email address as their username. The administrative API can create user accounts that specify arbitrary usernames.

For external IDPs, the username is mapped from the assertion received by UAA.

- **SAML:** UAA retrieve the username from the `nameID` claim. For example:

  ```
  <saml2:NameID> Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
  SAML-USERNAME
  </saml2:NameID>
  ```

  Where `SAML-USERNAME` is the username UAA retrieves.

- **LDAP:** UAA derives the username from user input.

- **OIDC1.0/OAuth2:** UAA derives usernames from OpenID Connect and OAuth2 providers from the `id_token`, userinfo endpoint, or the access token. The usernames are returned in JSON Web Token (JWT) format. The name of the claim holding the username value can be configured and defaults to `preferred_username`.

User groups

A user can belong to one or more groups. A group is a way to express a common group-based or role-based access control model. A group has a display name. This name is an arbitrary string that corresponds directly to a scope in a JWT access token and is used for access control by OAuth2 resource servers.

The common group attributes are:

- `type`: This can be one of two membership types, `DIRECT` and `INDIRECT`. `DIRECT` means that the user is directly associated with the group. `INDIRECT` means that the membership has been inherited from the nested membership of groups.

- `display`: The `displayName` of the group the user or the group belongs to. The `displayName` is an identifier that is unique to a given identity zone and is a representation of the access given to the user.

Note: To create a group, see Groups in the UAA API documentation. To add a user or group to a group, see Add Member in the UAA API documentation.

Default user groups

You can configure UAA to have one or more default groups. These are groups that every user in the system is a member of, even if there is no direct relationship between user and group in the database.

Shadow users

Users that have authenticated through an external IDP still get assigned a record in the users table in the UAA database. These users are frequently called *shadow users*.

UAA internal users have a `user.origin` of `uaa`. Shadow users are differentiated from internal users with a different origin corresponding to the external IDP. Each time an external user is authenticated and the assertion is passed to UAA, UAA refreshes the user information. This means that the information about a shadow user in UAA is accurate up to the last time UAA received an assertion with user information.

Shadow users have a different type of group membership. A shadow user can be associated with a group through its origin. This membership may also change each time a new assertion is received.

A shadow user can also have a group membership defined using `group_membership.origin='uaa'`. These are memberships that remain persistent and do not change when assertions report a change in external group memberships. It also allows UAA operators to assign privileges to users that are not known by external providers or cannot be mapped to external groups.

Client

UAA is an OAuth2 authorization server. Before apps can obtain access tokens, developers must perform a one-time registration process to create a *client* in the UAA.

A client usually represents an app with its own set of permissions and configurations. Clients are protected by simple credentials, such as client ID and secret, that apps use to authenticate themselves to the UAA in order to obtain tokens.

There are two types of clients:

- Clients that access resources and request tokens from UAA with which to do so
- Clients that represent resources and accept and validate access tokens

Clients are created in UAA through *client registration*. You can define clients in UAA using the UAA configuration files or create them using the UAA API.

Choosing authorization grant types

To create a client, the developer must specify which grant types should be permitted using their client. The grant type determines how your client can interact with UAA. Each grant types corresponds to one of the four different authorization flows defined in the OAuth2 2.0 Authorization Framework. For more information, see the Authorization Code section of the OAuth2 2.0 Authorization Framework.

The available grant types on UAA include:

- `authorization_code`
- `password`
- `implicit`
- `client_credentials`

For increased security, use only the grant types your app requires. However, you can assign multiple grant types to one client if necessary.

Selecting the client grant type

To help you choose a grant type for your use case, see the table below:

Grant type	User	Details

`authorization_code`	Developers building web apps	In the authorization code grant flow, the user is directed to a UAA page where they grant approvals to the client. After the user approves the requested scopes, they are redirected back to the client app with an authorization code in the URL parameters. The client app may then exchange the authorization code with UAA to obtain an access token.
`password`	Developers building native desktop or mobile apps	The name `password` refers to the resource owner password grant type. The user provides their username and password to the client app, which may then use them to obtain an `access_token`.
`implicit`	Developers building a single page web app with no server back end	The user is taken to a page on UAA where they are asked to grant approvals to the client. After doing so, they are redirected to the `redirect_uri` with the access token in the URI fragment.
`client_credentials`	Developers when the client app needs to perform actions in UAA on its own behalf	Actions where a `client_credentials` grant type might be appropriate include creating or destroying user groups, managing user group membership, or creating or destroying other clients. The `client_credentials` grant can be likened to service accounts in legacy app ecosystems.
`refresh_token`	Developers must use the `refresh_token` grant type with either the `authorization_code` or `password` grant type. `refresh_token` cannot be used by itself.	Clients typically use the `refresh_token` to obtain a new access token without the need for the user to authenticate again. They do this by calling `/oauth/token` with `grant_type=refresh_token`. A refresh token is only issued to clients that have `refresh_token` in their list of `authorized_grant_types`.

client.client_id

A client is identified with a value up to 255 characters, called the `client_id`. Unlike the `user.id`, the `client_id` is often a human-readable identifier. For example, an app's name could be its `client_id`. This identifier is unique within the identity zone.

client.secret

Client authentication occurs through a password mechanism called a `client_secret`.

UAA allows client credentials to be asserted in two different ways:

- With an HTTP authorization header using Basic authentication.
- By passing the `client_id` and `client_secret` as request parameters as part of an HTTP POST body using content type `application/x-www-form-urlencoded`.

For more information, see the Client Password section of the OAuth 2.0 Authorization Framework.

client.redirect-uri

The `authorization_code` and implicit grant types rely on a user agent. A user agent like a web browser is responsible for performing HTTP redirects to UAA and receiving a response from UAA.

That response can be in the form of an access token or a code that is exchanged for an access token later.

A client that supports either of these two flows must have at least one URL in the client configuration. Alternatively, you can use multiple URLs and wildcards (*) for ant path matching. For more information, see the Class AntPathMatcher section of the Spring Framework documentation.

URLs are registered using a comma-separated string.

client.access-token-validity

UAA validates access tokens up to the time those tokens expire. Clients do the same, if they can validate tokens offline. The access token validity is the value in seconds from the time the token was created to when it expires.

client.refresh-token-validity

UAA validates refresh tokens up to the time those tokens expire. Clients do the same, if they can validate tokens offline. The refresh token validity is the value in seconds from the time the token was created to when it expires.

Choosing scopes and authorities

Client scopes are used to populate the scope claim when constructing an access token where the client acts on behalf of the user.

When an access token is created, UAA takes the user groups and intersects them with the client scopes. The intersection of these two fields are scopes that are eligible to be populated in the access token. There are two more validations that can further limit the scopes that get populated in the access token after the intersection has been decided:

1. Did the user approve these scopes?
2. Did the client request these scopes in the authorization request?

The token can never contain more scopes than the intersection between client scopes and user groups.

client.autoapprove

Scopes in an access token must be approved by the granting entity.

During a `client_credentials` grant, the client itself is the granting entity and automatically assumes the client authorities are approved.

During a password grant, the user shares their password with the client app. The client app assumes this sharing to be implicit approval of the scopes that the client wants to be populated in the access token.

Two grant types, `authorization_code` and `implicit`, require specific user approval for the scopes to populate in the access token. UAA provides a UI that lets the user approve or deny scopes from being populated in the access token.

During client registration, the operator can configure the client to bypass this approval process by setting the auto-approve values to a single string with its value set to `true`. This results in any requested scope being approved automatically.

The value can also be a comma-separated list of selected scopes that do not require user approval.

client.additional_information

Clients can store custom attributes in a field called `additional_information`. This is a simple key value store. The table below describes these custom attributes:

Key	Value
`allowed providers`	You can limit which users can use which apps. For example, in a Ops Manager deployment, you may have set up multiple IDPs. Or, you could be using Facebook and your organization's LDAP system. You can limit UAA to only issue an app tokens if the users are from a certain provider. To do this, configure the app's client with `allowed providers="ldap"`. The value is a comma-separated string of `Identity Provider.origin` values.
`created with`	UAA stores the scope `zones.write` in this field if the client was created using the `/identity-zones` endpoint. UAA uses this field to allow clients to be deleted by the same endpoint. This is not a configurable field.
`name`	Various tooling in the Ops Manager ecosystem creates clients with generated `Client.client_id` values. These tools often store a human-readable name in this field. For more information, see client.client_id.
`approvals_deleted`	Contains a boolean value if performing an operation on the client resulted in all the client's user approvals were deleted. For example, changing the `client.client_secret` value causes UAA to delete all the approvals. UAA then stores a value of `true` in this field.
`token_salt`	Tokens, even stateless JWT, can be revoked. Revoked tokens do not pass a UAA token validation when the token is passed to the `/introspect` endpoint. UAA revokes a token if the client's secret has changed. There may be occasions to revoke all tokens for a certain client without having to change the client secret. You can do this by changing the `token_salt`. `token_salt` is an arbitrary string value that is used to generate a hash.

Identity providers in UAA

You can configure identity providers (IDPs) with your Cloud Foundry User Account and Authentication (UAA) server.

LDAP providers

The Lightweight Directory Access Protocol communicates with directory servers. Compared with the other external IDPs, LDAP is a very simple integration. It only requires configuration on UAA. Other types of provider require that you make configuration changes on both UAA and on the external provider.

LDAP supports three different connection types:

- `ldap://`: Cleartext LDAP, typically over port 389
- `ldaps://`: An SSL/TLS connection, typically over port 636

- `ldap://` with StartTLS: A TLS handshake performed on a cleartext LDAP connection, typically over port 389

For more information, see User Account and Authentication LDAP Integration in the UAA repository on GitHub.

SAML providers

The SAML v2 standard is a dominant player in the federated authentication space. It is also one of the harder integrations to configure. When configuring SAML integration on UAA, you must configure both UAA and the SAML IDP. A mistake on either side causes assertions to be rejected and authentication to fail.

OpenID Connect providers

The OpenID Connect standard is a simple identity layer on top of the OAuth 2.0 protocol. Clients can verify the identity of the end-user based on the authentication performed by the OIDC IDP, as well as to obtain basic profile information about the end-User in an interoperable and REST-like manner. When configuring OIDC integration on UAA, you must configure both UAA and the OIDC IDP. A mistake on either side causes tokens to be rejected and authentication to fail.

Service provider versus identity provider

The SAML specification defines two players in a SAML interaction:

- **Service provider (SP)**, the server that receives the assertion. This is typically UAA.
- **Identity provider (IDP)**, the server that receives the authentication request, authenticates the user and sends the assertion to the SP.

UAA can be configured to act as an SP or IDP. In most scenarios, UAA is the SP, and an external provider, such as Okta or ADFS, is the IDP.

View metadata

A SAML provider, either the SP or the IDP, presents a set of metadata. This metadata contains information about the server and is used to configure the opposing provider.

If you use UAA as an SP, you can download metadata through the `/saml/metadata` endpoint of your SAML provider. For example, http://login.example.com/saml/metadata. Metadata is in XML format. Using this information, you can configure the IDP.

If you use UAA as an IDP, fetch the metadata from the `/saml/idp/metadata` endpoint. For example, http://login.example.com/saml/idp/metadata.

SAML IDP workflow

SAML provides two commonly-used workflows, SP-initiated and IDP-initiated. These sections describe each workflow.

SP-Initiated

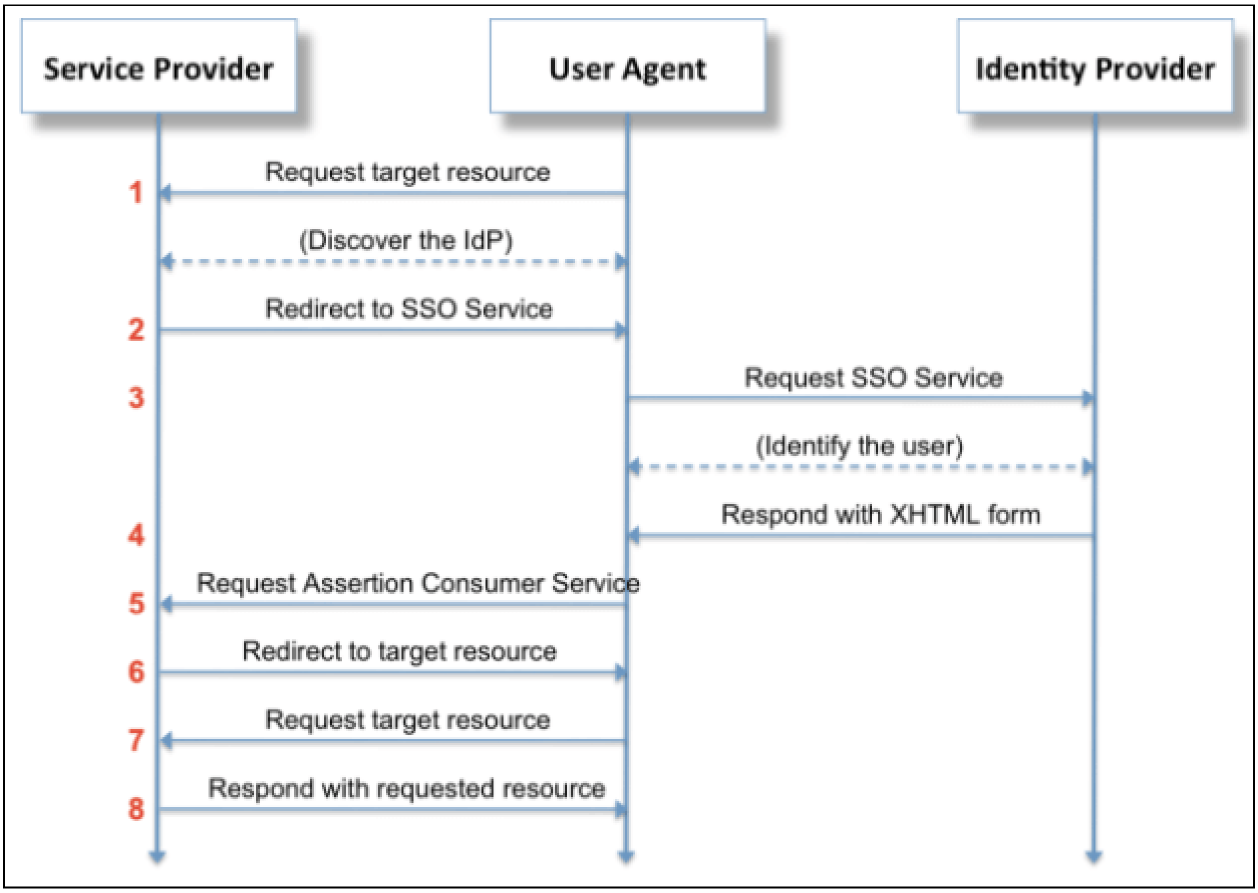
SAML SP-initiated

If you go to the UAA login page and click a link to authenticate using your organization's SAML provider, you are using an SP-initiated flow.

The SP sends an authentication request to the IDP. See the following example:

```
xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
    AssertionConsumerServiceURL="https://..host../saml/SSO/alias/login.identity.cf-app.com"
    Destination="http://simplesamlphp.identity.cf-app.com/saml2/idp/SSOService.php"
    ForceAuthn="false"
    ID="a17j41337a9835i93h78hihc9a89j4b"
    IsPassive="false" IssueInstant="2017-05-03T21:16:15.989Z"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Version="2.0"
>
  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
      Login.identity.cf-app.com
  saml2:Issuer>
  <saml2p:NameIDPolicy Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress" />
</saml2p:AuthnRequest>
```

After the IDP has authenticated the user, it sends an assertion back to the user. This step takes place in both SP-initiated and IDP-initiated flows. For information about the SAML response, see IDP-Initiated.



SAML IDP-initiated

You can start your authentication process with the IDP.

The IDP sends a SAML assertion to the SP, which in this case is UAA, like the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    Destination="https://...url.../saml/SSO/alias/login.identity.cf-app.com"
    ID="_de497bc8a79e5f17202a30112181ea7c99325f8827"
    InResponseTo="a17j41337a9835i93h78hihc9a89j4b"
    IssueInstant="2017-05-03T21:51:01Z" Version="2.0">
  <saml:Issuer>
    http://simplesamlphp.identity.cf-app.com/saml2/idp/metadata.php
  </saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sh
a1" />
      <ds:Reference URI="#_de497bc8a79e5f17202a30112181ea7c99325f8827">
        <ds:Transforms>
          <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
```

```
/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /
>
        ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /
>
        <ds:DigestValue>QQZHHBC4KDxZ0H4FrbTGFOnNR1E=<\/ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>...signature value...<\/ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>...x509 certificate value<\/ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
  </samlp:Status>
  <saml:Assertion
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ID="_ec6adc4fca9e3dc72fff4a1fd99561007777ac7afd"
    IssueInstant="2017-05-03T21:51:01Z"
    Version="2.0">
<saml:Issuer>http://simplesamlphp.identity.cf-app.com/saml2/idp/metadata.php<
\/saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:SignedInfo>
        <ds:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1" />
        <ds:Reference URI="#_ec6adc4fca9e3dc72fff4a1fd99561007777ac7afd">
          <ds:Transforms>
            <ds:Transform
              Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signatur
e" />
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
/>
          <ds:DigestValue>dxt7OSy1/Xf6+BHgDU4YOi4ogMg=<\/ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>...signature value...<\/ds:SignatureValue>
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>...x509 certificate value<\/ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </ds:Signature>
    <saml:Subject>
      <saml:NameID
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
        SPNameQualifier="login.identity.cf-app.com">
          marissa@test.org
      </saml:NameID>
      <saml:SubjectConfirmation
          Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData
            InResponseTo="a9e378fia1g38da5503ahgih430fah"
            NotOnOrAfter="2017-05-03T21:56:01Z"
            Recipient="https://..url../saml/SSO/alias/login.identity.cf-app.c
om" />
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions
        NotBefore="2017-05-03T21:50:31Z"
        NotOnOrAfter="2017-05-03T21:56:01Z">
      <saml:AudienceRestriction>
        <saml:Audience>login.identity.cf-app.com<\/saml:Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement
        AuthnInstant="2017-05-03T21:16:21Z"
        SessionIndex="_a3ab48bcfbb76ad78d4ac28231240af256be171c9f"
        SessionNotOnOrAfter="2017-05-04T05:51:01Z">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
           urn:oasis:names:tc:SAML:2.0:ac:classes:Password
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute
          Name="uid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:
basic">
        <saml:AttributeValue xsi:type="xs:string">
            marissa@test.org
        </saml:AttributeValue>
      </saml:Attribute>
      <saml:Attribute Name="eduPersonAffiliation"
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">member<\/saml:AttributeValu
e>
        <saml:AttributeValue xsi:type="xs:string">marissa<\/saml:AttributeVal
ue>
      </saml:Attribute>
      <saml:Attribute Name="emailAddress"
          NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <saml:AttributeValue xsi:type="xs:string">
            marissa@test.org
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
</samlp:Response>
```

SAML configuration

UAA provides a limited set of attributes to configure when setting up an external IDP. You configure these attributes for each individual SAML IDP you set up in UAA.

nameID

The `nameID` attribute is the element that provides the username in the SAML assertion. This is sometimes called the *subject* of the assertion. UAA links the external user to the internal shadow user record by matching the `nameID` to the `user.username`.

For example:

- `nameID: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
- `nameID: urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- `nameID: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

The most common configuration is `unspecified`, where the IDP decides which attribute is the username.

Below is an example of what the subject element looks like in an assertion, with the username `EXAMPLE@test.org`:

```
<saml:Subject>
  <saml:NameID
      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
      SPNameQualifier="login.identity.cf-app.com">
    EXAMPLE@test.org
  </saml:NameID>
  <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <saml:SubjectConfirmationData InResponseTo="a17j41337a9835i93h78hihc9a89j
4b"
      NotOnOrAfter="2017-05-03T21:21:21Z"
      Recipient="https://..url../saml/SSO/alias/login.identity.cf-app.com"
    />
  saml:SubjectConfirmation>
saml:Subject>
```

AssertionConsumerService

The `AssertionConsumerService` attribute indicates which assertion consumer service URL the external IDP should use. This is used by the SP when it sends a SAML request for authentication to the IDP. The SP metadata contains one assertion consumer service by default, but can contain more.

Below are examples of assertion consumer services in SP metadata, organized by index:

```
<md:AssertionConsumerService
  Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
  Location="https://login.run.pivotal.io/saml/SSO/alias/login.run.pivotal.io"
  index="0"
  isDefault="true"/>
<md:AssertionConsumerService
  Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
  Location="https://login.run.pivotal.io/saml/SSO/alias/login.run.pivotal.io"
  index="1"/>
<md:AssertionConsumerService
  Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
  Location="https://login.run.pivotal.io/oauth/token/alias/login.run.pivotal.
io"
  index="2"/>
```

Below is an example of a SAML request to the IDP with the correct assertion consumer service selected:

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest
    AssertionConsumerServiceURL="https://..../saml/SSO/alias/login.identity.c
f-app.com"
    Destination="http://simplesamlphp.identity.cf-app.com/saml2/idp/SSOServic
e.php"
    ForceAuthn="false"
    ID="a17j41337a9835i93h78hihc9a89j4b"
    IsPassive="false"
    IssueInstant="2017-05-03T21:16:15.989Z"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Version="2.0"
    xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml2:Issuer
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
      Login.identity.cf-app.com
  </saml2:Issuer>
  <saml2p:NameIDPolicy
    Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"/>
</saml2p:AuthnRequest>
```

idpMetadata

The `idpMetadata` attribute holds either an XML string or a URL.

Below is an example of IDP metadata configuration in UAA:

```
idpMetadata: \http://simplesamlphp.identity.cf-app.com/saml2/idp/metadata.php
```

If the IDP is configured using a URL, UAA refreshes the metadata content from that URL every 10 minutes. This is useful for providers that rotate their keys frequently.

If the IDP is configured using metadata as an XML string, the metadata remains static. Rotating keys on the IDP requires you to reconfigure UAA with the new data.

Below is an example of the XML metadata configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor
    xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="http://www.okta.com/k2lw4l5bPODCMIIDBRYZ">
  <md:IDPSSODescriptor
      WantAuthnRequestsSigned="true"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>..x509 certificateds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddressm
d:NameIDFormat>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecifiedmd:
NameIDFormat>
    <md:SingleSignOnService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
      Location="https://s.okta.com/app/env_1/k2lw4l5bPODCMIIDBRYZ/sso/saml" /
>
    <md:SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
        Location="https://s.okta.com/app/env_1/k2lw4l5bPODCMIIDBRYZ/sso/saml"
/>
  </md:IDPSSODescriptor>
</md:EntityDescriptor>
```

showSamlLoginLink

The `showSamlLoginLink` shows the login link on the UAA home page. This appears when `showSamlLoginLink` is set to `true`. When you click this link, it starts the SP-initiated flow. This is frequently set to `false` when you use the IDP-initiated flow.

Below is an example of a SAML login link text configuration:

```
showSamlLoginLink: true
```

linkText

The `linkText` attribute allows you to specify the text on a link on the UAA home page. This shows when `showSamlLoginLink` is set to `true`.

Below is an example of a link text configuration:

```
linkText: Log in with SSO
```

metadataTrustCheck

The `metadataTrustCheck` attribute requires UAA to verify signatures of incoming messages. This requirement exists when `metadataTrustCheck` is set to `true`.

Below is an example of a metadata trust check configuration:

```
metadataTrustCheck: true
```

UAA performance metrics

Cloud Foundry User Accounts and Authentication (UAA) emits metrics constantly. These metrics help you to understand performance, troubleshoot problems, and assess the health of your installation in real time. These metrics are emitted through Loggregator.

This topic describes various metrics that UAA, virtual machines (VMs), and Java Virtual Machines (JVMs) emit.

About UAA performance

The following tables explain different types of UAA and UAA-related metrics you can view. There are three different types of metrics discussed in these tables:

- Global performance metrics: Metric values that UAA emits
- Virtual Machine vital statistics: Metric values that UAA reads from the VM
- Java Virtual Machine vital statistics: Metric values that UAA reads from the JVM

Each table contains:

- **Name:** The name of the metric.
- **Type:** How the metric is displayed. For example, counters, gauges, or timers.
- **Description:** An explanation of what values this metric displays.
- **Example:** A code sample of this metric’s output.
- **Indicator:** What changes in the metric’s value may indicate over time.
- **Status**: Whether a metric is active or static. Active metrics may change between metrics emissions. Static metrics are fixed and do not change.

Note: If consuming UAA metrics through the Firehose, incremental metrics, or metrics that capture an increment or decrement in a value since UAA's last emission, are expressed as cumulative values. For more information, see the statsd-injector repository on GitHub.

Global performance metrics

The table below describes metrics that UAA emits:

Name	Type	Description	Example	Indicator	Status
`requests.global.completed.count`	Counter	Number of HTTP requests the server has processed since last metric emission. This metric includes all requests sent to the server, including health checks.	`uaa.requests.global.completed.count:1|c`	Use this metric to calculate request load and throughput.	Active
`requests.global.completed.time`	Gauge	Average time in milliseconds spent per HTTP request. This metric is calculated as an average across all completed requests, including health checks.	`uaa.requests.global.completed.time:60|g`	A rise may indicate problems with server or database.	Active
`server.inflight.count`	Gauge	Number of requests the server is currently processing, also called in-flight requests.	`uaa.server.inflight.count:1|g`	If this number climbs continuously, it can indicate that servers are getting saturated and are unable to handle the incoming load.	Active
`requests.global.unhealthy.count`	Counter in UAA v4.8 and earlier; gauge in UAA v4.9.0 and later	Number of completed requests that exceeded the tolerable response time since last metric emission. Each URL group can have a different tolerable completion time, which is preconfigured in each UAA release. These values are currently not configurable.	`uaa.requests.global.unhealthy.count:1|c`	If the number of requests not meeting tolerable completion time is growing, than either the tolerable request time must be configured to account for false negatives, or the server does not have enough capacity to handle the request load. The cause for this can be a need for an increase in server or database resources. In order to make a scaling decision, you need further metrics.	Active
`requests.global.unhealthy.time`	Gauge	Average time in milliseconds per completed HTTP request that did not finish within the set tolerable time since startup.	`uaa.requests.global.unhealthy.time:250|g`	It can be useful to compare this metric to `uaa.requests.global.completed.time`.	Active
`requests.global.status_4xx.count`	Counter	Number of HTTP requests that have returned `400` codes, or client errors, since UAA's last metrics emission. These do not indicate server errors. A `400` code may indicate an invalid request to the server.	`uaa.requests.global.status_4xx.count:1|c`	This metric gives the client the ability to calculate error rates. It is often used to detect faulty apps that may be causing unnecessary processing on the server.	Active
`requests.global.status_5xx.count`	Counter	Number of HTTP requests that have returned `500` codes, or server errors, since UAA's last metrics emission.	`uaa.requests.global.status_5xx.count:1|c`	This metric gives the client the ability calculate error rates and determine if further investigation is needed.	Active
`server.up.time`	Timer	Number of milliseconds that have elapsed since this server instance started.	`uaa.server.up.time:42346751|g`	This metric indicates the time since last startup.	Active

Name	Type	Description	Example	Indicator	Status
server.idle.time	Timer	Number of milliseconds that the server has spent in an idle state, when no requests were being processed. This allows a client to calculate the amount of actual, rather than cumulative, time the server has spent processing requests with up.time-idle.time.	uaa.server.idle.time:2346751|g	This metric allows the client to calculate when the server is receiving load time.	Active
database.global.completed.count	Counter	Number of database queries the server has processed since UAA's last metrics emission.	uaa.database.global.completed.count:1|c	Use this metric to track the number of queries that have reached and been processed by the server over a period of time.	Active
database.global.completed.time	Gauge	Average amount of time in milliseconds per database query.	uaa.database.global.completed.time:248|g	Use this metric to track the time to complete a database query on average.	Active
database.global.unhealthy.count	Counter	Number of database queries that failed or did not meet the tolerated response time since UAA's last metrics emission. The response time is not configurable during runtime. By default, it is set to three seconds.	uaa.database.global.unhealthy.count:1|c	Use this metric to monitor database query success and failure over time.	Active
database.global.unhealthy.time	Timer	Average amount of time in milliseconds per database query that was not within the tolerated response time.	uaa.database.global.unhealthy.time:4678623|g	Use this metric to monitor database response time.	Active

About UAA vital statistics

These sections describe metrics that the UAA VM and JVM emit.

Virtual Machine vital statistics

The table below describes metrics that the UAA VM emits:

Name	Type	Description	Example	Indicator	Status
vitals.vm.cpu.count	Gauge	How many CPUs are on this VM as reported by the JVM. This metric is useful when you want to read system load average. The number reported by load average must be correlated to the number of CPUs.	uaa.vitals.vm.cpu.count:4|g	This metric is required for a proper CPU load calculation.	Active

Name	Type	Description	Example	Indicator	Status
`vitals.vm.cpu.load`	Gauge	Average system CPU load as reported by the JVM. The value is reported as a whole number multiplied by .01. For example, a value of 163 is read as 1.63.	`uaa.vitals.vm.cpu.load:50|g`	If the value of `(cpu.load / 100.0 / cpu count) is more than 2.0, this indicates that the system may be overloaded and processing data slowly.	Active
`vitals.vm.memory.total`	Gauge	Total OS memory, in bytes, as reported by the JVM.	`uaa.vitals.vm.memory.total:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.vm.memory.committed`	Gauge	OS memory, in bytes, committed to UAA processes, as reported by the JVM.	`uaa.vitals.vm.memory.committed:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.vm.memory.free`	Gauge	Free OS memory, in bytes, as reported by the JVM.	`uaa.vitals.vm.memory.free:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active

Java Virtual Machine vital statistics

The table below describes metrics that the UAA JVM emits:

Name	Type	Description	Example	Indicator	Status
`vitals.jvm.cpu.load`	Gauge	UAA process CPU load average as reported by the JVM. This value is multiplied by 100 and reported as a whole number representing the CPU load on the VM incurred by the UAA process, excluding any other processes on the VM.	`uaa.vitals.jvm.cpu.load:25|g`	Health and scaling. If CPU load is showing as high, this metric can be used to confirm that it is indeed the UAA using up the CPU and not other jobs on the same VM.	Active
`vitals.jvm.thread.count`	Gauge	Number of threads running inside the UAA process, as reported by the JVM.	`uaa.vitals.jvm.thread.count:53|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.jvm.heap.init`	Gauge	Minimum amount of OS memory, in bytes, requested by the UAA JVM process to be used as part of the Java heap memory, as reported by the JVM.	`uaa.vitals.jvm.heap.init:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Static
`vitals.jvm.heap.committed`	Gauge	Guaranteed amount of Java heap memory, in bytes, committed to the UAA JVM process, as reported by the JVM.	`uaa.vitals.jvm.heap.committed:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.jvm.heap.used`	Gauge	Java heap memory, in bytes, currently in use by the UAA process as reported by the JVM.	`uaa.vitals.jvm.heap.used:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Static

Name	Type	Description	Example	Indicator	Status
`vitals.jvm.heap.max`	Gauge	Java heap memory, in bytes, that is the upper limit for the UAA processes, as reported by the JVM.	`uaa.vitals.jvm.heap.max:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Static
`vitals.jvm.non-heap.init`	Gauge	Minimum non-Java memory, in bytes, acquired by the UAA process, as reported by the JVM.	`uaa.vitals.jvm.non-heap.init:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.jvm.non-heap.committed`	Gauge	Guaranteed non-Java memory, in bytes, committed by the UAA process, as reported by the JVM.	`uaa.vitals.jvm.non-heap.committed:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.jvm.non-heap.used`	Gauge	Current non-Java memory, in bytes, that the UAA process can use, as reported by the JVM.	`uaa.vitals.jvm.non-heap.used:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active
`vitals.jvm.non-heap.max`	Gauge	Upper limit of non-Java memory, in bytes, that the UAA process can use, as reported by the JVM.	`uaa.vitals.jvm.non-heap.max:1073741824|g`	Use this metric in conjunction with other performance metrics to assess system health.	Active

For more information about JVM memory, see the Oracle documentation.

Configuring Authentication and Enterprise SSO for TAS for VMs

In this section:

- Configuring Authentication and Enterprise SSO for TAS for VMs
- Configuring CA as an Identity Provider
- Configuring PingFederate as an Identity Provider

Configuring Authentication and Enterprise SSO for TAS for VMs

This topic describes VMware Tanzu Application Service for VMs (TAS for VMs) authentication and single sign-on configuration with Lightweight Directory Access Protocol (LDAP) and Security Assertion Markup Language (SAML).

Overview

Connecting TAS for VMs to either the LDAP or SAML external user store allows the User Account and Authentication (UAA) server to delegate authentication to existing enterprise user stores.

If your enterprise user store is exposed as a SAML or LDAP identity provider (IDP) for single sign-on (SSO), you can configure SSO to allow users to access Apps Manager and the Cloud Foundry

Command Line Interface (cf CLI) without creating a new account or, if using SAML, without re-entering credentials.

For information about managing user identity and pre-provisioning user roles with SAML or LDAP, see Adding Existing SAML or LDAP Users to an Ops Manager Deployment.

For an explanation of the process used by the UAA server when it attempts to authenticate a user through LDAP, see Configuring LDAP Integration with Ops Manager in the Knowledge Base.

> **Note:** When integrating with an external IDP, such as LDAP, authentication within the UAA becomes chained. An authentication attempt with a user's credentials is first attempted against the UAA user store before the external provider, LDAP. For more information, see Chained Authentication in the *User Account and Authentication LDAP Integration* documentation on GitHub.

Follow one of the procedures below to configure your deployment with SAML or LDAP.

Configure TAS for VMs to Use a SAML Identity Provider

In SAML terminology, the SAML protocol communicates user data between an IDP and a service provider (SP).

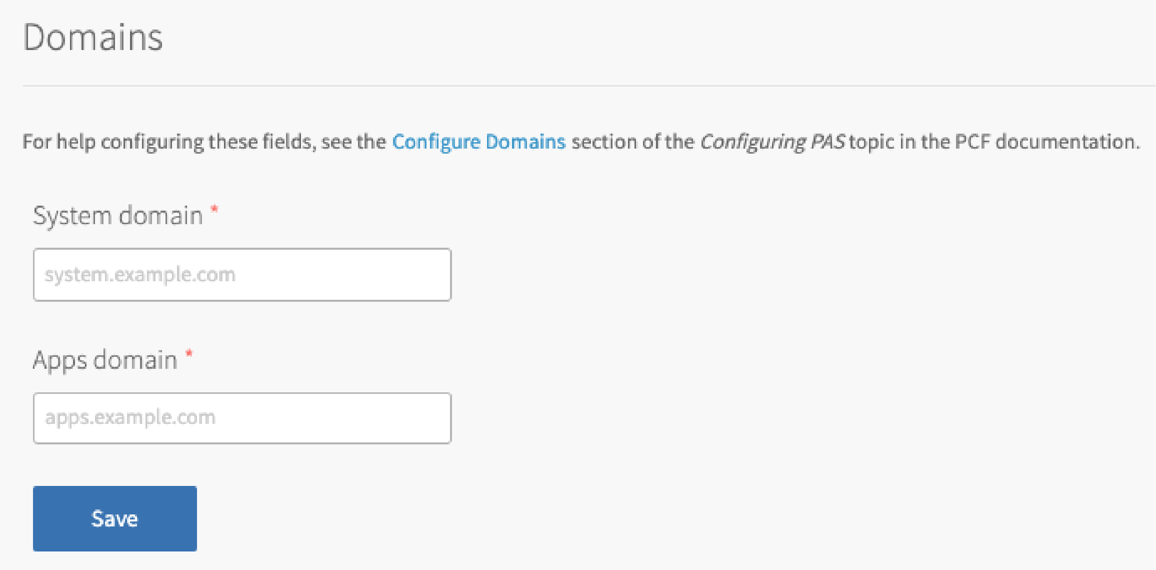
To connect TAS for VMs with SAML, follow both of these procedures:

- Configure TAS for VMs as a Service Provider for SAML
- Configure SAML as an Identity Provider for TAS for VMs

Configure TAS for VMs as a Service Provider for SAML

To configure TAS for VMs to use a SAML IDP:

1. Go to the VMware Tanzu Operations Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Domains**.
4. Record your system domain.



Domains

For help configuring these fields, see the Configure Domains section of the *Configuring PAS* topic in the PCF documentation.

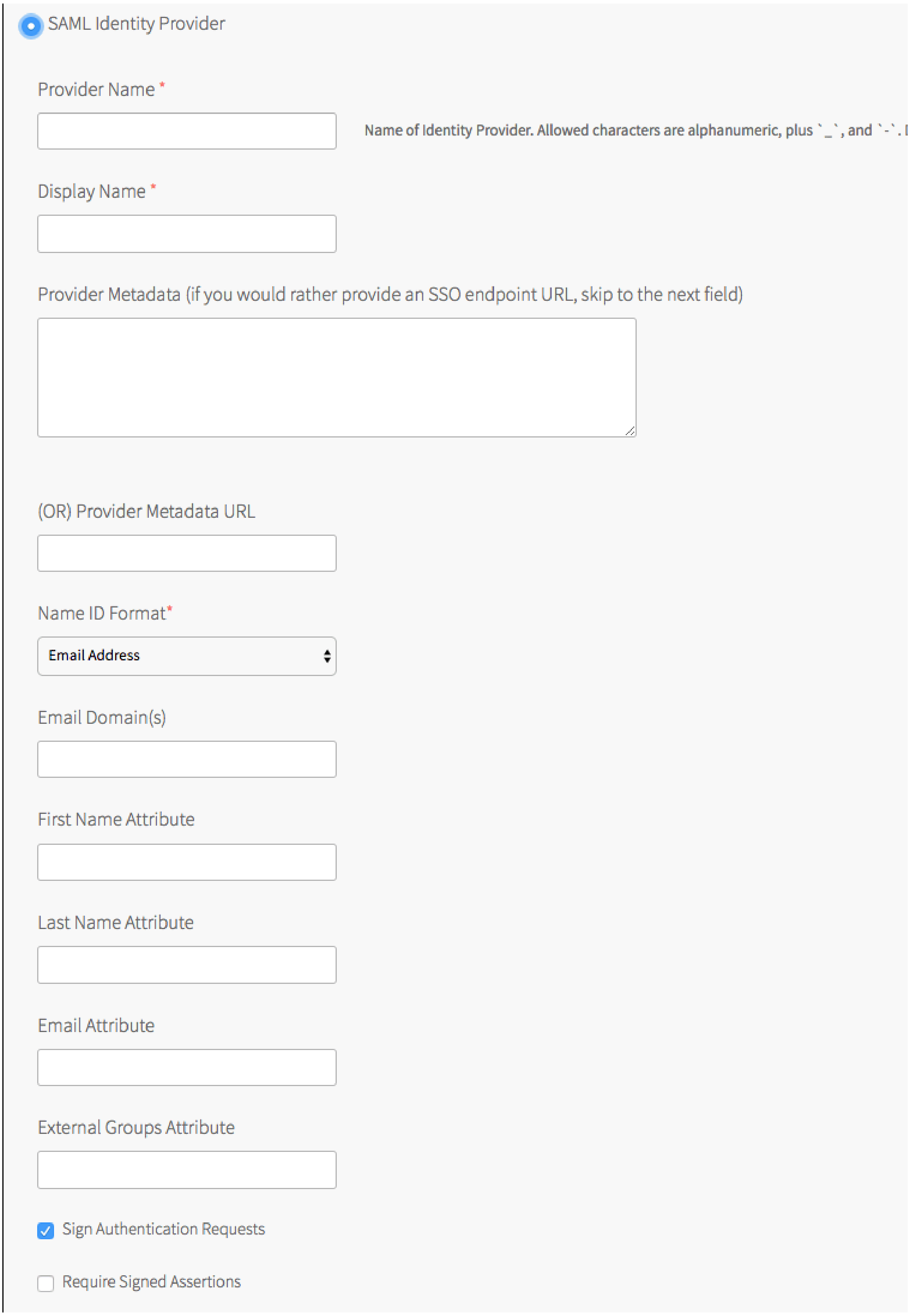
System domain *

system.example.com

Apps domain *

apps.example.com

5. Select **Authentication and Enterprise SSO**.
6. Select **SAML identity provider**.



● SAML Identity Provider

Provider Name *

Name of Identity Provider. Allowed characters are alphanumeric, plus `_`, and `-`. I

Display Name *

Provider Metadata (if you would rather provide an SSO endpoint URL, skip to the next field)

(OR) Provider Metadata URL

Name ID Format*

Email Address

Email Domain(s)

First Name Attribute

Last Name Attribute

Email Attribute

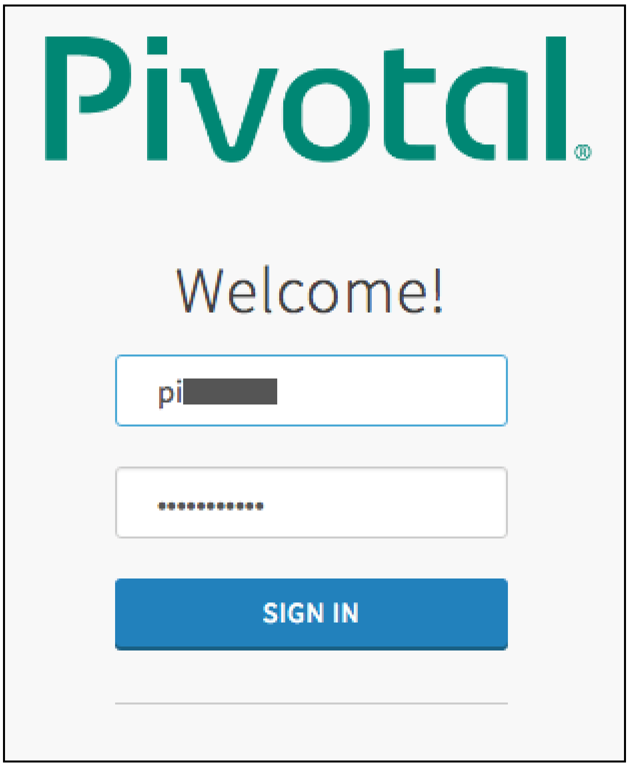
External Groups Attribute

☑ Sign Authentication Requests

□ Require Signed Assertions

Note: You must manually deactivate a SAML IDP created by TAS for VMs when you no longer require it.

7. Set the **Provider name**. This is a unique name you create for the IDP. This name can include only alphanumeric characters, `+`, `_`, and `-`. You should not change this name after deployment because all external users use it to link to the provider.
8. Enter a **Display name**. Your provider display name appears as a link on your Ops Manager login page. You can access this link at `https://login.SYSTEM-DOMAIN`, where `SYSTEM-DOMAIN` is the system domain you recorded from the **Domains** pane.



Welcome!

pi[redacted]

••••••••••••

9. Retrieve the metadata from your IDP and copy it into either the **Provider metadata** or the **Provider metadata URL** fields, depending on whether your IDP exposes a metadata URL or not. For more information, see Configure SAML as an Identity Provider for TAS for VMs. VMware recommends that you use the provider metadata URL rather than provider metadata because the metadata can change. You can do this in either of the following ways:
    - If your IDP exposes a metadata URL, provide the metadata URL.
    - Download your IDP metadata and paste this XML into the **Provider metadata** field.

> **Note:** You only need to select one of the above configurations. If you configure both, your IDP defaults to the **Provider metadata URL**.

> **Note:** For information about onboarding SAML users and mapping them to TAS for VMs user roles, see Adding Existing SAML or LDAP Users to an Ops Manager Deployment.

10. Select the **Name ID format** for your SAML IDP. This translates to `username` in TAS for VMs. The default is **Email Address**.

11. For **Email domains**, enter a comma-separated list of the email domains for external users who are to receive invitations to Apps Manager.
12. For **First name attribute** and **Last name attribute**, enter the attribute names in your SAML database that correspond to the first and last names in each user record. For example, `first_name` and `last_name`. This field is case-sensitive.
13. For **Email attribute**, enter the attribute name in your SAML assertion that corresponds to the email address in each user record. For example, `EmailID`. This field is case-sensitive.
14. For **External groups attribute**, enter the attribute name in your SAML database that defines the groups that a user belongs to. For example, `group_memberships`. To map the groups from the SAML assertion to admin roles in TAS for VMs, see Grant Admin Permissions to an External Group (SAML or LDAP) in *Creating and Managing Users with the UAA CLI (UAAC)*. This field is case-sensitive.
15. By default, all SAML authentication requests from TAS for VMs are signed. To change this, deactivate the **Sign authentication requests** checkbox and configure your IDP to verify SAML authentication requests.
16. To validate the signature for the incoming SAML assertions, enable the **Required signed assertions** checkbox and configure your IDP to send signed SAML assertions.
17. Click **Save**.
18. Return to the Ops Manager Installation Dashboard.
19. Click **Review Pending Changes**.
20. Click **Apply Changes**.

Configure SAML as an Identity Provider for TAS for VMs

To configure a SAML IDP to designate TAS for VMs as an SP:

1. Download the SP metadata from `https://login.SYSTEM-DOMAIN/saml/metadata`, where `SYSTEM-DOMAIN` is the system domain you recorded from the **Domains** pane. For configuration instructions, see the documentation from your IDP.

   For configuration instructions, see the documentation from your IDP.

2. See the table below for information about certain industry-standard IDPs and how to integrate them with TAS for VMs:

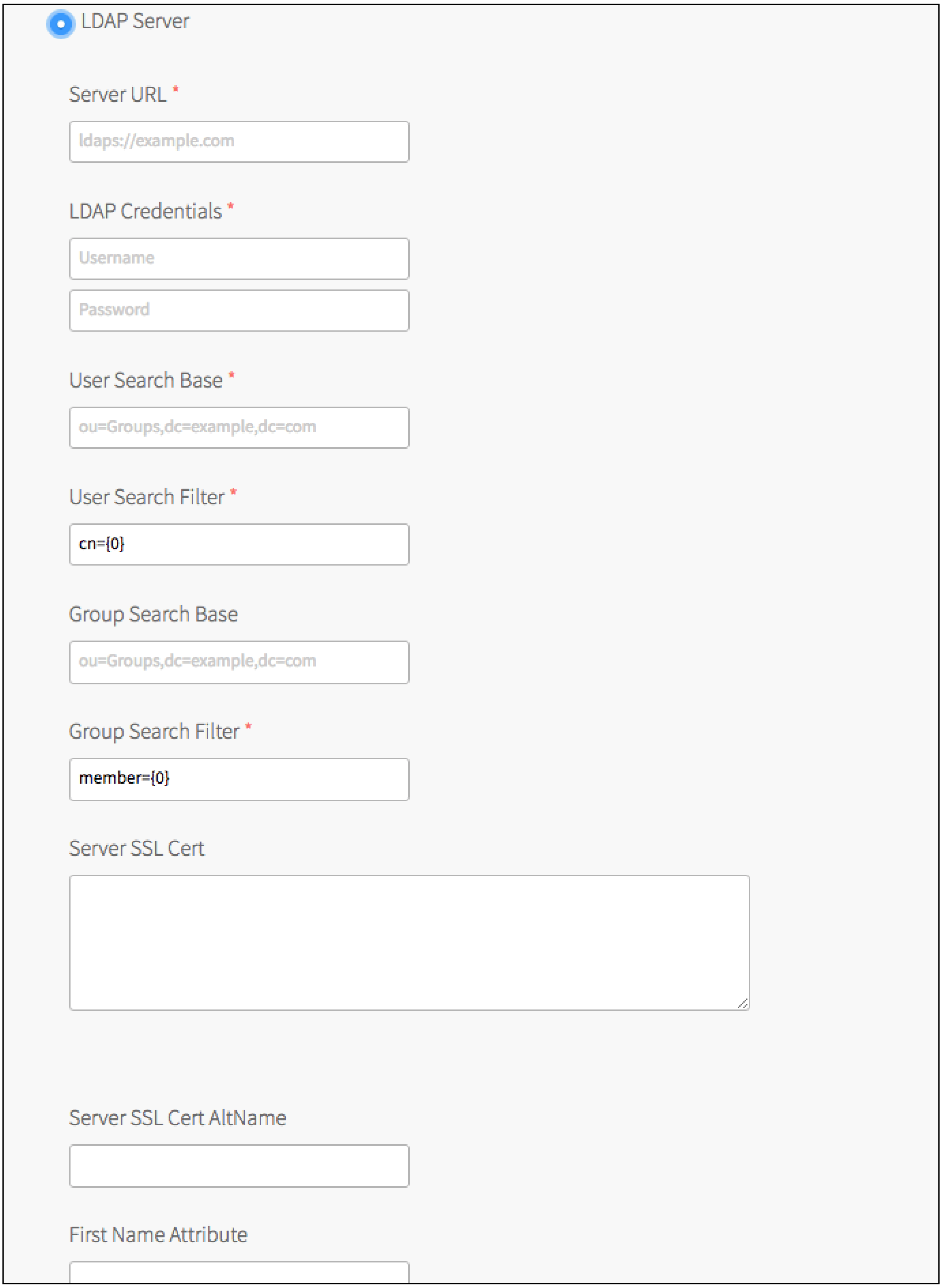
Solution Name	Integration Guide
CA Single Sign-On aka CA SiteMinder	Configuring CA as an Identity Provider
Ping Federate	Configuring PingFederate as an Identity Provider
Active Directory Federation Services (AD FS)	Configuring AD FS as an Identity Provider

Note: Some IDPs allow uploads of SP metadata. Other providers require you to manually enter the SP metadata into a form. If your IDP requires manual entry but is not listed above, see Configuring CA as an Identity Provider.

Configure LDAP as an Identity Provider for TAS for VMs

To integrate the UAA with one or more LDAP servers:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Authentication and Enterprise SSO**.



● LDAP Server

Server URL *

ldaps://example.com

LDAP Credentials *

Username

Password

User Search Base *

ou=Groups,dc=example,dc=com

User Search Filter *

cn={0}

Group Search Base

ou=Groups,dc=example,dc=com

Group Search Filter *

member={0}

Server SSL Cert

Server SSL Cert AltName

First Name Attribute

4. Under **Configure your UAA user account store with either internal or external authentication mechanisms**, select **LDAP server**.
5. For **Server URLs**, enter the URL that points your LDAP server. For multiple LDAP servers, enter a space-separated list. Each URL must include one of the following protocols:
   - `ldap://`: This specifies that the LDAP server uses an unencrypted connection.
   - `ldaps://`: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.
6. For **LDAP credentials**, enter the LDAP distinguished name (DN) and password for binding to the LDAP server. For example, `cn=administrator,ou=Users,dc=example,dc=com`.

   Note: VMware recommends that you provide LDAP credentials that grant read-only permissions on the LDAP search base and the LDAP group search base. Additionally, if the bind user belongs to a different search base, you must use the full DN.

   Caution: VMware recommends against reusing LDAP service accounts across environments. LDAP service accounts should not be subject to manual lockouts, such as lockouts that result from users utilizing the same account. Also, LDAP service accounts should not be subject to automated deletions, since disruption to these service accounts could prevent user logins.

7. For **User search base**, enter the location in the LDAP directory tree from which any LDAP user search begins. The typical LDAP search base matches your domain name.

   For example, a domain named “cloud.example.com” typically uses the following LDAP user search base: `ou=Users,dc=example,dc=com`
8. For **User search filter**, enter a string that defines LDAP user search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

   In the LDAP search filter string that you use to configure TAS for VMs, use `{0}` instead of the username. For example, use `cn={0}` to return all LDAP objects with the same common name as the username.

   In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

   Note: For instructions for testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry in the Knowledge Base.

9. For **Group search base**, enter the location in the LDAP directory tree from which the LDAP group search begins.

   For example, a domain named "cloud.example.com" typically uses the following LDAP group search base: `ou=Groups,dc=example,dc=com`

   This is required if you are mapping LDAP groups to an admin role. To map the groups under this search base to admin roles in TAS for VMs, see Grant Admin Permissions to an External Group (SAML or LDAP) in *Creating and Managing Users with the UAA CLI (UAAC)*.

   > **Note:** To onboard individual LDAP users and map them to TAS for VMs roles, see Adding Existing SAML or LDAP Users to a Ops Manager Deployment.

10. For **Group search filter**, enter a string that defines LDAP group search criteria. The standard value is `member={0}`. This is required if you are mapping LDAP groups to an admin role.
11. For **Maximum group search depth**, enter a value between `1` and `10` that sets the maximum LDAP group search depth. The default value, `1`, turns off nested group search.
12. For **Server SSL certificate**, paste in the root certificate from your CA certificate or your self-signed certificate. This is required only for `ldaps://` URLs.
13. For **First name attribute** and **Last name attribute**, enter the attribute names in your LDAP directory that correspond to the first and last names in each user record. For example, `cn` and `sn`.
14. For **Email attribute**, enter the attribute name in your LDAP directory that corresponds to the email address in each user record. For example, `mail`.
15. For **Email domains**, enter a comma-separated list of the email domains for external users who are to receive invitations to Apps Manager.
16. For **LDAP referrals**, select how UAA handles LDAP server referrals out to other external user stores. UAA can:
    - **Automatically follow any referrals**.
    - **Ignore referrals and return partial result**.
    - **Throw exception for each referral and abort**.
17. Click **Save**.
18. Return to the Ops Manager Installation Dashboard.
19. Click **Review Pending Changes**.
20. Click **Apply Changes**.

Configure TAS for VMs to Use an OIDC Identity Provider

To integrate the UAA with an external OIDC IDP:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Authentication and Enterprise SSO**.
4. Select **OIDC identity provider**.
5. Set the **Provider name**. This is a unique name you create for the IDP. This name can include only alphanumeric characters, `+`, `_`, and `-`. You must not change this name after deployment because all external users use this name to link to the provider.
6. Enter a **Display name**. Your identity provider display name appears as a link on your Ops Manager login page.
7. For **Provider metadata URL**, enter the OpenID server endpoint URL that returns the configuration metadata for your OIDC identity provider. This URL usually has this format: `https://example-oidc-provider.com/.well-known/openid-configuration`. It serves as a JSON that contains the OIDC provider specifications. For example, its `issuer`, `authorization_endpoint`, and `token_endpoint` values.
8. For **Client ID**, enter the ID of the client registered with the external OIDC provider for use by the UAA. This external OIDC client must be configured according to the following general requirements. If applicable, different OIDC providers might use slightly different naming for these configuration options:
    - **Client**: Per security best practices, register a dedicated client for use by TAS for VMs. Some providers might refer to a “client” as an “app”. Save the client secret for the next step.
    - **Grant type**: Make sure that the client is allowed to perform the `Authorization Code` OAuth flow.
    - **Sign-in redirect URIs**: Include a URL in this format: `https://login.SYSTEM-DOMAIN/login/callback/PROVIDER-NAME`, where:
        - `SYSTEM-DOMAIN` is the domain for system level components. You can find this value under the **Domains** section of the TAS for VMs tile.
        - `PROVIDER-NAME` is the unique name you create for the IDP in the UAA. This can match the **Provider name** value that you previously entered.
    - **Users**: Make sure that the users to whom you want to grant access to TAS for VMs are allowed access to this client or app.
    - **Groups**: Make sure that this client is allowed to obtain all relevant OIDC user groups information that you need to perform mappings from external OIDC user groups to internal TAS for VMs scopes. For example, if you wish to grant TAS for VMs admin scopes to all external OIDC users belonging to the `example-inc-operators` group, you must make sure that this client is allowed to access this group memberships.

    > **Note:** For more information about how to perform external group mappings, see [Grant Admin Permissions to an External Group (SAML, LDAP, or OIDC)](../uaa/uaa-user-management.html#external-group).

9. For **Client secret**, enter the client secret of the client whose ID is entered in the previous step.
10. For **Scopes**, enter a comma-separated list of scopes to request from the external OIDC provider. Per OIDC 1.0 specifications, requesting certain scopes from an OIDC provider can result in receiving additional claims. For example, requesting `openid, profile, email` scopes (the current default value for this input field) can allow the UAA to obtain claims containing the user's first name, last name, and email. Additionally, please include the scope or scopes, for example, `groups`) that allows the UAA to obtain the user's group memberships. Consult your OIDC provider's documentation to decide the exact scopes to request to obtain sufficient claims.
11. For **Email domains**, enter a comma-separated list of the email domains for external users to invite to Apps Manager.
12. For **Username claim**, enter the attribute in your OIDC provider claim that corresponds to the TAS for VMs username.
13. For **First name claim**, enter the attribute in your OIDC provider claim that corresponds to the user's first name in TAS for VMs.
14. For **Last name claim**, enter the attribute in your OIDC provider claim that corresponds to the user's last name in TAS for VMs.
15. For **Email claim**, enter the attribute in your OIDC provider claim that corresponds to the user's email in TAS for VMs.
16. For **External groups claims**, enter the attribute in your OIDC provider claim that defines the groups to which a user belongs. This can be a comma-separated list, and can be used to map to internal TAS for VMs scopes.
17. Click **Save**.
18. Return to the Ops Manager Installation Dashboard.
19. Click **Review Pending Changes**.
20. Click **Apply Changes**.

⚠ **Caution:** As of TAS for VMs v4.0, when an external OIDC provider login is configured, and when you successfully log out from the UAA UI or other TAS for VMs UIs, the UAA attempts to log the users out of the OIDC provider as well. This external OIDC provider logout operation might fail due to a known issue, that results in producing an error page from the provider. If you do not want to log out of the external OIDC provider, you can ignore this error. If you want to log out of the external OIDC provider, navigate to the provider login page directly and log out from there.

Configuring CA as an Identity Provider

This topic explains how to configure single sign-on (SSO) between CA and Ops Manager.

Overview

Partnership creation between CA and Ops Manager involves the following steps:

1. Installing and configuring the prerequisites. For more information, see Prerequisites.
2. Configuring CA SSO as an identity provider (IDP). For more information, see Configure CA as the SAML Identity Provider for Ops Manager.
3. Configuring the service provider (SP). For more information, see Configure Ops Manager as the SAML Service Provider for CA Single Sign-On.

Prerequisites

To configure SSO between CA and Ops Manager, you must have:

- An installation of CA SSO v12.52 or later.
- Configured user store and session store.
- A signed certificate by a certificate authority (CA).
- A protected IDP URL with CA SSO by creating:
    - Authentication scheme
    - Domain
    - Realm
    - Rules and policy
- An Ops Manager environment at https://console.SYSTEM-DOMAIN, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.

Configuring CA as the SAML Identity Provider for Ops Manager

To configure CA SSO as the SAML IDP for Ops Manager:

1. Download the SP metadata.
    1. Go to `https://login.SYSTEM-DOMAIN/saml/metadata`, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.
    2. Log in to CA SSO.
    3. Go to **Federation**.
    4. Select **Partnership Federation**
    5. In the **Actions** menu, select **Export Metadata**.
    6. Save the exported metadata in an XML file.
2. Follow the procedure in Configure TAS for VMs as a Service Provider for SAML in *Configuring Authentication and Enterprise SSO for TAS for VMs* to set the IDP metadata on Ops Manager.
3. Paste the contents of the XML file into the **Provider metadata** field.
4. Click **Save**.

5. Return to the Ops Manager Installation Dashboard.
6. Click **Review Pending Changes**.
7. Click **Apply Changes**.

Configuring Ops Manager as the SAML Service Provider for CA Single Sign-On

This section explains how to configure Ops Manager as the SAML SP for CA SSO.

Configure Identity Provider and Service Provider Entities

To configure IDP and SP entities in CA SSO:

1. Go to `https://login.SYSTEM-DOMAIN/`, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.
2. Log in to CA SSO.
3. Go to **Federation**.
4. Click **Partnership Federation**.
5. Click **Entity**.
6. Click **Create Entity**.
7. To create a local entity, configure the fields with the following values:
    - **Entity Location:** Local
    - **Entity Type:** SAML2 IDP
    - **Entity ID:** Enter an ID for your local identity provider. For example, `https://ca-technologies.xxx.com`.
    - **Entity Name:** Create a name for your local identity provider.
    - **Base URL:** Enter the fully-qualified domain name for the host service CA SSO Federation Web Services.
    - **Signing Private Key Alias:** Select the private key alias or import a private key.
    - **Signed Authentication Requests Required:** Select **No**.
    - **Supported NameID format:** Enter `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` and `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` to select both email address and unspecified as supported NameID formats.
8. To create a remote entity:
    1. Click **Import Metadata Button**.
    2. Download the SP metadata from https://login.SYSTEM-DOMAIN/saml/metadata, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.
    3. Save the SP metadata to an XML file.
    4. Browse and select the saved XML metadata you downloaded in the previous step.

5. Provide a name for the **Remote Service Provider Entity**.
6. Provide an alias for the **Signing Certificate** imported from the metadata.

   > **Note:** Ops Manager signs the outgoing SAML authentication requests.

7. Click **Save**.

Configure Partnership Between CA SSO and Ops Manager

To configure a partnership between CA SSO and Ops Manager:

1. Go to `https://login.SYSTEM-DOMAIN/`, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.
2. Log in to CA SSO.
3. Go to **Federation**.
4. Click **Partnership Federation**.
5. Click **Create Partnership**.
6. To configure the partnership, configure the fields with the following values:
   - **Add Partnership Name:** Enter a name for your partnership.
   - (Optional) **Description:** Enter a relevant description for your partnership.
   - **Local IPD ID:** Enter the Local Service Provider ID you created in Configure Identity Provider and Service Provider Entities.
   - **Remote SP ID:** Enter the Remote SP ID you created in Configure Identity Provider and Service Provider Entities.
   - **Base URL:** This field will be pre-populated.
   - **Skew Time:** Enter any skew time required by your environment.
   - **User Directories and Search Order:** Select the required directories in the required search order.
7. Click **Next**.
8. On the Federation Users page, accept the default values.
9. Click **Next**.
10. To complete the **Name ID Format** section:
    1. Select **Email Address** from the **Name ID Format** dropdown.
    2. Select **User Attribute** from the **Name ID Type** dropdown.

       > **Note:** Ops Manager does not support processing SAML Assertion Attributes at this time. You can skip filling out the **Assertion Attributes** fields.

11. Click **Next**.
12. To complete the **SSO and SLO** section:
    1. Enter the **Authentication URL** that is protected by CA SSO under prerequisites.
    2. For **SSO Binding**, click **HTTP-POST**.
    3. In the **Audience** field, enter http://login.SYSTEM-DOMAIN, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.

       > **Note:** The **Audience** field requires `http://` instead of `https://`. This is only a naming convention within the schema and does not determine connection security.

    4. Select **Both IDP and SP Initiated** from the **Transactions Allowed** dropdown.
    5. The **Assertion Consumer Service URL** field is be pre-populated using information from the SP entity.
13. Click **Next**.
14. To complete the **Configure Signature and Encryption** section:
    1. In the **Signing Private Key Alias** dropdown, verify that the correct **Private Key Alias** is selected.
    2. Verify that the correct **Verification Certificate Alias** is selected in the **Verification Certificate Analysis** dropdown. This alias should be the same certificate created when you imported the remote SP entity ID in Remote Service Provider Entity ID.
    3. Select **Sign Both** from the **Post Signature Options** dropdown.

       > **Note:** Ops Manager does not support encryption options at this time.

    4. Click **Finish**.
15. To activate the partnership, expand the **Action** dropdown for your partnership and click **Activate**.

Configuring PingFederate as an Identity Provider

This topic explains how to configure single sign-on (SSO) between PingFederate and VMware Tanzu Application Service for VMs (TAS for VMs).

Overview

Partnership creation between PingFederate and TAS for VMs involves the following steps:

1. Configuring PingFederate as an identity provider (IDP). For more information, see Configure PingFederate as the SAML Identity Provider for TAS for VMs.

2. Configuring the service provider (SP). For more information, see Configure TAS for VMs as the SAML Service Provider for PingFederate.

Configure PingFederate as the SAML Identity Provider for TAS for VMs

To configure PingFederate as the SAML IDP for your TAS for VMs tile:

1. Download your IDP metadata from PingFederate Server:
    1. Log in to PingFederate Administrative Console.
    2. Select **Administrative Functions**.
    3. Click **Metadata Export**.
    4. If your PingFederate server is configured to act as both an IDP and an SP, indicate which type of configuration you want to export. The Signing key can be exported. You can skip the options related to encryption keys and metadata attribute contract because they are not supported at this time.
    5. Click **Next**.
2. Follow the procedure in Configure TAS for VMs as a Service Provider for SAML in *Configuring Authentication and Enterprise SSO for TAS for VMs* to set the IDP metadata on TAS for VMs.

Configure TAS for VMs as the SAML Service Provider for PingFederate

To configure TAS for VMs as the SAML SP for PingFederate:

1. Download the SP metadata from https://login.SYSTEM-DOMAIN/saml/metadata, where `SYSTEM-DOMAIN` is the system domain of your Ops Manager installation.
2. Save the SP metadata to an XML file.
3. Import the SP metadata to PingFederate:
    1. Log in to PingFederate Administrative Console.
    2. Under **Main Menu**, select **IdP Configuration**.
    3. Select **SP Connection**.
    4. Click **Import**.
    5. In the **Import Connection** pane, browse and select the `.xml` file downloaded in the previous step.
    6. Click **Import**.
    7. Click **Done**.
4. TAS for VMs expects the NameID format to be an email address, such as `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`, and the value to be the email address of the currently logged-in user. The SSO does not function without this setting.

    1. Under **Main Menu**, click the connection name. To see a full list of connections, click **Manage All SP**.
    2. Under the **SP Connection**, select **Browser SSO**.
    3. Click **Configure Browser SSO**.
    4. Under **Browser SSO**, select **Assertion Creation**.
    5. Select **Configure Assertion Creation**.
    6. On the **Summary** pane, select **Identity Mapping**.
    7. Select **Standard**.
    8. For the NameID format, select **Email Address** and enter the email address of the user.

5. Select the Authentication Source:
    1. Under **SP Connection**, select **Browser SSO**.
    2. Select **Configure Browser SSO**.
    3. Under **Browser SSO**, select **Assertion Creation**.
    4. Select **Configure Assertion Creation**.
    5. On the **Summary** pane, select **IdP Adapter Mapping**.
    6. Select **Adapter Instance Name**.
    7. On the **Summary** pane, select **Adapter Instance**.
6. Enable the SSO Browser Profiles:
    1. Under **SP Connection**, select **Browser SSO**.
    2. Select **Configure Browser SSO**.
    3. On the **Summary** pane, select **SAML Profiles**.
    4. Ensure the **IdP-Initiated SSO** and **SP-Initiated SSO** check boxes are selected.

    > **Note:** TAS for VMs does not support SLO profiles at this time. You can leave them deactivated.

7. Activate the SP Connection.

Traffic and Security

In this section:

- **Front End**
    - Securing Traffic into TAS for VMs
    - Configuring SSL/TLS Termination at HAProxy
    - Securing System and App Endpoints
    - Configuring SSH Access for TAS for VMs

- App SSH Components and Processes
- Configuring Front End Idle Timeout for Gorouter and HAProxy

- **Load Balancing and Routing**
  - Using Your Own Load Balancer
  - Enabling and Configuring TCP Routing
  - Enabling IPv6 for Hosted Apps
  - Supporting HTTP/2
  - Configuring Proxy Settings for All Apps
  - Switching App Domains
  - Configuring Load Balancer Health Checks for TAS for VMs Routers
  - Configuring Route Service Lookup
- **Internal App Security**
  - Container Security
  - Container-to-Container Networking
  - App Security Groups
  - Restricting App Access to Internal TAS for VMs Components
- **Certificates and Credentials**
  - Rotating Runtime CredHub Encryption Keys
  - Rotating the Cloud Controller Database Encryption Key
  - Securing Service Instance Credentials with Runtime CredHub
  - Providing a Certificate for Your TLS Termination Point
  - Trusted System Certificates
- **Component Communications**
  - BOSH DNS Network Communications
  - Cloud Controller Network Communications
  - Container-to-Container Networking Communications
  - CredHub Network Communications
  - Diego Network Communications
  - Loggregator Network Communications
  - MySQL Network Communications
  - NATS Network Communications
  - Routing Network Communications
  - UAA Network Communications

Front End

In this section:

- Securing Traffic into TAS for VMs
- Configuring SSL/TLS Termination at HAProxy
- Securing System and App Endpoints
- Configuring SSH Access for TAS for VMs
- App SSH Components and Processes
- Configuring Front End Idle Timeout for Gorouter and HAProxy

Securing incoming traffic in TAS for VMs

You can secure HTTP traffic into your TAS for VMs deployment with TLS certificates. This article describes how. You can also configure the location where your deployment stops TLS depending on your needs and certificate restrictions.

Protocol support

The Gorouter supports HTTP/HTTPS requests only. For more information about features supported by the Gorouter, see HTTP Routing.

You can force HTTPS-only connections by enabling HSTS on HAProxy. For more information, see the Secure Apps Domain with HAProxy section of the *Securing System and App Endpoints* topic.

To secure non-HTTP traffic over TCP routing, terminate TLS at your load balancer or at the app. For more information, see Enabling TCP Routing.

TLS termination options for HTTP routing

There are several options for terminating TLS for HTTP traffic. You can terminate TLS at the Gorouter, your load balancer, or both.

The following table summarizes TLS termination options and which option to choose for your deployment.

> **Note:** To ensure traffic is sent to the platform securely, VMware recommends as a minimum that you terminate TLS at the Gorouter. You can optionally terminate TLS at the load balancer.

If the following applies to you:	Then configure TLS termination at:	Related topic and configuration procedure

• You want the optimum balance of performance and security, and • You want to make minimum changes to your load balancer, or • You are deploying TAS for VMs to AWS. For information about AWS limitations, see TLS Cipher Suite Support by AWS ELBs.	**Gorouter only**	Terminating TLS at the Gorouter Only
• You require TLS termination at a load balancer, or • You want the highest level of security, and • You do not mind a slightly less performant deployment.	**Load Balancer and Gorouter**	Terminating TLS at the Load Balancer and Gorouter
• You require TLS termination at a load balancer, and • You prefer unencrypted traffic between the Load Balancer and the Gorouter.	**Load Balancer only**	Terminating TLS at the Load Balancer Only
Optionally, if you are deploying HAProxy, and	**Then in addition, terminate SSL/TLS at:**	**Related topic and configuration procedure**
• You would like to secure traffic to the HAProxy.	**HAProxy**	Terminating SSL/TLS at HAProxy

Certificate requirements

The following requirements apply to the certificates you use to secure traffic into TAS for VMs:

- You must obtain at least one TLS certificate for your environment.
  - In a production environment, use a signed TLS certificate (trusted) from a known certificate authority (CA).
  - In a development or testing environment, you can use a trusted CA certificate or a self-signed certificate. You can generate a self-signed certificate with `openssl` or a similar tool.

> **Note**: Alternately, you can use the TAS for VMs Ops Manager interface to generate a certificate for you. Certificates generated in TAS for VMs are signed by the Ops Manager Certificate Authority. They are not technically self-signed, but they are sometimes referred to as "self-signed certificates" in the Ops Manager UI and throughout this documentation. For more information, see the Multiple certificates

In order to support custom domains on TAS for VMs, an operator has to configure the Gorouter with a certificate that represents the domain. VMware recommends that operators add a new certificate instead of reissuing a single certificate when adding TLS support for an additional domain. Using multiple certificates provides a security benefit in that it prevents clients from discovering all the custom domains of apps running on a TAS for VMs platform.

The Gorouter supports SNI and can be configured with multiple certificates, each which may optionally include wildcard and alternative names. The Gorouter uses SNI to determine the correct certificate to present in a TLS handshake. It requires clients to support the SNI protocol by sending a server name outside the encrypted request payload. For clients that do not support SNI, the Gorouter presents a default certificate. The default is the first certificate keypair in the Gorouter's configuration.

The Gorouter decides which certificate to provide in the TLS handshake as follows:

- If a client provides an SNI header with a ServerName that matches to a configured certificate keypair, the Gorouter returns the matching certificate.
- If a client provides an SNI header with a ServerName that does not match a configured certificate keypair, the Gorouter returns the default certificate.

The first certificate keypair listed is used as the default.

The Gorouter supports both RSA and ECDSA certificates in PEM encoding. In the case that a certificate chain is required, the order should be as follows: primary certificate, intermediate certificate, then root certificate.

How to configure multiple certificate keypairs

To configure multiple HTTPS certificate keypairs for TAS for VMs, add each keypair along with a meaningful name in the applicable **Certificates and private keys for the Gorouter and HAProxy** fields of the **Networking** pane in TAS for VMs. For more information, see Configure Networking in *Configuring TAS for VMs*.

In TAS for VMs, multiple certificates configured for the Gorouter are also configured for HAProxy.

TLS cipher suite support

Some TAS for VMs components like the Gorouter support additional TLS cipher suites to accommodate older clients. As a security best practice, only configure the TLS cipher suites that you need for your deployment.

Default Gorouter cipher suites

By default, the Gorouter supports the following TLS cipher suites:

TLS Version	RFC	OpenSSL
1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
1.3	TLS_AES_128_GCM_SHA256	TLS13-AES-128-GCM-SHA256
1.3	TLS_AES_256_GCM_SHA384	TLS13-AES-256-GCM-SHA384
1.3	TLS_CHACHA20_POLY1305_SHA256	TLS13-CHACHA20-POLY1305-SHA256

You can override the default cipher suites in the **Networking** pane of the TAS for VMs tile. For more information about using custom SSL ciphers, see Gorouter Only or Load Balancer and Gorouter.

TLS cipher suite support by AWS load balancers

AWS Classic Load Balancers (formerly referred to as ELBs) support configuration of cipher suites for front end connections with clients only. When configuring Classic Load Balancers to forward requests to Gorouters over TLS, operators may encounter a `Cipher Suite Mismatch` error. This is because the cipher suites supported by Classic Load Balancers for TLS handshakes with back ends (Gorouters in this case) are hardcoded, undocumented, and do not support the Gorouter default cipher suites.

You can configure TLS termination in one of the following ways:

- Configure Classic Load Balancer listeners in TCP mode. This allows TCP connections from clients to pass through the Classic Load Balancer to Gorouters on port 443. With this configuration, Gorouters are the first point of TLS termination.
- If you require TLS termination at an AWS load balancer in addition to terminating at the Gorouter, use AWS Application Load Balancers (ALBs) that support the Gorouter default cipher suites.

TLS v1.3

You cannot configure cipher suites for TLS v1.3. Gorouter only supports the defaults listed in Default Gorouter Cipher Suites.

Note: To support TLS v1.3, ensure the Gorouter is configured with certificates generated with a key larger than 512 bits.

TLS v1.2

The following cipher suites are optionally supported for TLS v1.2 only:

RFC	OpenSSL
TLS_RSA_WITH_AES_128_GCM_SHA256	AES128-GCM-SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384	AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	ECDHE-ECDSA-RC4-SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE-ECDSA-AES128-SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE-ECDSA-AES256-SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA	ECDHE-RSA-RC4-SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE-RSA-DES-CBC3-SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE-RSA-AES128-SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE-RSA-AES256-SHA
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE-ECDSA-AES128-GCM-SHA256

RFC	OpenSSL
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE-RSA-AES256-GCM-SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE-ECDSA-AES256-GCM-SHA384
TLS_RSA_WITH_AES_128_CBC_SHA256	AES128-SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE-ECDSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE-RSA-AES128-SHA256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305	ECDHE-RSA-CHACHA20-POLY1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305	ECDHE-ECDSA-CHACHA20-POLY1305

TLS v1.0 and v1.1

The following cipher suites are optionally supported for TLS v1.0 and TLS v1.1 only:

RFC	OpenSSL
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
TLS_RSA_WITH_AES_128_CBC_SHA	AES128-SHA
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA

You can override the default cipher suites in the **Networking** pane of the TAS for VMs tile. For more information about using custom SSL ciphers, see Gorouter Only or Load Balancer and Gorouter.

For more information about supported ciphers, see Golang Constants in the Golang repository on GitHub and Ciphers in the OpenSSL documentation.

Note: ECDSA ciphers require a certificate and key for DSA, as opposed to RSA.

Mutual authentication with clients

The Gorouter supports validation of client certificates in TLS handshakes with clients, also known as mutual authentication. Operators can choose whether the Gorouter requests client certificates and when requesting certificates, whether or not to require them.

By default, the Gorouter requests but does not require client certificates in TLS handshakes.

To configure Gorouter behavior for handling client certificates, select one of the options in the **Gorouter behavior for client certificate validation** field of the **Networking** pane in TAS for VMs:

- **The Gorouter does not request client certificates:** Client certificates are not requested, so the client does not provide them, and validation of client certificates does not occur. This option is incompatible with the **TLS termination point** options **HAProxy** and **Gorouter** because these options require mutual authentication.

- **The Gorouter requests but does not require client certificates:** The Gorouter requests client certificates in TLS handshakes and validates them when presented, but does not require them. This is the default configuration.
- **The Gorouter requires client certificates:** The Gorouter validates that the client certificate is signed by a Certificate Authority that the Gorouter trusts. If the Gorouter cannot validate the client certificate, the TLS handshake fails.

The behavior controlled by this property is global; it applies to all requests received by Gorouters so configured.

If the Gorouter is the first point of TLS termination (your load balancer does not terminate TLS, and passes the request through to Gorouter over TCP), consider:

- Only the **The Gorouter does not request client certificates** option should be used with TAS for VMs, as the Gorouters in TAS for VMs receive requests for the system domain. Many clients of Ops Manager platform APIs do not present client certificates in TLS handshakes, so the first point of TLS termination for requests to the system domain must not request them.
- All options may be used for Gorouters deployed with the Isolation Segment tile, as these only receive requests for app domains.
- The **The Gorouter requests but does not require client certificates** and **The Gorouter requires client certificates** options trigger browsers to prompt users to select a certificate if the browser is not already configured with a certificate signed by one of the CAs configured for the Gorouter.

If the Gorouter is not the first point of TLS termination, this property can be used to secure communications between the load balancer and Gorouter. The Gorouter must be configured with the CA used to sign the client certification the load balancer presents.

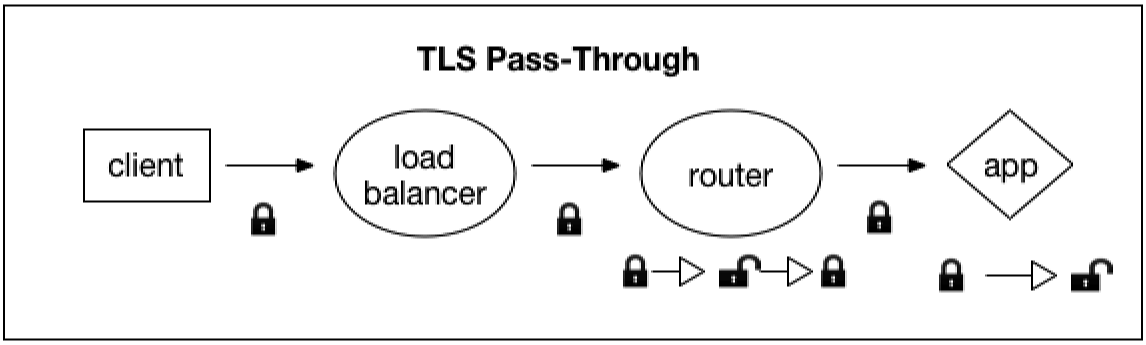
> ⚠ **Caution:** Requests to the platform fail upon upgrade if your load balancer is configured to present a client certificate in the TLS handshake with the Gorouter but the Gorouter has not been configured with the certificate authority used to sign it. To mitigate this issue, select **The Gorouter does not request client certificates** or configure the Gorouter with the appropriate CA.

Terminating TLS at the Gorouter only

In this configuration, the load balancer does not terminate TLS for TAS for VMs domains at all. Instead, it passes through the underlying TCP connection to the Gorouter.

VMware recommends this more performant option, establishing and terminating a single TLS connection.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



Traffic passes from the encrypted client, to the load balancer, to the router, and traffic terminates at the app. Traffic between the load balancer and the Gorouter is encrypted only if the client request is encrypted.

Note: Traffic between the Gorouter and app is encrypted with TLS, unless a Windows stemcell is being used.

About HTTP header forwarding

If you terminate TLS at the Gorouter only, your load balancer does not send HTTP headers.

The Gorouter appends the `X-Forwarded-For` and `X-Forwarded-Proto` headers to requests forwarded to apps and platform system components.

`X-Forwarded-For` is set to the IP address of the source. Depending on the behavior of your load balancer, this may be the IP address of your load balancer. For the Gorouter to deliver the IP address of the client to apps, configure your load balancer to forward the IP address of the client or configure your load balancer to send the client IP address using the PROXY protocol.

`X-Forwarded-Proto` provides the scheme of the HTTP request from the client. The scheme is HTTP if the client made a request on port 80 (unencrypted) or HTTPS if the client made a request on port 443 (encrypted). The Gorouter sanitizes the `X-Forwarded-Proto` header based on the settings of the `router.sanitize_forwarded_proto` and `router.force_forwarded_proto_https` manifest properties as follows:

- `router.sanitize_forwarded_proto: true` and `router.force_forwarded_proto_https: true`: The Gorouter sets the value of `X-Forwarded-Proto` header to `HTTPS` in requests forwarded to back ends.
- `router.sanitize_forwarded_proto: true` and `router.force_forwarded_proto_https: false`: The Gorouter strips the `X-Forwarded-Proto` header when present in requests from front end clients. When the request is received on port 80 (unencrypted), the Gorouter sets the value of this header to `HTTP` in requests forwarded to back ends, When the request is received on port 443 (encrypted), the Gorouter sets the value of this header to `HTTPS`.
- `router.sanitize_forwarded_proto: false` and `router.force_forwarded_proto_https: true`: The Gorouter passes through the header as received from the load balancer, without modification.

- `router.sanitize_forwarded_proto: false` and `router.force_forwarded_proto_https: false`: The Gorouter passes through the header as received from the load balancer, without modification.

> **Note:** If you configured TLS to terminate at the Gorouter only, VMware recommends setting `router.sanitize_forwarded_proto: true` and `router.force_forwarded_proto_https: false` to secure against header spoofing.

For more information about HTTP headers in TAS for VMs, see HTTP Headers in *HTTP Routing*. For information about configuring the forwarding of client certificates, see Forward Client Certificate to Apps in *HTTP Routing*.

Procedure: Gorouter only

To configure SSL termination on the Gorouter in TAS for VMs:

1. Configure your load balancer to pass through TCP requests from the client to the Gorouter.
2. Navigate to the Ops Manager Installation Dashboard.
3. Click the TAS for VMs tile.
4. Select **Networking**.
5. For TAS for VMs deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the **Gorouter IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see Configure Networking in *Configuring TAS for VMs*.
6. Under **Certificates and private keys**:
    1. Click **Add** to define at least one certificate and key pair for the Gorouter.
    2. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in TAS for VMs. To create a certificate for your wildcard domains, see Creating a Wildcard Certificate for Ops Manager Deployments in *Providing a Certificate for Your TLS Termination Point*.
7. In the **Minimum version of TLS supported by the Gorouter and HAProxy**, select the minimum version of TLS to use in Gorouter communications. The Gorouter uses TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.
8. Under **TLS termination point**, select **Gorouter**.
9. To use a specific set of TLS ciphers for the Gorouter, configure **TLS cipher suites for the Gorouter**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of

TLS, enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.

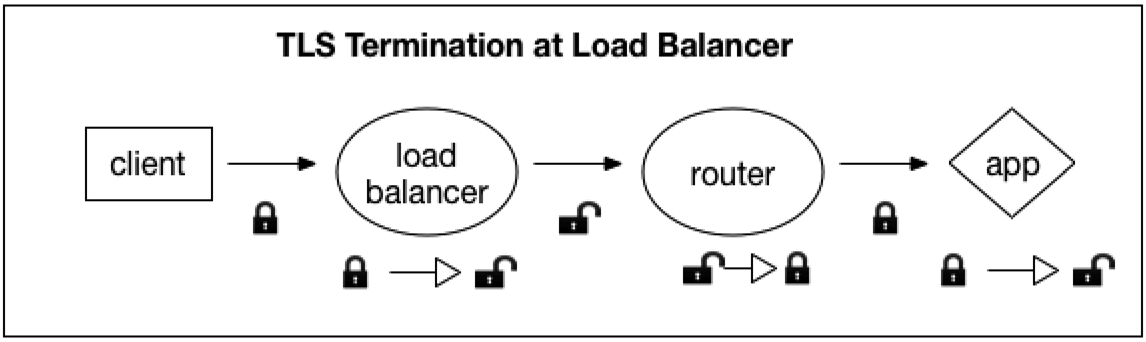
10. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Disable**.
11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on the Gorouter and HAProxy** checkbox.
12. Click **Save**.
13. Select **Resource Config**.
14. In the **Instances** dropdown for the **HAProxy** job, select `0` instances.
15. Click **Save**.

Terminating TLS at the load balancer only

In this configuration, your load balancer terminates TLS and passes unencrypted traffic to the Gorouter, which routes it to your app. Traffic between the load balancer and the Gorouter is not encrypted.

VMware recommends this option if you cannot use SAN certificates and if you do not require traffic to be encrypted between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



The diagram shows configuration of the the client, load balancer, Gorouter, and app when the TLS terminates at the load balancer with lock icons.

Traffic starts at the encrypted client, passes through the load balancer to the router, and terminates at the app. Traffic is not encrypted past the load balancer.

Note: Traffic between the Gorouter and app is encrypted with TLS, unless a Windows stemcell is being used.

About HTTP header forwarding

If you terminate TLS at your load balancer, you must also configure the load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` HTTP headers to the HTTP traffic it passes to the Gorouter.

For more information about HTTP headers in TAS for VMs, see HTTP Headers in *HTTP Routing*. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Apps in *HTTP Routing*.

Procedure: Load balancer only

To configure SSL termination on the load balancer only in TAS for VMs:

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System domain** and **Apps domain** that you configure in the **Domains** pane of the TAS for VMs tile with the IP address of your load balancer.

   For example, with `pcf.example.com` as the main subdomain for your Ops Manager deployment and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.pcf` to `198.51.100.1`.

Name	Type	Data	Domain
*.pcf	A	198.51.100.1	example.com

2. Navigate to the Ops Manager Installation Dashboard.
3. Click the TAS for VMs tile.
4. Select **Networking**.
5. For TAS for VMs deployments on OpenStack or vSphere, choose IP addresses for the Gorouters from the subnet configured for Ops Manager and enter them in the **Gorouter IPs** field. Then configure your load balancer to forward requests for the above domains to these IP addresses. For more information, see Configure Networking in *Configuring TAS for VMs*.
6. In the **Certificates and private keys for the Gorouter and HAProxy** field, click the **Add** button to define one certificate keypair for the Gorouter and HAProxy. Since you have opted for unencrypted traffic behind the load balancer, then you can generate an RSA certificate in TAS for VMs.
7. In the **Minimum version of TLS supported by the Gorouter and HAProxy**, select the minimum version of TLS to use in HAProxy communications. HAProxy use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see Cipher Suites.
8. Under **TLS termination point**, select **Infrastructure load balancer**.
9. To use a specific set of TLS ciphers for HAProxy, configure **TLS cipher suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, enter cipher suites supported by this version. Otherwise, leave the default values in this field.
10. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Disable**.
11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, enable the **Disable HTTP on the Gorouter and HAProxy** checkbox.

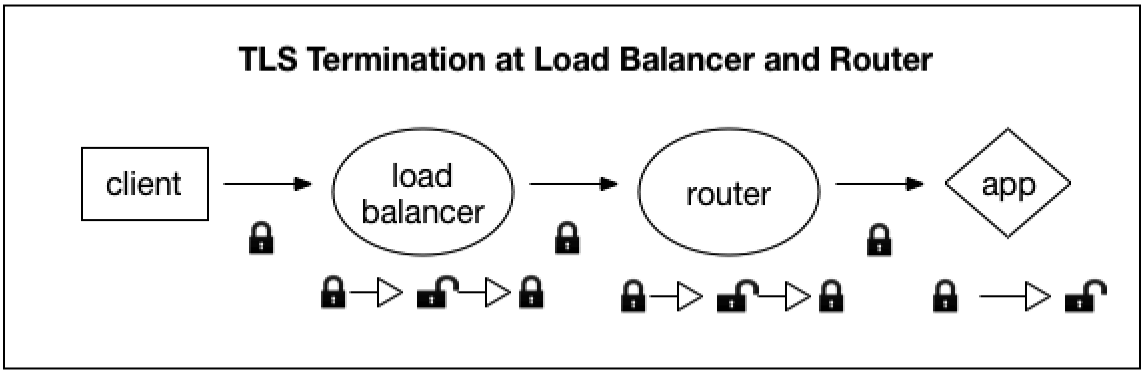
12. Click **Save**.
13. After you complete the configuration in TAS for VMs, add your certificate or certificates to your load balancer and configure its listening port. The procedures vary depending on your IaaS.
14. Configure your load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` headers to client requests.
    If the load balancer cannot be configured to provide the `X-Forwarded-For` header, the Gorouter appends it in requests forwarded to apps and system components, set to the IP address of the load balancer.

    > **Note:** If the load balancer accepts unencrypted requests, it **must** provide the `X-Forwarded-Proto` header. Conversely, if the load balancer cannot be configured to send the `X-Forwarded-Proto` header, it should not accept unencrypted requests. Otherwise, apps and platform system components that require encrypted client requests accept unencrypted requests when they should not accept them.

Terminating TLS at the load balancer and Gorouter

In this configuration, two TLS connections are established: one from the client to the load balancer, and another from the load balancer to the Gorouter. This configuration secures all traffic between the load balancer and the Gorouter.

The following diagram illustrates communication between the client, load balancer, Gorouter, and app.



Traffic starts at the encrypted client, moves through the load balancer to the router, and terminates at the unencrypted app. Traffic is encrypted between the client and load balancer, and between the load balancer and router.

This option is less performant, but allows for termination at a load balancer, as well as secure traffic between the load balancer and the Gorouter.

> **Note:** Traffic between the Gorouter and app is encrypted with TLS, unless a Windows stemcell is being used.

Certificate guidelines

In this deployment scenario, the following guidelines apply:

- Certificates for the TAS for VMs domains must be stored on the load balancer, as well as on the Gorouter.
- Generate certificates for your load balancer and the Gorouter with different keys. If the key for the certificate on the Gorouter is compromised, then the certificate on the load balancer is not at risk, and vice-versa.
- If you choose to host only one certificate on the Gorouter and many on your load balancer, configure your load balancer with the CA and host name with which to validate the certificate hosted by the Gorouter.

About host name verification

Host name verification between the load balancer and the Gorouter is unnecessary when the load balancer is already configured with the Gorouter's IP address to correctly route the request.

If the load balancer uses DNS resolution to route requests to the Gorouters, then you should enable host name verification.

About HTTP header forwarding

If you terminate TLS at your load balancer, you must configure the load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` HTTP headers to requests it sends to the Gorouter.

If you terminate TLS at your load balancer but it does not support HTTP, such that it cannot append HTTP headers, a workaround exists. You should use this workaround **only if your load balancer does not accept unencrypted requests**. Configure your load balancer to send the client IP address using the PROXY protocol, and enable PROXY in the Gorouter. As the `X-Forwarded-Proto` header is stripped, configure the Gorouter to force-set this header to 'HTTPS'.

For more information about HTTP headers in TAS for VMs, see HTTP Headers in *HTTP Routing*. If you are configuring the forwarding of client certificates, see Forward Client Certificate to Apps in *HTTP Routing*.

Procedure: Load balancer and Gorouter

To configure SSL termination on the Gorouter and load balancer in TAS for VMs:

1. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System domain** and **Apps domain** that you configure in the **Domains** pane of the TAS for VMs tile with the IP address of your load balancer.

   For example, with `pcf.example.com` as the main subdomain for your Ops Manager deployment and a load balancer IP address `198.51.100.1`, you must create an A record in your DNS that serves `example.com` and points `*.pcf` to `198.51.100.1`.

Name	Type	Data	Domain
*.pcf	A	198.51.100.1	example.com

2. Navigate to the Ops Manager Installation Dashboard.
3. Click the TAS for VMs tile.
4. Select **Networking**.
5. For TAS for VMs deployments on OpenStack or vSphere:
    1. Choose IP addresses for the Gorouters from the subnet configured for Ops Manager.
    2. In **Gorouter IPs**, enter the IP addresses you chose in the previous step.
    3. Configure your load balancer to forward requests for the above domains to the IP addresses you configured in the previous step. For more information, see Configure Networking in *Configuring TAS for VMs*.
6. Under **Certificates and private keys**:
    1. Click **Add** to define at least one certificate and key pair for the Gorouter.
    2. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in TAS for VMs. To create a certificate for your wildcard domains, see Creating a Wildcard Certificate for Ops Manager Deployments in *Providing a Certificate for Your TLS Termination Point*.
7. In the **Minimum version of TLS supported by the Gorouter and HAProxy**, select the minimum version of TLS to use in HAProxy and Gorouter communications. The Gorouter uses TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites.
8. If you are using HAProxy:
    1. Under **TLS termination point**, select **Infrastructure load balancer**.
    2. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Enable**.
    3. In the **Certificate authority for HAProxy back end** field, specify the certificate authority (CA) that signed the certificate you configured in the **Certificates and private keys for the Gorouter and HAProxy** field.

    > **Note:** If you used the **Generate RSA Certificate** link to generate a self-signed certificate, then the CA to specify is the Ops Manager CA, which you can locate using the `/api/v0/certificate_authorities` endpoint in the Ops Manager API.

    4. If you want to use a specific set of TLS ciphers for HAProxy, configure **TLS cipher suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, you can enter cipher suites supported by this version. Otherwise, leave the default values in this field.

    5. Proceed to step 11.

9. To use a specific set of TLS ciphers for the Gorouter, configure **TLS cipher suites for the Gorouter**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, enter cipher suites supported by this version. For a list of TLS ciphers supported by the Gorouter, see Cipher Suites. Otherwise, leave the default values in this field.

10. If you are not using HAProxy:

    1. Under **TLS termination point**, select any of the available options depending on your client app needs. For more information about XFCC header forwarding, see Forward Client Certificate to Apps in *HTTP Routing*.

    2. Under **HAProxy forwards all requests to the Gorouter over TLS**, select **Disable**.

    3. In the TAS for VMs tile, select **Resource Config**.

    4. In the **Instances** dropdown for the **HAProxy** job, select `0` instances.

    5. Click **Save**.

11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on the Gorouter and HAProxy** checkbox.

12. Click **Save**.

13. After you complete the configuration in TAS for VMs, add your certificate or certificates to your load balancer and configure its listening port. The procedures vary depending on your IaaS.

14. Configure your load balancer to append the `X-Forwarded-For` and `X-Forwarded-Proto` headers to client requests.
    If you cannot configure the load balancer to provide the `X-Forwarded-For` header, the Gorouter appends it in requests forwarded to apps and system components, set to the IP address of the load balancer.

    > **Note:** If the load balancer accepts unencrypted requests, it **must** provide the `X-Forwarded-Proto` header. Conversely, if the load balancer cannot be configured to send the `X-Forwarded-Proto` header, it should not accept unencrypted requests. Otherwise, apps and platform system components that require encrypted client requests accept unencrypted requests when they should not accept them.

Configuring SSL/TLS Termination at HAProxy

This topic describes how to configure SSL/TLS termination at HAProxy in VMware Tanzu Application Service for VMs (TAS for VMs) and Isolation Segment.

Overview

Both TAS for VMs and Isolation Segment include an HAProxy instance.

HAProxy is appropriate to use in a deployment when features are needed that are offered by HAProxy but are not offered by the Gorouters or IaaS-provided load balancers, such as with Azure load balancers. These include filtering of protected domains from trusted networks.

While HAProxy instances provide load balancing for the Gorouters, HAProxy is not itself highly available. For production environments, use a highly-available load balancer to scale HAProxy horizontally. The load balancer does not need to terminate TLS or even operate at Layer 7 (HTTP), as it can provide Layer 4 load balancing of TCP connections. Use of HAProxy does not remove the need for Gorouters. The Gorouter must always be deployed for HTTP apps, and the TCP router for non-HTTP apps.

You can generate a self-signed certificate for HAProxy if you do not want to obtain a signed certificate from a certificate authority (CA).

Terminate SSL/TLS at HAProxy

To configure SSL termination on HAProxy in TAS for VMs:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Configure these fields based on the IaaS of your TAS for VMs deployment:

If your TAS for VMs deployment is on:	Then configure:	See also:

OpenStack or vSphere	Decide whether you want your HAProxy to be highly available. ○ If you need highly available HAProxy: 1. Choose an IP address for each HAProxy instance on the subnet where you deployed TAS for VMs. 2. In the **HAProxy IPs** field of the **Networking** pane, enter the IP addresses you have selected for your HAProxy instances. 3. Configure your load balancer, such as F5 or NSX, to forward domain names to the HAProxy IP addresses. ○ If you do not require high availability, such as if you are setting up a development environment: 1. Skip setting up the load balancer. 2. Choose one IP address for the single HAProxy instance. 3. Configure DNS to point at the IP address. For more information, see How to Set Up DNS for HAProxy.	For more information, see Configure Networking in *Configuring TAS for VMs.*
AWS, GCP or Azure	1. Leave the HAProxy IP address blank. 2. In the **Resource Config** pane, locate the HAProxy job. 3. In the **Load Balancer** column for the HAProxy job, specify the appropriate IaaS load balancer resource.	For more information, see Configure Networking in *Configuring TAS for VMs.*

5. In the **Certificates and private keys for the Gorouter and HAProxy** field, click the **Add** button to define at least one certificate keypair for the Gorouter and HAProxy. For each certificate keypair that you add, assign a name, enter the PEM-encoded certificate chain and PEM-encoded private key. You can either upload your own certificate or generate an RSA certificate in TAS for VMs. For options and instructions on creating a certificate for your wildcard domains, see Creating a Wildcard Certificate.

6. In the **Minimum version of TLS supported by the Gorouter and HAProxy**, select the minimum version of TLS to use in HAProxy communications. HAProxy use TLS v1.2 by default. If you need to accommodate clients that use an older version of TLS, select a lower minimum version. For a list of TLS ciphers supported by the HAProxy, see TLS Cipher Suites.

7. Under **HAProxy forwards all requests to the Gorouter over TLS**, leave **Enable** selected and provide the back end certificate authority.

8. To use a specific set of TLS ciphers for HAProxy, configure **TLS cipher suites for HAProxy**. Enter an ordered, colon-separated list of TLS cipher suites in the OpenSSL format. For example, if you have selected support for an earlier version of TLS, enter cipher suites supported by this version. For a list of TLS ciphers supported by the HAProxy,see TLS Cipher Suites.

9. If you expect requests larger than the default maximum of 16.384 KB, enter a new value in bytes for **HAProxy request maximum buffer size**. You may need to do this, for example,

to support apps that embed a large cookie or query string values in headers.

10. To force browsers to use HTTPS when making requests to HAProxy, select **Enable** in the **HAProxy support for HSTS** field and complete these optional configuration steps:
    - **Maximum age** in seconds for the HSTS request. HAProxy forces HTTPS requests from browsers for the duration of this setting. The maximum age is one year, or 31536000 seconds.
    - Enable the **Include subdomains** check box to force browsers to use HTTPS requests for all component subdomains.
    - Select the **Enable preload** check box to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from HAProxy.
11. (Optional) If you do not want the Gorouter to accept any non-encrypted HTTP traffic, select the **Disable HTTP on the Gorouter and HAProxy** check box.
12. Under **TLS termination point**, select **Infrastructure load balancer**.
13. (Optional) If your TAS for VMs deployment uses HAProxy and you want it to receive traffic only from specific sources, configure these fields:
    - **HAProxy protected domains**: Enter a comma-separated list of domains from which TAS for VMs can receive traffic.
    - (Optional) **HAProxy trusted CIDRs**: Enter a space-separated list of CIDRs to limit which IP addresses from the protected domains can send traffic to TAS for VMs.
14. Click **Save**.

Set Up DNS for HAProxy

You only need to perform this procedure if you are using one instance of HAProxy such as in a development environment. If you would like HAProxy to be highly available, you must have a load balancer in front of it. In this case, you point DNS at the load balancer.

To use a single-instance HAProxy load balancer in a vSphere or OpenStack deployment:

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the **System domain** and **Apps domain** that you configure in the **Domains** pane of the Ops Manager tile with the HAProxy IP address.

   For example, with `pcf.example.com` as the main subdomain for your Ops Manager deployment and an HAProxy IP address `203.0.113.1`, you must create an A record in your DNS that serves `example.com` and points `*.pcf` to `203.0.113.1`.

Name	Type	Data	Domain
*.pcf	A	203.0.113.1	example.com

2. To test your DNS entry, run:

```
host
```

The `host` command should return your HAProxy IP address.

Securing System and App Endpoints

This topic describes how to create a public-facing website and an intranet microservice in the same app.

Overview

The configuration described in this topic may be ideal for those in financial or health sectors who need extra security and convenience managing internal and external services. If the configuration is successful, developers can only use the Cloud Foundry Command Line Interface (cf CLI) command `cf push` from approved network paths. For more information about `cf push`, see the push section of the cf CLI documentation.

Prerequisites

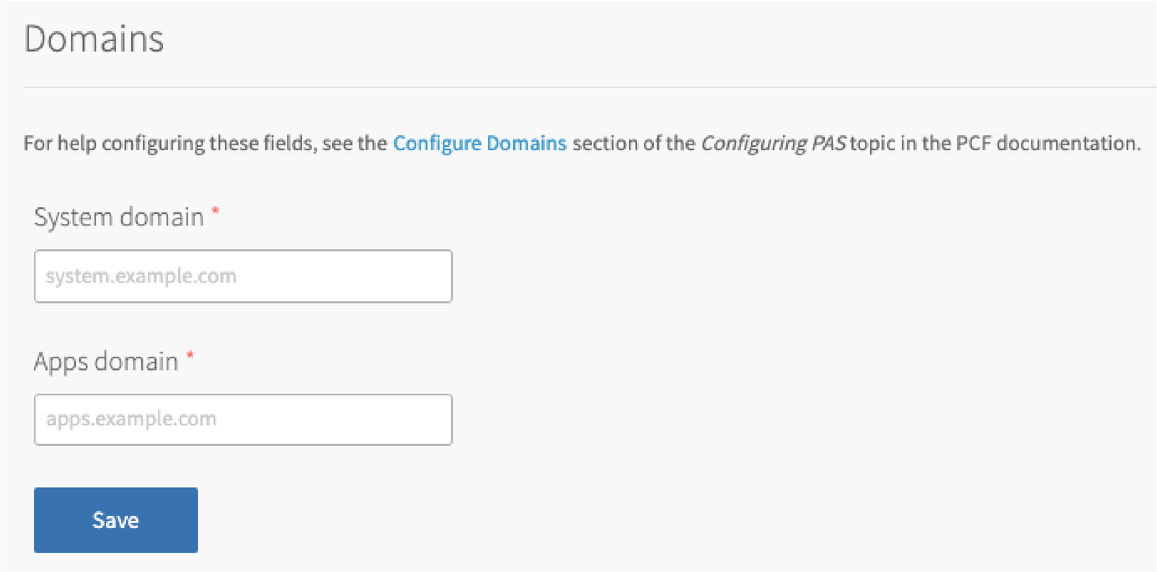
Before beginning your configuration, you must register three domains through your trusted DNS provider:

- A **system domain** for VMware Tanzu Application Service for VMs (TAS for VMs) and its accompanying tile services
- An **apps domain** for your intranet microservice
- An additional domain for the public-facing portions of your app

> **Note:** VMware recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes.

Step 1: Configure System and Apps Domains

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Domains**.
4. Enter the **System domain** and **Apps domain** defined in Prerequisites.



Domains

For help configuring these fields, see the Configure Domains section of the *Configuring PAS* topic in the PCF documentation.

System domain *

system.example.com

Apps domain *

apps.example.com

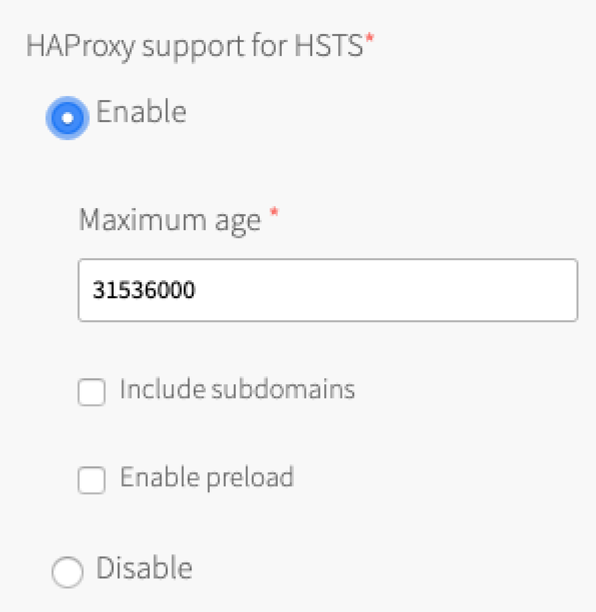
5. Navigate to your DNS provider to create A records that point from your apps domains to the public IP address of your load balancer.
6. Click **Save**.

Step 2: Secure Apps Domain with HAProxy

Caution: This method is easy to implement but may not be the best option for your security needs. Contact Support if you seek a custom solution to building secure, internal services.

Note: VMware supports HAProxy by default. However, you may configure an internal domain using any load balancer you choose that supports host header filtering.

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Enter **HAProxy protected domains**. The domain you enter should be the same domain as the **Apps domain** defined in Prerequisites.
5. Enter **HAProxy trusted CIDRs**, separated by spaces. This specifies CIDRs allowed to make requests to the domains listed in the **HAProxy protected domains** field. For example, entering `10.0.1.0/24` would allow any requests originating at a host IP in that range to reach apps or services hosted on the **HAProxy protected domains** list.
6. (Optional) If you want to force browsers to use HTTPS when making requests to HAProxy, select **Enable** under **HAProxy support for HSTS** and complete these optional configuration steps:



HAProxy support for HSTS*

● Enable

Maximum age *

31536000

□ Include subdomains

□ Enable preload

○ Disable

- Enter a **Maximum age** in seconds for the HSTS request. HAProxy forces HTTPS requests from browsers for the duration of this setting. The maximum age is one year, or 31536000 seconds.
- Enable the **Include subdomains** check box to force browsers to use HTTPS requests for all component subdomains.
- Select the **Enable preload** check box to force instances of Google Chrome, Firefox, and Safari that access your HAProxy to refer to their built-in lists of known hosts that require HTTPS, of which HAProxy is one. This ensures that the first contact a browser has with your HAProxy is an HTTPS request, even if the browser has not yet received an HSTS header from HAProxy.
- (Optional) Configure an additional HAProxy for BOSH to secure your public-facing domain with credentials. For more information, see proxy-boshrelease on GitHub.

Step 3: (Optional) Enable an Authentication Service

For added security, you may want to install an authentication service to your private domain like Single Sign-on (SSO) for VMware Tanzu. For more information about SSO for VMware Tanzu, see Single Sign-On Overview.

Configuring SSH Access for TAS for VMs

This topic describes how to configure SSH access for VMware Tanzu Application Service for VMs (TAS for VMs).

Overview

To help troubleshoot apps hosted by a deployment, TAS for VMs supports SSH access into running apps. This document describes how to configure a TAS for VMs deployment to allow SSH access to app instances, and how to configure load balancing for those app SSH sessions.

Configure TAS for VMs SSH Access

This section describes how to configure TAS for VMs to activate or deactivate deployment-wide SSH access to app instances. In addition to this deployment-wide configuration, Space Managers have SSH access control over their Space, and Space Developers have SSH access control over their to their apps. For details about SSH access permissions, see App SSH Overview.

To configure TAS for VMs SSH access for app instances:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **App Containers**.
4. Activate or deactivate the **Allow SSH access to app containers** checkbox.
5. Optionally, select the **Enable SSH when an app is created** checkbox to enable SSH access for new apps by default in spaces that allow SSH. If you deselect this checkbox, developers can still enable SSH after pushing their apps by running `cf enable-ssh APP-NAME`, where `APP-NAME` is the name of the app for which they want to enable SSH.

Configure an SSH Load Balancer

For IaaSes where load-balancing is available as a service, you should provision a load balancer to balance load across SSH proxy instances. Configure this load balancer to forward incoming TCP traffic on port 2222 to a target pool where you deploy `diego_brain` instances.

For AWS, Azure, and GCP IaaSes, you configure SSH load balancers in the **Resource Config** pane. To register SSH proxies with a load balancer:

1. Select **Resource Config**.
2. In the **Diego Brain** row, enter your load balancer name in the **Load Balancers** field.

Ops Manager supports an API-only `nsx_lbs` field. You can configure load balancers in vSphere using this field.

App SSH components and processes

This topic tells you about the VMware Tanzu Application Service for VMs (TAS for VMs) SSH components that are used for access to deployed app instances. TAS for VMs supports native SSH access to apps and load balancing of SSH sessions with the load balancer for your TAS for VMs deployment.

For procedural and configuration information about app SSH access, see SSH Overview.

SSH components

TAS for VMs SSH includes two central components: an implementation of an SSH proxy server and a lightweight SSH daemon. If these components are deployed and configured correctly, they provide a simple and scalable way to access containers apps and other long-running processes (LRPs).

SSH daemon

SSH daemon is a lightweight implementation that is built around the Go SSH library. It supports command execution, interactive shells, local port forwarding, and secure copy. The daemon is self-contained and has no dependencies on the container root file system.

The daemon is focused on delivering basic access to app instances in TAS for VMs. It is intended to run as an unprivileged process, and interactive shells and commands run as the daemon user. The daemon only supports one authorized key, and it is not intended to support multiple users.

The daemon is available on a file server and Diego LRPs that want to use it can include a download action to acquire the binary and a run action to start it. TAS for VMs apps download the daemon as part of the lifecycle bundle.

SSH proxy authentication

The SSH proxy hosts the user accessible SSH endpoint and is responsible for authentication, policy enforcement, and access controls in the context of TAS for VMs. After you successfully authenticate with the proxy, the proxy attempts to locate the target container and create an SSH session to a daemon running inside the container. After both sessions have been established, the proxy manages the communication between your SSH client and the container SSH Daemon.

Configuring Front End Idle Timeout for Gorouter and HAProxy

This topic describes how to configure the **Front end idle timeout for the Gorouter and HAProxy** field in the **Networking** pane of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.

You can optionally use the **Front end idle timeout for the Gorouter and HAProxy** field to help prevent connections from your load balancer to the Gorouter or HAProxy from being closed prematurely. The value you enter sets the duration, in seconds, that the Gorouter or HAProxy maintains an idle open connection from a load balancer that supports keep-alive.

In general, set the value higher than your load balancer's back end idle timeout to avoid the race condition where the load balancer sends a request before it discovers that the Gorouter or HAProxy has closed the connection.

For specific guidance and exceptions to this rule, see the table below:

IaaS	Guidance
AWS	AWS ELB has a default timeout of 60 seconds, so VMware recommends a value greater than `60`.
Azure	By default, Azure load balancer times out at 240 seconds without sending a TCP RST to clients, so as an exception, VMware recommends a value lower than `240` to force the load balancer to send the TCP RST.
GCP	GCP has a default timeout of 600 seconds. For GCP HTTP load balancers, VMware recommends a value greater than `600`. For GCP TCP load balancers, VMware recommends a value less than `600` to force the load balancer to send a TCP RST.
Other	Set the timeout value to be greater than that of the load balancer's back end idle timeout.

Load Balancing and Routing

In this section:

Using Your Own Load Balancer

This topic describes how to use your own load balancer and forward traffic to your VMware Tanzu Application Service for VMs (TAS for VMs) router IP address.

Overview

TAS for VMs includes a tier of reverse proxies that dynamically track the location of app containers and system components, enabling routing of requests to those endpoints even as IPs and ports change.

In order for the TAS for VMs routers to be horizontally scalable and highly available, a load balancer must be deployed in front of them. The simplest solution is to use a Layer 4 TCP load balancer, provided by your IaaS or IT team, which passes all HTTP and TLS handling to the TAS for VMs routers. For more information about TLS termination, see Securing Traffic into TAS for VMs. For a description of features supported by the TAS for VMs routing tier, see HTTP Routing.

If you have requirements that are not fulfilled by the TAS for VMs routers alone, you can choose to use your own Layer 7 load balancer provided by your IaaS or IT team, or the HAProxy load balancer included with TAS for VMs. If you use HAProxy, you must use a Layer 4 TCP load balancer in front of it in order for HAProxy itself to be highly available. Singleton instances of HAProxy are only for use in lab and test environments.

If you use your own Layer 7 load balancer, it must fulfill the following requirements:

- Provides load balancing to each of the TAS for VMs router IP addresses.
- Supports TLS termination for wildcard hostnames.
- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers to incoming requests.
- Sets an HTTP keepalive connection timeout greater than five seconds.
- (Optional) Supports WebSocket.

The choice to use HAProxy or your own load balancer depends on what features you need out of a load balancer, and whether you want the ability to configure it yourself.

> **Note:** App logging with Loggregator requires WebSocket. To use another logging service, see Streaming App Logs to Log Management Services.

For information about how to install an F5 Local Traffic Manager (LTM) as a load balancer for Ops Manager and TAS for VMs, see Configuring an F5 Load Balancer for TAS for VMs. For more information about F5 LTMs, see the F5 documentation.

Prerequisites

To integrate your own load balancer with TAS for VMs, you must ensure:

- WebSocket connections are not blocked for Loggregator functionality.
- The load balancer must be able to reach the Gorouter IP addresses.

Follow the procedure below to use your own load balancer.

Step 1: Deploy Ops Manager Installation VM

Deploy an Ops Manager Installation VM. For more information, see Deploying Ops Manager on vSphere.

Step 2: Register Ops Manager IP Address

In your load balancer, register the IP addresses that you assigned to Ops Manager.

Step 3: Configure Ops Manager and BOSH Director

Configure Ops Manager and the BOSH Director as described in Configuring BOSH Director on vSphere, then add TAS for VMs.

Do not click **Install** after adding TAS for VMs.

Step 4: Configure Networking

Configure the **Networking** pane in TAS for VMs. Load balancer configuration in TAS for VMs varies depending on which IaaS you are using for Ops Manager. For more information, see Configure Networking in *Configuring TAS for VMs*.

Step 5: Finalize Changes

To finalize the changes to your deployment:

1. Return to the Ops Manager Installation Dashboard.
2. Click **Install**.

Enabling and configuring TCP routing in TAS for VMs

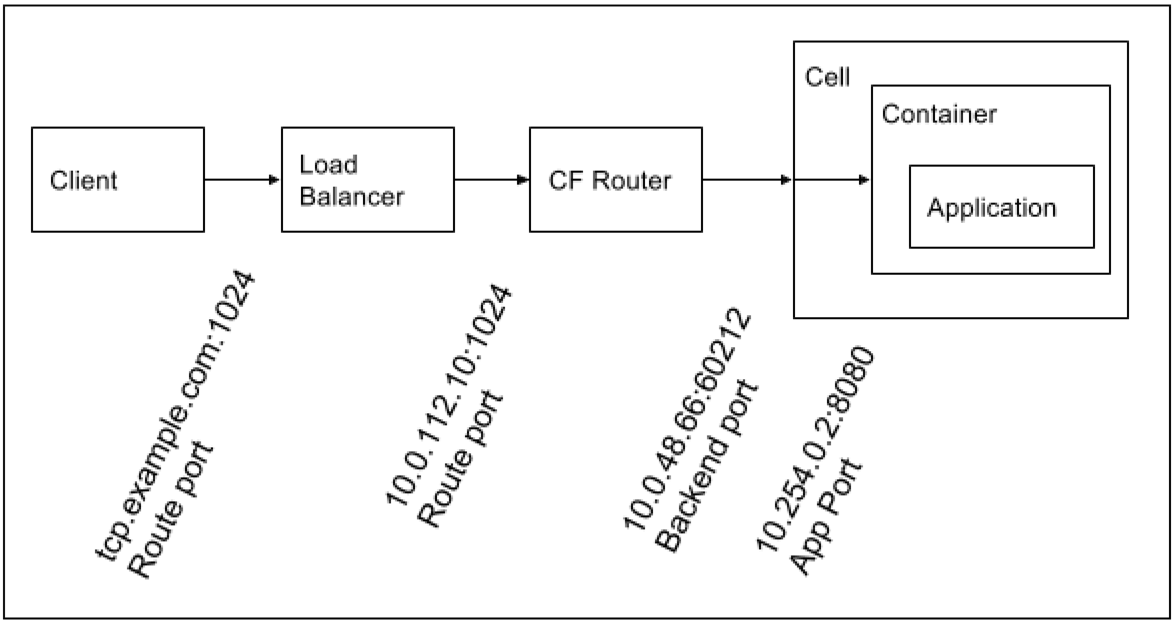
You can enable TCP routing in TAS for VMs. This article describes TCP routing and walks you through the steps of enabling and configuring it.

TCP routing overview

TCP routing enables developers to run apps that serve requests on non-HTTP TCP protocols. You can use TCP routing to comply with regulatory requirements to terminate TLS as close to apps as possible so that packets are not decrypted before reaching the app level.

TCP routing architecture

This diagram shows the layers of network address translation that occur in VMware Tanzu Application Service for VMs (TAS for VMs) in support of TCP routing.



The following is an example workflow that includes route ports, back end ports, and app ports:

1. A developer creates a TCP route for their app based on a TCP domain and a route port, and maps this route to one or more apps. For more information, see Create a Route in *Configuring Routes and Domains*.
2. Clients make requests to the route. DNS resolves the domain name to the load balancer.
3. The load balancer listens on the port and forwards requests for the domain to the TCP routers. The load balancer must listen on a range of ports to support multiple TCP route creation. Additionally, TAS for VMs must be configured with this range, so that the platform knows which ports can be reserved when developers create TCP routes.
4. The TCP router can be dynamically configured to listen on the port when the route is mapped to an app. The domain the request was originally sent to is no longer relevant to the routing of the request to the app. The TCP router keeps a dynamically updated record of the back ends for each route port. The back ends represent instances of an app mapped to the route. The TCP router chooses a back end using a round-robin load balancing algorithm for each new TCP connection from a client. Because the TCP router is protocol-agnostic, it does not recognize individual requests, only TCP connections. All client requests

transit the same connection to the selected back end until the client or back end closes the connection. Each subsequent connection triggers the selection of a back end.

5. Because containers each have their own private network, the TCP router does not have direct access to app containers. When a container is created for an app instance, a port on the Diego Cell VM is randomly chosen and iptables are configured to forward requests for this port to the internal interface on the app container. The TCP router then receives a mapping of the route port to the Diego Cell IP and port.
6. By default, the Diego Cell only routes requests to port `8080`, the app port, on the app container internal interface. The app port is the port on which apps must listen. Developers can use the Cloud Controller API to update the ports an app can receive requests on. For more information, see Configuring Apps to Listen on Custom Ports (Beta).

Prerequisites for Enabling TCP Routing

> **Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing. For more information, see TLS to Apps and Other Back End Services in *HTTP Routing*.

Before enabling TCP routing, you must set up networking requirements. To set up networking requirements:

1. Choose a domain from which your developers are to create TCP routes for their apps. For example, create a domain which is similar to your app domain but prefixed by the TCP subdomain, such as `tcp.APP-DOMAIN.com`, where `APP-DOMAIN` is the name of your app domain.
2. Configure DNS to resolve this domain name to the IP address of a highly-available load balancer that can forward traffic for the domain to the TCP routers. For more information, see Domains in *Configuring Routes and Domains*. If you are operating an environment that does not require high availability, configure DNS to resolve the TCP domain name you have chosen directly to a single instance of the TCP router.
3. (Optional) Choose IP addresses for the TCP routers and configure your load balancer to forward requests for the domain you chose in the first step above to these addresses. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP router. To configure your IP addresses for your TAS for VMs deployment, follow the procedure for enabling TCP routing in Configure Networking in *Configuring TAS for VMs*.
4. (Optional) Decide how many TCP routes you want to support. For each TCP route, you must reserve a port. Configure your load balancer to forward the range of ports to the TCP routers. Skip this step if you have configured DNS to resolve the TCP domain name to an instance of the TCP router. For more information about configuring port reservations, see Modify TCP Port Reservations below.
5. To configure your ports for your TAS for VMs deployment, enable TCP routing in TAS for VMs. For more information, see Configure Networking in *Configuring TAS for VMs*.

Enable TCP routing

To enable TCP routing:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **VMware Tanzu Application Service for VMs** tile.
3. Select **Networking**.
4. Under **TCP routing**, select **Enable**.
5. For **TCP router IPs**, enter the IP addresses to assign to the TCP routers as multiple values as a comma-separated list or as a range. For example, `10.254.0.1, 10.254.0.2` or `10.254.0.1-10.254.0.2`. These addresses must be within your subnet CIDR block. The addresses are the same IP addresses with which you configured your load balancer in Prerequisites for Enabling TCP Routing above, unless you configured DNS to resolve the TCP domain name directly to an IP for the TCP router.
6. For **TCP routing ports**, enter one or more ports to which the load balancer forwards requests. To support multiple TCP routes, VMware recommends allocating multiple ports. Do one of the following:
    - To allocate a single port or range of ports, enter a single port or a range of ports.

      > **Note:** If you configured AWS for TAS for VMs manually, enter `1024-1123`. This range of ports corresponds to the rules you created for `-tcp-elb`.

    - To allocate a list of ports:
        1. Enter a single port in the **TCP routing ports** field.
        2. After deploying TAS for VMs, follow the procedure in Modify TCP Port Reservations below to add TCP routing ports using the cf CLI.
7. (Optional) For **TCP request timeout**, modify the default value of `300` seconds. This field determines when the TCP router closes idle connections from clients to apps that use TCP routes. You may want to increase this value to enable developers to push apps that require long-running idle connections with clients.
8. For AWS, Azure, or GCP Ops Manager deployments, add the name of your load balancer to the **TCP Router** field in the **Resource Config** pane of the TAS for VMs tile. For more information, see Configuring Load Balancing for TAS for VMs.

Configure TCP Routing After Deploying TAS for VMs

After you enable TCP routing and deploy TAS for VMs, you must add the TCP shared domain and configure org quotas to enable developers to create TCP routes. To do this, you must use the Cloud Foundry Command Line Interface (cf CLI) and have an admin user account.

For more information about the cf CLI, see Using the Cloud Foundry Command Line Interface (cf CLI).

Configure TAS for VMs with Your TCP Domain

After deploying TAS for VMs, you must configure TAS for VMs with the domain that you configured in Prerequisites for Enabling TCP Routing. This is the domain from which developers create TCP routes.

To configure TAS for VMs with your TCP domain:

1. List your router groups by running:

   ```
   cf router-groups
   ```

   You should see `default-tcp` as a response.

2. Create a shared domain and associate it with the `default-tcp` router group by running:

   ```
   cf create-shared-domain tcp.APP-DOMAIN.com --router-group default-tcp
   ```

   Where `APP-DOMAIN` is the name of your app domain.

3. Verify that `TCP` appears under `type` next to your TCP domain by running:

   ```
   cf domains
   ```

Configure a Quota for TCP Routes

Since TCP route ports are a limited resource in some environments, quotas are configured to allow creation of zero TCP routes by default. After you deploy TAS for VMs, you can increase the maximum number of TCP routes for all orgs or for particular orgs and spaces. Because you reserve a route port for each TCP route, you manage the quota for route ports using the `--reserved-route-ports` cf CLI command option. For more information, see Creating and Modifying Quota Plans.

You can configure quotas for TCP routes in the following ways:

- If you have a default quota that applies to all orgs, you can update it to configure the number of route ports that can be reserved by each org by running:

  ```
  cf update-org-quota QUOTA --reserved-route-ports NUMBER-OF-ROUTE-PORTS
  ```

  Where:

  - `QUOTA` is the maximum number of TCP routes you want to allocate to all orgs.
  - `NUMBER-OF-ROUTE-PORTS` is the number of route ports you want to allow each org to reserve.

- To create a new quota that governs the number of route ports that can be created in a particular org:

  1. Target the org for which you want to create the quota by running:

     ```
     cf target -o ORG-NAME
     ```

     Where `ORG-NAME` is the name of the org for which you want to create the quota.

2. Run:

   ```
   cf create-quota QUOTA --reserved-route-ports NUMBER-OF-ROUTE-PORTS
   ```

   Where:

   - QUOTA is the maximum number of TCP routes you want to allocate to particular orgs.
   - NUMBER-OF-ROUTE-PORTS is the number of route ports you want to allow each org to reserve.

- To create a new quota that governs the number of route ports that can be created in a particular space:

  1. Target the space for which you want to create the quota by running:

     ```
     cf target -s SPACE-NAME
     ```

     Where SPACE-NAME is the name of the space for which you want to create the quota.

  2. Run:

     ```
     cf create-space-quota QUOTA --reserved-route-ports NUMBER-OF-ROUTE-PORTS
     ```

     Where:

     - QUOTA is the maximum number of TCP routes you want to allocate to a particular space.
     - NUMBER-OF-ROUTE-PORTS is the number of route ports you want to allow the space to reserve.

Create a TCP route

For information about creating a TCP route, see Create a TCP Route with a Port in *Configuring Routes and Domains*.

Modify TCP port reservations

After deploying TAS for VMs, you can modify the range of ports available for TCP routes using cf curl commands, as demonstrated with the commands below. These commands require you to have an admin user account with the routing.router_groups.read and routing.router_groups.write scopes.

1. In a terminal window, view the reservable_ports by running:

   ```
   cf curl /routing/v1/router_groups
   ```

   Record the guid from the output.

2. To configure a new port, run:

   ```
   cf curl /routing/v1/router_groups/GUID
   ```

Where `GUID` is the GUID you recorded in the previous step.

To configure multiple ports, enter a comma-separated list of ports or port ranges by running:

```
cf curl \
-X PUT -d '{"reservable_ports":"PORTS-OR-PORT-RANGES"}' \
/routing/v1/router_groups/f7392031-a488-4890-8835-c4a038a3bded
```

Where `PORTS-OR-PORT-RANGES` is a comma-separated list of ports or port ranges. For example, `"reservable_ports":"1024-1199,1234-1248,1312"`.

Note: Do not enter `reservable_ports` that conflict with other TCP router instances or ephemeral port ranges. VMware recommends using port ranges within `1024-2047` and `18000-32767` on default installations. Check which ports are available on the TCP router VMs to verify that no additional ports are in use. For more information, see TCP router fails to configure routes when there is a port conflict with a local process in the Routing Release repository on GitHub.

Disable TCP Routing

To deactivate TCP routing:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **VMware Tanzu Application Service for VMs** tile.
3. Select **Networking**.
4. Under **TCP routing**, select **Disable**.
5. Remove the TCP routing domain by running:

   ```
   cf delete-domain DOMAIN
   ```

   Where `DOMAIN` is the name of your TCP routing domain.

Enabling IPv6 for hosted apps on TAS for VMs

You can enable IPv6 support for hosted apps on TAS for VMs. This article walks you through the steps.

Overview

The procedure described below allows apps deployed to VMware Tanzu Application Service for VMs (TAS for VMs) to be reached using IPv6 addresses.

Note: Amazon Web Services (AWS) EC2 instances currently do not support IPv6.

TAS for VMs system components use a separate DNS subdomain from hosted apps. These components currently support only IPv4 DNS resolved addresses. This means that although an IPv6 address can be used for app domains, the system domain must resolve to an IPv4 address.

Enable IPv6 Support for Hosted Apps

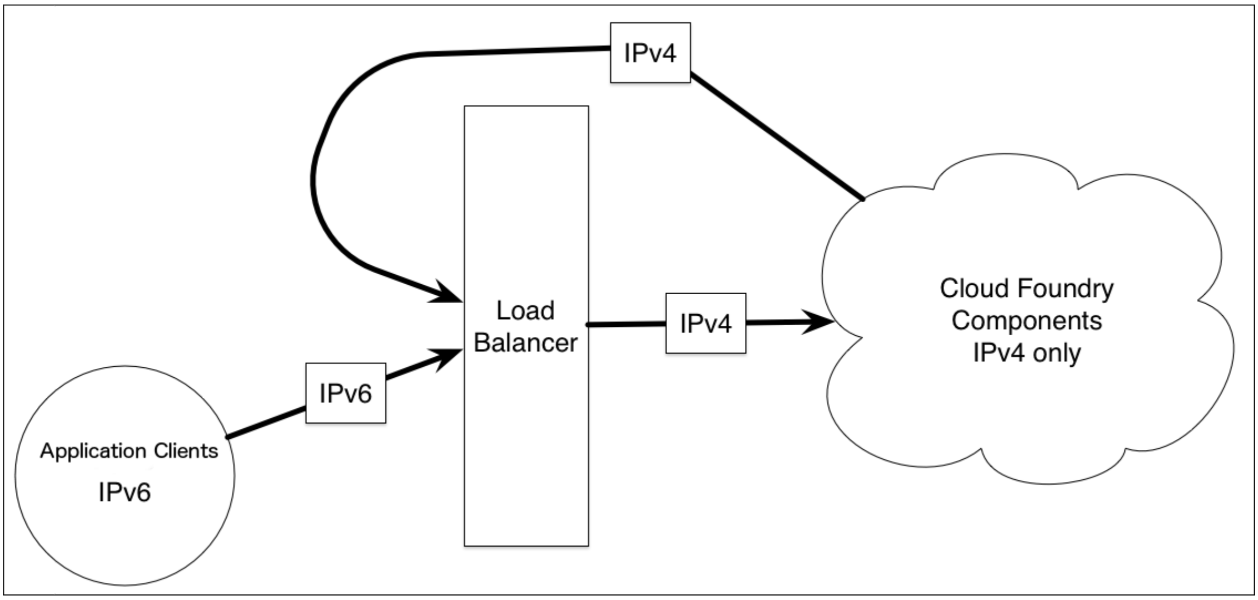
To enable support for IPv6 app domains:

1. Set up an external load balancer for your TAS for VMs deployment. For more information, see Using Your Own Load Balancer.
2. Configure DNS to resolve app domains to an IPv6 address on your external load balancer.

   > **Note:** Your IPv4 interface for the system domain and IPv6 interface for app domain can be configured on the same or different load balancers.

3. Configure the external load balancer to route requests for an IPv6 address to an IPv4 address:
   - If you are using the HAProxy load balancer for SSL termination, route to its IPv4 address.
   - Otherwise, route directly to the IPv4 addresses of the Gorouters.

The following diagram illustrates how a single load balancer can support traffic on both IPv4 and IPv6 addresses for a TAS for VMs installation:



For more information about domains in TAS for VMs, see Routes and Domains.

"Configuring HTTP vars TAS for VMs

You can configure your VMware Tanzu Application Service for VMs (TAS for VMs) deployment to support HTTP/2 from ingress to egress. This article tells you how.

HTTP/2 support overview

For information about how HTTP/2 can benefit apps running on TAS for VMs, see Routing HTTP/2 and gRPC Traffic to Apps.

Apps can benefit from HTTP/2 even if not all network segments use HTTP/2. Headers are compressed and requests are multiplexed for HTTP/2 segments even if other network hops do not use HTTP/2.

However, some features like gRPC require all segments to use HTTP/2. While browsers and other clients might indicate a request is being served over HTTP/2, operators must ensure that all network hops use HTTP/2 to support gRPC.

Prerequisites

Before you can configure your TAS for VMs deployment to support HTTP/2, you must enable TLS.

Most implementations of HTTP/2 require TLS with Application-Layer Protocol Negotiation (ALPN). For more information about ALPN, see RFC 7301: Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension.

Security considerations

If you use a firewall or other tools to monitor network traffic, ensure that those tools support HTTP/2 connections. If your network monitoring tools do not support HTTP/2 connections, configuring HTTP/2 can cause problems with protecting and analyzing traffic on your network.

VMware recommends that you become familiar with common vulnerabilities in apps that support HTTP/2. For example, HTTP/1.1 has vulnerabilities such as request smuggling and desync attacks, but these might be more prevalent in HTTP/2 environments.

Enable end-to-end HTTP/2

This section describes how to enable HTTP/2 in your deployment from ingress to egress. To enable end-to-end HTTP/2, you must configure the load balancers, the TAS for VMs platform, and the app.

Configure load balancers

To support HTTP/2, operators must configure platform load balancers to enable HTTP/2 ingress and egress.

Load balancers in front of TAS for VMs can be either Layer 4 (TCP) or Layer 7 (Application). Layer 4 load balancers tend to be simpler, while Layer 7 load balancers offer more features by inspecting the contents of HTTP requests. For example, a Layer 7 load balancer might send requests to different TAS for VMs deployments based on the resources that are being requested. You can configure many load balancers to function in either Layer 4 or Layer 7 mode.

For more information, see the following sections:

- Configure Layer 4 TCP Load Balancers
- Configure Layer 7 Application Load Balancers

Configure layer 4 TCP load balancers

Layer 4 load balancers do not terminate HTTP connections and support passing HTTP/2 traffic.

If you are terminating TLS traffic at a Layer 4 load balancer, configure your load balancer to advertise support for HTTP/2 over ALPN. ALPN ensures that a client making an HTTP request knows that the app server that is servicing the request can support HTTP/2.

If a load balancer terminates TLS without advertising HTTP/2 over ALPN, then clients must be configured to use HTTP/2 with prior knowledge. For more information, see Starting HTTP/2 with Prior Knowledge in *RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2)*.

Configure layer 7 application load balancers

Layer 7 load balancers terminate the incoming HTTP connection and initiate new HTTP connections to their back ends. For end-to-end HTTP/2 support, Layer 7 load balancers must have HTTP/2 enabled for both ingress and egress HTTP connections.

The HAProxy BOSH release contains the canonical example of how to set up HTTP/2 load balancing for TAS for VMs. See the BOSH release for HAProxy on GitHub.

When HTTP/2 is enabled, HAProxy advertises support for HTTP/2 over ALPN, accepts HTTP/2 ingress traffic for all connections, and negotiates using HTTP/2 over ALPN when connecting to the Gorouter.

Gorouter and many Layer 7 load balancers do not support WebSockets over HTTP/2. For more information, see RFC 8441. TAS for VMs uses WebSockets for streaming logs and metrics as well as apps that serve WebSocket traffic.

To continue supporting WebSockets in TAS for VMs when you enable HTTP/2, you can do either of the following:

- Configure load balancers to forward WebSocket traffic over HTTP/1.1.
- Use a Layer 4 load balancer for WebSocket traffic.

For more information, see Supporting WebSockets.

Configure the TAS for VMs platform

In TAS for VMs v2.12 and later, HTTP/2 support is enabled by default. To deactivate HTTP/2 support, set the `router.enable_http2` property to `false`.

When HTTP/2 is enabled, the Gorouter accepts HTTP/2 ingress traffic for all apps, but does not connect to app instances over HTTP/2 unless configured on app routes.

Configure TAS for VMs apps

Before the Gorouter can send HTTP/2 traffic to apps, the operator must configure HTTP/2 when mapping the route to the app. This is because the Gorouter defaults to HTTP/1.1 for compatibility unless it knows that a given route and app combination supports HTTP/2.

After you map a route with HTTP/2 support enabled, the Gorouter sends all traffic to that app over HTTP/2, even traffic that ingresses to the Gorouter over HTTP/1.1.

To map a route to the app with HTTP/2 support:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI) by running:

```
cf login -a API-URL -u USERNAME -p PASSWORD -o ORG -s SPACE
```

Where:

- `API-URL` is your API endpoint, the URL of the Cloud Controller in your TAS for VMs instance.
- `USERNAME` is your username.
- `PASSWORD` is your password. VMware discourages using the `-p` option, as it may record your password in your shell history.
- `ORG` is the org where your app is deployed.
- `SPACE` is the space in the org where your app is deployed.

2. Run the following command to map the route to your app:

```
cf map-route MY-APP EXAMPLE.COM --destination-protocol=http2
Creating route MY-APP.EXAMPLE.COM for org my-org / space my-space as admin...
OK
```

Where:

- `MY-APP` is the name of your app.
- `EXAMPLE.COM` is the route you want to map to your app.

Configuring Proxy Settings for All Apps

This topic describes how to globally configure proxy settings for all apps in your Ops Manager deployment.

Overview

Some environments restrict access to the Internet by requiring traffic to pass through an HTTP or HTTPS proxy. Ops Manager operators can use the Cloud Foundry Command Line Interface (cf CLI) to provide the proxy settings to all apps, including system apps and service brokers.

> **Note:** Incorrectly configuring proxy settings can prevent apps from connecting to the Internet or accessing required resources. They can also cause errands to fail and break system apps and service brokers. Although errands, system apps, and service brokers do not need to connect to the Internet, they often need to access other resources in Ops Manager. Incorrect proxy settings can break these connections.

Set Environment Variables

To globally configure proxy settings for Ops Manager apps, you must set three environment variables for both the staging environment variable group and the running environment variable group.

For more information about variable groups, see Environment Variable Groups in *TAS for VMs Environment Variables*.

This procedure explains how to set proxy information for both staging and running apps. However, you can also set proxy settings for only staging or only running apps.

To globally configure proxy settings for Ops Manager apps:

1. Target your Cloud Controller with the cf CLI. If you have not installed the cf CLI, see Installing the cf CLI. Run:

   ```
   cf api api.SYSTEM-DOMAIN
   ```

   Where `SYSTEM-DOMAIN` is your system domain.

2. Log in with your UAA administrator credentials.

   1. To retrieve these credentials:
      1. Navigate to the Ops Manager Installation Dashboard.
      2. Click the VMware Tanzu Application Service for VMs (TAS for VMs) tile.
      3. Select the **Credentials** tab.
      4. Under **UAA**, next to **Admin Credentials**, click **Link to Credential**.
      5. Record the password.
   2. Run:

      ```
      cf login
      ```

3. To configure proxy access for apps that are staging, run:

   ```
   cf set-staging-environment-variable-group '{"http_proxy": "http://PROXY:8080/",
   "https_proxy": "http://PROXY:8080/", "no_proxy": "DNS-SUFFIX-OR-IP-ADDRESS,DNS-
   SUFFIX-OR-IP-ADDRESS"}'
   ```

   Where:

   - `http_proxy` is the proxy to use for HTTP requests.
   - `https_proxy` is the proxy to use for HTTPS requests. In most cases, this is the same as `http_proxy`.
   - `PROXY` is your proxy.
   - `no_proxy` is a comma-separated list of DNS suffixes or IP addresses that can be accessed without passing through the proxy. VMware recommends this value contains the domain `.cf.internal`, which is used by the platform to communicate with CredHub.
   - `DNS-SUFFIX-OR-IP-ADDRESS` is a DNS suffix or IP address that can be accessed without passing through the proxy.

   From now on, the proxy settings are applied to apps during staging.

4. To configure proxy access for apps that are running:

   - For non-Java-based apps, run:

```
cf set-running-environment-variable-group '{"http_proxy": "http://PROXY:8080/", "https_proxy": "http://PROXY:8080/", "no_proxy": "DNS-SUFFIX-OR-IP-ADDRESS,DNS-SUFFIX-OR-IP-ADDRESS"}'
```

Where:

- `http_proxy` is the proxy to use for HTTP requests.
- `https_proxy` is the proxy to use for HTTPS requests. In most cases, this is the same as `http_proxy`.
- `PROXY` is your proxy.
- `no_proxy` is a comma-separated list of DNS suffixes or IP addresses that can be accessed without passing through the proxy. VMware recommends this value contains the domain `.cf.internal`, which is used by the platform to communicate with CredHub.
- `DNS-SUFFIX-OR-IP-ADDRESS` is a DNS suffix or IP address that can be accessed without passing through the proxy.

- For Java-based apps, run:

```
cf set-running-environment-variable-group '{"JAVA_OPTS": "-Dhttp.proxyHost=PROXY -Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts=DNS-SUFFIX-OR-IP-ADDRESS|DNS-SUFFIX-OR-IP-ADDRESS"}'
```

Where:

- `-Dhttp.proxyHost` is the proxy to use for HTTP requests.
- `PROXY` is your proxy.
- `-Dhttp.proxyPort` is the port your proxy uses.
- `-Dhttp.nonProxyHosts` is a pipe-separated list of DNS suffixes or IP addresses that can be accessed without passing through the proxy. VMware recommends this value contains the domain `.cf.internal`, which is used by the platform to communicate with CredHub.
- `DNS-SUFFIX-OR-IP-ADDRESS` is a DNS suffix or IP address that can be accessed without passing through the proxy.

For more information about these Java proxy settings, see Java Networking and Proxies in the Oracle documentation.

5. To apply the proxy configuration for the running environment variable group, restart each app that you want to use the new configuration.

Troubleshooting

This section describes solutions to try if an app fails after you apply the global proxy settings.

Exclude an App From Global Proxy Settings

If your app fails, try instructing the app to ignore the global proxy settings.

To manually unset the proxy environment variables for the failing app:

1. Set the proxy environment variables for `http_proxy` to an empty value by running:

   ```
   cf set-env APP-NAME http_proxy ''
   ```

   Where `APP-NAME` is the name of your app.

2. Set the proxy environment variables for `https_proxy` to an empty value by running:

   ```
   cf set-env APP-NAME https_proxy ''
   ```

   Where `APP-NAME` is the name of your app.

3. Set the proxy environment variables for `no_proxy` to an empty value by running:

   ```
   cf set-env APP-NAME no_proxy ''
   ```

   Where `APP-NAME` is the name of your app.

Change Case of HTTP

Your app and language runtime may be case-sensitive. Try performing the steps in Set Environment Variables using uppercase for `HTTP_PROXY`, `HTTPS_PROXY`, and `NO_PROXY` instead of lowercase, as in the following example:

```
cf set-staging-environment-variable-group '{"HTTP_PROXY": "http://PROXY:8080/", "HTTPS_PROXY": "http://PROXY:8080/", "NO_PROXY": "DNS-SUFFIX-OR-IP-ADDRESS,DNS-SUFFIX-OR-IP-ADDRESS"}'.
```

Where:

- `PROXY` is your proxy.
- `DNS-SUFFIX-OR-IP-ADDRESS` is a DNS suffix or IP address that can be accessed without passing through the proxy.

Check Proxy Settings

If you have set up your proxy so that it can only send traffic to the Internet, a request to an internal resource like Ops Manager fails. You must set `no_proxy` so that traffic destined for Ops Manager and other internal resources is sent directly and does not go through the proxy. For instance, setting `no_proxy` to include your system and app domains will ensure that requests destined for those domains are sent directly.

Verify Interpretation

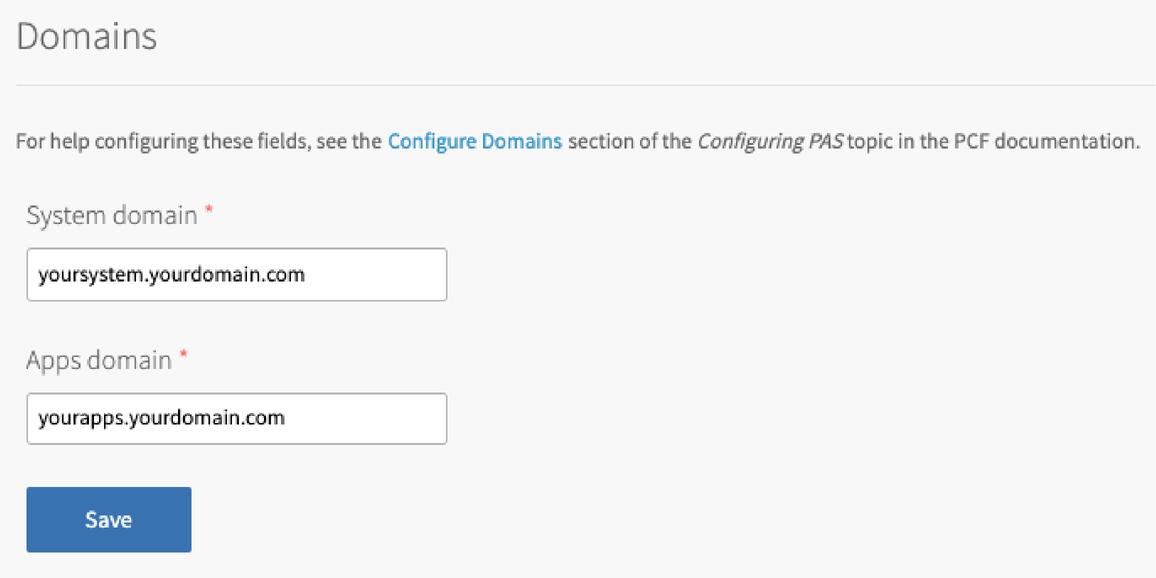
The interpretation of `no_proxy` depends on the app and the language runtime. Most support `no_proxy`, but the specific implementation may vary. For example, some match DNS names that end with the value set in `no_proxy`: `example.com` would match `test.example.com`. Others support the use of the asterisk as a wildcard to provide basic pattern matching in DNS names: `*.example.com` would match `test.example.com`. Most apps and language runtimes do not support pattern matching and wildcards for IP addresses.

Switching App Domains

This topic describes how to change the domain of an existing Ops Manager installation.

To change the domain of an existing Ops Manager installation:

1. Go to the Ops Manager Installation Dashboard.
2. Click the VMware Tanzu Application Service for VMs (TAS for VMs) tile.
3. Select **Domains** to see the current **Apps domain** for your TAS for VMs deployment.

   Domains

   For help configuring these fields, see the Configure Domains section of the *Configuring PAS* topic in the PCF documentation.

   System domain *

   yoursystem.yourdomain.com

   Apps domain *

   yourapps.yourdomain.com

4. In the terminal, run:

   ```
   cf login -a API-ENDPOINT
   ```

   Where `API-ENDPOINT` is the API endpoint for your TAS for VMs instance.
   The cf CLI prompts you for your Ops Manager username and password, as well as the org and space you want to access. If you do not know your API endpoint, see Identifying the API Endpoint for Your TAS for VMs Instance.

5. To view the domains in the space, run:

   ```
   cf domains
   ```

   The command returns output similar to the following example:

   ```
   $ cf domains
   Getting domains in org your-org as admin...

   name                    status
   yourapps.yourdomain.com   shared
   ```

   If you have more than one shared domain, ensure that the domain you want to change is at the top of the list before you apply the new domain to your TAS for VMs tile configuration. You can delete and re-create the other shared domains as necessary to push the domain you want to change to the top of the list. If you do this, make sure to re-map the routes for each domain. For more information, see Configuring Routes and Domains.

6. To confirm that your apps are assigned to the domain you plan to change, run:

   ```
   cf routes
   ```

   The command returns output similar to the following example:

   ```
   $ cf routes
   Getting routes as admin ...

   space      host    domain                  apps
   your-space yourapp yourapps.yourdomain.com yourapp
   ```

7. To create the new domain you want to use, run:

   ```
   cf create-shared-domain NEW-DOMAIN
   ```

   Where `NEW-DOMAIN` is the name you give the new domain.
   The command returns output similar to the following example:

   ```
   $ cf create-shared-domain newapps.yourdomain.com

   Creating shared domain newapps.yourdomain.com as admin...
   OK
   ```

8. To map the new domain to your app, run:

   ```
   cf map-route APP-NAME NEW-DOMAIN -n HOST-NAME
   ```

   Where:

   - `APP-NAME` is the name of the app you want to map to the new domain.
   - `NEW-DOMAIN` is the domain to which you want to map your app.
   - `HOST-NAME` is the hostname of the URL for the route you are mapping.

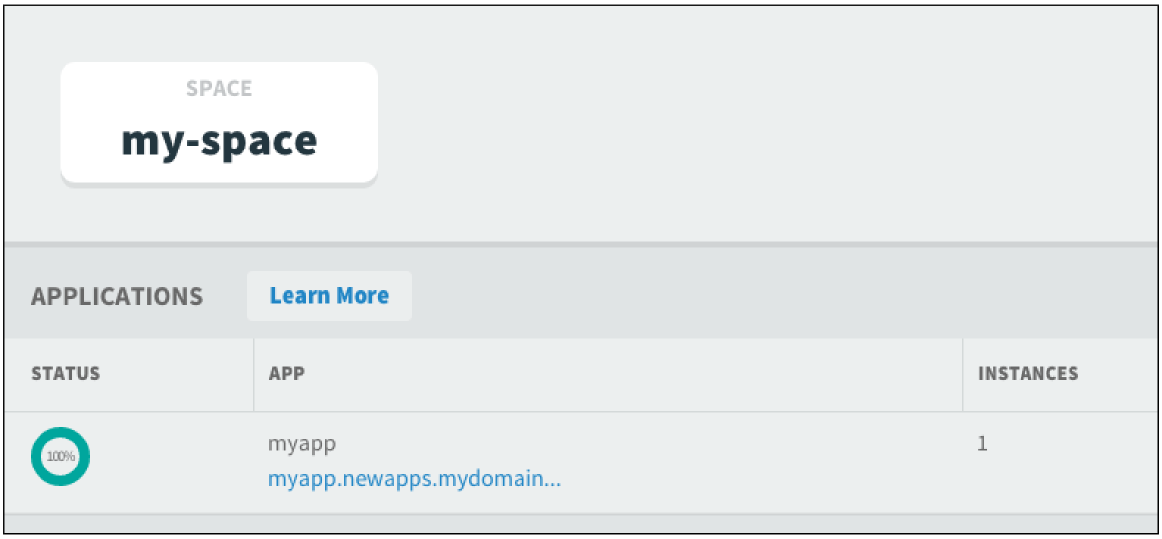
   The command returns output similar to the following example:

   ```
   $ cf map-route yourapp newapps.yourdomain.com -n yourapp

   Creating route yourapp.newapps.yourdomain.com for org your-org / space
   your-space as admin...
   OK
   Adding route yourapp.newapps.yourdomain.com to app yourapp in org your-
   org / space your-space as admin...
   OK
   ```

   In this example, both the `APP-NAME` and `HOST-NAME` arguments are `yourapp`, since this is both the name of the app to which you are mapping a route and the intended hostname for the URL.

9. Repeat the previous step for each app in this space. Afterward, check Apps Manager to confirm that the route URL has updated correctly for each app:



SPACE

my-space

APPLICATIONS

STATUS	APP	INSTANCES
100%	myapp myapp.newapps.mydomain...	1

10. Repeat the above steps for each space in your Ops Manager installation except for the System org, beginning with logging into the org and space and ending with confirming the URL update.

> **Note:** Ordinarily the System org contains only Ops Manager apps that perform utility functions for your installation. VMware does not recommend pushing apps to this org. However, if you have pushed apps to the System org, you must also repeat the above steps for these apps.

11. Once you confirm that every app in every space has been mapped to the new domain, delete the old domain by running:

```
cf delete-shared-domain OLD-DOMAIN
```

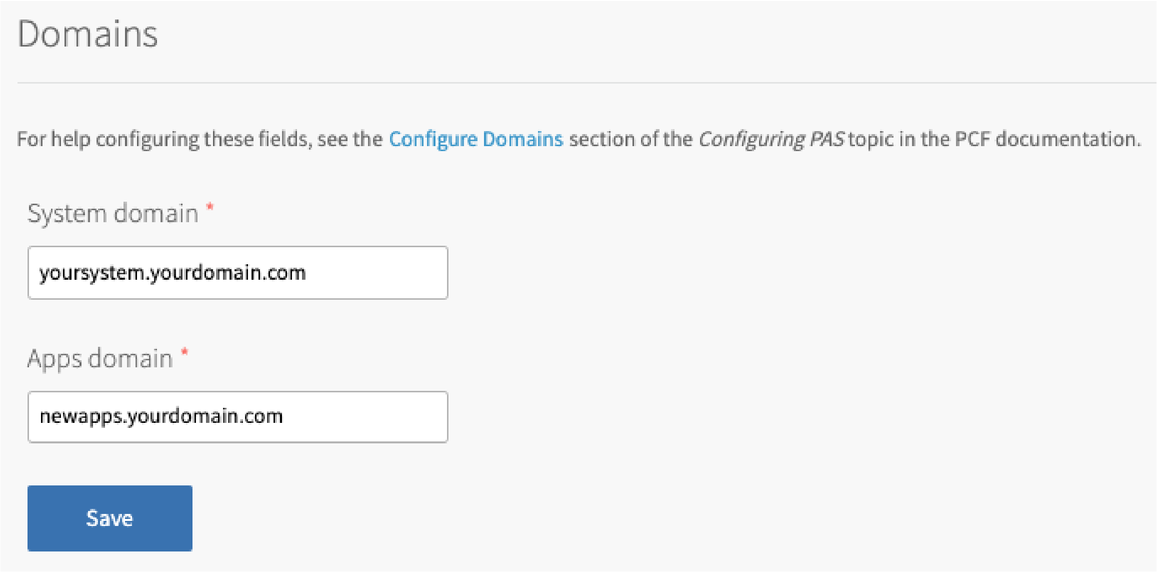
Where `OLD-DOMAIN` is the old domain to delete.
The command returns output similar to the following example:

```
$ cf delete-shared-domain yourapps.yourdomain.com
Deleting domain yourapps.yourdomain.com as admin...

This domain is shared across all orgs.
Deleting it removes all associated routes, and makes any app with this
domain unreachable.
Are you sure you want to delete the domain yourapps.yourdomain.com?
```

When prompted, type `yes` to confirm that you want to delete the domain.

12. In the **Domains** pane of the TAS for VMs tile, enter the new domain in the **Apps domain** field. This configures TAS for VMs to use the new domain.



Domains

For help configuring these fields, see the Configure Domains section of the *Configuring PAS* topic in the PCF documentation.

System domain *

yoursystem.yourdomain.com

Apps domain *

newapps.yourdomain.com

13. Return to the Ops Manager Installation Dashboard.
14. Click **Review Pending Changes**.
15. Click **Apply Changes**. Apps that you push after your update finishes use this new domain.

Configuring load balancer health checks for TAS for VMs routers

You can configure load balancer health checks for TAS for VMs routers so requests go only to healthy router instances. Learn more on this page.

You can also configure a health check for your HAProxy if your deployment uses the HAProxy component.

In environments that require high availability, operators must configure their own redundant load balancer to forward traffic directly to the VMware Tanzu Application Service for VMs (TAS for VMs) routers. In environments that do not require high availability, operators can skip the load balancer and configure DNS to resolve the TAS for VMs domains directly to a single instance of a router.

Add health check endpoints for routers

Configure your load balancer to use the following HTTP health check endpoints. Add the IP addresses of all router instances along with their corresponding port and path.

- Gorouter (HTTP router): `http://GOROUTER_IP:8080/health`
- TCP router: `http://TCP_ROUTER_IP:80/health`

Add a Health Check Endpoint for HAProxy

If you have deployed one or more instances of HAProxy between your infrastructure load balancer and Gorouters, configure your infrastructure load balancer to use the following HTTP health check endpoint: `http://HAPROXY_IP:8080/health`.

The HAProxy is an optional component that provides some features that the Gorouter does not and can be helpful for demonstrating horizontal scalability of the TAS for VMs routers in environments where an infrastructure load balancer is not available.

Set the healthy and unhealthy threshold properties for the Gorouter

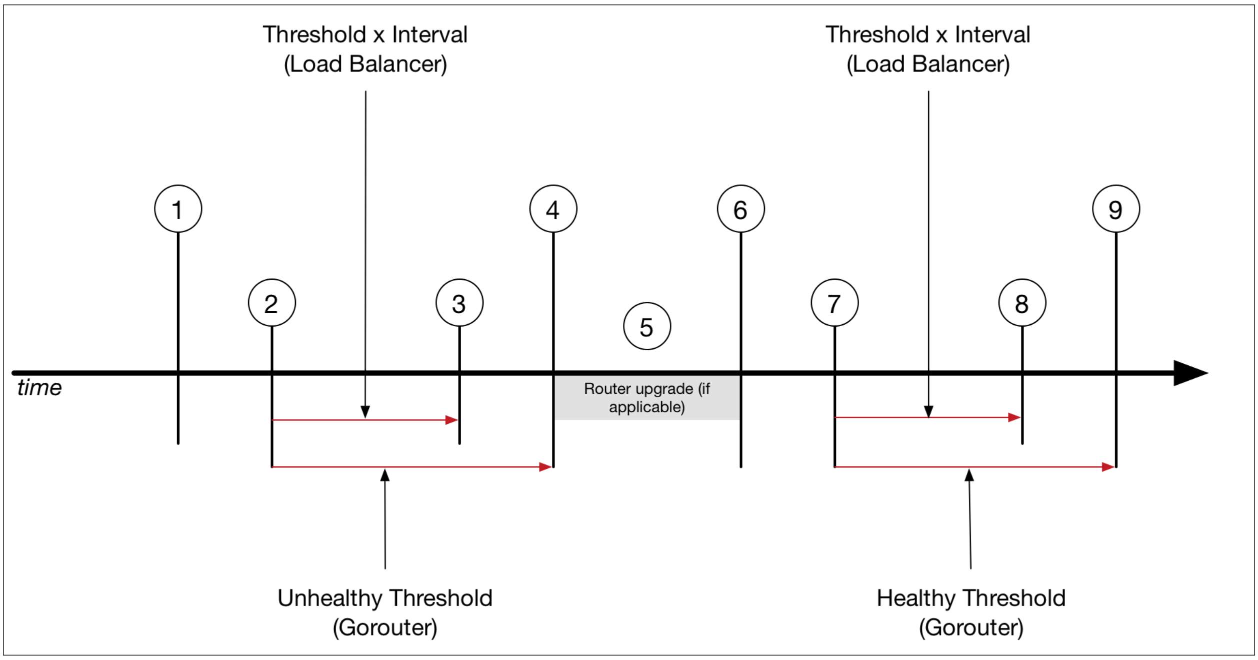
To maintain high availability during upgrades to the Gorouter, each router is upgraded on a rolling basis. During upgrade of a highly available environment with multiple routers, each router is shut down, upgraded, and restarted before the next router is upgraded. This ensures that any pending HTTP request passed to the Gorouter are handled correctly.

TAS for VMs uses these properties:

- **Unhealthy threshold**: Specifies the amount of time, in seconds, that the Gorouter continues to accept connections before shutting down. During this period, the health check reports `unhealthy` to cause load balancers to fail over to other Gorouters. You should set this value greater than or equal to the maximum amount of time it could take your load balancer to consider a Gorouter instance unhealthy, given contiguous failed health checks.
- **Healthy threshold**: Specifies the amount of time, in seconds, to wait until declaring the Gorouter instance started. This allows an external load balancer time to register the instance as `healthy`.

You can configure these properties in the **Networking** pane. For more information, see Configure Networking in *Configuring TAS for VMs*.

The image and table below describe the behavior of the load balancer health checks when a router shuts down and is restarted.



Step	Description
1	A shutdown request is sent to the router.

2	The router receives shutdown request, which causes the following: • The router begins sending Service Unavailable responses to the load balancer health checks. • The load balancer continues sending HTTP request to the router
3	The load balancer considers the router to be in an unhealthy state, which causes the load balancer to stop sending HTTP requests to the router. The time between step 2 and 3 is defined by the values of the health check interval and threshold configured on the load balancer.
4	The router shuts down. The interval between step 2 and 4 is defined by the Unhealthy Threshold property of the Gorouter. In general, the value of this property should be longer than the value of the interval and threshold values (interval x threshold) of the load balancer. This additional interval ensures that any remaining HTTP requests are handled before the router shuts down.
5	If the router shutdown is initiated by an upgrade, the Gorouter software is upgraded.
6	The router restarts. The router will return Service Unavailable responses for load balancer health checks for 20 seconds; during this time the routing table is preloaded.
7	The routers begins returning Service Available responses to the load balancer health check.
8	The load balancer considers the router to be in a healthy state. The time between step 7 and 8 is specified by the health check interval and threshold configured for your load balancer (health check threshold x health check interval).
9	Shutdown and upgrade of the other router begins.

Configuring Route Service Lookup

This topic describes configuring route service lookup in VMware Tanzu Application Service for VMs (TAS for VMs).

Overview

You can bind your app to a route service to preprocess requests before they reach an app. Example use cases include authentication, rate limiting, and caching services. For more information, see Route Services.

The **Bypass security checks for route service lookup** check box in the **Networking** pane of the TAS for VMs tile allows you to configure how the Gorouter handles traffic for apps that are bound to route services. The configuration options are:

- **Default lookup:** Default lookup is configured when you deactivate the **Bypass security checks for route service lookup** check box. In this case, the Gorouter does not check for an existing route. It sends traffic back through the load balancer when the traffic is for an internal route service.
- **Bypass mode:** Bypass mode is configured when you enable the **Bypass security checks for route service lookup** check box. The Gorouter checks for an existing route. If the Gorouter finds the route and the route service is internal, it sends the traffic directly to the

route service and skips the load balancer. This improves performance, but introduces the security risk described in Bypass Mode and External Route Service (Security Risk).

For more information, see Summary of Behavior in Different Configurations.

For configuration guidance and procedures, see Configure Route Service Lookup.

Configure Route Service Lookup

These sections provide guidance and configuration steps for route service lookup.

Guidance

VMware recommends that you do not configure TAS for VMs for bypass mode because of the security risk described in Bypass Mode and External Route Service (Security Risk). However, you might need to do so if your load balancer requires mutual TLS from clients.

If your load balancer requires mutual TLS from clients and TAS for VMs is configured for default lookup, the Gorouter cannot handle traffic successfully for internal route services. This is because the Gorouter does not have the necessary certificates from the client to communicate back with the load balancer for DNS lookup. Therefore you must configure bypass mode so the Gorouter can send the traffic directly to the route service.

Configure TAS for VMs for Bypass Mode

To configure bypass mode:

1. Navigate to the VMware Tanzu Operations Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **Route services**, select **Enable**.
5. Enable the **Bypass security checks for route service lookup** check box.
6. Follow the procedure in Mitigate Security Risk.

Mitigate Security Risk

To prevent users from intercepting traffic for externally-hosted route services:

1. Create an org for use by the TAS for VMs admin.
2. Register all external route service domains as private domains in the org you created.
3. Monitor TAS for VMs for the addition of new external route services and ensure you follow the same process for those external route services. One way to do this is by using `cf curl` to regularly view a list of user-provided service instances. Run:

   ```
   cf curl /v2/user_provided_service_instances
   ```

   Note: Since route services can be added by any space developer, this can be difficult to manage.

Configure TAS for VMs for Default Lookup

To configure TAS for VMs for default lookup behavior, with bypass mode deactivated:

1. G to the VMware Tanzu Operations Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **Route services**, select **Enable**.
5. Deactivate the **Bypass security checks for route service lookup** check box.
6. Communicate to developers of route services that the domain name for their internally-hosted route services must resolve to the load balancer.
7. If your load balancer or Gorouter terminates TLS:
    1. Work with developers of route services to ensure the load balancer or Gorouter have TLS certificates that are valid for the route service URL.
    2. Ensure that the TLS certificate from your load balancer is either signed by a well-known Certificate Authority (CA), or the CA has been added to the **Certificate Authorities trusted by the Gorouter and HAProxy** field in the **Networking** pane of the TAS for VMs and Isolation Segment tiles. The CA for the TLS certificate provided by the load balancer must be trusted by the Gorouter.
8. Work with developers of route services to verify that their internal route service apps are reachable. You can do this by visiting the HTTPS URL of the route service directly and confirming that the app received the request with the `cf logs` output for the route service app.

Summary of Behavior in Different Configurations

These sections describe how the Gorouter behaves when bypass mode is activated or deactivated and when a route service is internal or external.

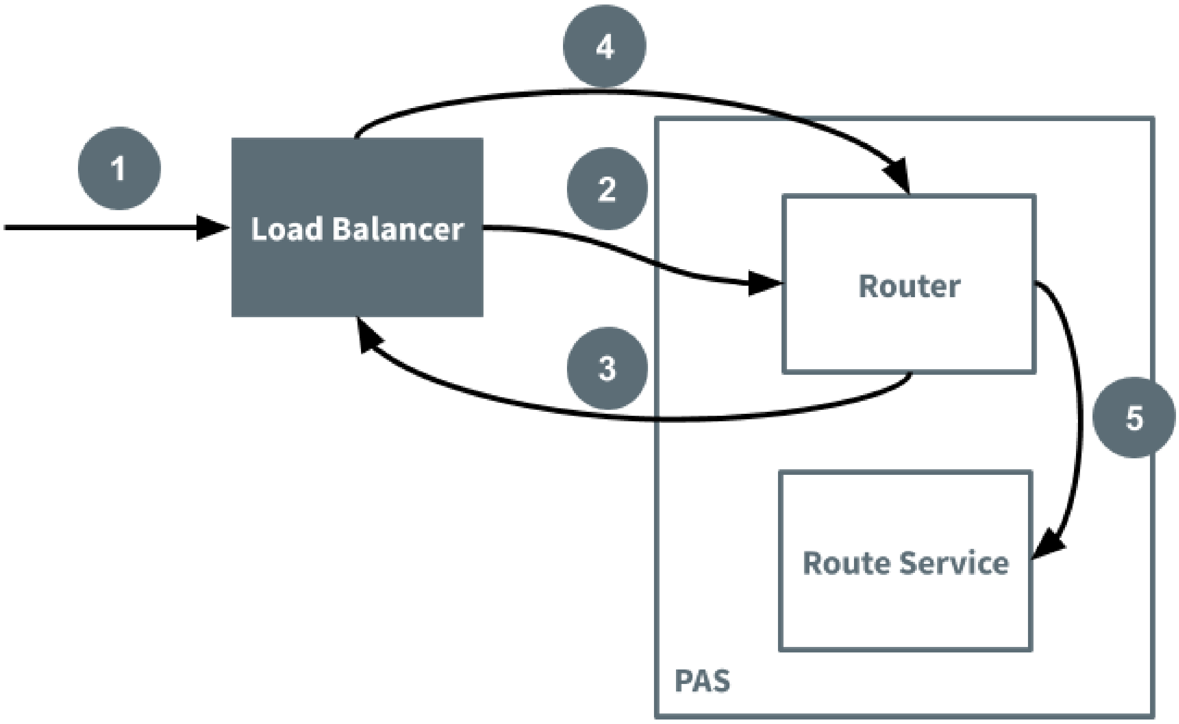
Default Lookup and Internal Route Service

This section describes how the Gorouter handles app requests when:

- The **Bypass security checks for route service lookup** check box in TAS for VMs is deactivated.
- The app is bound to a route service that is hosted on TAS for VMs.

In this case, when the Gorouter receives the request, it sends the traffic back to the load balancer to resolve DNS. The load balancer then sends the traffic back to the Gorouter.

The diagram below illustrates the flow of the request and numbers the steps to indicate order of occurrence.



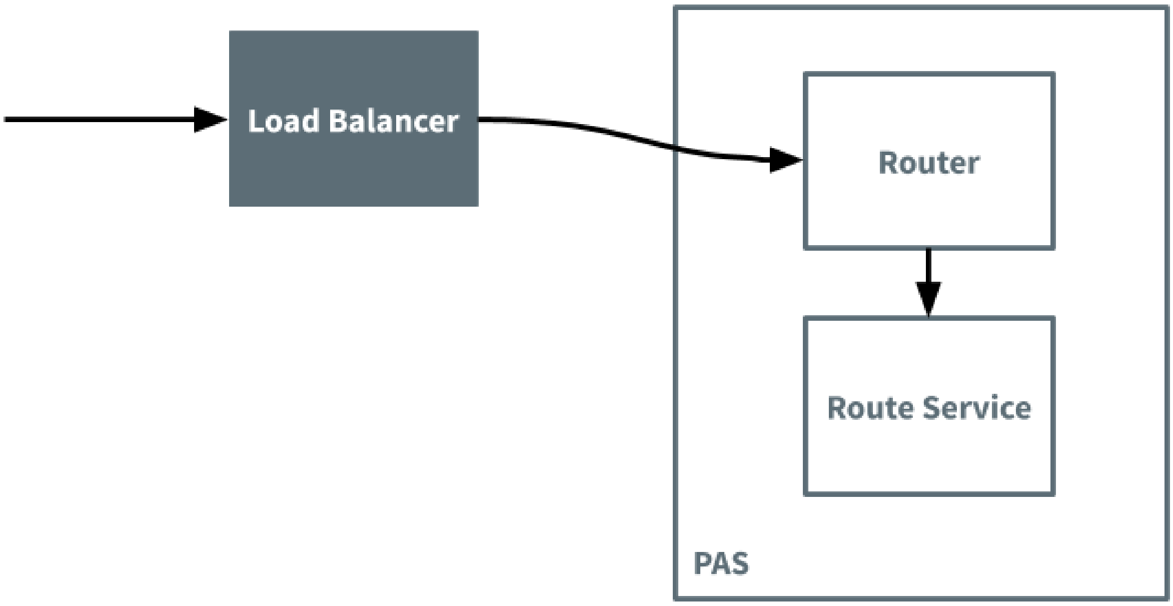
Bypass Mode and Internal Route Service

This section describes how the Gorouter handles app requests when:

- The **Bypass security checks for route service lookup** check box in TAS for VMs is enabled.
- The app is bound to a route service that is hosted on TAS for VMs.

In this case, when the Gorouter receives the request, it sends it directly to the route service. This assumes the Gorouter finds an existing route for the route service.

The diagram below illustrates the flow of the request. Arrows indicate the flow of the request through platform components. A request goes directly from the load balancer to the Gorouter to the route service.



Bypass Mode and External Route Service (Security Risk)

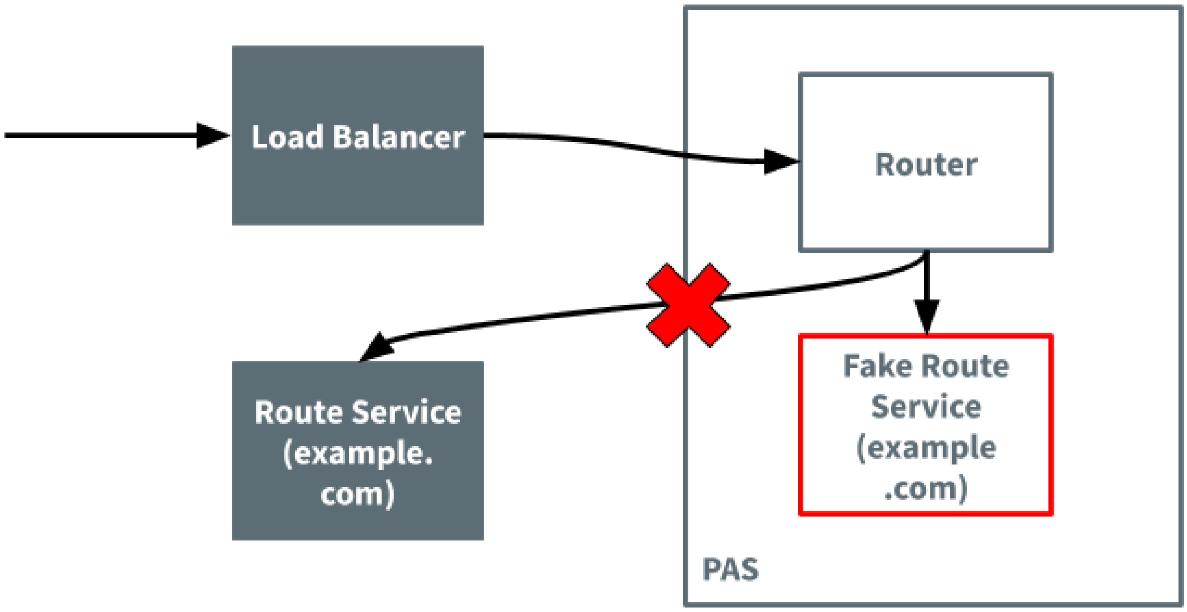
This section describes how the Gorouter handles app requests when:

- The **Bypass security checks for route service lookup** check box in TAS for VMs is enabled.
- The app is bound to a route service that is hosted outside of TAS for VMs.

In this case, when the Gorouter receives the request, it checks for an existing route and then sends the request directly to the route service. This enables external clients to intercept route service traffic. A developer can register the external route service domain as a private domain in TAS for VMs and map it to their own, malicious app. When the Gorouter receives a request for the original app bound to the external route service, it finds the domain internally and sends the request to the malicious app.

> **Note:** This vulnerability exists for both externally hosted route services and route services hosted on a separate foundation. If all of your route services are hosted internally on the same foundation, you are not at risk. However, you would be at risk if externally hosted route services are later configured.

The following diagram illustrates the flow of the request in the case that it is intercepted:


Load Balancer
Router
Route Service (example. com)
Fake Route Service (example .com)
PAS

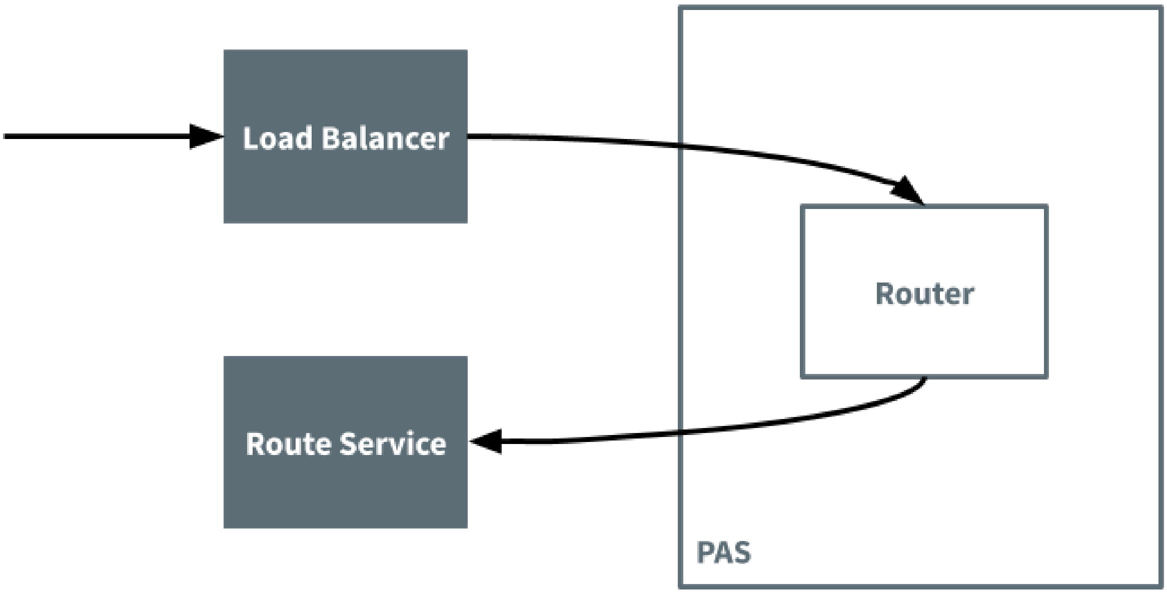
Default Lookup and External Route Service

This section describes how the Gorouter handles app requests when:

- The **Bypass security checks for route service lookup** check box in TAS for VMs is deactivated.
- The app is bound to a route service that is hosted outside of TAS for VMs.

In this case, the Gorouter sends traffic directly to the external route service without checking for an existing route.

The diagram below illustrates the flow of the request. The arrows indicate the flow of the request through platform components. A request goes directly from the load balancer to the Gorouter to the route service.



Internal App Security

In this section:

- Container Security
- Container-to-Container Networking
- App Security Groups
- Restricting App Access to Internal TAS for VMs Components

Container security

This topic tells you about how VMware Tanzu Application Service for VMs (TAS for VMs) secures the containers that host app instances on Linux.

For an overview of other TAS for VMs security features, see TAS for VMs Security.

The sections in this topic provide the following information:

- Container Mechanics provides an overview of container isolation.
- Inbound and Outbound Traffic from TAS for VMs provides an overview of container networking and describes how TAS for VMs admins customize container network traffic rules for their deployment.
- Container Security describes how TAS for VMs secures containers by running app instances in unprivileged containers and by hardening them.

Container mechanics

Each instance of an app deployed to TAS for VMs runs within its own self-contained environment, a Garden container. This container isolates processes, memory, and the filesystem using operating system features and the characteristics of the virtual and physical infrastructure where TAS for VMs is deployed. For more information about Garden containers, see Garden.

TAS for VMs achieves container isolation by namespacing kernel resources that would otherwise be shared. The intended level of isolation is set to prevent multiple containers that are present on the same host from detecting each other. Every container includes a private root filesystem, which includes a Process ID (PID), namespace, network namespace, and mount namespace.

TAS for VMs creates container filesystems using the Garden Rootfs (GrootFS) tool. It stacks the following using OverlayFS:

- A **read-only base filesystem**: This filesystem has the minimal set of operating system packages and Garden-specific modifications common to all containers. Containers can share the same read-only base filesystem because all writes are applied to the read-write layer.
- A **container-specific read-write layer**: This layer is unique to each container and its size is limited by XFS project quotas. The quotas prevent the read-write layer from overflowing into unallocated space.

For more information about GrootFS, see Garden RootFS (GrootFS) in *Garden*.

Resource control is managed using Linux control groups. Associating each container with its own cgroup or job object limits the amount of memory that the container may use. Linux cgroups also require the container to use a fair share of CPU compared to the relative CPU share of other containers.

Note: TAS for VMs does not support a RedHat Enterprise Linux OS stemcell. This is due to an inherent security issue with the way RedHat handles user namespacing and container isolation.

CPU

Each container is placed in its own cgroup. Cgroups make each container use a fair share of CPU relative to the other containers. This prevents oversubscription on the host VM where one or more containers hog the CPU and leave no computing resources to the others.

The way cgroups allocate CPU time is based on shares. CPU shares do not work as direct percentages of total CPU usage. Instead, a share is relative in a given time window to the shares held by the other containers. The total amount of CPU that can be overall divided among the cgroups is what is left by other processes that may run in the host VM.

Generally, cgroups offers two possibilities for limiting the CPU usage: *CPU affinity* and *CPU bandwidth*, the latter of which is used in TAS for VMs.

- CPU affinity consists of binding a cgroup to certain CPU cores. The actual amount of CPU cycles that can be consumed by the cgroup is thus limited to what is available on the bound CPU cores.
- CPU bandwidth sets the weight of a cgroup with the process scheduler. The process scheduler divides the available CPU cycles among cgroups depending on the shares held by each cgroup, relative to the shares held by the others. For example, consider two cgroups, one holding two shares and one holding four. Assuming the process scheduler gets to administer 60 CPU cycles, the first cgroup with two shares will get one third of those

available CPU cycles, as it holds one third of the overall shares. Similarly, the second cgroup will get 40 cycles, as it holds two thirds of the collective shares.

The calculation of the CPU usage based on the percentage of the total CPU power available is quite sophisticated and is performed regularly as the CPU demands of the various containers fluctuates. Specifically, the percentage of CPU cycles a cgroup gets can be calculated by dividing the `cpu.shares` it holds by the sum of the `cpu.shares` of all the cgroups that are currently doing CPU activity, as shown in the following calculation:

```
process_cpu_share_percent = cpu.shares / sum_cpu_shares * 100
```

In TAS for VMs, cgroup shares range from 10 to 1024, with 1024 being the default. The actual amount of shares a cgroup gets can be read from the `cpu.shares` file of the cgroup configurations pseudo-file-system available in the container at `/sys/fs/cgroup/cpu`.

The amount of shares given to an apps cgroup depends on the amount of memory the app declares to need in the manifest. TAS for VMs scales the number of allocated shares linearly with the amount of memory, with an app instance requesting 8 GB of memory getting the upper limit of 1024 shares, as shown in the following calculation:

```
process_cpu.shares = min( 1024*(application_memory / 8 GB), 1024)
```

The next example helps to illustrate this better. Consider three processes: P1, P2 and P3, which are assigned `cpu.shares` of 5, 20 and 30, respectively.

P1 is active, while P2 and P3 require no CPU. Hence, P1 may use the whole CPU. When P2 joins in and is doing some actual work, such as when a request comes in, the CPU share between P1 and P2 is calculated as follows:

- P1 -> 5 / (5+20) = 0.2 = 20%
- P2 -> 20 / (5+20) = 0.8 = 80%
- P3 (idle)

At some point, process P3 joins in. Then the distribution is recalculated again:

- P1 -> 5 / (5+20+30) = 0.0909 = ~9%
- P2 -> 20 / (5+20+30) = 0.363 = ~36%
- P3 -> 30 / (5+20+30) = 0.545 = ~55%

Should P1 become idle, the following recalculation between P2 and P3 takes place:

- P1 (idle)
- P2 -> 20 / (20+30) = 0.4 = 40%
- P3 -> 30 / (20+30) = 0.6 = 60%

If P3 finishes or becomes idle, then P2 can consume all the CPU, as another recalculation is performed.

In summary, the cgroup shares are the minimum guaranteed CPU share that the process can get. This limitation becomes effective only when processes on the same host compete for resources.

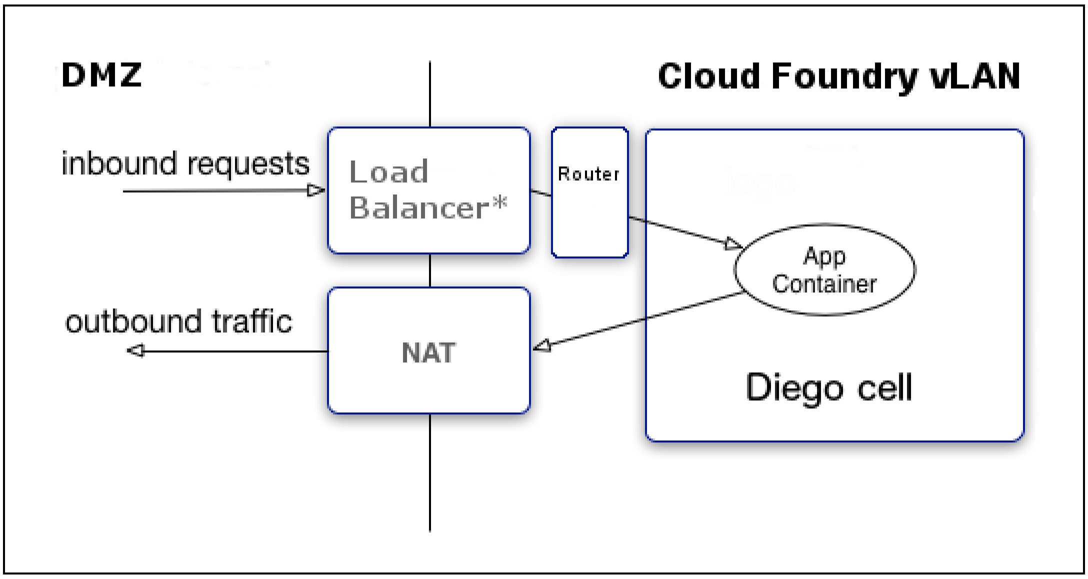
Inbound and outbound traffic from TAS for VMs

Learn about container networking and how TAS for VMs admins customize container network traffic rules for their deployment.

Networking overview

A host VM has a single IP address. If you configure the deployment with the cluster on a VLAN, as VMware recommends, then all traffic goes through the following levels of network address translation, as shown in the diagram below.

- **Inbound** requests flow from the load balancer through the router to the host Diego Cell, then into the app container. The router determines which app instance receives each request.
- **Outbound** traffic flows from the app container to the Diego Cell, then to the gateway on the Diego Cell's virtual network interface. Depending on your IaaS, this gateway may be a NAT to external networks.



The networking diagram shows the following:

- DMZ on left side
  - Inbound requests go to Load Balancer
  - Outbound traffic come from NAT
- Cloud Foundry on right side
  - Load balancer goes to router and to the app container inside the Diego Cell
  - App container response goes to NAT

Network traffic rules

Admins configure rules to govern container network traffic. This is how containers send traffic outside of TAS for VMs and receive traffic from outside, the Internet. These rules can prevent system access from external networks and between internal components and determine if apps can establish connections over the virtual network interface.

Admins configure these rules at two levels:

- Application Security Groups (ASGs) apply network traffic rules at the container level. For information about creating and configuring ASGs, see App Security Groups.
- Container-to-container networking policies determine app-to-app communication. Within TAS for VMs, apps can communicate directly with each other, but the containers are isolated from outside TAS for VMs. For information about administering container-to-container network policies, see Administering Container-to-Container Networking.

Container security

TAS for VMs secures containers through the following measures:

- Running app instances in unprivileged containers by default. For more information, see Types.
- Hardening containers by limiting functionality and access rights. For more information, see Hardening.
- Allowing admins to configure ASGs to block outbound connections from app containers. For information about creating and configuring ASGs, see App Security Groups.

Types

Garden has the following container types:

- unprivileged
- privileged

Currently, TAS for VMs runs all app instances and staging tasks in unprivileged containers by default. This measure increases security by eliminating the threat of root escalation inside the container.

Hardening

TAS for VMs mitigates against container breakout and denial of service attacks in the following ways:

- TAS for VMs uses the full set of Linux namespaces - IPC, Network, Mount, PID, User, UTS - to provide isolation between containers running on the same host. The User namespace is not used for privileged containers. For more information about Linux namespaces, see namespaces - overview of Linux namespaces in the Ubuntu documentation.
- In unprivileged containers, Ops Manager maps UID/GID 0 (root) inside the container user namespace to a different UID/GID on the host to prevent an app from inheriting UID/GID 0 on the host if it breaks out of the container.
    - TAS for VMs uses the same UID/GID for all containers.
    - TAS for VMs maps all UIDs except UID 0 to themselves. TAS for VMs maps UID 0 inside the container namespace to `MAX_UID-1` outside of the container namespace.
    - Container Root does not grant Host Root permissions.
- TAS for VMs mounts `/proc` and `/sys` as read-only inside containers.

- TAS for VMs disallows `dmesg` access for unprivileged users and all users in unprivileged containers.
- TAS for VMs uses `chroot` when importing docker images from docker registries.
- TAS for VMs establishes a container-specific overlay filesystem mount. TAS for VMs uses `pivot_root` to move the root filesystem into this overlay, in order to isolate the container from the host system's filesystem. For more information about `pivot_root`, see pivot_root - change the root filesystem in the Ubuntu documentation.
- TAS for VMs does not call any binary or script inside the container filesystem, in order to eliminate any dependencies on scripts and binaries inside the root filesystem.
- TAS for VMs avoids side-loading binaries in the container through bind mounts or other methods. Instead, it re-executes the same binary by reading it from `/proc/self/exe` whenever it needs to run a binary in a container.
- TAS for VMs establishes a virtual ethernet pair for each container for network traffic. For more information, see Container Network Traffic. The virtual ethernet pair has the following features:
  - One interface in the pair is inside the container's network namespace, and is the only non-loopback interface accessible inside the container.
  - The other interface remains in the host network namespace and is bridged to the container-side interface.
  - Egress allow list rules are applied to these interfaces according to ASGs configured by the admin.
  - First-packet logging rules may also be enabled on TCP allow list rules.
  - DNAT rules are established on the host to enable traffic ingress from the host interface to allowed ports on the container-side interface.
- TAS for VMs applies disk quotas using container-specific XFS quotas with the specified disk-quota capacity.
- TAS for VMs applies a total memory usage quota through the memory cgroup and destroys the container if the memory usage exceeds the quota.
- TAS for VMs applies a fair-use limit to CPU usage for processes inside the container through the `cpu.shares` control group.
- TAS for VMs allows admins to rate limit the maximum bandwidth consumed by single-app containers, configuring `rate` and `burst` properties on the `silk-cni` job.
- TAS for VMs limits access to devices using cgroups but explicitly allows the following safe device nodes:
  - `/dev/full`
  - `/dev/fuse`
  - `/dev/null`
  - `/dev/ptmx`
  - `/dev/pts/*`

    - `/dev/random`
    - `/dev/tty`
    - `/dev/tty0`
    - `/dev/tty1`
    - `/dev/urandom`
    - `/dev/zero`
    - `/dev/tap`
    - `/dev/tun`

- TAS for VMs drops the following Linux capabilities for all container processes. Every dropped capability limits the actions the root user can perform.
    - `CAP_DAC_READ_SEARCH`
    - `CAP_LINUX_IMMUTABLE`
    - `CAP_NET_BROADCAST`
    - `CAP_NET_ADMIN`
    - `CAP_IPC_LOCK`
    - `CAP_IPC_OWNER`
    - `CAP_SYS_MODULE`
    - `CAP_SYS_RAWIO`
    - `CAP_SYS_PTRACE`
    - `CAP_SYS_PACCT`
    - `CAP_SYS_BOOT`
    - `CAP_SYS_NICE`
    - `CAP_SYS_RESOURCE`
    - `CAP_SYS_TIME`
    - `CAP_SYS_TTY_CONFIG`
    - `CAP_LEASE`
    - `CAP_AUDIT_CONTROL`
    - `CAP_MAC_OVERRIDE`
    - `CAP_MAC_ADMIN`
    - `CAP_SYSLOG`
    - `CAP_WAKE_ALARM`
    - `CAP_BLOCK_SUSPEND`
    - `CAP_SYS_ADMIN` for unprivileged containers

For more information about Linux capabilities, see capabilities - overview of Linux capabilities in the Ubuntu documentation.

Container to container networking

This topic provides you with an overview of how container-to-container networking works in VMware Tanzu Application Service for VMs (TAS for VMs).

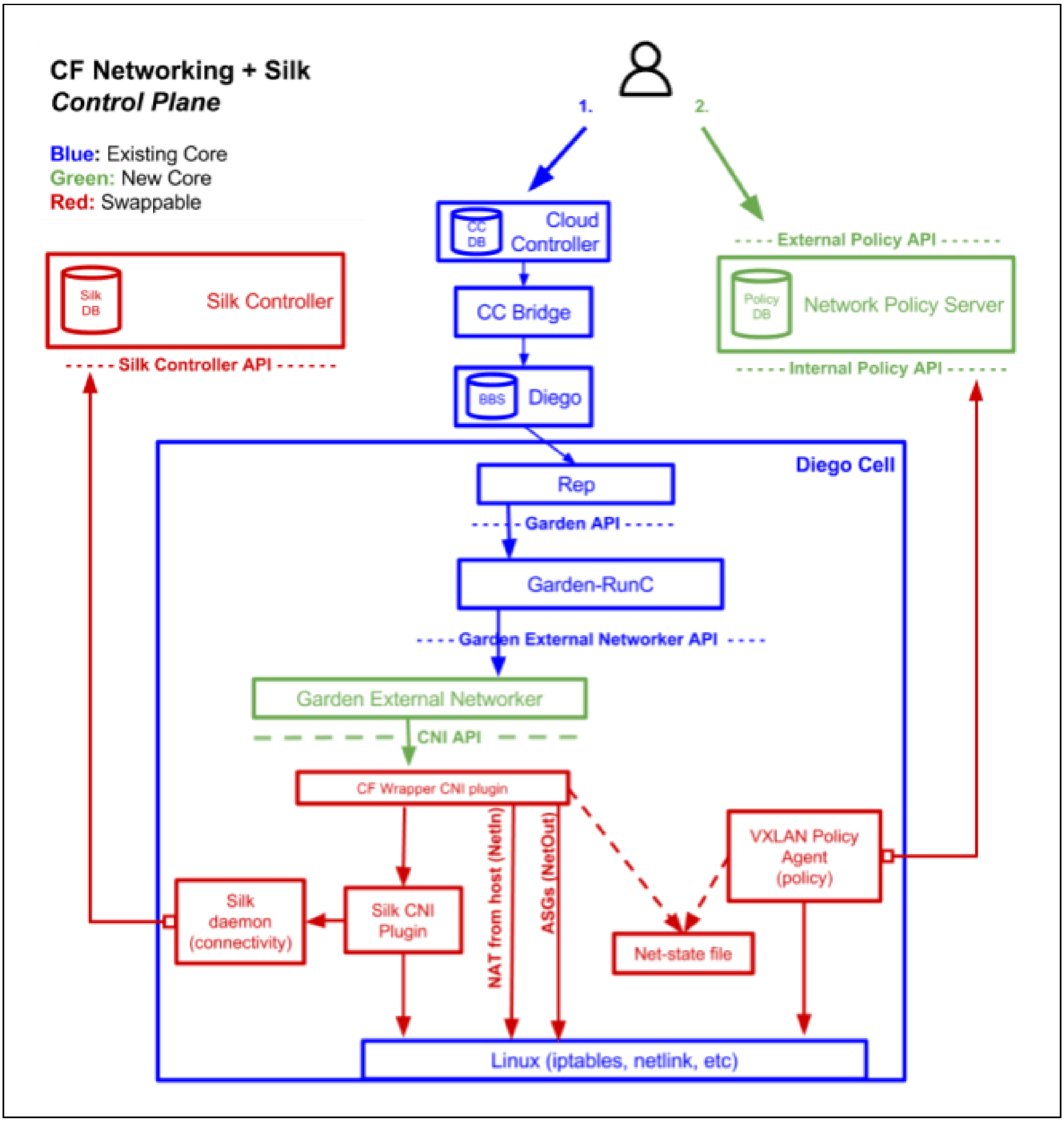
> **Note:** Container-to-container networking is not available for apps hosted on Windows.

The container-to-container networking feature enables app instances to communicate with each other directly. Container-to-container networking is always enabled in TAS for VMs. For more information about how to configure container-to-container networking, see Configuring Container-to-Container Networking.

Architecture

Container-to-container networking integrates with [Garden-runC] in a Diego deployment. The CF Networking Release includes several core components, as well as swappable components. For more information about Garden-runC, see Garden-runC in *Garden*. For more information about Diego, see Diego Components and Architecture. For more information about the CF Networking release, see CF Networking Release on GitHub.

To understand the components and how they work, see the diagram and tables below. The diagram highlights TAS for VMs components in blue and green. The diagram also highlights swappable components in red.



Core components

The container-to-container networking BOSH release includes the following core components:

Part	Function
Policy Server	A central management node that: • Maintains a database of policies for traffic between apps. The cf CLI calls an API to create or update a record in the policy database whenever you create or remove a policy. • Exposes a JSON REST API used by the cf CLI. This API serves traffic using TLS.

Garden External Networker	A Garden-runC add-on deployed to every Diego Cell that: • Invokes the CNI plugin component to set up the network for each app. • Forwards ports to support incoming connections from the Gorouter, TCP Router, and Diego SSH Proxy. This keeps apps externally reachable. • Installs outbound allow list rules to support Application Security Groups (ASGs).

Swappable components

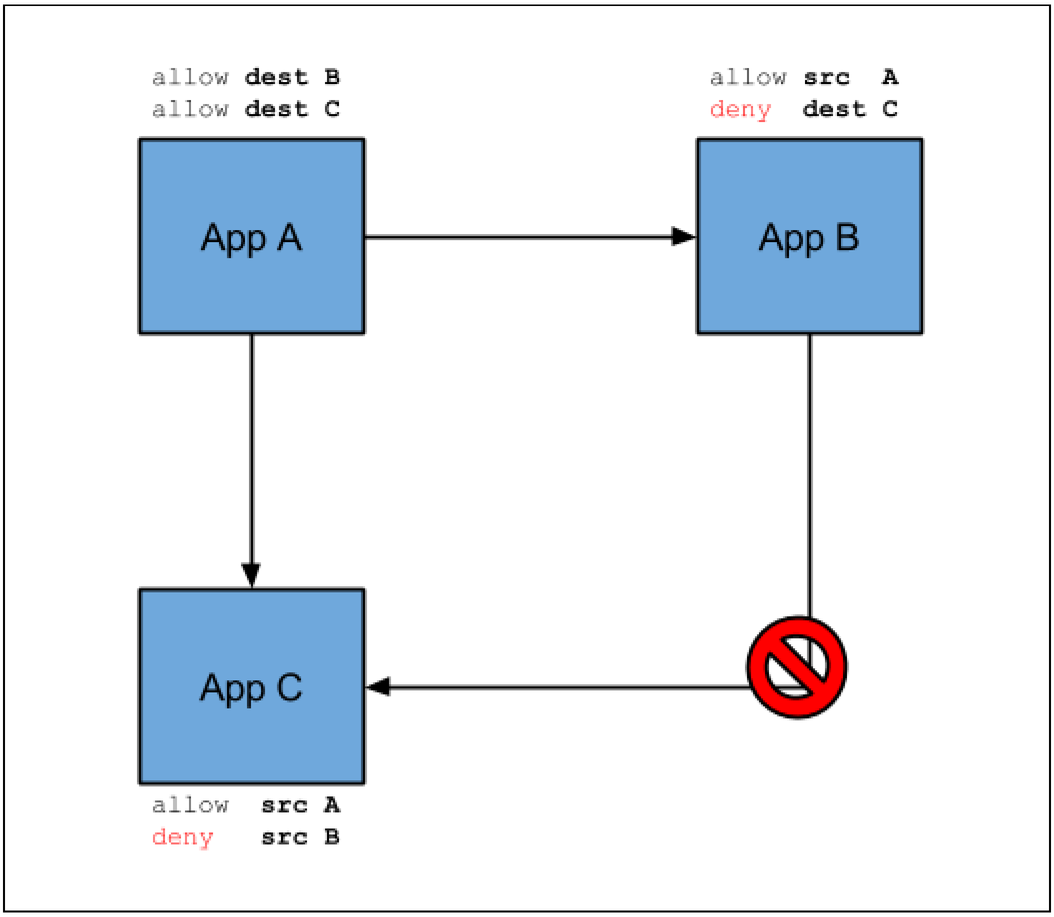
Part	Function
Silk CNI plugin	A plugin that provides IP address management and network connectivity to app instances as follows: • Uses a shared VXLAN overlay network to assign each container a unique IP address. • Installs network interface in container using the Silk VXLAN back end. This is a shared, flat L3 network.
VXLAN Policy Agent	Enforces network policy for traffic between apps as follows: • Discovers desired network policies from the Policy Server Internal API. • Updates iptables rules on the Diego Cell to allow approved inbound traffic. • Tags outbound traffic with the unique identifier of the source app using the VXLAN Group-Based Policy (GBP) header.

App instance communication

The diagram below illustrates how app instances communicate in a deployment with container-to-container networking enabled. In this example, the operator creates two policies to regulate the flow of traffic between **App A**, **App B**, and **App C**.

- Allow traffic from **App A** to **App B**
- Allow traffic from **App A** to **App C**

If traffic and its direction is not explicitly allowed, it is denied. For example, **App B** cannot send traffic to **App C**.



Overlay network

Container-to-container networking uses an overlay network to manage communication between app instances.

Overlay networks are not externally routable, and traffic sent between containers does not exit the overlay. You can use the same overlay network range for different TAS for VMs deployments in your environment.

The overlay network range defaults to `10.255.0.0/16`. You can modify the default to any RFC 1918 range that meets the following requirements:

- The range is not used by services that app containers access.
- The range is not used by the underlying TAS for VMs infrastructure.

All Diego Cells in your TAS for VMs deployment share this overlay network. By default, each Diego Cell is allocated a /24 range that supports 254 containers per Diego Cell, one container for each of the usable IP addresses, `.1` through `.254`. To modify the number of Diego Cells your overlay network supports, see Overlay Network in *Configuring Container-to-Container Networking*.

TAS for VMs container networking is currently supported only on Linux.

⚠ **Warning:** The overlay network IP address range must not conflict with any other IP addresses in the network. If a conflict exists, Diego Cells cannot reach any endpoint that has a conflicting IP address.

Note: Traffic to app containers from the Gorouter or from app containers to external services uses Diego Cell IP addresses and NAT, not the overlay network.

Policies

Enabling container-to-container networking for your deployment allows you to create policies for communication between app instances. The container-to-container networking feature also provides a unique IP address to each app container and provides direct IP reachability between app instances.

The policies you create specify a source app, destination app, protocol, and port so that app instances can communicate directly without going through the Gorouter, a load balancer, or a firewall. Container-to-container networking supports UDP and TCP, and you can configure policies for multiple ports. These policies apply immediately without having to restart the app.

Additionally, policies use and and track the GUIDs of the apps. The policies continue to work when apps redeploy, or if they crash and Diego places them in a new container. Pushing a brand new app requires a new policy, but not updates to an existing app because an app always retains its GUID.

App Service Discovery

The TAS for VMs platform supports DNS-based service discovery that lets apps find each other's internal addresses. For example, a front end app instance can use the service discovery mechanism to establish communications with a back end app instance. To set up and use app service discovery, see App Service Discovery in *Configuring Container-to-Container Networking*.

Container-to-container app service discovery does not provide client-side load balancing or circuit-breaking, and it does not apply to `cf marketplace` services or require app binding. It just lets apps publish service endpoints to each other, unbrokered and unmediated.

Alternative network stacks

The BOSH release that contains the container-to-container networking feature is composed of a pluggable network stack. Advanced users or third-party vendors can integrate a different network stack. For more information about third-party plugins, see 3rd Party Plugin Development for Container Networking in the CF Networking Release repository on GitHub.

Container-to-Container networking versus ASGs

Both app security groups (ASGs) and container-to-container networking policies affect traffic from app instances. The following table highlights differences between ASGs and container-to-container networking policies.

	ASGs	Container-to-Container Networking Policies
Policy granularity	From a space to an IP address range	From a source app to a destination app
Scope	For a space, org, or deployment	For app to app only

Traffic direction	Outbound control	Policies apply for incoming packets from other app instances
Source app	Is not known	Is identified because of direct addressability
Policies take affect	After app restart	Immediately

Securing container-to-container traffic

TLS encapsulation for container-to-container traffic is disabled by default.

To secure communication between the source and destination containers on the overlay network, you can enable TLS encapsulation using either of the following options:

- **Automatic**: Use the automatic option when you only care that traffic between the containers cannot be sniffed on the overlay network.
- **Manual**: Use the manual option when your app also needs to use TLS capabilities for its operation. For example, the destination app can examine the client certificate and reject service for those that are not permitted.

> **Note:** The source app can make use of its own platform-provisioned certificate when it opens a TLS connection to the destination container.

For more information about the TLS encapsulation options, see the following table:

Option	Configuration	Description
Automatic	Connect to port 61443 on the destination container.	• The platform provisions certificates for each app. • The platform ensures that TLS terminates within the destination container and passes cleartext traffic to the app. • The destination app does not need special configuration.
Manual	Implement your own TLS termination in the app for the port provided.	Configure the destination app to implement its own TLS termination.

Specific Cloud Foundry vendors might provide additional methods for securing container-to-container traffic.

> For information about how the platform can encrypt public traffic to apps, see TLS to Apps and Other Back End Services in *HTTP Routing*.

App security groups

App Security Groups (ASGs) are a collection of egress rules that enable you to specify the protocols, ports, and IP address ranges where app or task instances send traffic.

ASGs define **allow** rules, and their order of evaluation is unimportant when multiple ASGs apply to the same space or deployment. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

Staging and running ASGs

Admins can define a `staging` ASG for app and task staging, and a `running` ASG for app and task runtime.

When apps or tasks begin staging, they require traffic rules permissive enough to allow them to pull resources from the network. A running app or task no longer needs to pull resources, so traffic rules can be more restrictive and secure. To distinguish between these two security requirements, you can define a `staging` ASG for app and task staging with more permissive rules, and a `running` ASG for app and task runtime with less permissive rules.

Platform wide and space scoped ASGs

To provide granular control when securing a deployment, you can assign platform-wide ASGs that apply to all app and task instances for the entire deployment, or space scoped ASGs that apply only to apps and tasks in a particular space.

In environments with both platform-wide and space-specific ASGs, the ASGs for a particular space are combined with the platform ASGs to determine the effective rules for that space. Any of the permitted traffic in the set of applicable ASGs is then effectively permitted for that app.

Dynamic ASGs

Dynamic ASGs allow ASGs to be applied to apps without the need for an application restart. Only containers using the Silk CNI will have their ASGs updated dynamically; Windows apps and apps using other CNIs (e.g., NSX-T) do not support Dynamic ASGs at this time. Dynamic ASGs are enabled with cf deployment v20.0.0 and higher through cf-networking-release 3.0.0. It is recommended to use Dynamic ASGs with capi-release v1.126.0 or later for improved performance on the /v3/security_groups APIs.

To deactivate Dynamic ASGs: use the ops file `operations/disable-dynamic-asgs.yml` from cf-deployment v20.0.0 or higher.

Simplifying ASGs with a services subnet

ASGs can be complicated to configure correctly, especially when the specific IP addresses listed in a group change.

To simplify securing a deployment while still allowing apps reach external services, operators can deploy the services into a subnet that is separate from their TAS for VMs deployment, then create ASGs for the apps that allow those service subnets, while denying access to any VM hosting other apps.

For examples of typical ASGs, see Typical ASGs.

Default ASGs

TAS for VMs defines one default ASG, `default_security_group`. This group allows all outbound traffic from app containers on public and private networks except for the link-local range, `169.254.0.0/16`, which is blocked.

> **Caution:** For security, TAS for VMs administrators must modify the default ASGs so that outbound network traffic cannot access internal components. For more information, see Restricting App Access to Internal TAS for VMs Components.

The ASG is defined in the Cloud Controller configuration as follows:

```
security_group_definitions:
- name: default_security_group
  rules:
  - protocol: all
    destination: 0.0.0.0-169.253.255.255
  - protocol: all
    destination: 169.255.0.0-255.255.255.255
```

ASG sets

ASGs are applied by configuring ASG sets differentiated by *scope*, platform-wide or space specific, and *lifecycle*, staging or running.

Currently, the following ASG sets exist in TAS for VMs:

- Platform-wide staging ASG set, also called “default-staging”
- Platform-wide running ASG set, also called “default-running”
- Space-scoped staging ASG set
- Space-scoped running ASG set

In environments with both platform-wide and space-specific ASG sets, combine the ASG sets for a particular space with the platform ASG sets to determine the rules for that space.

The following table indicates the differences between the four sets:

When an ASG is bound to the...	the ASG rules are applied to...
Platform-wide staging ASG set	the staging lifecycle for all apps and tasks.
Platform-wide running ASG set	the running lifecycle for all app and task instances.
Space-scoped staging ASG set	the staging lifecycle for apps and tasks in a particular space.
Space-scoped running ASG set	the running lifecycle for app and task instances in a particular space.

Typically, ASGs applied during the staging lifecycle are more permissive than the ASGs applied during the running lifecycle. This is because staging often requires access to different resources, such as dependencies.

You use different commands to apply an ASG to each of the four sets.

For more information, see Managing ASGs with the cf CLI.

> **Note:** To apply a staging ASG to apps within a space, you must use cf CLI v6.28.0 or later.

Structure and attributes of ASGs

ASG rules are specified as a JSON array of ASG objects. An ASG object has the following attributes:

Attribute	Description	Notes
`protocol`	`tcp`, `udp`, `icmp`, or `all`	Required
`destination`	A single IP address, an IP address range like `192.0.2.0-192.0.2.50`, or a CIDR block that can receive traffic	
`ports`	A single port, multiple comma-separated ports, or a single range of ports that can receive traffic. Examples: `443`, `80,8080,8081`, `8080-8081`	Only possible if `protocol` is `tcp` or `udp`.
`code`	ICMP code	Required when `protocol` is `icmp`. A value of `-1` allows all codes.
`type`	ICMP	Required when `protocol` is `icmp`. A value of `-1` allows all types.
`log`	Set to `true` to enable logging. For more information about how to configure system logs to be sent to a syslog drain, see Using Log Management Services.	Logging is only supported with protocol type `tcp`.
`description`	An optional field for operators managing ASG rules	

Managing ASGs

The following table outlines the flow of tasks that you implement over the lifecycle of ASGs.

For information about procedures for each of these tasks, see Managing ASGs with cf CLI.

	Task	For more information, see:
1.	Review the existing ASGs. If this is a new deployment, these consist of only the default ASGs.	View ASGs
2.	Create new ASGs.	Create ASGs
3.	Update the existing ASGs.	Update ASGs
4.	Bind ASGs to an ASG set.	Bind ASGs
5.	If you need to delete an ASG, first unbind it, then delete it.	Unbind ASGs and Delete ASGs

Managing ASGs with cf CLI

Learn about the commands you need to create and manage ASGs.

Many of the following procedures require the Cloud Foundry Command Line Interface (cf CLI). To download the cf CLI, see Installing the cf CLI.

Viewing ASGs

To view information about existing ASGs, run the following commands:

Command	Output
`cf security-groups`	All ASGs
`cf staging-security-groups`	All ASGs applied to the platform-wide staging ASG set
`cf running-security-groups`	All ASGs applied to the platform-wide running ASG set
`cf security-group ASG`	All rules in the specified ASG, where `ASG` is the name of the ASG

Note: You can also view ASGs in Apps Manager under the **Settings** tab of a space or an app.

Creating ASGs

To create an ASG:

1. Create a rules file: a JSON-formatted single array containing objects that describe the rules. See the following example, which allows ICMP traffic of code `1` and type `0` to all destinations, and TCP traffic to `10.0.11.0/24` on ports `80` and `443`. For more information, see The Structure and Attributes of ASGs.

```
[
  {
    "protocol": "icmp",
    "destination": "0.0.0.0/0",
    "type": 0,
    "code": 0
  },
  {
    "protocol": "tcp",
    "destination": "10.0.11.0/24",
    "ports": "80,443",
    "log": true,
    "description": "Allow http and https traffic to ZoneA"
  }
]
```

2. Run:

```
cf create-security-group ASG PATH-TO-RULES-FILE.json
```

Where:

- `ASG` is the name of your ASG.
- `PATH-TO-RULES-FILE.json` is the absolute or relative path to a rules file.

In the following example, `EXAMPLE-ASG` is the name of an ASG, and `~/workspace/example-asg.json` is the path to a rules file:

```
cf create-security-group EXAMPLE-ASG ~/workspace/example-asg.json
```

After the ASG is created, you must bind it to an ASG set before it takes effect. For more information, see Bind ASGs.

Bind ASGs

Note: Binding an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

To apply an ASG, you must first bind it to an ASG set.

Platform-wide Staging ASG Set

To bind an ASG to the platform-wide staging ASG set:

1. Run:

   ```
   cf bind-staging-security-group ASG
   ```

   Where `ASG` is the name of your ASG.

Platform-wide Running ASG Set

To bind an ASG to the platform-wide running ASG set:

1. Run:

   ```
   cf bind-running-security-group ASG
   ```

   Where `ASG` is the name of your ASG.

Space-scoped Running ASG Set

To bind an ASG to a space-scoped running ASG set:

1. Run:

   ```
   cf bind-security-group ASG ORG --space SPACE
   ```

   Where:

   - `ASG` is the name of your ASG.
   - `ORG` is the name of the org where you want to bind the ASG set.
   - `SPACE` is the name of the space where you want to bind the ASG set.

Space-scoped Staging ASG Set

To bind an ASG to a space-scoped staging ASG set:

1. Run:

```
cf bind-security-group ASG ORG --space SPACE --lifecycle staging
```

Where:

- `ASG` is the name of your ASG.
- `ORG` is the name of the org where you want to bind the ASG set.
- `SPACE` is the name of the space where you want to bind the ASG set.

Update ASGs

To update an existing ASG:

> **Note:** Updating an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

1. Edit the ASG rules in the JSON file you created in Create ASGs.
2. Run:

```
cf update-security-group ASG PATH-TO-RULES-FILE.json
```

Where:

- `ASG` is name of the existing ASG you want to change.
- `PATH-TO-RULES-FILE.json` is the absolute or relative path to a rules file.

In the following example, `example-asg` is the name of an ASG, and `~/workspace/example-asg-v2.json` is the path to a rules file:

```
cf update-security-group example-asg ~/workspace/example-asg-v2.json
```

Unbind ASGs

> **Note:** Updating an ASG does not affect started apps until you restart them. To restart all of the apps in an org or a space, use the app-restarter cf CLI plugin.

Platform-wide Staging ASG Set

To unbind an ASG from the platform-wide staging ASG set:

1. Run:

```
cf unbind-staging-security-group ASG
```

Where `ASG` is the name of your ASG.

Platform-wide Running ASG Set

To unbind an ASG from the platform-wide running ASG set:

1. Run:

   ```
   cf unbind-running-security-group ASG
   ```

   Where `ASG` is the name of your ASG.

Space-scoped ASG Set

To unbind an ASG from a specific space:

1. Run:

   ```
   cf unbind-security-group ASG ORG --space SPACE --lifecycle running
   ```

   Where:

   - `ASG` is the name of your ASG.
   - `ORG` is the org where you want to unbind the ASG set.
   - `SPACE` is the space where you want to unbind the ASG set.
   - To unbind from the staging ASG set, replace `running` with `staging`.

Delete ASGs

Note: You can only delete unbound ASGs. To unbind ASGs, see Unbind ASGs.

To delete an ASG:

1. Run:

   ```
   cf delete-security-group ASG
   ```

   Where `ASG` is the name of your ASG.

Typical ASGs

The following table describes examples of typical ASGs. Configure your ASGs in accordance with your organization's network access policy for untrusted apps.

ASG	For access to:
`dns`	DNS, either public or private
`public-networks`	Public networks, excluding IaaS metadata endpoints
`private-networks`	Private networks in accordance with RFC-1918
`load-balancers`	The internal TAS for VMs load balancer and others
`internal-proxies`	Internal proxies
`internal-databases`	Internal databases

DNS

To resolve hostnames to IP addresses, apps require DNS server connectivity, which typically use port 53. Admins should create or update a `dns` ASG with appropriate rules. Admins can further restrict the DNS servers to specific IP addresses or ranges of IP addresses.

The following is an example `dns` ASG:

```
[
  {
    "protocol": "tcp",
    "destination": "0.0.0.0/0",
    "ports": "53"
  },
  {
    "protocol": "udp",
    "destination": "0.0.0.0/0",
    "ports": "53"
  }
]
```

Public Networks

Apps often require public network connectivity to retrieve app dependencies, or to integrate with services available on public networks. Example app dependencies include public Maven repositories, NPM, RubyGems, and Docker registries.

Note: You should exclude IaaS metadata endpoints, such as `169.254.169.254`, because the metadata endpoint can expose sensitive environment information to untrusted apps. The `public_networks` example below accounts for this recommendation.

The following is an example `public_networks` ASG:

```
[
  {
    "destination": "0.0.0.0-9.255.255.255",
    "protocol": "all"
  },
  {
    "destination": "11.0.0.0-169.253.255.255",
    "protocol": "all"
  },
  {
    "destination": "169.255.0.0-172.15.255.255",
    "protocol": "all"
  },
  {
    "destination": "172.32.0.0-192.167.255.255",
    "protocol": "all"
  },
  {
    "destination": "192.169.0.0-255.255.255.255",
    "protocol": "all"
  }
]
```

Private Networks

Network connections that are commonly allowable in private networks include endpoints such as proxy servers, Docker registries, load balancers, databases, messaging servers, directory servers, and file servers. Configure appropriate private network ASGs as appropriate. You might find it helpful to use a naming convention with `private_networks` as part of the ASG name, such as `private_networks_databases`.

Note: You should exclude any private networks and IP addresses that app and task instances should not have access to.

The following is an example `private_networks` ASG:

```
[
  {
    "protocol": "tcp",
    "destination": "10.0.0.0-10.255.255.255",
    "ports": "443"
  },
  {
    "protocol": "tcp",
    "destination": "172.16.0.0-172.31.255.255",
    "ports": "443"
  },
  {
    "protocol": "tcp",
    "destination": "192.168.0.0-192.168.255.255",
    "ports": "443"
  }
]
```

Marketplace Services

Each installed Marketplace Service requires its own set of ASG rules to function properly. To determine which ASG rules it requires, see the installation instructions for each installed Marketplace Service. For more information about how to provision and integrate services, see Services Overview.

About the ASG Creator Tool

The ASG Creator is a command line tool that you can use to create JSON rules files. The ASG Creator lets you specify IP addresses, CIDRs, and IP address ranges that you want to disallow traffic to, as well as the addresses that you want to allow traffic to. Based on these disallow/allow (exclude/include) lists that you provide as input, the ASG Creator formulates a JSON file of allow rules.

In turn, the JSON file is the input for the `cf create-security-group` command that creates an ASG.

To download the latest release of the ASG Creator, see the Cloud Foundry Incubator repository on GitHub.

ASG Logging

For information about how you can use ASGs to correlate emitted logs back to an app, see How to use Application Security Group (ASG) logging in the Knowledge Base.

Restricting App Access to Internal TAS for VMs Components

This topic describes how to secure the component virtual machines (VMs) of your VMware Tanzu Application Service for VMs (TAS for VMs) deployment from being accessed by apps.

Overview

See the following list to understand the concepts for this topic:

- **How TAS for VMs determines where apps can send traffic**:
  - TAS for VMs uses *App Security Groups (ASGs)*, which are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps. For more information, see App Security Groups.
- **Why you must create new rules for outbound app traffic**:
  - TAS for VMs installs with a default ASG that allows apps running on your deployment to send traffic to almost any IP address. This means apps are not blocked from initiating connections to most network destinations unless an administrator takes action to update the ASGs with a more restrictive policy. For more information, see the Default ASGs section of the *App Security Groups* topic.
- **How you can set up new rules**:
  - To help secure your component VMs against apps while ensuring your apps can access the services they need, follow the steps in the Procedure section, which includes:

Step	Description
1	Determine Your Network Layout: The procedure for securing your deployment with ASGs varies depending on your network layout, which you can determine using Ops Manager.
2	Ensure Access for TAS for VMs System Apps: Bind the default ASG to the `system` org so that TAS for VMs system apps can continue accessing the system components they need after you remove the deployment-wide default ASG in Step 4.
3	Create New ASGs: Block apps from sending traffic to system components, but allow them to send traffic to the services they need.

4	Remove the Default ASG: After you create and bind new ASGs, you no longer need the deployment-wide default ASG bindings that allow apps to send traffic to any IP.
5	Restart Your Apps: To apply the ASG changes, you must restart all of the apps in your deployment.

- **When to set up new rules**:
  - VMware recommends that you complete this procedure directly after installing TAS for VMs and before developers push apps to the platform. If you complete the procedure after apps have been pushed to the platform, you must restart all the apps in your deployment.

Prerequisites

The procedure below requires that you have the latest release of ASG Creator from the Cloud Foundry incubator repository on GitHub. For more information, see the About the ASG Creator Tool section of the *App Security Groups* topic.

Procedure

This procedure outlines how to apply ASGs that prevent apps running on your deployment from accessing internal TAS for VMs components.

Step 1: Determine Your Network Layout

The procedure for securing your deployment with ASGs varies depending on your network layout. To determine your network layout:

1. Log in to Ops Manager.
2. For each tile, click **Assign AZs and Networks** and record the selected **Network** that the tile is installed on.
3. Based on the information you gathered, determine which of the following network layouts you have:

Layout Name	Layout Description
One Network	○ One network for Ops Manager and the BOSH Director, TAS for VMs, and services. **Note:** You cannot secure your deployment with ASGs if you have this network layout. Because Ops Manager dynamically allocates IPs, they cannot be easily excluded in the case of a single network.
Two Networks	○ One network for Ops Manager and the BOSH Director. ○ One network for TAS for VMs and Services.

Three Networks	○ One network for Ops Manager and the BOSH Director. ○ One network for TAS for VMs. ○ One network for all services.
Three or More Networks	○ One network for Ops Manager and the BOSH Director. ○ One network for TAS for VMs. ○ One network for each service.

4. If your network layout includes two or more networks, continue to Step 2: Ensure Access for TAS for VMs System Apps.

Step 2: Ensure Access for TAS for VMs System Apps

Applying the default ASG to the `system` org provides network access to TAS for VMs system apps without restrictions, which enables them to continue functioning properly after you perform Step 4: Remove the Deployment-wide Default ASG Binding. To apply the default ASG to the `system` org:

1. Bind the default ASG to the staging set in the `system` org by running:

```
cf bind-staging-security-group default_security_group system
```

2. Bind the default ASG to the running set in the `system` org by running:

```
cf bind-running-security-group default_security_group system
```

For more information about staging and running sets, see the ASG Sets section of the *App Security Groups* topic.

Step 3: Create New ASGs

This step outlines how to create ASGs that block apps from accessing TAS for VMs components and create any additional ASGs that allow apps to access the services they require.

Part A: Record CIDRs

To gather the CIDRs for each network in your deployment:

1. From the BOSH Director tile, click **Create Networks** within the **Settings** tab.
2. In the **Networks** section, expand each network in your deployment by clicking its name.
3. Record the **CIDR** for each network.

Part B: Create and Bind ASGs that Block Network Access

To create ASGs that block apps from sending traffic to the networks that host Ops Manager, TAS for VMs, and, optionally, any installed services:

1. Create a `config.yml` containing the appropriate content for your network layout and replace the indicated values with the CIDRs you gathered:
   - **Two Network Layout**:

```
exclude:
- YOUR-OPS-MANAGER-CIDR
- YOUR-TAS-AND-SERVICES-CIDR
```

- **Three Network Layout**:

  > **Note:** If you want to secure only the Ops Manager and TAS for VMs components, you can remove the services CIDR from the `exclude` section.

  ```
  exclude:
  - YOUR-OPS-MANAGER-CIDR
  - YOUR-TAS-CIDR
  - YOUR-SERVICES-CIDR
  ```

- **Three or More Network Layout**:

  > **Note:** If you want to secure only the Ops Manager and TAS for VMs components, you can remove the services CIDRs from the `exclude` section.

  ```
  exclude:
  - YOUR-OPS-MANAGER-CIDR
  - YOUR-TAS-CIDR
  - YOUR-SERVICE-CIDR-1
  - YOUR-SERVICE-CIDR-2
  etc...
  ```

2. To generate the default `public-networks.json` and `private-networks.json` files that contain your ASG rules, specifying the location of the `config.yml` file as input, run:

   ```
   asg-creator create --config config.yml
   ```

3. Create the `public-networks` ASG by running:

   ```
   cf create-security-group public-networks public-networks.json
   ```

4. Bind the ASG to the default staging set by running:

   ```
   cf bind-staging-security-group public-networks
   ```

5. Bind the ASG to the default running set by running:

   ```
   cf bind-running-security-group public-networks
   ```

6. Create the `private-networks` ASG by running:

   ```
   cf create-security-group private-networks private-networks.json
   ```

7. Bind the ASG to the default staging set by running:

```
cf bind-staging-security-group private-networks
```

8. Bind the ASG to the default running set by running:

```
cf bind-running-security-group private-networks
```

Note: To create and bind additional ASGs, see the Create ASGs and Bind ASGs sections of the *App Security Groups* topic.

Part C: Create and Bind ASGs for Service Access

Note: This part is only necessary if you blocked apps from a network that hosts services in the previous part. If you did not block apps from a network that hosts services, proceed to Step 4: Remove the Default ASG.

⚠ **Caution:** In the two network layout, TAS for VMs and services share the same network. This means that each time you create an ASG that allows apps to access a new port and protocol within the network, you further expose the TAS for VMs component VMs. This is a limitation of a two network layout. For guidance on network topology, see Reference Architecture.

Now that you have created ASGs to secure the Ops Manager, TAS for VMs, and service components, work with developers to create additional ASGs that give apps access to the services they need.

For more information about creating and binding ASGs, see the Managing ASGs with the cf CLI and Typical ASGs sections of the *App Security Groups* topic.

Step 4: Remove the Default ASG

Now that you have bound new ASGs to determine outbound traffic rules, you no longer need the default ASG bindings that allow apps to send traffic to any IP address. To remove the default ASG:

1. Unbind the default ASG from the staging set by running:

```
cf unbind-staging-security-group default_security_group
```

2. Unbind the default ASG from the running set by running:

```
cf unbind-running-security-group default_security_group
```

Step 5: Restart your Apps

To apply the ASG changes, you must restart all of the apps in your deployment. To mitigate app downtime during the restart, VMware recommends a blue-green deployment strategy. For more information, see Using Blue-Green Deployment to Reduce Downtime and Risk.

To restart all of the apps in your deployment:

> **Note:** You do not need to restart the apps in the `system` org.

1. Work with developers to restart a few of their apps individually and test that they still work correctly with the new ASGs in place. If an app does not work as expected, you likely must create another ASG that allows the app to send traffic to a service it requires.

   > **Note:** To quickly roll back to the original overly-permissive state, you can re-bind the `default_security_group` ASG to the `default-staging` and `default-running` sets. You must then restart your apps to re-apply the original ASGs.

2. Restart the rest of the apps running on your deployment. Optionally, you can use the app-restarter cf CLI plugin to restart all apps in a particular space, org, or deployment.

Certificates and Credentials

In this section:

- Rotating Runtime CredHub Encryption Keys
- Rotating the Cloud Controller Database Encryption Key
- Securing Service Instance Credentials with Runtime CredHub
- Providing a Certificate for Your TLS Termination Point
- Trusted System Certificates

Rotating Runtime CredHub Encryption Keys

This topic discusses how to rotate CredHub encryption keys for VMware Tanzu Application Service for VMs (TAS for VMs). Encryption keys are values that CredHub uses to obscure stored secrets. When an operator marks an additional key as primary, CredHub can rotate in that additional key as the encryption key.

During this credential rotation process, the initial encryption key is used to access the hidden value, That value is then stored again by the additional encryption key.

> ⚠ **Caution:** If you remove an encryption key and click **Apply Changes** before the rotation completes, the deployment breaks. If this happens, you can no longer access data stored with the deleted key.

Rotate TAS for VMs Encryption Keys

To rotate TAS for VMs encryption keys:

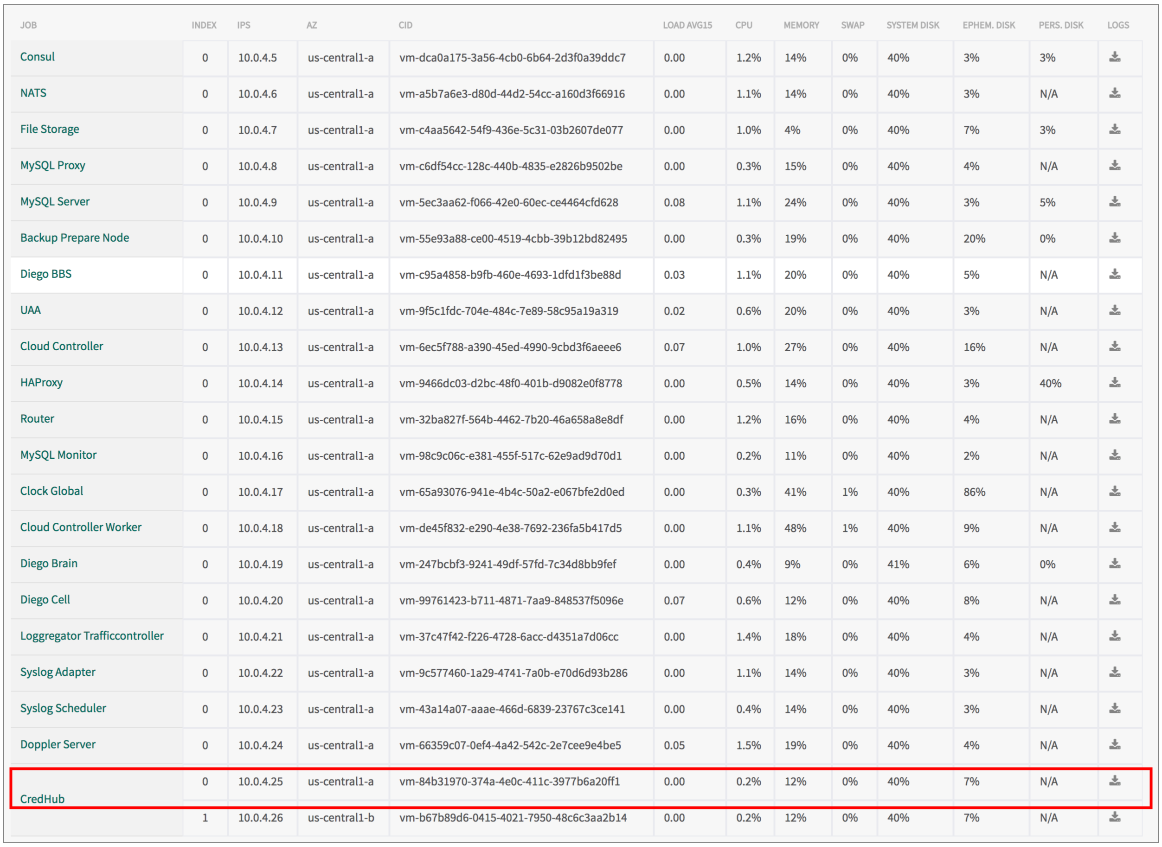
1. Go to the **Ops Manager Installation Dashboard**.
2. Click the TAS for VMs tile.
3. Select **CredHub**.

4. In the **Encryption Keys** section, click **Add**.
5. For **Name**, enter the name of your new encryption key.
6. For **Key**, enter your new encryption key.
7. Select the **Primary** check box.
8. Click **Save**.
9. Go to **Ops Manager Installation Dashboard**.
10. Click **Review Pending Changes**, then **Apply Changes**.

Verify Encryption Key Rotation

To verify that the rotation completes:

1. Click the TAS for VMs tile.
2. Select the **Status** tab.
3. Within the **CredHub** job, locate **Index 0**.



JOB	INDEX	IPS	AZ	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
Consul	0	10.0.4.5	us-central1-a	vm-dca0a175-3a56-4cb0-6b64-2d3f0a39ddc7	0.00	1.2%	14%	0%	40%	3%	3%	⤓
NATS	0	10.0.4.6	us-central1-a	vm-a5b7a6e3-d80d-44d2-54cc-a160d3f66916	0.00	1.1%	14%	0%	40%	3%	N/A	⤓
File Storage	0	10.0.4.7	us-central1-a	vm-c4aa5642-54f9-436e-5c31-03b2607de077	0.00	1.0%	4%	0%	40%	7%	3%	⤓
MySQL Proxy	0	10.0.4.8	us-central1-a	vm-c6df54cc-128c-440b-4835-e2826b9502be	0.00	0.3%	15%	0%	40%	4%	N/A	⤓
MySQL Server	0	10.0.4.9	us-central1-a	vm-5ec3aa62-f066-42e0-60ec-ce4464cfd628	0.08	1.1%	24%	0%	40%	3%	5%	⤓
Backup Prepare Node	0	10.0.4.10	us-central1-a	vm-55e93a88-ce00-4519-4cbb-39b12bd82495	0.00	0.3%	19%	0%	40%	20%	0%	⤓
Diego BBS	0	10.0.4.11	us-central1-a	vm-c95a4858-b9fb-460e-4693-1dfd1f3be88d	0.03	1.1%	20%	0%	40%	5%	N/A	⤓
UAA	0	10.0.4.12	us-central1-a	vm-9f5c1fdc-704e-484c-7e89-58c95a19a319	0.02	0.6%	20%	0%	40%	3%	N/A	⤓
Cloud Controller	0	10.0.4.13	us-central1-a	vm-6ec5f788-a390-45ed-4990-9cbd3f6aeee6	0.07	1.0%	27%	0%	40%	16%	N/A	⤓
HAProxy	0	10.0.4.14	us-central1-a	vm-9466dc03-d2bc-48f0-401b-d9082e0f8778	0.00	0.5%	14%	0%	40%	3%	40%	⤓
Router	0	10.0.4.15	us-central1-a	vm-32ba827f-564b-4462-7b20-46a658a8e8df	0.00	1.2%	16%	0%	40%	4%	N/A	⤓
MySQL Monitor	0	10.0.4.16	us-central1-a	vm-98c9c06c-e381-455f-517c-62e9ad9d70d1	0.00	0.2%	11%	0%	40%	2%	N/A	⤓
Clock Global	0	10.0.4.17	us-central1-a	vm-65a93076-941e-4b4c-50a2-e067bfe2d0ed	0.00	0.3%	41%	1%	40%	86%	N/A	⤓
Cloud Controller Worker	0	10.0.4.18	us-central1-a	vm-de45f832-e290-4e38-7692-236fa5b417d5	0.00	1.1%	48%	1%	40%	9%	N/A	⤓
Diego Brain	0	10.0.4.19	us-central1-a	vm-247bcbf3-9241-49df-57fd-7c34d8bb9fef	0.00	0.4%	9%	0%	41%	6%	0%	⤓
Diego Cell	0	10.0.4.20	us-central1-a	vm-99761423-b711-4871-7aa9-848537f5096e	0.07	0.6%	12%	0%	40%	8%	N/A	⤓
Loggregator Trafficcontroller	0	10.0.4.21	us-central1-a	vm-37c47f42-f226-4728-6acc-d4351a7d06cc	0.00	1.4%	18%	0%	40%	4%	N/A	⤓
Syslog Adapter	0	10.0.4.22	us-central1-a	vm-9c577460-1a29-4741-7a0b-e70d6d93b286	0.00	1.1%	14%	0%	40%	3%	N/A	⤓
Syslog Scheduler	0	10.0.4.23	us-central1-a	vm-43a14a07-aaae-466d-6839-23767c3ce141	0.00	0.4%	14%	0%	40%	3%	N/A	⤓
Doppler Server	0	10.0.4.24	us-central1-a	vm-66359c07-0ef4-4a42-542c-2e7cee9e4be5	0.05	1.5%	19%	0%	40%	4%	N/A	⤓
CredHub	0	10.0.4.25	us-central1-a	vm-84b31970-374a-4e0c-411c-3977b6a20ff1	0.00	0.2%	12%	0%	40%	7%	N/A	⤓
	1	10.0.4.26	us-central1-b	vm-b67b89d6-0415-4021-7950-48c6c3aa2b14	0.00	0.2%	12%	0%	40%	7%	N/A	⤓

4. Within the **Logs** column, click the correlating download icon.
5. Select the **Logs** tab.
6. Click the corresponding link to the retrieve the downloaded log file.
7. Unzip the log file.
8. Unzip the larger of the two nested directories.

9. Ops Manager generates a compressed file for each CredHub VM that exists on your deployment. Unzip each of these compressed files.
10. Open the `credhub` directory.
11. Open the `credhub.log` file. If the credential rotation completed successfully, the CredHub log contains the following string: `Successfully rotated NUMBER-OF-CREDENTIALS items`
12. Remove the old encryption key.
13. Click the trashcan icon that corresponds to the old encryption key.
14. Click **Save**.
15. Go to the **Ops Manager Installation Dashboard**.
16. Click **Review Pending Changes**, then **Apply Changes**.

Rotating the Cloud Controller Database Encryption Key

This topic tells you how to rotate the Cloud Controller database (CCDB) encryption key.

Overview

The Cloud Controller is the primary API of VMware Tanzu Application Service for VMs (TAS for VMs). It has its own database, the CCDB, in which it stores information about objects in TAS for VMs such as apps. The CCDB encryption key is used to encrypt sensitive data at rest in the CCDB such as app environment variables.

The CCDB encryption key is automatically generated when you first deploy TAS for VMs. You can rotate the CCDB encryption key as described in this topic. For example, you might want to perform this procedure if your existing key is leaked.

Rotate the CCDB Encryption Key

> ⚠ **Caution:** Do not modify the **Cloud Controller database encryption key** field in the TAS for VMs tile. This field is only for use when restoring TAS for VMs from a backup. If you modify this value at a time other than restore, commands such as `cf push` might fail. Instead, change keys using the **Encryption key ledger** as described below.

To rotate the CCDB encryption key:

1. Go to the TAS for VMs tile in Ops Manager and select the **Cloud Controller** pane.
2. Review the **Encryption key ledger** field. If you previously added a key and enabled the **Primary** check box, deselect the check box. The TAS for VMs deploy fails if there is more than one key with the **Primary** check box enabled.
3. In the **Encryption key ledger** field:

   > **Note:** Do not remove any existing keys until after you complete the last step of this procedure. Removing keys now causes the TAS for VMs deploy to

> fail.

   1. Click **Add**.
   2. Enter values for **Encryption key label** and **Encryption key**.
   3. Enable the check box to mark this as your **Primary** key.

4. Click **Save**.
5. Select the **Errands** pane.
6. Set the **Rotate CC Database Key** errand to **On**.
7. Click **Save**.
8. Redeploy TAS for VMs. After TAS for VMs deploys and the errand runs successfully, you can remove any previously specified CCDB encryption keys in the ledger.

Troubleshoot Failed Rotation

This section describes how to troubleshoot errors that you may encounter while rotating the CCDB encryption key.

Error: Encryption key(s) have had their values changed

Error	The TAS for VMs deploy fails with a Cloud Controller error. The Cloud Controller logs include the following message: `Encryption key(s) mykey have had their values changed. Label and value pairs should not change, rather a new label and value pair should be added.`
Cause	TAS for VMs does not support editing key names or values in the ledger. A previously existing key was edited in the TAS for VMs UI. For example, a key with the name `key-name` previously had the value `key-value-1` and now it is `key-value-2`.
Solution	Replace the value for the key with its original value and follow the rotation procedure to add a new key. You may need to find the original value of the key using the steps in View Encryption Keys.

Error: Encryption key(s) are still in use but not present in ‘cc.database_encryption.keys’

Error	The TAS for VMs deploy fails with a Cloud Controller error. The Cloud Controller logs include the following message: `Encryption key(s) mykey are still in use but not present in 'cc.database_encryption.keys'`
Cause	A previous key was removed from the ledger in the TAS for VMs UI before the key rotation errand ran.
Solution	Add the previously used key back to the ledger and redeploy TAS for VMs. You may need to find the value of the previously used key using the steps in View Encryption Keys.

View Encryption Keys

The section describes how to retrieve the value of an existing or previous encryption key. You may need to do this when troubleshooting a failed key rotation.

Ops Manager actual-installation.yml

The Ops Manager `actual-installation.yml` file shows properties from the most recent successful deploy. To view the encryption keys listed in this file:

1. Follow the steps in Modify the Installation Files in *Modifying Your Ops Manager Installation and Product Template Files* to decrypt the `actual-installation.yml` file.
2. Open the `actual-installation.yml` file and view the `cloud_controller_encryption_keys` property.

CredHub

To view current and previous encryption keys using CredHub:

1. Follow the steps in How to access CredHub with the CredHub CLI in the Knowledge Base to log in to the CredHub CLI.
2. Run the following command to view the five most recent encryption keys:

   ```
   credhub get -n /opsmgr/TAS-GUID/cloud_controller/INDEX/encryption_keys --versions=5
   ```

   Where:

   - `TAS-GUID` is the GUID of your TAS for VMs BOSH deployment.
   - `INDEX` is the zero-based index of the key. This corresponds to the order of the **Encryption key ledger** in the Ops Manager UI.

Securing Services Instance Credentials with Runtime CredHub

This topic describes how operators can ensure service instance credentials are securely stored in runtime CredHub.

Overview

The VMware Tanzu Application Service for VMs (TAS for VMs) tile includes its own CredHub component, separate from the CredHub component included with the BOSH Director tile. For more information about this centralized credential management component, see the CredHub documentation.

Runtime CredHub exists to securely store service instance credentials. Previously, TAS for VMs could only use the Cloud Controller database for storing these credentials.

When developers want their app to use a service, such as those provided by Spring Cloud Services for VMware Tanzu, they must bind their app to an instance of that service. Service bindings include credentials that developers can use to access the service from their app. For more information, see Binding Credentials.

To ensure that service instance credentials are stored in runtime CredHub, you must configure TAS for VMs to enable this functionality. Not all services support the use of runtime CredHub.

TAS for VMs supports services that do and do not use runtime CredHub. Services that do not use runtime CredHub continue to pass their credentials to the Cloud Controller database.

Runtime CredHub supports credential rotation. For more information, see Rotating Runtime CredHub Encryption Keys.

Services that Use Secure Binding Credentials

The procedures in this document are only effective for services that support storing their instance credentials in runtime CredHub. To learn whether a service supports this feature, see the table below or the documentation for that service.

Service	Versions with Secure Binding Credentials	VMware Tanzu Network Listing
VMware Tanzu GemFire and Pivotal Cloud Cache	v1.7 and later	VMware Tanzu GemFire
CredHub Service Broker	All	CredHub Service Broker
VMware Tanzu SQL [MySQL]	v2.3 and later (available to user groups only)	VMware Tanzu SQL [MySQL]
RabbitMQ for VMware Tanzu [VMs]	v1.12 and later	RabbitMQ for VMware Tanzu [VMs]
Redis for VMware Tanzu	v1.13 and later	Redis for VMware Tanzu
Spring Cloud Services for VMware Tanzu	v1.5 and later	Spring Cloud Services for VMware Tanzu

Prerequisites

> **Breaking Change:** If you opt out of the BOSH DNS feature, your Ops Manager deployment cannot support Secure Service Instance Credentials. For more information, see BOSH DNS Service Discovery for Application Containers (Beta) in the Pivotal Application Service v2.0 release notes.

Runtime CredHub allows you to use one or more Hardware Security Modules (HSMs) to store encryption keys. If you wish to use an HSM with CredHub, you must configure the HSM before completing the following procedures. For more information, see Preparing CredHub HSMs for Configuration.

Step 1: Configure the TAS for VMs Tile

To configure the TAS for VMs tile to support securing service instance credentials in CredHub, you must:

- In the **CredHub** pane, provide at least one encryption key. CredHub supports multiple encryption key providers.
- In the **Resource Config** pane, set the number of **CredHub** instances to at least one.

CredHub configuration options include:

- Internal or external databases
- Encryption keys stored internally, externally in a Hardware Security Module (HSM), or both

To configure the TAS for VMs tile, see Configuring TAS for VMs.

Step 2: Create App Security Groups

App Security Groups (ASGs) are network policy rules specifying protocols, ports, and IP ranges that apply to outbound network connections initiated from apps.

> **Note:** The default ASGs in Ops Manager allow apps running on your deployment to send traffic to almost any IP address. This step is only required if your ASGs restrict network access from apps to the network on which TAS for VMs runs. For more information, see Restricting App Access to Internal TAS for VMs Components.

To ensure the ASGs for your deployment allow apps to communicate with the runtime CredHub API:

1. From the TAS for VMs tile, click **Assign AZs and Networks** and record the selected network where the tile is installed.
2. From the BOSH Director tile, in the **Settings** tab, click **Create Networks**.
3. In the **Networks** section, click the name of the TAS for VMs network to expand it.
4. Record the CIDR for the TAS for VMs network.
5. Create a file named `runtime-credhub.json` for specifying your ASG rules. Copy the content below into the file. Replace `YOUR-TAS-CIDR` with the CIDR you recorded in the previous step.

```
[
  {
  "protocol": "tcp",
  "destination": "YOUR-TAS-CIDR",
  "ports": "8844"
  }
]
```

6. Create an ASG that allows apps to access the CredHub API by running:

```
cf create-security-group runtime-credhub ~/workspace/runtime-credhub runtime-cr
edhub.json
```

7. Bind this ASG to your deployment or the specific space in which you want apps to access CredHub. For more information about binding ASGs, see the Bind ASGs section of the *App Security Groups* topic. Ensure that apps deployed as part of the service tile installation process have access to CredHub in addition to the apps pushed to the platform by developers. For example, the Spring Cloud Services for VMware Tanzu tile deploys the `spring-cloud-broker` app to the `p-spring-cloud-services` space of the `system` org.

8. Restart apps for the ASGs to take effect. Optionally, you can use the app-restarter cf CLI plugin to restart all apps in a particular space, org, or deployment.

Step 3: Unbind and Rebind Service Instances

For any service instance bindings that existed before runtime CredHub was supported for that service, you must work with your developers to unbind and rebind the service instances to their apps. If you do not unbind and rebind the service, apps continue functioning as normal and fetching credentials from the Cloud Controller database.

> **Note:** This step is not required for bindings created after you installed the new version of the service tile that supports CredHub and you completed the procedures in steps 1 and 2 of this topic.

To unbind and rebind your service instances to their apps:

1. Unbind the service instance from the app by running:

   ```
   cf unbind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

   Where:

   - `YOUR-APP` is the app to which you bound the service instance.
   - `YOUR-SERVICE-INSTANCE` is the service instance you want to unbind from the app.

2. Rebind the service instance to the app by running:

   ```
   cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

   Where:

   - `YOUR-APP` is the app from which you unbound the service instance.
   - `YOUR-SERVICE-INSTANCE` is the service instance you want to rebind to the app.

3. Review the `VCAP_SERVICES` environment variable to verify that the new service instance binding includes CredHub pointers by running:

   ```
   cf env YOUR-APP
   ```

   Where `YOUR-APP` is the app to which you bound the service instance.

   For help parsing the output of the `cf-env` command, see VCAP_SERVICES in *TAS for VMs Environment Variables*.

4. Restart the app to apply the service instance binding by running:

   ```
   cf restart YOUR-APP
   ```

   Where `YOUR-APP` is the app to which you bound the service instance.

   If you run `cf-env` again, you can see the `VCAP-SERVICES` environment variable now contains the credentials for the service instance binding.

Providing a Certificate for Your TLS Termination Point

This topic describes how to configure Transport Layer Security (TLS) termination for HTTP traffic in VMware Tanzu Application Service for VMs (TAS for VMs) with a TLS certificate, as part of the process of configuring TAS for VMs for deployment.

Configure TLS Termination

When you deploy Ops Manager, you must configure the TLS termination for HTTP traffic in your TAS for VMs configuration. You can terminate TLS at all of these points:

- Load balancer
- Load balancer and the Gorouter
- The Gorouter

To choose and configure the TLS termination option for your deployment, see TLS Termination Options for HTTP Routing in *Securing Traffic into TAS for VMs*.

> **Note**: If you are using HAProxy in a TAS for VMs deployment, you can choose to terminate SSL/TLS at HAProxy in addition to any of the SSL/TLS termination options above. For more information, see Configuring SSL/TLS Termination at HAProxy.

Obtain TLS Certificates

To secure traffic into Ops Manager, you must obtain at least one TLS certificate. For general certificate requirements for deploying Ops Manager, see Certificate Requirements in *Securing Traffic into TAS for VMs*.

For additional IaaS-specific certificate requirements:

- **AWS:** Certificate Requirements on AWS
- **Azure:** Certificate Requirements on Azure
- **GCP:** Certificate Requirements on GCP
- **OpenStack:** Certificate Requirements on OpenStack
- **vSphere:** Certificate Requirements on vSphere

Create a Wildcard Certificate for Ops Manager Deployments

This section describes how to create or generate a certificate for your TAS for VMs environment. If you are deploying to a production environment, you must obtain a certificate from a trusted Certificate Authority (CA).

For internal development or testing environments, you have two options for creating a required TLS certificates:

- You can create a self-signed certificate, or
- You can have TAS for VMs generate the certificate for you.

To create a certificate, you can use a wide variety of tools including OpenSSL, Java's keytool, Adobe Reader, and Apple's Keychain to generate a Certificate Signing Request (CSR).

In either case for either self-signed or trusted single certificates, apply these rules when creating the CSR:

- Specify your registered wildcard domain as the `Common Name`, where `DOMAIN` is your registered wildcard domain. For example, `*.DOMAIN.com`.
- VMware recommends using a split domain configuration that separates the domains for `apps` and `sys` components. To use a split domain configuration, enter these values in the `Subject Alternative Name` of the certificate, where `DOMAIN` is your registered wildcard domain:
    - `*.apps.DOMAIN.com`
    - `*.sys.DOMAIN.com`
    - `*.login.sys.DOMAIN.com`
    - `*.uaa.sys.DOMAIN.com`
- If you are using a single domain configuration, use these values as the `Subject Alternative Name` of the certificate, where `DOMAIN` is your registered wildcard domain:
    - `*.login.sys.DOMAIN.com`
    - `*.uaa.sys.DOMAIN.com`

> **Note:** TLS certificates generated for wildcard DNS records only work for a single domain name component or component fragment. For example, a certificate generated for `*.DOMAIN.com` does not work for `*.apps.DOMAIN.com` and `*.sys.DOMAIN.com`. The certificate must have both `*.apps.DOMAIN.com` and `*.sys.DOMAIN.com` attributed to it.

Generate an RSA Certificate in TAS for VMs

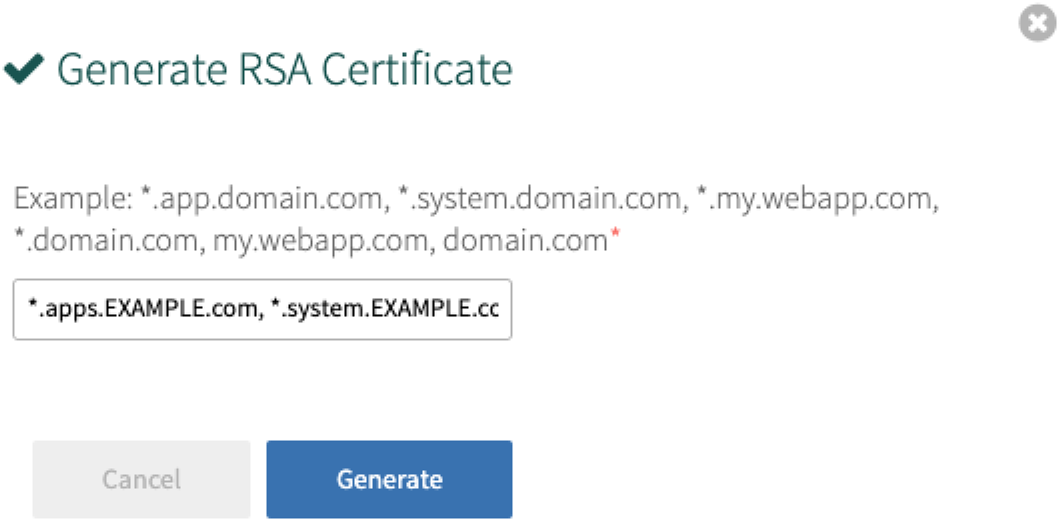
To generate an RSA certificate in TAS for VMs:

1. Go to the VMware Tanzu Operations Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Networking**.
4. Under **Certificate and private keys for the Gorouter and HAProxy**:
    1. Under **Certificate and private key**, click **Change**.
    2. Click **Generate RSA Certificate** to populate the **Certificate and private key** fields with RSA certificate and private key information.
5. VMware recommends using a split domain configuration that separates the domains for `apps` and `sys` components. To use a split domain configuration, enter these following

domains for the certificate, where `DOMAIN` is your registered wildcard domain:

- `*.DOMAIN.com`
- `*.apps.DOMAIN.com`
- `*.sys.DOMAIN.com`
- `*.login.sys.DOMAIN.com`
- `*.uaa.sys.DOMAIN.com`

6. Click **Generate**.



✔ Generate RSA Certificate

Example: *.app.domain.com, *.system.domain.com, *.my.webapp.com, *.domain.com, my.webapp.com, domain.com*

*.apps.EXAMPLE.com, *.system.EXAMPLE.cc

Trusted system certificates

As a Ops Manager admin, you can deploy a set of trusted system certificates. These trusted certificates are available in Linux-based app instances running on the Diego back end. Such instances include buildpack-based apps using the `cflinuxfs[3|4]` stack and Docker image-based apps.

If the admin configures these certificates, they are available inside the instance containers as files with extension `.crt` in the read-only `/etc/cf-system-certificates` directory.

For `cflinuxfs3`-based apps, these certificates are also installed directly in the `/etc/ssl/certs` directory, and are available automatically to libraries such as `openssl` that respect that trust store. If the administrator configure these certificates, the location of the certificates is provided in the environment variable `CF_SYSTEM_CERT_PATH` on the instance container.

Bulletin Board System (BBS) data store encryption

You must configure Diego Release with a set of encryption keys to encrypt data in the BBS data store. This page tells you how to configure and rotate encryption keys.

The BBS Data Store encrypts all stored data. Diego automatically encrypts or re-encrypts all of the stored data using the active key on boot. This ensures an operator can rotate out a key without manually rewriting all of the records.

Configuring encryption keys

Diego uses multiple keys for decryption while allowing only one for encryption.

To configure encryption, set the `diego.bbs.encryption_keys` and `diego.bbs.active_key_label` properties.

Replace the placeholders in the manifest below with values appropriate for your deployment.

```
properties:
  diego:
    bbs:
      active_key_label: KEY-LABEL-NAME
      encryption_keys:
      - label: 'KEY-LABEL-NAME'
        passphrase: 'MY-PASSPHRASE'
```

In the example below, the operator configures two encryption keys and selects one of them to be the active key. The active key is used for encryption while all configured keys are used for decryption.

```
properties:
  diego:
    bbs:
      active_key_label: key-2017-10
      encryption_keys:
      - label: 'key-2017-10'
        passphrase: 'my september passphrase'
      - label: 'key-2017-09'
        passphrase: 'my august passphrase'
```

Key label restrictions

Key labels have the following restrictions:

- 127 character limit
- Must not include a `:` (colon) character

Passphrases have no enforced character limit.

Rotating encryption keys

You can rotate encryption keys without downtime by following a two-deploy procedure. All the records are re-encrypted with the new active key, using the old key for decryption only. After the decryption is successful, you can remove the old key.

The following example rotates `key-2017-09` to `key-2017-10`.

Given the following starting manifest, use this procedure to rotate your encryption keys:

```
properties:
  diego:
    bbs:
      active\_key\_label: key-2017-09
      encryption_keys:
```

```
      - label: 'key-2017-09'
        passphrase: 'my september passphrase'
```

1. Add the new encryption key `key-2017-10` and set it as the active key.

   ```
   properties:
     diego:
       bbs:
         active\_key\_label: key-2017-10
         encryption_keys:
         - label: 'key-2017-09'
           passphrase: 'my september passphrase'
         - label: 'key-2017-10'
           passphrase: 'my october passphrase'
   ```

2. Redeploy Diego release.
3. If the first deploy is successful, update the manifest to remove the old key `key-2017-09`.

   ```
   properties:
     diego:
       bbs:
         active\_key\_label: key-2017-10
         encryption_keys:
         - label: 'key-2017-10'
           passphrase: 'my october passphrase'
   ```

4. Redeploy Diego release.

After the second deployment is complete, the encryption keys are rotated.

You must complete the second deployment to remove the old key. If not removed, you can continue to decrypt information from the BBS data store using the old key.

Component Communications

In this section:

- BOSH DNS Network Communications
- Cloud Controller Network Communications
- Container-to-Container Networking Communications
- CredHub Network Communications
- Diego Network Communications
- Loggregator Network Communications
- MySQL Network Communications
- NATS Network Communications
- Routing Network Communications
- UAA Network Communications

BOSH DNS network communications

In this topic, you can learn about BOSH DNS internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about BOSH DNS, see Native DNS Support in the BOSH documentation.

The table below lists network communication paths for BOSH DNS.

Note: Port 8853 is the destination port for communications between BOSH DNS health processes. You must allow TCP traffic on 8853 for all VMs running BOSH DNS.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any VM running BOSH DNS	backup_restore	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	clock_global	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	cloud_controller	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	cloud_controller_worker	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	credhub	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	diego_brain	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	diego_cell	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	diego_database	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	doppler	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	ha_proxy	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	loggregator_traffic controller	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	mysql	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	mysql_monitor*	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	mysql_proxy*	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	nats	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any VM running BOSH DNS	nfs_server[†]	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	router	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	tcp_router	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	uaa	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	Service instance VMs	53	TCP and UDP	DNS	Unencrypted. This communication happens inside the VM.
Any VM running BOSH DNS	backup_restore	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	clock_global	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	cloud_controller	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	cloud_controller_worker	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	credhub	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	diego_brain	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	diego_cell	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	diego_database	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	doppler	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	ha_proxy	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	log_cache	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	loggregator_trafficcontroller	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	mysql	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	mysql_monitor[*]	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	mysql_proxy[*]	8853	TCP	HTTPS	Mutual TLS

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any VM running BOSH DNS	nats	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	nfs_server[†]	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	router	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	tcp_router	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	uaa	8853	TCP	HTTPS	Mutual TLS
Any VM running BOSH DNS	Service instance VMs	8853	TCP	HTTPS	Mutual TLS

[*] Applies only to deployments where internal MySQL is selected as the database.

[†] Applies only to deployments where the internal NFS server is selected for file storage.

Cloud Controller Network communications

The tables here show Cloud Controller internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about Cloud Controller, see Cloud Controller.

Inbound communications

The table below lists network communication paths that are inbound to the Cloud Controller:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
clock_global (Syslog Binding Cache)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
cloud_controller	cloud_controller (Routing API)	443	TCP	HTTPS	OAuth 2.0
diego_brain	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	cloud_controller	9024	TCP	HTTPS	OAuth 2.0
diego_cell (Rep)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_database (BBS)	cloud_controller	9023	TCP	HTTPS	Mutual TLS

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
doppler (Syslog Drain Binder)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
loggregator_trafficcontroller	cloud_controller	9023	TCP	HTTPS	Mutual TLS
router	cloud_controller	9024	TCP	HTTPS	OAuth 2.0

Outbound communications

The table below lists network communication paths that are outbound from the Cloud Controller:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	mysql_proxy[*]	3306	TCP	MySQL	MySQL authentication[**]
cloud_controller	nfs_server or other blobstore[†]	4443	TCP	HTTPS	TLS and basic authentication
cloud_controller	uaa	8443	TCP	HTTPS	OAuth 2.0 or none
cloud_controller	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
cloud_controller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
cloud_controller_worker	mysql_proxy[*]	3306	TCP	MySQL	MySQL authentication[**]
cloud_controller_worker	nfs_server or other blobstore[†]	4443	TCP	HTTPS	TLS and basic authentication
clock_global	mysql_proxy[*]	3306	TCP	MySQL	MySQL authentication[**]

[*] Applies only to deployments where internal MySQL is selected as the database.

[**] MySQL authentication uses the MySQL native password method.

[†] The destination depends on your file storage or blobstore configuration.

The authentication method depends on the type of request.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

Container-to-container networking communications

The tables here show the network communication paths to container-to-container networking in VMware Tanzu Application Service for VMs (TAS for VMs).

For more information about container-to-container networking, see Container-to-container networking.

Inbound communications

The table below lists network communication paths that are inbound to container-to-container networking.

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_cell (Silk CNI)	diego_cell (Silk Daemon)	23954	TCP	HTTP	None
diego_cell (Silk Daemon)	diego_api (Silk Controller)	4103	TCP	HTTP	Mutual TLS
diego_cell (VXLAN Policy Agent)	diego_database (api - Policy Server Internal)	4003	TCP	HTTP	Mutual TLS
diego_cell (BOSH DNS Adapter)	diego_brain (Service Discovery Controller)	8054	TCP	HTTP	Mutual TLS

Outbound communications

The table below lists network communication paths that are outbound from container-to-container networking:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_database (API - Policy Server)	uaa	8443	TCP	HTTPS	TLS
diego_database (API - Policy Server)	cloud_controller (api - Cloud Controller)	9022	TCP	HTTP	OAuth 2.0
diego_database (API - Policy Server)	mysql_proxy*	3306	TCP	MySQL	MySQL authentication
diego_brain (Service Discovery Controller)	nats (NATS)	4222	TCP	HTTP	Basic authentication
diego_cell (BOSH DNS)	diego_cell (BOSH DNS Adapter)	8053	TCP	HTTP	None
diego_cell (VXLAN Policy Agent)	mysql_proxy*	3306	TCP	MySQL	MySQL authentication

*Applies only to deployments where internal MySQL is selected as the database.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

CredHub Network Communications

This topic describes CredHub internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about CredHub, see CredHub.

Inbound Communications

The following table lists network communication paths that are inbound to CredHub:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller (API)	credhub	8844	TCP	HTTPS	OAuth 2.0
diego_cell	credhub	8844	TCP	HTTPS	Mutual TLS†
windows_cell	credhub	8844	TCP	HTTPS	Mutual TLS†
windows2016_cell	credhub	8844	TCP	HTTPS	Mutual TLS†

†Diego Cells use the certificate pairs generated for individual containers to authenticate with CredHub on behalf of apps.

Outbound Communications

The following table lists network communication paths that are outbound from CredHub:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
credhub	uaa	8443	TCP	HTTPS	N/A
credhub	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**

*Applies only to deployments where internal MySQL is selected as the database.

**MySQL authentication uses the MySQL native password method.

†Diego Cells use the certificate pairs generated for individual containers to authenticate with CredHub on behalf of apps.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

Diego network communications

The tables here show Diego internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about Diego components and architecture, see How Diego pushes an app in *Diego Components and Architecture*.

Inbound communications

The table below lists network communication paths that are inbound to Diego:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS

Diego internal communications

The table below lists network communication paths that are internal for Diego:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_brain (Auctioneer)	diego_cell (Rep)	1801	TCP	HTTPS	Mutual TLS
diego_brain (Auctioneer)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_brain (Auctioneer)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	diego_cell (App instances)	Varies‡	TCP	SSH	SSH
diego_brain (TPS Watcher)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_cell (local Route Emitter)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_cell (Rep)	diego_brain (CC Uploader)	9091	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	diego_brain (File Server)[⁂]	8447	TCP	HTTPS	TLS
diego_cell (Rep)	diego_database (BBS)	8889	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_brain (Auctioneer)	9016	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_cell (Rep)	1801	TCP	HTTPS	Mutual TLS
diego_database (BBS)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS

[‡] These are the host-side ports that map to port 2222 in app instance containers and are typically within the range 61001 to 65534.

[⁂] The Diego File Server is responsible for distributing non-sensitive, static platform assets to internal platform components.

Outbound communications

The table below lists network communication paths that are outbound from Diego:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_brain	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_brain (SSH Proxy)	cloud_controller	9024	TCP	HTTPS	OAuth 2.0
diego_brain (SSH Proxy)	uaa	443	TCP	HTTPS	TLS and OAuth 2.0
diego_cell (local Route Emitter)	nats	4222, 4223, 4224, 4225	TCP	NATS	Basic authentication
diego_cell (Rep)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_cell (Rep)	nfs_server or other blobstore[*]	Varies	TCP	HTTP	Signed URLs/TLS
diego_database (BBS)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
diego_database (BBS)	mysql_proxy[†]	3306	TCP	MySQL	MySQL authentication[**]

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_database (Locket)	mysql_proxy[†]	3306	TCP	MySQL	MySQL authentication[**]

[*]The destination depends on your TAS for VMs blobstore configuration. If you use the internal blobstore, the Diego Cell communicates to the blobstore using TLS on port 4443.

[**]MySQL authentication uses the MySQL native password method.

[†]Applies only to deployments where internal MySQL is selected as the database.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

Loggregator network communications

The tables here show Loggregator internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about Loggregator components and architecture, see Loggregator components and architecture.

Loggregator communications

The table below lists network communication paths for Loggregator:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
Any*	loggregator_trafficcontroller	8081	TCP	HTTP/WebSocket	OAuth
Any VM running Loggregator Agent	doppler	8082	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller	doppler	8082	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller	uaa	8443	TCP	HTTPS	TLS
loggregator_trafficcontroller	cloud_controller	9023	TCP	HTTPS	Mutual TLS
loggregator_trafficcontroller (Reverse Log Proxy)	doppler	8082	TCP	gRPC over HTTP/2	Mutual TLS

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
loggregator_trafficcontroller (Route Registrar)	nats	4222	TCP	NATS	Basic authentication
loggregator_trafficcontroller (Metrics Forwarder)	BOSH Director (Metrics Server)	25555 and 8443	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller	doppler (Log Cache)	8080	TCP	gRPC over HTTP/2	Mutual TLS
loggregator_trafficcontroller (Reverse Log Proxy Gateway)	cloud_controller	9023	TCP	HTTPS	Mutual TLS
Any*	loggregator_trafficcontroller (Reverse Log Proxy Gateway)	8088	TCP	HTTP/Server Sent Events	OAuth

*Any source VM can send requests to the specified destination within its subnet.

**Any host configured through a user-provided service binding with a syslog URL.

***Any port configured through a user-provided service binding with syslog URL.

****Basic authentication only supported for HTTPS syslog drains.

Log Cache communications

The table below lists network communication paths for Log Cache:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
loggregator_trafficcontroller (Reverse Log Proxy)	log-cache (Nozzle)	8082	TCP	gRPC over HTTP/2	Mutual TLS
Any*	log-cache	8080	TCP	gRPC over HTTP/2	Mutual TLS
gorouter	log-cache (Auth Proxy)	8083	TCP	HTTP	OAuth
log-cache (Auth Proxy)	uaa	8443	TCP	HTTPS	TLS
log-cache (Auth Proxy)	cloud_controller	9024	TCP	HTTPS	TLS

*Any source VM can send requests to the specified destination within its subnet.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

MySQL network communications

The tables here show MySQL internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

> **Note:** These communications only apply to deployments where internal MySQL is selected as the TAS for VMs database.

Inbound communications

The table below lists network communication paths that are inbound to MySQL VMs:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
cloud_controller_worker	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
clock_global	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
credhub	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
diego_cell (VXLAN Policy Agent)	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
diego_database (Policy Server)	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
diego_database (BBS)	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
diego_database (Locket)	mysql_proxy	3306	TCP	MySQL	MySQL authentication*
uaa	mysql_proxy	3306	TCP	MySQL	MySQL authentication*

* MySQL authentication uses the MySQL native password method.

Internal communications

The table below lists network communication paths that are internal to MySQL VMs:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
mysql	mysql (Galera)	4567	TCP	MySQL	MySQL authentication*
mysql_monitor	mysql (MySQL Server)	3306	TCP	HTTP	Basic authentication
mysql_monitor	mysql_proxy (Proxy health check)	443/8080**	TCP	HTTP	Basic authentication
mysql_proxy	mysql (MySQL Server)	3306	TCP	HTTP	MySQL authentication*
mysql_proxy	mysql (Galera health check)	9200	TCP	HTTP	Basic authentication

*MySQL authentication uses the MySQL native password method.

**Port 443 is used if mysql_proxy is registered with the Gorouter. If not registered, mysql_proxy uses port 8080 instead.

Outbound communications

The table below lists network communication paths that are outbound from MySQL:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
mysql_monitor	uaa	8443	TCP	HTTPS	OAuth
mysql_proxy (Route Registrar)	nats	4222	TCP	NATS	Basic authentication

> **Note:** If you select the **Enable inactive mysql port** checkbox on the **Internal MySQL** pane of the TAS for VMs tile, you can run auditing and reporting queries on an inactive MySQL node over port 3336. For more information, see Configure Internal MySQL in *Configuring TAS for VMs.*

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

NATS network communications

The tables here show NATS internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about how Cloud Foundry integrates NATS, see TAS for VMs Routing Architecture.

Publish communications

The table below lists network communications that are published to NATS:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller (Route Registrar)	nats	4222-4225	TCP	NATS	Basic authentication
loggregator_trafficcontroller (Route Registrar)	nats	4222-4225	TCP	NATS	Basic authentication
mysql_proxy (Route Registrar)*	nats	4222-4225	TCP	NATS	Basic authentication
nfs_server (Route Registrar)†	nats	4222-4225	TCP	NATS	Basic authentication
uaa (Route Registrar)	nats	4222-4225	TCP	NATS	Basic authentication
diego_cell (local Route Emitter)	nats	4222-4225	TCP	NATS	Basic authentication

*Applies only to deployments where internal MySQL is selected as the database.

†Applies only to deployments where the internal NFS server is selected for file storage.

Subscribe communications

The table below lists network communications that are subscribed to NATS:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
router	nats	4222-4225	TCP	NATS	Basic authentication

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

Routing Network Communications

The tables here show the internal network communication paths of the routing subsystem with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

HTTP routing

The table below lists network communication paths for HTTP routing:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
diego_cell (local Route Emitter)	nats	4222	TCP	NATS	Basic authentication
Load balancer	router (Gorouter)	80	TCP	HTTP	None
Load balancer	router (Gorouter)	443	TCP	HTTPS	TLS
router (Gorouter)	nats	4222	TCP	NATS	Basic authentication
router (Gorouter)	System components	Varies	TCP	Varies	None
router (Gorouter)	App containers	Varies	TCP	Varies	Optional TLS
haproxy	router (Gorouter)	80	TCP	HTTP	None
haproxy	router (Gorouter)	443	TCP	HTTPS	TLS
Load balancer	haproxy	80	TCP	HTTP	None
Load balancer	haproxy	443	TCP	HTTPS	TLS

TCP routing (optional)

The table below lists network communication paths for TCP routing:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	cloud_controller (Routing API)[*]	443	TCP	HTTPS	TLS and OAuth 2.0
cloud_controller (Routing API)	diego_database (Locket)	8891	TCP	HTTPS	Mutual TLS
cloud_controller (Routing API)	mysql_proxy	3306	TCP	MySQL	MySQL authentication[**]
cloud_controller (Routing API)	uaa	8443	TCP	HTTPS	TLS
diego_brain (global TCP Emitter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
diego_brain (global TCP Emitter)	uaa	8443	TCP	HTTPS	TLS
diego_cell (local Route Emitter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
diego_cell (local Route Emitter)	uaa	8443	TCP	HTTPS	TLS
Load balancer	tcp_router	1024-65535[†]	TCP	TCP	None

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
router (Gorouter)	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
router (Gorouter)	uaa	8443	TCP	HTTPS	TLS
tcp_router	cloud_controller (Routing API)	3000	TCP	HTTP	OAuth 2.0
tcp_router	uaa	8443	TCP	HTTPS	TLS

[*] This communication happens through a load balancer and a Gorouter. Requests are received by Routing API on port 3000.

[†] You can use this port range to configure the port in the TAS for VMs tile.

[**] MySQL authentication uses the MySQL native password method.

Service Mesh (optional)

The table below lists network communication paths for service mesh:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller (cloud_controller_ng)	istio_control (Copilot)	9001	TCP	GRPC	Mutual TLS
istio_control (Copilot)	diego_database (BBS)	8889	TCP	HTTP	Mutual TLS
istio_control (Pilot-Discovery)	istio_control (Copilot)	9009	TCP	GRPC	Mutual TLS
istio_router (Envoy)	App containers	Varies	TCP	HTTP/HTTPS	Optional TLS
istio_router (Envoy)	istio_control (Pilot-Discovery)	15010	TCP	GRPC	None
Load balancer	istio_router (Envoy)	80	TCP	HTTP	None
Load balancer	istio_router (Envoy)	443	TCP	HTTPS	TLS
Load balancer (health check)	istio_router (Envoy)	8002	TCP	HTTP	None
route_syncer (CC Route Syncer)	istio_control (Copilot)	9001	TCP	GRPC	Mutual TLS
route_syncer (CC Route Syncer)	mysql_proxy*	3306	TCP	MySQL	MySQL authentication*
N/A (admin)	istio_router (Envoy)	8001	TCP	HTTP	None
N/A (for Envoy secure GRPC communication)	istio_control (Pilot-Discovery)	15012	TCP	GRPC	Mutual TLS

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
N/A (for HTTP discovery service)	istio_control (Pilot-Discovery)	8080	TCP	HTTP	None
N/A (for Pilot's self-monitoring)	istio_control (Pilot-Discovery)	9093	TCP	HTTP	None

[*]Applies only to deployments where internal MySQL is selected as the database.

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

UAA Network Communications

The tables here show the User Account and Authentication (UAA) internal network communication paths with other VMware Tanzu Application Service for VMs (TAS for VMs) components.

For more information about UAA, see User Account and Authentication (UAA) Server.

Inbound communications

The table below lists network communication paths that are inbound to UAA:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
cloud_controller	uaa	8443	TCP	HTTPS	OAuth 2.0 or none[*]
diego_brain (SSH Proxy)	uaa	443	TCP	HTTPS	OAuth 2.0
loggregator_trafficcontroller	uaa	8443	TCP	HTTPS	TLS
mysql_monitor	uaa	8443	TCP	HTTPS	OAuth
router	uaa	8443	TCP	HTTPS	OAuth 2.0

[*]The authentication method depends on the type of request.

Outbound communications: Internal to TAS for VMs

The table below lists network communication paths that are outbound from UAA:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
uaa	mysql_proxy*	3306	TCP	MySQL	MySQL authentication**
uaa (Route Registrar)	nats	4222	TCP	NATS	Basic authentication

*Applies only to deployments where internal MySQL is selected as the database.

** MySQL authentication uses the MySQL native password method.

Outbound communications: External to TAS for VMs

The table below lists network communication paths from UAA that are outbound to external systems:

Source VM	Destination VM	Port	Transport Layer Protocol	App Layer Protocol	Security and Authentication
uaa	LDAP	LDAP server communication port	TCP	LDAP/LDAPS	Basic authentication (LDAP bind)
uaa	SAML/OIDC	80 or 443 (HTTP port)	TCP	HTTP/HTTPS	Key

BOSH DNS communications

By default, TAS for VMs components and app containers look up services using the BOSH DNS service discovery mechanism. To support this lookup, BOSH Director co-locates a BOSH DNS server on every deployed VM. For more information, see BOSH DNS network communications.

Managing apps and their stacks

These articles are your source for learning about managing apps and their stacks in VMware Tanzu Application Service for VMs (TAS for VMs):

- Using the Stack Auditor Plug-In
- Changing Stacks
- Changing Windows Stacks

Using Stack Auditor in TAS for VMs

Stack Auditor is a cf CLI plug-in that allows you to list apps and their stacks, migrate apps to a new stack, and delete a stack. Learn how to use it on this page.

One use case for Stack Auditor is when you must migrate a large number of apps to a new stack. This includes moving from `cflinuxfs2` to `cflinuxfs3` in preparation to upgrade your deployment to a version that does not contain `cflinuxfs2`. The following table describes the workflow you can use:

Stage	Description
1	Operator audits stack usage to determine which apps need to be migrated. See List apps and their stacks.
2	Operator communicates with developers that they must migrate their existing apps to a new stack and begin pushing all new apps to the a new stack.
3	Developers migrate their apps to a new stack. See Change stacks.
4	Operator confirms apps have been migrated.
5	Operator deletes buildpacks associated with the old stack.
6	Operator deletes the old stack. See Delete a stack. **Note**: If you upgrade your deployment to a version that contains the stack you deleted, the stack returns on upgrade.
7	If applicable, operator upgrades the deployment to the version that does not contain the old stack.

Install Stack Auditor

To install Stack Auditor, do the following:

1. Download the Stack Auditor binary for your OS from Buildpack Extensions on VMware Tanzu Network.
2. Unzip the binary file you downloaded:

   ```
   tar xvzf PATH-TO-BINARY
   ```

3. Install the plugin with the cf CLI:

   ```
   cf install-plugin PATH-TO-BINARY
   ```

Use Stack Auditor

The sections below describe how to use Stack Auditor.

List apps and their stacks

This section describes how to see the apps in each org and space and what stack they are using.

1. To see which apps are using which stack, run the following command. It lists apps for each org you have access to. To see all the apps in your deployment, ensure that you are logged in to the cf CLI as a user who can access all orgs.

   ```
   cf audit-stack
   ```

   See the following example output:

```
$ cf audit-stack
first-org/development/first-app cflinuxfs2
first-org/staging/first-app cflinuxfs2
first-org/production/first-app cflinuxfs2
second-org/development/second-app cflinuxfs3
second-org/staging/second-app cflinuxfs3
second-org/production/second-app cflinuxfs3
...
```

Change stacks

This section describes how to change the stack that an app uses. Stack Auditor rebuilds the app onto the new stack without a change in the source code of the app. If you want to move the app to a new stack with updated source code, follow the procedure in the Changing Stacks topic.

⚠ **Warning**: After successfully staging the app on `cflinuxfs3`, Stack Auditor attempts to restart the app on `cflinuxfs3`. This causes brief downtime. To avoid this brief downtime, use a blue-green strategy. See Using Blue-Green Deployment to Reduce Downtime and Risk.

To change the stack an app uses, do the following:

1. Target the org and space of the app:

   ```
   cf target ORG SPACE
   ```

   Where: * `ORG` is the org the app is in * `SPACE` is the space the app is in

2. Run the following command:

   Note: If the app is in a `stopped` state, it remains stopped after changing stacks.

   Note: When attempting to change stacks, your app is stopped. If the app fails on `cflinuxfs3`, Stack Auditor attempts to restage your app on `cflinuxfs2`.

   ```
   cf change-stack APP-NAME STACK-NAME
   ```

   Where: * `APP-NAME` is the app that you want to move to a new stack * `STACK-NAME` is the stack you want to move the app to

   See the following example output:

   ```
   $ cf change-stack my-app cflinuxfs3
   Attempting to change stack to cflinuxfs3 for my-app...
   Starting app my-app in org pivotal-pubtools / space pivotalcf-staging a
   s ljarzynski@pivotal.io...
   ```

```
Downloading staticfile_buildpack...

. . .

requested state: started
instances: 1/1
usage: 64M x 1 instance
urls: example.com
last uploaded: Thu Mar 28 17:44:46 UTC 2019
stack: cflinuxfs3
buildpack: staticfile_buildpack

         state     since                    cpu    memory         disk
details
#0   running   2019-04-02 03:18:57 PM   0.0%   8.2M of 64M   6.9M of 1G


Application my-app was successfully changed to Stack cflinuxfs3
```

Delete a stack

This section describes how to delete a stack from your deployment. You must be an admin user to complete this step.

1. To delete a stack, run the following command. This action cannot be undone, with the following exception: If you upgrade your deployment to a version that contains the stack you deleted, the stack returns on upgrade.

   ```
   cf delete-stack STACK-NAME
   ```

   Where `STACK-NAME` is the name of the stack you want to delete.

   ```
   $ cf delete-stack cflinuxfs2
   Are you sure you want to remove the cflinuxfs2 stack? If so, type the n
   ame of the stack [cflinuxfs2]
   >cflinuxfs2
   Deleting stack cflinuxfs2...
   Stack cflinuxfs2 has been deleted.
   ```

   If you have any apps still running on `cflinuxfs2`, the command returns the following error:

   ```
   Failed to delete stack cflinuxfs2 with error: Please delete the app ass
   ociations for your stack.
   ```

Changing stacks

You can restage apps on a new stack. Here you will find a description of stacks and lists of stacks that are supported on .

To restage a Windows app on a new Windows stack, see Changing Windows Stacks.

You can also use the Stack Auditor plugin for the Cloud Foundry Command Line Interface (cf CLI) when changing stacks. See Using the Stack Auditor Plugin.

Overview

A stack is a prebuilt root file system (rootfs) that supports a specific operating system. For example, Linux-based systems need `/usr` and `/bin` directories at their root. The stack works in tandem with a buildpack to support apps running in compartments. Under Diego architecture, cell VMs can support multiple stacks.

Note: Docker apps do not use stacks.

Available stacks

TAS for VMs includes support for `cflinuxfs3`. The Linux `cflinuxfs3` stack is derived from Ubuntu Bionic 18.04. For more information about supported libraries, see the GitHub stacks page.

Restaging apps on a new stack

For security, stacks receive regular updates to address Common Vulnerabilities and Exposures (CVEs). Apps pick up on these stack changes through new releases of TAS for VMs. However, if your app links statically to a library provided in the rootfs, you may have to manually restage it to pick up the changes.

It can be difficult to know what libraries an app statically links to, and it depends on the languages you are using. One example is an app that uses a Ruby or Python binary, and links out to part of the C standard library. If the C library requires an update, you may need to recompile the app and restage it.

To restage an app on a new stack, do the following:

1. Use the `cf stacks` command to list the stacks available in a deployment.

   ```
   $ cf stacks
   Getting stacks in org MY-ORG / space development as developer@example.c
   om...
   OK

   name            description
   cflinuxfs2      Cloud Foundry Linux-based filesystem - Ubuntu Trusty 1
   4.04 LTS
   cflinuxfs3      Cloud Foundry Linux-based filesystem - Ubuntu Bionic 1
   8.04 LTS
   ```

2. To change your stack and restage your app, run the following command:

   ```
   cf push MY-APP -s STACK-NAME
   ```

   Where: - MY-APP is the name of the app. - STACK-NAME is the name of the new stack.

For example, to restage your app on the stack `cflinuxfs3`, run `cf push MY-APP -s cflinuxfs3`:

```
 $ cf push MY-APP -s cflinuxfs3 Using stack cflinuxfs3… OK Creating app
MY-APP in org MY-ORG / space development as developer@example.com… OK …
requested state: started instances: 1/1 usage: 1G x 1 instances urls: M
Y-APP.cfapps.io last uploaded: Wed Apr 8 23:40:57 UTC 2015 state    sin
ce                      cpu    memory        disk #0   running   2015-04-08
04:41:54 PM   0.0%   57.3M of 1G   128.8M of 1G
```

Stacks API

For API information, see the *Stacks* section of the Cloud Foundry API Documentation.

Restaging your apps on a Windows stack

This topic explains how you can restage apps on a new Windows stack. It also describes what stacks are and lists the supported Windows stacks on .

To restage a Windows app on a new Linux stack, see Changing Stacks.

You can also use the Stack Auditor plugin for the Cloud Foundry Command Line Interface (cf CLI) when changing stacks. See Using the Stack Auditor Plugin.

Overview

A stack is a prebuilt root file system (rootfs) that supports a specific operating system. For example, Linux-based systems need `/usr` and `/bin` directories at their root and Windows needs `/windows`. The stack works in tandem with a buildpack to support apps running in compartments. Under Diego architecture, cell VMs can support multiple stacks.

Note: Docker apps do not use stacks.

Available stacks

If you push your TAS for VMs [Windows] app to a Windows stack, you must use `windows`. The `windows2016` stack is not supported on TAS for VMs [Windows].

Restaging apps on a new stack

For security, stacks receive regular updates to address Common Vulnerabilities and Exposures (CVEs). Apps pick up on these stack changes through new releases of TAS for VMs [Windows]. However, if your app links statically to a library provided in the rootfs, you may have to manually restage it to pick up the changes.

It can be difficult to know what libraries an app statically links to, and it depends on the languages you are using. One example is an app that uses a Ruby or Python binary, and links out to part of the C standard library. If the C library requires an update, you may need to recompile the app and restage it.

To restage an app on a new stack, do the following:

1. Use the `cf stacks` command to list the stacks available in a deployment.

```
$ cf stacks
Getting stacks in org MY-ORG / space development as developer@example.c
om...
OK

name            description
windows2016     Windows Server 2016
windows         Windows Server
```

2. To change your stack and restage your app, run the following command:

```
cf push MY-APP -s STACK-NAME
```

Where: - MY-APP is the name of the app. - STACK-NAME is the name of the new stack.

For example, to restage your app on the `windows` stack, run `cf push MY-APP -s windows`:

```
$ cf push MY-APP -s windows
Using stack windows...
OK
Creating app MY-APP in org MY-ORG / space development as developer@exam
ple.com...
OK
...
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: MY-APP.cfapps.io
last uploaded: Wed Apr 8 23:40:57 UTC 2015
    state     since                    cpu    memory         disk
\#0   running   2015-04-08 04:41:54 PM   0.0%   57.3M of 1G    128.8M of 1
G
```

Stacks API

For API information, see the “Stacks” section of the Cloud Foundry API Documentation.

Developing Apps

In this section:

Designing and running your app in the cloud

These guidelines represent best practices for developing modern apps for cloud platforms. For more detailed reading about good app design for the cloud, see The Twelve-Factor App.

Apps written in supported frameworks often run unmodified on Ops Manager if the app design follows a few simple guidelines. Following these guidelines facilitates app deployment to Ops Manager and other cloud platforms.

For more information about the features of HTTP routing handled by the Gorouter in Ops Manager, see HTTP routing. For more information about the lifecycle of app containers, see App container lifecycle.

Avoid Writing to the Local File System

Apps running on Ops Manager should not write files to the local file system for the following reasons:

- **Local file system storage is short-lived.** When an app instance crashes or stops, the resources assigned to that instance are reclaimed by the platform, including any local disk changes made since the app started. When the instance is restarted, the app starts with a new disk image. Although your app can write local files while it is running, the files disappear after the app restarts.
- **Instances of the same app do not share a local file system.** Each app instance runs in its own isolated container. Therefore, a file written by one instance is not visible to other instances of the same app. If the files are temporary, this should not be a problem. However, if your app needs the data in the files to persist across app restarts, or the data needs to be shared across all running instances of the app, the local file system should not be used. VMware recommends using a shared data service like a database or blobstore for this purpose.

For example, instead of using the local file system, you can use a Ops Manager service such as the MongoDB document database or a relational database like MySQL or PostgreSQL. Another option is to use cloud storage providers such as Amazon S3, Google Cloud Storage, Dropbox, or Box. If your app needs to communicate across different instances of itself, consider a cache like Redis or a messaging-based architecture with RabbitMQ.

If you must use a file system for your app because, for example, your app interacts with other apps through a network attached file system or because your app is based on legacy code that you cannot rewrite, consider using volume services to bind a network attached file system to your app. For more information, see Using an External File System (Volume Services).

Cookies Accessible Across Apps

In an environment with shared domains, cookies might be accessible across apps.

Many tracking tools such as Google Analytics and Mixpanel use the highest available domain to set their cookies. For an app using a shared domain such as `example.com`, a cookie set to use the highest domain has a `Domain` attribute of `.example.com` in its HTTP response header. For example, an app at `my-app.shared-domain.example.com` might be able to access the cookies for an app at `your-app.shared-domain.example.com`.

You should decide whether or not you want your apps or tools that use cookies to set and store the cookies at the highest available domain.

Port Considerations

Clients connect to apps running on Ops Manager by making requests to URLs associated with the app. Ops Manager allows HTTP requests to apps on ports 80 and 443. For more information, see Routes and Domains.

Ops Manager also supports WebSocket handshake requests over HTTP containing the `Upgrade` header. The Ops Manager router handles the upgrade and initiates a TCP connection to the app to form a WebSocket connection.

To support WebSockets, the operator must configure the load balancer correctly. Depending on the configuration, clients may have to use a different port for WebSocket connections, such as port 4443, or a different domain name. For more information, see Supporting WebSockets.

Ops Manager Updates and Your App

For app management purposes, Ops Manager may need to stop and restart your app instances. If this occurs, Ops Manager performs the following steps:

1. Ops Manager sends a single `termination signal` to the root process that your start command invokes.
2. Ops Manager waits 10 seconds to allow your app to cleanly shut down any child processes and handle any open connections.
3. After 10 seconds, Ops Manager forcibly shuts down your app.

Your app should accept and handle the termination signal to ensure that it shuts down gracefully. To achieve this, the app is expected to follow the steps below when shutting down:

1. App receives termination signal
2. App closes listener so that it stops accepting new connections
3. App finishes serving in-flight requests
4. App closes existing connections as their requests complete
5. App is stopped or shut down

For an implementation of the expected shutdown behavior in Golang, see the Sample HTTP App repository on GitHub.

Ignore unnecessary files when pushing

By default, when you push an app, all files in the app's project directory tree are uploaded to your Ops Manager instance, except version control and configuration files or folders with the following names:

- `.cfignore`
- `_darcs`

- `.DS_Store`
- `.git`
- `.gitignore`
- `.hg`
- `manifest.yml`
- `.svn`

In addition to these, if API request diagnostics are directed to a log file and the file is within the project directory tree, it is excluded from the upload. You can direct these API request diagnostics to a log file using `cf config --trace` or the `CF_TRACE` environment variable.

If the app directory contains other files, such as `temp` or `log` files, or complete subdirectories that are not required to build and run your app, you might want to add them to a `.cfignore` file to exclude them from upload. Especially with a large app, uploading unnecessary files can slow app deployment.

To use a `.cfignore` file, create a text file named `.cfignore` in the root of your app directory structure. In this file, specify the files or file types you wish to exclude from upload. For example, these lines in a `.cfignore` file exclude the "tmp" and "log" directories.

```
tmp
log
```

The file types you might want to exclude vary, based on the app frameworks you use. For examples of commonly-used `.gitignore` files, see the gitignore repository on GitHub.

Run multiple instances to increase availability

Singleton apps may become temporarily unavailable for reasons that include:

- During an upgrade, Ops Manager gracefully shuts down the apps running on each Diego Cell and restarts them on another Diego Cell. Single app instances may become temporarily unavailable if the replacement instance does not become healthy within the Diego Cell's evacuation timeout, which defaults to 10 minutes.
- Unexpected faults in Ops Manager system components or underlying infrastructure, such as container-host VMs or IaaS availability zones, might cause lone app instances to disappear or become unroutable for a minute or two.

To avoid the risk of an app becoming temporarily unavailable, developers can run more than one instance of the app.

Using buildpacks

A buildpack consists of bundles of detection and configuration scripts that provide framework and runtime support for your apps. When you deploy an app that needs a buildpack, Ops Manager installs the buildpack on the Diego Cell where the app runs.

For more information, see Buildpacks.

TAS for VMs Environment Variables

Environment variables are the means by which TAS for VMs communicates with a deployed app about its environment.

For information about setting your own app-specific environment variables, see the Environment Variable section of the *Deploying with App Manifests* topic.

> **Note:** Do not use user-provided environment variables for security sensitive information such as credentials as they might unintentionally show up in cf CLI output and Cloud Controller logs. Use [user-provided service instances](../services/user-provided.html) instead. The system-provided environment variable [VCAP_SERVICES](#VCAP_SERVICES) is properly redacted for user roles such as Space Supporter and in Cloud Controller log files.

View environment variables

Using the Cloud Foundry Command Line Interface (cf CLI), you can run the `cf env` command to view the TAS for VMs environment variables for your app. The `cf env` command displays the following environment variables:

- The `VCAP_APPLICATION` and `VCAP_SERVICES` variables provided in the container environment
- The user-provided variables set using the `cf set-env` command

For more information about the `cf env` command, see env in the cf CLI documentation. For more information about the `cf set-env` command, see set-env in the cf CLI documentation.

The following example demonstrates the environment variables `cf env` displays:

```
$ cf env my-app
Getting env variables for app my-app in org my-org / space my-space as
admin...
OK
System-Provided:

{
 "VCAP_APPLICATION": {
  "application_id": "fa05c1a9-0fc1-4fbd-bae1-139850dec7a3",
  "application_name": "my-app",
  "application_uris": [
    "my-app.example.com"
  ],
  "application_version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca",
  "cf_api": "https://api.example.com",
  "limits": {
    "disk": 1024,
    "fds": 16384,
    "mem": 256
  },
  "name": "my-app",
  "organization_id": "c0134bad-97a9-468d-ab9d-e97547e3aed5",
  "organization_name": "my-org",
  "space_id": "06450c72-4669-4dc6-8096-45f9777db68a",
  "space_name": "my-space",
  "uris": [
    "my-app.example.com"
  ],
  "users": null,
  "version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca"
  }
}

User-Provided:
MY_DRAIN: http://drain.example.com
MY_ENV_VARIABLE: 100
```

App-specific system variables

This section describes the environment variables that TAS for VMs makes available to your app container. Some of these variables are the same across instances of a single app, and some vary from instance to instance.

You can access environment variables programmatically, including variables defined by the buildpack. For more information, see the buildpack documentation for Java, Node.js, and Ruby.

The table below lists the system variables available to your app container.

Environment Variable	Running	Staging	Task
`CF_INSTANCE_ADDR`	x	x	x
`CF_INSTANCE_GUID`	x		x
`CF_INSTANCE_INDEX`	x		
`CF_INSTANCE_INTERNAL_IP`	x	x	x
`CF_INSTANCE_IP`	x	x	x
`CF_INSTANCE_PORT`	x	x	x
`CF_INSTANCE_PORTS`	x	x	x
`CF_STACK`		x	
`DATABASE_URL`	x		x
`HOME`	x	x	x
`INSTANCE_GUID`	x		
`INSTANCE_INDEX`	x		
`LANG`	x	x	x
`MEMORY_LIMIT`	x	x	x
`PATH`	x	x	x

Environment Variable	Running	Staging	Task
PORT	x		
PWD	x	x	x
TMPDIR	x		x
USER	x	x	x
VCAP_APP_HOST	x		
VCAP_APP_PORT	x		
VCAP_APPLICATION	x	x	x
VCAP_SERVICES	x	x	x

CF_INSTANCE_ADDR

The `CF_INSTANCE_IP` and `CF_INSTANCE_PORT` of the app instance, in the format `IP:PORT`.

For example: `CF_INSTANCE_ADDR=1.2.3.4:5678`

For more information, see CF_INSTANCE_IP and CF_INSTANCE_PORT.

CF_INSTANCE_GUID

The UUID of the app instance.

For example: `CF_INSTANCE_GUID=41653aa4-3a3a-486a-4431-ef258b39f042`

CF_INSTANCE_INDEX

The index number of the app instance.

For example: `CF_INSTANCE_INDEX=0`

CF_INSTANCE_IP

The external IP address of the host running the app instance.

For example: `CF_INSTANCE_IP=1.2.3.4`

CF_INSTANCE_INTERNAL_IP

The internal IP address of the container running the app instance.

For example: `CF_INSTANCE_INTERNAL_IP=5.6.7.8`

CF_INSTANCE_PORT

The external, or *host-side*, port corresponding to the internal, or *container-side*, port with value `PORT`. This value is usually different from the `PORT` of the app instance.

For example: `CF_INSTANCE_PORT=61045`

For more information, see PORT.

CF_INSTANCE_PORTS

The list of mappings between internal, or *container-side*, and external, or *host-side*, ports allocated to the container running the app instance. Not all of the internal ports are necessarily available for the app to bind to, as some of them may be used by system-provided services that also run inside the container. These internal and external values may differ.

For example: `CF_INSTANCE_PORTS=[{external:61045,internal:8080},{external:61046,internal:2222}]`

DATABASE_URL

For apps bound to certain services that use a database, TAS for VMs creates a `DATABASE_URL` environment variable based on the `VCAP_SERVICES` environment variable.

TAS for VMs uses the structure of the `VCAP_SERVICES` environment variable to populate the `DATABASE_URL` environment variable. TAS for VMs recognizes any service containing a JSON object like the example below as a candidate for the `DATABASE_URL` environment variable and uses the first candidate it finds.

```
{
  "some-service": [
    {
      "credentials": {
        "uri": "SOME-DATABASE-URL"
      }
    }
  ]
}
```

For example, see the following `VCAP_SERVICES` environment variable example:

```
VCAP_SERVICES =
{
  "elephantsql": [
    {
      "name": "elephantsql-c6c60",
      "label": "elephantsql",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampled
b"
      }
    }
  ]
}
```

Based on this `VCAP_SERVICES` environment variable, TAS for VMs creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampledb
```

For more information, see VCAP_SERVICES.

HOME

The root folder for the deployed app.

For example: `HOME=/home/vcap/app`

LANG

Required by buildpacks to ensure consistent script load order.

For example: `LANG=en_US.UTF-8`

MEMORY_LIMIT

The maximum amount of memory that each instance of the app can consume. You specify this value in an app manifest or with the cf CLI when pushing an app. The value is limited by space and org quotas.

If an instance exceeds the maximum limit, it is restarted. If TAS for VMs is asked to restart an instance too frequently, the instance is terminated.

For example: `MEMORY_LIMIT=512M`

PORT

The port on which the app should listen for requests. TAS for VMs allocates a port dynamically for each instance of the app, so code that obtains or uses the app port should refer to it using the `PORT` environment variable.

For example: `PORT=8080`

PWD

The present working directory where the buildpack that processed the app ran.

For example: `PWD=/home/vcap/app`

TMPDIR

The directory location where temporary and staging files are stored.

For example: `TMPDIR=/home/vcap/tmp`

USER

The user account under which the app runs.

For example: `USER=vcap`

VCAP_APP_PORT

Deprecated name for the `PORT` variable.

VCAP_APPLICATION

This environment variable contains the associated attributes for a deployed app. Results are returned in JSON format. The table below lists the attributes that are returned.

Attribute	Description
`application_id`	The GUID identifying the app.
`application_name`	The name assigned to the app when it was pushed.
`application_uris`	The URIs assigned to the app.
`application_version`	The GUID identifying a version of the app. Each time an app is pushed or restarted, this value is updated.
`cf_api`	The location of the Cloud Controller API for the TAS for VMs deployment where the app runs.
`host`	Deprecated. The IP address of the app instance.
`limits`	The limits to disk space, number of files, and memory permitted to the app. Memory and disk space limits are supplied when the app is deployed, either on the command line or in the app manifest. The number of files allowed is operator-defined.
`name`	Identical to `application_name`.
`organization_id`	The GUID identifying the org where the app is deployed.
`organization_name`	The human-readable name of the org where the app is deployed.
`process_id`	The UID identifying the process. Only present in running app containers.
`process_type`	The type of process. Only present in running app containers.
`space_id`	The GUID identifying the space where the app is deployed.
`space_name`	The human-readable name of the space where the app is deployed.
`start`	The human-readable timestamp for the time the instance was started. Not provided on Diego Cells.
`started_at`	Identical to `start`. Not provided on Diego Cells.
`started_at_timestamp`	The Unix epoch timestamp for the time the instance was started. Not provided on Diego Cells.
`state_timestamp`	Identical to `started_at_timestamp`. Not provided on Diego Cells.
`uris`	Identical to `application_uris`. You must ensure that both `application_uris` and `uris` are set to the same value.
`users`	Deprecated. Not provided on Diego Cells.
`version`	Identical to `application_version`.

VCAP_SERVICES

For bindable services, TAS for VMs adds connection details to the `VCAP_SERVICES` environment variable when you restart your app, after binding a service instance to your app. For more information about bindable services, see Services Overview.

TAS for VMs returns the results as a JSON document that contains an object for each service for which one or more instances are bound to the app. The service object contains a child object for each instance of the service that is bound to the app.

The table below defines the attributes that describe a bound service. The key for each service in the JSON document is the same as the value of the “label” attribute.

Attribute	Description
`binding_guid`	The guid of the service binding.
`binding_name`	The name assigned to the service binding by the user.
`instance_guid`	The guid of the service instance.
`instance_name`	The name assigned to the service instance by the user.
`name`	The `binding_name`, if it exists. Otherwise, the `instance_name`.
`label`	The name of the service offering.
`tags`	An array of strings an app can use to identify a service instance.
`plan`	The service plan selected when the service instance was created.
`credentials`	A JSON object containing the service-specific credentials needed to access the service instance.
`syslog_drain_url`	The service-specific syslog drain url.
`volume_mounts`	An array of service-specific volume mounts.

To see the value of the `VCAP_SERVICES` environment variable for an app pushed to TAS for VMs, see View Environment Variable Values.

The example below shows the value of the `VCAP_SERVICES` environment variable for bound instances of several services available in the Marketplace.

```
VCAP_SERVICES=
{
  "elephantsql": [
    {
      "name": "elephantsql-binding-c6c60",
      "binding_guid": "44ceb72f-100b-4f50-87a2-7809c8b42b8d",
      "binding_name": "elephantsql-binding-c6c60",
      "instance_guid": "391308e8-8586-4c42-b464-c7831aa2ad22",
      "instance_name": "elephantsql-c6c60",
      "label": "elephantsql",
      "tags": [
        "postgres",
        "postgresql",
        "relational"
      ],
      "plan": "turtle",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampleu
ser"
      },
      "syslog_drain_url": null,
      "volume_mounts": []
    }
  ],
  "sendgrid": [
    {
      "name": "mysendgrid",
      "binding_guid": "6533b1b6-7916-488d-b286-ca33d3fa0081",
      "binding_name": null,
      "instance_guid": "8c907d0f-ec0f-44e4-87cf-e23c9ba3925d",
      "instance_name": "mysendgrid",
      "label": "sendgrid",
      "tags": [
        "smtp"
      ],
      "plan": "free",
      "credentials": {
        "hostname": "smtp.sendgrid.net",
        "username": "QvsXMbJ3rK",
        "password": "HCHMOYluTv"
      },
      "syslog_drain_url": null,
      "volume_mounts": []
    }
  ]
}
```

Environment variable groups

Environment variable groups are system-wide variables that enable operators to apply a group of environment variables to all running apps and all staging apps separately.

An environment variable group consists of a single hash of name-value pairs that are later inserted into an app container at runtime or at staging. These values can contain information such as HTTP proxy information. The values for variables set in an environment variable group are case-sensitive.

When creating environment variable groups:

- Only the TAS for VMs operator can set the hash value for each group.
- All authenticated users can get the environment variables assigned to their app.
- All variable changes take effect after the operator restarts or restages the apps.
- Any user-defined variable takes precedence over environment variables provided by these groups.

The table below lists the commands for environment variable groups.

CLI Command	Description
`running-environment-variable-group` or `revg`	Retrieves the contents of the running environment variable group.
`staging-environment-variable-group` or `sevg`	Retrieves the contents of the staging environment variable group.
`set-staging-environment-variable-group` or `ssevg`	Passes parameters as JSON to create a staging environment variable group.
`set-running-environment-variable-group` or `srevg`	Passes parameters as JSON to create a running environment variable group.

The following examples demonstrate how to retrieve the environment variables:

```
$ cf revg
Retrieving the contents of the running environment variable group as
sampledeveloper@example.com...
OK
Variable Name   Assigned Value
HTTP Proxy      198.51.100.130

$ cf sevg
Retrieving the contents of the staging environment variable group as
sampledeveloper@example.com...
OK
Variable Name   Assigned Value
HTTP Proxy      203.0.113.105
EXAMPLE-GROUP   2001

$ cf apps
Getting apps in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

name    requested state   instances   memory   disk   urls
my-app  started           1/1         256M     1G     my-app.com

$ cf env APP-NAME
Getting env variables for app APP-NAME in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

System-Provided:

{
  "VCAP_APPLICATION": {
  "application_name": "APP-NAME",
  "application_uris": [
    "my-app.example.com"
  ],
  "application_version": "7d0d64be-7f6f-406a-9d21-504643147d63",
  "limits": {
  "disk": 1024,
  "fds": 16384,
  "mem": 256
  },
  "name": "APP-NAME",
  "organization_id": "c0134bad-97a9-468d-ab9d-e97547e3aed5",
  "organization_name": "my-org",
  "space_id": "37189599-2407-9946-865e-8ebd0e2df89a",
  "space_name": "dev",
  "uris": [
    "my-app.example.com"
  ],
  "users": null,
  "version": "7d0d64be-7f6f-406a-9d21-504643147d63"
 }
}

Running Environment Variable Groups:
HTTP Proxy: 198.51.100.130

Staging Environment Variable Groups:
EXAMPLE-GROUP: 2001
HTTP Proxy: 203.0.113.105
```

The following examples demonstrate how to set environment variables:

```
$ cf ssevg '{"test":"198.51.100.130","test2":"203.0.113.105"}'
Setting the contents of the staging environment variable group as admin...
OK
$ cf sevg
Retrieving the contents of the staging environment variable group as admin...
OK
Variable Name   Assigned Value
test            198.51.100.130
test2           203.0.113.105

$ cf srevg '{"test3":"2001","test4":"2010"}'
Setting the contents of the running environment variable group as admin...
OK
$ cf revg
Retrieving the contents of the running environment variable group as admin...
OK
Variable Name   Assigned Value
test3           2001
test4           2010
```

Available Cloud Controller API client libraries

Here is a list of the client libraries you can use with the Cloud Foundry API (CAPI) for VMware Tanzu Application Service for VMs (TAS for VMs).

CAPI overview

CAPI is the entry point for most operations within the TAS for VMs platform. You can use it to manage orgs, spaces, and apps, which includes user roles and permissions. You can also use CAPI to manage the services provided by your TAS for VMs deployment, including provisioning, creating, and binding them to apps.

For more information, see the CAPI documentation.

Client libraries

While you can develop apps that consume CAPI by calling it directly as in the API documentation, you might want to use an existing client library. See the available client libraries below.

Supported

TAS for VMs currently supports the following clients for CAPI:

- Java
- Scripting with the Cloud Foundry Command Line Interface (cf CLI)

Experimental

The following client is experimental and a work in progress:

- Golang

Unofficial

TAS for VMs does not support the following clients, but they may be supported by third-parties:

- Golang:
  - cloudfoundry-community/go-cfclient
- Python:
  - cloudfoundry-community/cf-python-client
  - hsdp/python-cf-api
  - cloudfoundry-community/cf-python-client

Routes and Domains

In this section:

- Configuring Routes and Domains
- Routing HTTP/2 and gRPC Traffic to Apps
- Configuring TAS for VMs to Route Traffic to Apps on Custom Ports

Configuring routes and domains

Developers and administrators can configure routes and domains for their apps using the Cloud Foundry Command Line Interface (cf CLI). This topic describes how routes and domains work in VMware Tanzu Application Service for VMs (TAS for VMs). For more information about routing capabilities in TAS for VMs, see HTTP Routing.

Routes

The TAS for VMs Gorouter routes requests to apps by associating an app with an address, known as a route. This is known as a **mapping**. Use the cf CLI cf map-route command to associate an app and route.

The routing tier compares each request with a list of all the routes mapped to apps and attempts to find the best match. For example, the Gorouter would make the following matches for the two routes `myapp.shared-domain.example.com` and `myapp.shared-domain.example.com/products`:

Request	Matched Route
http://myapp.shared-domain.example.com	myapp.shared-domain.example.com
http://myapp.shared-domain.example.com/contact	myapp.shared-domain.example.com
http://myapp.shared-domain.example.com/products	myapp.shared-domain.example.com/products
http://myapp.shared-domain.example.com/products/123	myapp.shared-domain.example.com/products
http://products.shared-domain.example.com	No match; 404

The Gorouter does not use a route to match requests until the route is mapped to an app. In the above example, `products.shared-domain.example.com` may have been created as a route in TAS for VMs, but until it is mapped to an app, requests for the route receive a `404` error.

The routing tier knows the location of instances for apps mapped to routes. Once the routing tier determines a route as the best match for a request, it makes a load-balancing calculation using a round-robin algorithm, and forwards the request to an instance of the mapped app.

Developers can map many apps to a single route, resulting in load-balanced requests for the route across all instances of all mapped apps. This approach enables the blue/green rolling deployment strategy. Developers can also map an individual app to multiple routes, enabling access to the app from many URLs. The number of routes that can be mapped to each app is approximately 1000 (128 KB).

Routes belong to a space, and developers can only map apps to a route in the same space.

Note: Routes are globally unique. Developers in one space cannot create a route with the same URL as developers in another space, regardless of which orgs control these spaces.

HTTP vs. TCP routes

Note: By default, TAS for VMs only supports routing of HTTP requests to apps.

Routes are considered HTTP if they are created from HTTP domains, and TCP if they are created from TCP domains. For more information, see HTTP vs. TCP Shared Domains.

HTTP routes include a domain, an optional hostname, and an optional context path. `shared-domain.example.com`, `myapp.shared-domain.example.com`, and `myapp.shared-domain.example.com/products` are all examples of HTTP routes. Apps should listen to the `localhost` port defined by the `$PORT` environment variable, which is `8080` on Diego. As an example, requests to `myapp.shared-domain.example.com` would be routed to the app container at `localhost:8080`.

Note: Developers can use the Cloud Controller API to update the ports an app can receive requests on. For more information, see Configuring Apps to Listen on

Custom Ports (Beta).

- Requests to HTTP routes must be sent to ports 80 or 443.
- Ports cannot be reserved for HTTP routes.

TCP routes include a domain and a route port. A route port is the port clients make requests to. This is not the same port as what an app pushed to Cloud Foundry listens on. `tcp.shared-domain.example.com:60000` is an example of a TCP route. Just as for HTTP routes, apps should listen to the `localhost` port defined by the `$PORT` environment variable, which is `8080` on Diego. As an example, requests to `tcp.shared-domain.example.com:60000` would be routed to the app container at `localhost:8080`.

- Once a port is reserved for a route, it cannot be reserved for another route.
- Hostname and path are not supported for TCP routes.

Internal container-to-container routes

TAS for VMs apps can communicate with each other securely and directly over internal routes that never leave the platform.

Note: Apps running on Windows cells cannot use internal, container-to-container routes.

To create an internal route:

1. Use the cf map-route command with an internal domain. For example:

   ```
   $ cf map-route app apps.internal --hostname app
   ```

   - After an internal route is mapped to an app, the route resolves to IP addresses of the app instances. The IP addresses are visible in the app container:

     ```
     $ cf ssh app
     vcap@1234:~$ host app.apps.internal
     app.apps.internal has address 10.255.169.200
     app.apps.internal has address 10.255.49.7
     app.apps.internal has address 10.255.49.77
     ```

   - To resolve individual instances, prepend the index to the internal route.

     ```
     vcap@1234:~$ host 1.app.apps.internal
     1.app.apps.internal has address 10.255.49.7
     ```

2. Create a network policy that allows your apps to communicate with each other. By default, apps cannot communicate over the container network. For more information, see Configuring Container-to-Container Networking and add-network-policy in the *Cloud Foundry CLI Reference Guide*.

Create a route

When a developer creates a route using the cf CLI, TAS for VMs determines whether the route is an HTTP or a TCP route based on the domain. To create a HTTP route, a developer must choose an HTTP domain. To create a TCP route, a developer must choose a TCP domain.

Domains in TAS for VMs provide a namespace from which to create routes. To list available domains for a targeted organization, use the cf domains command. For more information about domains, see Domains.

The following sections describe how developers can create HTTP and TCP routes for different use cases.

Create an HTTP route with hostname

In TAS for VMs, a hostname is the label that indicates a subdomain of the domain associated with the route. Given a domain `shared-domain.example.com`, a developer can create the route `myapp.shared-domain.example.com` by specifying the hostname `myapp` with the cf create-route command as shown in this example:

- **cf CLI v7**

  ```
  $ cf create-route shared-domain.example.com --hostname myapp
  Creating route myapp.shared-domain.example.com for org my-org / space m
  y-space as username@example.com...
  OK
  ```

  > **Note:** The cf CLI v7 `create-route` command does not require the space as an argument. It uses the space you are currently targeting.

- **cf CLI v6**

  ```
  $ cf create-route my-space shared-domain.example.com --hostname myapp
  Creating route myapp.shared-domain.example.com for org my-org / space m
  y-space as username@example.com...
  OK
  ```

This command instructs TAS for VMs to only route requests to apps mapped to this route for the following URLs:

- `http://myapp.shared-domain.example.com`
- `https://myapp.shared-domain.example.com`
- Any path under either of the above URLs, such as `http://myapp.shared-domain.example.com/bar`

Create an HTTP route without hostname

This approach creates a route with the same address as the domain itself and is permitted for private domains only. For more information, see Private Domains.

A developer can create a route from the domain `private-domain.example.com` with no hostname with the cf create-route command:

- **cf CLI v7**

```
$ cf create-route private-domain.example.com
Creating route private-domain.example.com for org my-org / space my-spa
ce as username@example.com...
OK
```

- **cf CLI v6**

  ```
  $ cf create-route my-space private-domain.example.com
  Creating route private-domain.example.com for org my-org / space my-spa
  ce as username@example.com...
  OK
  ```

If DNS has been configured correctly, this command instructs TAS for VMs to route requests to apps mapped to this route from these URLs:

- `http://private-domain.example.com`
- `https://private-domain.example.com`
- Any path under either of the above URLs, such as `http://private-domain.example.com/foo`

If there are no other routes for the domain, requests to any subdomain, such as `http://foo.private-domain.example.com`, will fail.

A developer can also create routes for subdomains with no hostnames. The following command creates a route from the subdomain `foo.private-domain.example.com`:

- **cf CLI v7**

  ```
  $ cf create-route foo.private-domain.example.com
  Creating route foo.private-domain.example.com for org my-org / space my
  -space as username@example.com...
  OK
  ```

- **cf CLI v6**

  ```
  $ cf create-route my-space foo.private-domain.example.com
  Creating route foo.private-domain.example.com for org my-org / space my
  -space as username@example.com...
  OK
  ```

Assuming DNS has been configured for this subdomain, this command instructs TAS for VMs to route requests to apps mapped to this route from these URLs:

- `http://foo.private-domain.example.com`
- `https://foo.private-domain.example.com`
- Any path under either of the above URLs, such as `http://foo.private-domain.example.com/foo`

Create an HTTP route with wildcard hostname

An app mapped to a wildcard route acts as a fallback app for route requests if the requested route does not exist. To create a wildcard route, use an asterisk for the hostname.

A developer can create a wildcard route from the domain `foo.shared-domain.example.com` by running:

- **cf CLI v7**

  ```
  $ cf create-route foo.shared-domain.example.com --hostname '*'
  Creating route *.foo.shared-domain.example.com for org my-org / space m
  y-space as username@example.com...
  OK
  ```

- **cf CLI v6**

  ```
  $ cf create-route my-space foo.shared-domain.example.com --hostname '*'
  Creating route *.foo.shared-domain.example.com for org my-org / space m
  y-space as username@example.com...
  OK
  ```

If a client sends a request to `http://app.foo.shared-domain.example.com` by accident, attempting to reach `myapp.foo.shared-domain.example.com`, TAS for VMs routes the request to the app mapped to the route `*.foo.shared-domain.example.com`.

Create an HTTP route with a path

Developers can use paths to route requests for the same hostname and domain to different apps.

A developer can create three routes using the same hostname and domain in the space `my-space` by running:

- **cf CLI v7**

  ```
  $ cf create-route shared-domain.example.com --hostname store --path pro
  ducts
  Creating route store.shared-domain.example.com/products for org my-org
  / space my-space as username@example.com...
  OK
  ```

  ```
  $ cf create-route shared-domain.example.com --hostname store --path ord
  ers
  Creating route store.shared-domain.example.com/orders for org my-org /
  space my-space as username@example.com...
  OK
  ```

  ```
  $ cf create-route shared-domain.example.com --hostname store
  Creating route store.shared-domain.example.com for org my-org / space m
  y-space as username@example.com...
  OK
  ```

- **cf CLI v6**

  Note: If you want to create a route without a path using cf CLI v6, you must run the commands in the order below. Create the route without a path first, and then create any routes with a path. If you do not need a route without a path, then this order is not required.

```
$ cf create-route my-space shared-domain.example.com --hostname store
Creating route store.shared-domain.example.com for org my-org / space m
y-space as username@example.com...
OK
```

```
$ cf create-route my-space shared-domain.example.com --hostname store -
-path products
Creating route store.shared-domain.example.com/products for org my-org
/ space my-space as username@example.com...
OK
```

```
$ cf create-route my-space shared-domain.example.com --hostname store -
-path orders
Creating route store.shared-domain.example.com/orders for org my-org /
space my-space as username@example.com...
OK
```

The developer can then map the new routes to different apps by following the steps in Map a Route to Your App.

If the developer maps the first route with path `products` to the `products` app, the second route with path `orders` to the `orders` app, and the last route to the `storefront` app. After this:

- TAS for VMs routes requests to `http://store.shared-domain.example.com/products` to the `products` app.
- TAS for VMs routes requests to `http://store.shared-domain.example.com/orders` to the `orders` app.
- TAS for VMs routes requests to `http://store.shared-domain.example.com` to the `storefront` app.

TAS for VMs attempts to match routes with a path first, and then attempts to match host and domain.

Note: Routes with the same domain and hostname but different paths can only be created in the same space. Private domains do not have this limitation.

Note: TAS for VMs does not route requests for context paths to the root context of an app. Apps must serve requests on the context path.

Create a TCP route with a port

cf CLI v7: To create an arbitrary port when the `--port` is not used, run the command shown below. Note that the `--random-port` flag is no longer supported, and the behavior it provided is now the default.

```
$ cf create-route my-space tcp.shared-domain.example.com
Creating route tcp.shared-domain.example.com for org my-org / space my-space
as user@example.com...
OK
Route tcp.shared-domain.example.com:60034 has been created
```

cf CLI v6: A developer can create a TCP route for `tcp.shared-domain.example.com` on an arbitrary port. If the clients of the app can accommodate addressing an arbitrary port, then developers should use the `--random-port` to instruct TAS for VMs to pick a port for your route. To create a TCP route for `tcp.shared-domain.example.com` on an arbitrary port, run:

```
$ cf create-route my-space tcp.shared-domain.example.com --random-port
Creating route tcp.shared-domain.example.com for org my-org / space my-space
as user@example.com...
OK
Route tcp.shared-domain.example.com:60034 has been created
```

In this example, TAS for VMs routes requests to `tcp.shared-domain.example.com:60034` to apps mapped to this route.

To request a specific port, a developer can use the `--port` flag, so long as the port is not reserved for another space. To create a TCP route for `tcp.shared-domain.example.com` on port 60035, run:

```
$ cf create-route my-space tcp.shared-domain.example.com --port 60035
Creating route tcp.shared-domain.example.com:60035 for org my-org / space my-
space as user@example.com...
OK
```

List routes

Developers can list routes for the current space with the cf routes command. A route is uniquely identified by the combination of hostname, domain, and path. Note that, as of cf CLI v7, `port` no longer appears in the table.

```
$ cf routes
Getting routes as user@private-domain.example.com ...

space      host     domain                                    port   path       typ
e       apps
my-space   myapp    shared-domain.example.com
myapp
my-space   myapp    private-domain.example.com
myapp
my-space   store    shared-domain.example.com              /products        p
roducts
my-space   store    shared-domain.example.com              /orders          o
rders
my-space   store    shared-domain.example.com                           store
front
```

```
my-space                            shared-domain.example.com   60000
tcp   tcp-app
```

Developers can only see routes in spaces where they are members.

> **Note:** cf CLI v7 removes the `port` and `path` columns from the output.

Check routes

Developers cannot create a route that is already taken. To check whether a route is available, developers can use the cf check-route command.

To check whether a route with the hostname `store` and the domain `shared-domain.example.com` and the path `products` exists, run:

- **cf CLI v7**:

  ```
  $ cf check-route shared-domain.example.com --hostname store --path /pro
  ducts
  Checking for route...
  OK
  Route store.shared-domain.example.com/products does exist
  ```

- **cf CLI v6**:

  ```
  $ cf check-route store shared-domain.example.com --path /products
  Checking for route...
  OK
  Route store.shared-domain.example.com/products does exist
  ```

Map a route to your app

For an app to receive requests to a route, developers must map the route to the app with the cf map-route command. If the route does not already exist, this command creates it.

> **Note:** Any app that is not routed to port `80` or port `443` must be explicitly mapped using the `cf map-route` command . Otherwise, the route is automatically mapped to port `443`.

Developers can create and reserve routes for later use by following the steps in Manually Map a Route. Or they can map routes to their app immediately as part of a push by following the steps in Map a Route with App Push.

> **Note:** Changes to route mappings are executed asynchronously. On startup, an app will be accessible at its route within a few seconds. Similarly, upon mapping a new route to a running app, the app will be accessible at this route within a few seconds of the CLI exiting successfully.

Manually map a route

Given the following routes and apps:

Route	Apps
store.shared-domain.example.com/products	products
store.shared-domain.example.com/orders	orders
store.shared-domain.example.com	storefront
tcp.shared-domain.example.com:60000	tcp-app

The following commands map the above routes to their respective apps. Developers use hostname, domain, and path to uniquely identify a route to map their apps to.

```
$ cf map-route products shared-domain.example.com --hostname store --path pro
ducts
$ cf map-route orders shared-domain.example.com --hostname store --path order
s
$ cf map-route storefront shared-domain.example.com --hostname store
$ cf map-route tcp-app tcp.shared-domain.example.com --port 60000
```

The following command maps the wildcard route `*.foo.shared-domain.example.com` to the app `myfallbackapp`.

```
$ cf map-route myfallbackapp foo.shared-domain.example.com --hostname '*'
```

In cf CLI v8, the following command maps the route `h2app.shared-domain.example.com` as an HTTP/2 route to the HTTP/2 app `h2app`.

```
$ cf map-route h2app shared-domain.example.com --hostname h2app --destination
-protocol http2
```

In cf CLI v8, the following command maps the route `h2app.shared-domain.example.com` as an HTTP/2 route to the HTTP/2 app `h2app`.

```
$ cf map-route h2app shared-domain.example.com --hostname h2app --destination
-protocol http2
```

Map a route with app push

Developers can map a route to their app with the `cf push` command.

If a domain or hostname is not specified, then a route will be created using the app name and the default shared domain (see Shared Domains). The following command pushes the app `myapp`, creating the route `myapp.shared-domain.example.com` from the default shared domain `shared-domain.example.com`. If the route has not already been created in another space this command also maps it to the app.

```
$ cf push myapp
```

To customize the route during `push`:

cf CLI v6: Specify the domain using the `-d` flag and the hostname with the `--hostname` flag. The following command creates the `foo.private-domain.example.com` route for `myapp`:

```
$ cf push myapp -d private-domain.example.com --hostname foo
```

cf CLI v7: The following command creates the `foo.private-domain.example.com` route for `myapp`. Domain and hostname must be specified in the manifest using the `routes` property.

```
$ cf push myapp private-domain.example.com
```

To map a TCP route during `push`, specify a TCP domain and request a random port using `--random-route`. To specify a port, push the app without a route, then create and map the route manually by following the steps in Create a TCP Route with a Port. If you are using cf CLI v6, use the `-d` flag to specify the domain.

```
$ cf push tcp-app tcp.shared-domain.example.com --random-route
```

Map a route using app manifest

Developers can map a route to their app with a manifest by editing the `route` attribute to specify the host, domain, port and/or path components of the route. For more information, see Deploying with App Manifests.

Map a route to multiple apps

TAS for VMs allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

For more information about troubleshooting this problem, see the Routing Conflict section of the *Troubleshooting App Deployment and Health* topic.

Map multiple routes to one app

You can have multiple routes to an app, but those routes cannot have different context paths.

The following routes are valid for a single app:

Route 1	Route 2
`myapp.example.com`	`myapp.apps.cf.example.com`
`myapp.example.com/foo`	`myapp.apps.cf.example.com/foo`

The following routes are **not** valid for a single app:

Route 1	Route 2
`myapp.example.com/foo`	`myapp.apps.cf.example.com/bar`

myapp.apps.cf.example.com/foo	myapp.example.com/bar

Map an internal route to an app

You can map an internal route to any app. This internal route allows your app to communicate with other apps without leaving the platform. Once mapped, this route becomes available to all other apps on the platform.

This example creates a `foo.apps.internal` internal route for `myapp`:

```
$ cf map-route myapp apps.internal --hostname foo
```

Unmap a route

Developers can remove a route from an app using the `cf unmap-route` command. The route remains reserved for later use in the space where it was created until the route is deleted.

To unmap an HTTP route from an app, identify the route using the hostname, domain, and path:

```
$ cf unmap-route tcp-app private-domain.example.com --hostname myapp --path mypath
```

To unmap a TCP route from an app, identify the route using the domain and port:

```
$ cf unmap-route tcp-app tcp.shared-domain.example.com --port 60000
```

Delete a route

Developers can delete a route from a space using the `cf delete-route` command.

To delete a HTTP route, identify the route using the hostname, domain, and path:

```
$ cf delete-route private-domain.example.com --hostname myapp --path mypath
```

To delete a TCP route, identify the route using the domain and port.

```
$ cf delete-route tcp.private-domain.example.com --port 60000
```

Routing requests to a specific app instance

Users can route HTTP requests to a specific app instance using the header `X-Cf-App-Instance`.

Use of the `X-Cf-App-Instance` header is only available for users on the Diego architecture.

The format of the header is `X-Cf-App-Instance: APP_GUID:APP_INDEX`.

`APP_GUID` is an internal identifier for your app. Use the `cf APP-NAME --guid` command to discover the `APP_GUID` for your app.

```
$ cf app myapp --guid
```

APP_INDEX, for example 0,1, 2, or 3, is an identifier for a particular app instance. Use the CLI command cf app APP-NAME to get statistics on each instance of a particular app.

```
$ cf app myapp
```

The following example shows a request made to instance 9 of an app with GUID 5cdc7595-2e9b-4f62-8d5a-a86b92f2df0e and mapped to route myapp.private-domain.example.com.

```
$ curl myapp.private-domain.example.com -H "X-Cf-App-Instance: 5cdc7595-2e9b-
4f62-8d5a-a86b92f2df0e:9"
```

If the X-Cf-App-Instance header is set to an invalid value, Gorouter returns a 400 status code and the response from Gorouter contains a X-Cf-Routererror header with more information about the error. Before the routing release v0.197.0, Gorouter returned a 404 error.

The following table describes the possible error responses:

X-Cf-Routererror Value	Reason for Error	Response Body
invalid_cf_app_instance_header	The value provided for X-Cf-App-Instance was an incorrectly formatted GUID.	None
unknown_route	The value provided for X-Cf-App-Instance is a correctly formatted GUID, but there is no instance found with that GUID for the route requested.	400 Bad Request: Requested instance ('1') with guid ('aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa') does not exist for route ('myroute.cf.com')

Domains

Note: The term domain in this topic differs from its common use and is specific to TAS for VMs. Likewise, shared domain and private domain refer to resources with specific meaning in TAS for VMs. The use of domain name, root domain, and subdomain refers to DNS records.

Domains indicate to a developer that requests for any route created from the domain are routed to TAS for VMs. This requires DNS to be configured out-of-band to resolve the domain name to the IP address of a load balancer configured to forward requests to the TAS for VMs routers. For more information about configuring DNS, see DNS for Domains.

List domains for an org

When creating a route, developers select from domains available to them. Use the cf domains command to view a list of available domains for the targeted org:

```
$ cf domains
Getting domains in org my-org as user@example.com... OK
name                     status   type
shared-domain.example.com         shared
```

```
tcp.shared-domain.example.com    shared   tcp
private-domain.example.com       owned
```

This example displays three available domains: a shared HTTP domain `shared-domain.example.com`, a shared TCP domain `tcp.shared-domain.example.com`, and a private domain `private-domain.example.com`. For more information, see Shared Domains and Private Domains.

HTTP vs. TCP domains

HTTP domains indicate to a developer that only requests using the HTTP protocol are routed to apps mapped to routes created from the domain. Routing for HTTP domains is layer 7 and offers features like custom hostnames, sticky sessions, and TLS termination.

TCP domains indicate to a developer that requests over any TCP protocol, including HTTP, are routed to apps mapped to routes created from the domain. Routing for TCP domains is layer 4 and protocol agnostic, so many features available to HTTP routing are not available for TCP routing. TCP domains are defined as being associated with the TCP Router Group. The TCP Router Group defines the range of ports available to be reserved with TCP Routes. Currently, only Shared Domains can be TCP.

Note: By default, TAS for VMs only supports routing of HTTP requests to apps.

Shared domains

Admins manage shared domains, which are available to users in all orgs of a TAS for VMs deployment. An admin can offer multiple shared domains to users. For example, an admin may offer developers the choice of creating routes for their apps from `shared-domain.example.com` and `cf.some-company.com`.

If a developer pushes an app without specifying a domain, a route is created for it from the first shared domain created in the system. All other operations involving route require the domain be specified (see Routes).

When using shared domains, you cannot have routes with the same hostname and domain across different orgs and spaces.

Shared domains are HTTP by default, but can be configured to be TCP when associated with the TCP router group.

Create a shared domain

Admins can create an HTTP shared domain with the `cf create-shared-domain` command:

```
$ cf create-shared-domain shared-domain.example.com
```

To create a TCP shared domain, first discover the name of the TCP router group.

Note: cf CLI v7 does not support TCP routing or creating shared domains with router groups.

```
$ cf router-groups
Getting router groups as admin ...
name          type
default-tcp   tcp
```

Then create the shared domain using the `--router-group` option to associate the domain with the TCP router group.

```
$ cf create-shared-domain tcp.shared-domain.example.com --router-group defaul
t-tcp
```

Delete a shared domain

Admins can delete a shared domain from TAS for VMs with the `cf delete-shared-domain` command:

```
$ cf delete-shared-domain example.com
```

Internal domain

The internal domain is a special type of shared domain used for app communication internal to the platform. When you enable service discovery, the internal domain `apps.internal` becomes available for route mapping.

Admins can configure multiple internal domains. First add a custom internal domain name to the `internal_domains` property on the `bosh-dns-adapter` job. Then create an internal domain using the `--internal` option:

```
$ cf create-shared-domain shared-domain.example.com --internal
```

The `--router-group` option is not used with internal domains.

Private domains

Org managers can add private domains, or custom domains, and give members of the org permission to create routes for privately registered domain names.

Private domains can be shared with other orgs and spaces. These are referred to as shared private domains and are not the same as shared domains. See Shared Domains above.

When using private domains, you can have routes with the same hostname and domain name across different orgs and spaces. This cannot be done with shared domains.

Private domains can be HTTP or HTTPS only. TCP Routing is supported for shared domains only.

Create a private domain

Org managers can create a private domain with the following command:

- **cf CLI v7**

  ```
  $ cf create-private-domain my-org private-domain.example.com
  ```

- **cf CLI v6**

  ```
  $ cf create-domain my-org private-domain.example.com
  ```

Org managers can create a private domain for a subdomain with the following command:

- **cf CLI v7**

  ```
  $ cf create-private-domain my-org foo.private-domain.example.com
  ```

- **cf CLI v6**

  ```
  $ cf create-domain my-org foo.private-domain.example.com
  ```

Sharing a private domain with one or more orgs

Org managers can grant or revoke access to a private domain to other orgs if they have permissions for these orgs with the following commands:

```
$ cf share-private-domain test-org private-domain.example.com

$ cf unshare-private-domain test-org private-domain.example.com
```

Note that, as of cf CLI v7, `cf unshare-private-domain` provides a warning and requires confirmation before proceeding. The optional `-f` flag forces unsharing without confirmation.

Delete a private domain

Org managers can delete a domain from TAS for VMs with the `cf delete-private-domain` command:

- **cf CLI v7**:

  ```
  $ cf delete-private-domain private-domain.example.com
  ```

  Note: cf CLI v7 renames the `delete-domain` command to `delete-private-domain`.

- **cf CLI v6**:

  ```
  $ cf delete-domain private-domain.example.com
  ```

Requirements for parent and child domains

In the domain `myapp.shared-domain.example.com`, `shared-domain.example.com` is the parent domain of subdomain `myapp`. Note the following requirements for domains:

- You can only create a private domain that is parent to a private subdomain.
- You can create a shared domain that is parent to either a shared or a private subdomain.

The domain `foo.myapp.shared-domain.example.com` is the child subdomain of `myapp.shared-domain.example.com`. Note the following requirements for subdomains:

- You can create a private subdomain for a private parent domain only if the domains belong to the same org.
- You can create a private subdomain for a shared parent domain.
- You can only create a shared subdomain for a shared parent domain.
- You cannot create a shared subdomain for a private parent domain.

DNS for domains

To create customized access to your apps, you can map specific or wildcard custom domains to TAS for VMs by using your DNS provider.

Mapping domains to your custom domain

To associate a registered domain name with a domain on TAS for VMs, configure a CNAME record with your DNS provider, pointing at any shared domain offered in TAS for VMs.

Mapping a single domain to your custom domain

To map a single domain to a custom domain to TAS for VMs, configure a CNAME record with your DNS provider.

The following table provides some example CNAME record mappings.

Record Set in Custom Domain	Type	Target in TAS for VMs
myapp.yourcustomdomain.com.	CNAME	myapp.shared-domain.example.com
www.yourcustomdomain.com.	CNAME	myapp.shared-domain.example.com

After you create the CNAME mapping, your DNS provider routes your custom domain to `myapp.shared-domain.example.com`.

> **Note:** See your DNS provider documentation to determine whether the trailing `.` is required.

Mapping multiple subdomains to your custom domain

Use a wildcard CNAME record to point all of the subdomains in your custom domain to shared-domain.example.com.

Each separately configured subdomain has priority over the wildcard configuration.

The following table provides some example wildcard CNAME record mappings.

Record Set in Custom Domain	Type	Target in TAS for VMs
*.yourcustomdomain.com.	CNAME	*.shared-domain.example.com
*.yourcustomdomain.com.	CNAME	*.myapp.shared-domain.example.com

If you use a wildcard as the subdomain name, then your DNS provider can route from `*.YOURCUSTOMDOMAIN` to any of the following:

- `*.shared-domain.example.com`
- `foo.myapp.shared-domain.example.com`
- `bar.foo.myapp.shared-domain.example.com`

Configuring DNS for your registered root domain

To use your root domain (for example, `example.com`) for apps on TAS for VMs you can either use custom DNS record types like ALIAS and ANAME, if your DNS provider offers them, or subdomain redirection.

> **Note:** Root domains are also called zone apex domains.

If your DNS provider supports using an ALIAS or ANAME record, configure your root domain with your DNS provider to point at a shared domain in TAS for VMs.

Record	Name	Target	Note
ALIAS or ANAME	empty or @	private-domain.example.com.	Refer to your DNS provider documentation to determine whether to use an empty or @ value for the Name entry.

If your DNS provider does not support ANAME or ALIAS records you can use subdomain redirection, also known as domain forwarding, to redirect requests for your root domain to a subdomain configured as a CNAME.

> **Note:** If you use domain forwarding, SSL requests to the root domain may fail if the SSL certificate only matches the subdomain. For more information about SSL certificates, see Configuring Trusted System Certificates for Apps.

Configure the root domain to point at a subdomain such as `www`, and configure the subdomain as a CNAME record pointing at a shared domain in TAS for VMs.

Record	Name	Target	Note
URL or Forward	private-domain.example.com	www.private-domain.example.com	This method results in a `301 permanent redirect` to the subdomain you configure.
CNAME	www	myapp.shared-domain.example.com	

Routing HTTP/2 and gRPC traffic to apps

Here are instructions for routing HTTP/2 and gRPC traffic to VMware Tanzu Application Service for VMs (TAS for VMs) apps. HTTP/2 is the second major version of the the HTTP protocol. In TAS for VMs v2.12 and later, HTTP/2 support is enabled by default.

For more information about the HTTP/2 protocol, see RFC 7540.

gRPC is a Remote Procedure Call (RPC) framework that uses HTTP/2 as its transport medium. It is especially useful for managing communication between apps in a microservices cluster. For apps to serve gRPC traffic, every network hop between the client and app must use HTTP/2.

For more information, see gRPC.

Performance

HTTP/2 can provide performance improvements for apps that are built to do the following:

- Load many resources in parallel. For example, an HTML page that loads many images or JavaScript files.
- Use large and repetitive headers.

Other apps might see minimal or no performance increases. Experiment with your app to see how serving HTTP/2 affects its performance.

For an example of an app that benefits from HTTP/2, see the Cloud Foundry HTTP/2 Tile Demo App.

Limitations

HTTP/2 support in TAS for VMs has the following limitations:

- End-to-end HTTP/2 is not available for Windows Diego Cells. Routes with HTTP/2 mappings continue to send HTTP/1.1 traffic to apps that run on Windows Diego Cells.
- Routes that have route services bound to them might not support end-to-end HTTP/2 depending on what protocols the bound service supports. For more information, see Route Services.

How Pushing Apps with HTTP/2 Works

This section describes how pushing apps with HTTP/2 works.

If your TAS for VMs deployment is configured to support HTTP/2, then all traffic ingressing to TAS for VMs supports HTTP/2. The traffic is forwarded as HTTP/1.1 before it reaches your app unless configured otherwise.

For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.

To serve gRPC traffic, your app must use HTTP/2 for all network hops. Configure your route to send HTTP/2 traffic to your app. When configured, traffic that matches the route is always sent to your app over HTTP/2 regardless of the original ingress protocol.

For example, external HTTP/1.1 requests are forwarded to your app over HTTP/2. Your app does not have the opportunity to negotiate what protocol it receives.

> **Note:** If a client is familiar with the HTTP/2 specification, your app receives HTTP/2 with prior knowledge. For more information, see Starting HTTP/2 with Prior Knowledge in *RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2).*

Push Apps with HTTP/2 Support

This section describes how to push TAS for VMs apps with support for HTTP/2.

You do not need to make changes to existing apps to support HTTP/2. For information about pushing an HTTP/1.1 app that serves HTTP/2 traffic, see Push an HTTP/1.1 App that Serves HTTP/2 below.

There are multiple ways to push an app with end-to-end HTTP/2. You can push the app using either the app manifest or the Cloud Foundry Command Line Interface (cf CLI). You can also use either method to push an app that serves gRPC traffic.

To push an app with end-to-end HTTP/2, see the following sections:

- Push an App with End-to-End HTTP/2 Using the App Manifest
- Push an App with End-to-End HTTP/2 Using the cf CLI
- Push a gRPC App

Push an HTTP/1.1 App that Serves HTTP/2

To push an HTTP/1.1 app that can serve HTTP/2 traffic:

1. Push the HTTP/1.1 app by running:

   ```
   cf push MY-APP
   ```

   Where `MY-APP` is the name of your app.

2. Send an HTTP/2 request to the app by running:

   ```
   curl MY-APP.EXAMPLE.COM --http2 -v
   ```

   Where `MY-APP.EXAMPLE.COM` is the route mapped to your app.

   The request and response are both issued over HTTP/2.

   > **Note:** To issue this request, you must use a version of `curl` that supports HTTP/2.

3. Review the app logs by running:

   ```
   cf logs MY-APP
   ```

   The request to the app is issued over HTTP/1.1.

Push an App with End-to-End HTTP/2 Using the App Manifest

To push an app that serves HTTP/2 traffic using the app manifest:

1. Use an app that supports serving HTTP/2 traffic with prior knowledge. For example, you can use the HTTP/2 test app from Cloud Foundry Acceptance Tests on GitHub.

2. Create an app manifest named `manifest.yml` with a route mapped to the app with HTTP/2:

```
---
applications:
- name: MY-APP
  routes:
    - route: MY-APP.EXAMPLE.COM
      protocol: http2
```

Where `MY-APP` is the name of your app and `MY-APP.EXAMPLE.COM` is the route you want to map to your app.

3. Push the app with the manifest by running:

   ```
   cf push -f manifest.yml
   ```

4. Send an HTTP/2 request to the app by running:

   ```
   curl MY-APP.EXAMPLE.COM --http2 -v
   ```

   The request and response are both sent over HTTP/2. The response from the test app includes the protocol.

   For example:

   ```
   Hello, /, TLS: false, Protocol: HTTP/2.0
   ```

   The response shows that the request was received over HTTP/2.

Push an App with End-to-End HTTP/2 Using the cf CLI

To push an app that serves HTTP/2 traffic using the cf CLI:

1. Use an app that supports serving HTTP/2 traffic with prior knowledge. For example, you can use the HTTP/2 test app from Cloud Foundry Acceptance Tests on GitHub.
2. Push the app without a default route by running:

   ```
   cf push --no-route
   ```

3. Map a route with destination protocol `http2` by running:

   ```
   cf map-route MY-APP EXAMPLE.COM --hostname host --destination-protocol http2
   ```

   Where `MY-APP` is the name of your app and `EXAMPLE.COM` is the route you want to map to your app.

4. Send an HTTP/2 request to the app by running:

   ```
   curl HOST.EXAMPLE.COM --http2 -v
   ```

   The request and response are both sent over HTTP/2. The response from the test app includes the protocol.

   For example:

```
Hello, /, TLS: false, Protocol: HTTP/2.0
```

The response shows that the request was received over HTTP/2.

Push a gRPC App

To push an app that serves gRPC traffic:

1. Use an app that supports serving gRPC traffic. For example, you can use the gRPC test app from Cloud Foundry Acceptance Tests on GitHub.
2. Push the app with HTTP/2 enabled using either the app manifest or the cf CLI. See Push an App with End-to-End HTTP/2 Using the App Manifest or Push an App with End-to-End HTTP/2 Using the cf CLI above.
3. Send a gRPC request to the app using `grpcurl` by running the command below. For more information, see grpcurl.

   ```
   grpcurl -vv -import-path ./test/ -proto test.proto MY-APP.EXAMPLE.COM:443 test.
   Test.Run
   ```

   Where `MY-APP.EXAMPLE.COM` is the route to your app.

   A successful response looks like the following:

   ```
   Response contents:
   {
     "body": "Hello"
   }
   ```

Configuring TAS for VMs to Route Traffic to Apps on Custom Ports

By default, apps only receive requests on port 8080 for both HTTP and TCP routing, and so must be configured, or hardcoded, to listen on this port. Configuring custom app ports allows developers to bring workloads onto TAS for VMs that listen on ports other than 8080. Some example use cases are:

- Serving web client requests on one port and offering stats/debug on another
- Using TCP protocols that require multiple ports
- Running Docker images on TAS for VMs

The procedure below describes how to use the `apps` and `route_mappings` Cloud Controller API endpoints to update the ports the app can receive requests on.

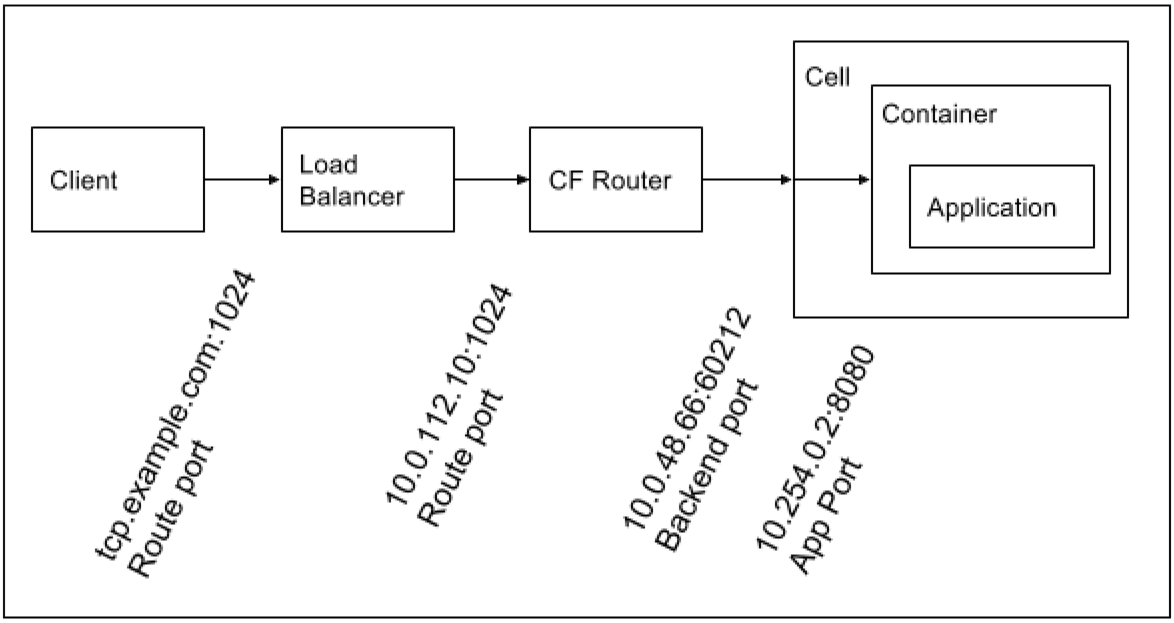
Flow of a Request to an App

Flow of a request to an app

The following table describes the Network Address Translation that occurs in the data path of a client request.

Port Type	Description	Network Hop
Route port	The port a client sends a request to	Client to load balancer, load balancer to Gorouter
Back end port	The port on the VM where an app container is hosted, which is unique to the container	Gorouter to Diego Cell
App port	The port on the container; this must match a port the app is configured to listen on	Diego Cell to app container

The following diagram provides an example data path and Network Address Translation for TCP routing. For HTTP routing, the route port is always 80 or 443.



Prerequisites

Before following the procedure to configure routing to your app on custom ports, you must have:

- An app pushed to TAS for VMs that can listen on one or more custom ports.
- Routes for which you want traffic forwarded to your app on custom ports.
    - If your app listens on two ports and you want clients to be able to send requests to both of them, create two routes. These can be from HTTP or TCP domains. Consider an example in which you have two routes: `foo.example.com` and `bar.example.com`. In the following procedure, you use API endpoints to map these routes to your app on the ports it is listening. For more information, see Routes and Domains.

Procedure

To configure your app to receive HTTP or TCP traffic on custom ports:

1. Retrieve the GUID of your app by running:

```
cf app APP-NAME --guid
```

Where `APP-NAME` is the name of your app.

2. Configure TAS for VMs with the ports your app is listening on by running:

```
cf curl /v2/apps/APP-GUID -X PUT -d '{"ports": [PORT1, PORT2, PORT3...]}'
```

Where:

- `APP-GUID` is the GUID of your app.
- `PORT1, PORT2, PORT3...` is a comma-separated list of the ports on which you want your app to receive traffic.

3. Retrieve the GUID of the route to which clients make requests, and for which TAS for VMs routes requests to the app on a custom port. Use one of the following options:

- For a TCP route with a hostname, retrieve its GUID by running:

```
cf curl /v2/routes?q=host:HOST-NAME
```

Where `HOST-NAME` is the hostname for the route. By default, this is the name of your app.

- For a TCP route without a hostname, retrieve its GUID by running:

```
cf curl /v2/apps/APP-GUID/routes
```

4. Check and update the route mappings for your app by running:

```
cf curl /v2/routes/ROUTE-GUID/route_mappings
cf curl /v2/route_mappings -X POST -d '{"app_guid": "APP-GUID", "route_guid":
"ROUTE-GUID", "app_port": PORT1}'
```

Where:

- `APP-GUID` is the GUID of your app.
- `ROUTE-GUID` is the GUID of the route at which the app serves.
- `PORT1` is the app port, or one of the app ports, that you added in the previous step.

5. Repeat the previous two steps for each port that you want your app to receive requests on.

Note: If you are trying to remove an app port, you need to delete the associated route mapping before you can update the app to remove the port.

Additional resources

For additional resources related to configuring custom app ports:

- For more information about making requests to the Cloud Controller `apps` endpoint, see Updating an App in the Cloud Controller API documentation.

- For more information about making requests to the Cloud Controller `route_mappings` endpoint, see Mapping an App and a Route in the Cloud Controller API documentation.
- For an example multi-port app, see the cf-acceptance-tests repository on GitHub.
- For a demo procedure written by an open source CF user, see the "Multiple App Ports" Demo on Cloud Foundry on GitHub.

Managing Apps with the cf CLI

In this section:

- Running Tasks
- Scaling an App Using cf scale
- Using App Health Checks
- App Revisions
- Configuring Container-to-Container Networking

Running tasks in your apps

A task is an app or script, the code of which is included as part of a deployed app, but runs independently in its own container. This topic describes how to run tasks in VMware Tanzu Application Service for VMs (TAS for VMs).

About Tasks

In contrast to a long-running process (LRP), tasks run for a finite amount of time, then stop. Tasks run in their own containers and are designed to use minimal resources. After a task runs, Cloud Foundry destroys the container running the task.

As a single-use object, a task can be checked for its state and for a success or failure message.

Note: Running a task consumes an app instance and is billed accordingly.

Use cases for tasks

Tasks are used to perform one-off jobs, which include:

- Migrating a database
- Sending an email
- Running a batch job
- Running a data processing script
- Processing images
- Optimizing a search index
- Uploading data
- Backing-up data

- Downloading content

How tasks are run

Tasks are always executed asynchronously, meaning that they run independently from the parent app or other tasks that run on the same app.

The lifecycle of a task is as follows:

1. A user initiates a task in TAS for VMs using one of the following mechanisms:
   - The `cf run-task APP-NAME "TASK"` command. For more information, see Running tasks in your apps.
   - A Cloud Controller v3 API call. For more information, see the Cloud Foundry API documentation.
   - The Cloud Foundry Java Client. For more information, see Cloud Foundry Java Client Library and the Cloud Foundry Java Client repository on GitHub.
2. Cloud Foundry creates a container specifically for the task.
3. Cloud Foundry runs the task on the container using the value passed to the `cf run-task` command.
4. Cloud Foundry destroys the container.

The container also inherits environment variables, service bindings, and security groups bound to the app.

Note: You cannot SSH into the container running a task.

Task logging and execution history

Any data or messages the task outputs to STDOUT or STDERR is available on the app's firehose logs. A syslog drain attached to the app receives the task log output.

The task execution history is retained for one month.

Manage tasks

At the system level, a user with admin-level privileges can use the Cloud Controller v3 API to view all tasks that are running within an org or space. For more information, see List tasks in the Cloud Foundry API documentation.

Admins can set the default memory and disk usage quotas for tasks on a global level. Tasks use the same memory and disk usage defaults as apps unless customized using run-task in the cf CLI.

You configure the default memory and disk allocations using the **Default app memory** and **Default disk quota per app** fields in the **App Developer Controls** pane of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.

Run a task on an app

You can use the Cloud Foundry Command Line Interface (cf CLI) to run a task in the context of an app.

> **Note:** To run tasks with the cf CLI, you must install cf CLI v6.23.0 or later, or install cf CLI v7. For information about downloading, installing, and uninstalling the cf CLI., see Installing the Cloud Foundry Command Line Interface.

> **Note:** To run a task on v6 cf CLI without starting the app, push the app with `cf push -i 0` and then run the task. You can run the app later by scaling up its instance count.

Run a task on an app with cf CLI v6

To run a task on an app with cf CLI v6:

1. Push your app by running:

   ```
   cf push APP-NAME
   ```

2. Run your task on the deployed app by running:

   ```
   cf run-task APP-NAME "TASK" --name TASK-NAME
   ```

   The following example runs a database migration as a task on the `my-app` app:

   ```
   $ cf run-task my-app "bin/rails db:migrate" --name my-task
   Creating task for app my-app in org jdoe-org / space development as jdoe@pivotal.io...
   OK
   Task 1 has been submitted successfully for execution.
   ```

   > **Note:** To re-run a task, you must run it as a new task using the above command.

3. To display the recent logs of the app and all its tasks, run:

   ```
   cf logs APP-NAME --recent
   ```

   The following example displays the logs of a successful task:

   ```
   $ cf logs my-app --recent
   2017-01-03T15:58:06.57-0800 [APP/TASK/my-task/0]OUT Creating container
   2017-01-03T15:58:08.45-0800 [APP/TASK/my-task/0]OUT Successfully created container
   2017-01-03T15:58:13.32-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:58:13.322258 #7] DEBUG -- :    (15.9ms)  CREATE TABLE "schema_migrations" ("version" character varying PRIMARY KEY)
   2017-01-03T15:58:13.33-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:5
   ```

```
8:13.337723 #7] DEBUG -- :    (11.9ms)  CREATE TABLE "ar_internal_metad
ata" ("key" character varying PRIMARY KEY, "value" character varying,
"created_at" timestamp NOT NULL, "updated_at" timestamp NOT NULL)
2017-01-03T15:58:13.34-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:5
8:13.340234 #7] DEBUG -- :    (1.6ms)  SELECT pg_try_advisory_lock(3720
865444824511725);
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:5
8:13.351853 #7] DEBUG -- :   ActiveRecord::SchemaMigration Load (0.7ms)
SELECT "schema_migrations".* FROM "schema_migrations"
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT I, [2017-01-03T23:5
8:13.357294 #7]  INFO -- : Migrating to CreateArticles (20161118225627)
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT D, [2017-01-03T23:5
8:13.359565 #7] DEBUG -- :    (0.5ms)  BEGIN
2017-01-03T15:58:13.35-0800 [APP/TASK/my-task/0]OUT == 20161118225627 C
reateArticles: migrating ==================================
2017-01-03T15:58:13.50-0800 [APP/TASK/my-task/0]OUT Exit status 0
2017-01-03T15:58:13.56-0800 [APP/TASK/my-task/0]OUT Destroying containe
r
2017-01-03T15:58:15.65-0800 [APP/TASK/my-task/0]OUT Successfully destro
yed container
```

The following example displays the logs of a failed task:

```
$ cf logs my-app --recent
2016-12-14T11:09:26.09-0800 [APP/TASK/my-task/0]OUT Creating container
2016-12-14T11:09:28.43-0800 [APP/TASK/my-task/0]OUT Successfully create
d container
2016-12-14T11:09:28.85-0800 [APP/TASK/my-task/0]ERR bash: bin/rails: co
mmand not found
2016-12-14T11:09:28.85-0800 [APP/TASK/my-task/0]OUT Exit status 127
2016-12-14T11:09:28.89-0800 [APP/TASK/my-task/0]OUT Destroying containe
r
2016-12-14T11:09:30.50-0800 [APP/TASK/my-task/0]OUT Successfully destro
yed container
```

If your task name is unique, you can `grep` the output of the `cf logs` command for the task name to view task-specific logs.

Run a task on an app with cf CLI v7

To run a task on an app with cf CLI v7:

1. Configure your v3 API manifest with a task as a process type. For more information, see The app manifest specification in the Cloud Foundry API documentation.

2. Push your app by running:

   ```
   cf push APP-NAME --task
   ```

3. Run your task on the deployed app by running:

   ```
   cf run-task APP-NAME --name TASK-NAME
   ```

> **Note:** `cf run-task` allows the optional flags `--process` and `--command`. If provided, `--command` overrides the manifest property.

The following example runs a database migration as a task on the `my-app` app:

```
$ cf run-task my-app --name my-task
Creating task for app my-app in org jdoe-org / space development as jdo
e@pivotal.io...
OK
Task 1 has been submitted successfully for execution.
```

4. To display the recent logs of the app and all its tasks, run:

```
cf logs APP-NAME --recent
```

List tasks running on an app

To list the tasks for a given app, run `cf tasks APP-NAME`. For example:

```
$ cf tasks my-app
Getting tasks for app my-app in org jdoe-org / space development as jdoe@pivo
tal.io...
OK

id   name       state       start time                      command
2    339044ef   FAILED      Wed, 23 Nov 2016 21:52:52 UTC   echo foo; sleep 1
00; echo bar
1    8d0618cf   SUCCEEDED   Wed, 23 Nov 2016 21:37:28 UTC   bin/rails db:migr
ate
```

Each task has one of the following states:

State	Description
RUNNING	The task is currently in progress.
FAILED	The task did not complete. This state occurs when a task does not work correctly or a user cancels the task.
SUCCEEDED	The task completed successfully.

Cancel a task

After running a task, you may be able to cancel it before it finishes. To cancel a running task, run `cf terminate-task APP-NAME TASK-ID`. For example:

```
$ cf terminate-task my-app 2
Terminating task 2 of app my-app in org jdoe-org / space development as jdoe@
pivotal.io...
OK
```

Scaling an app using Cloud Foundry CLI (cf scale)

Factors such as user load, or the number and nature of tasks performed by an app, can change the disk space and memory the app uses. This topic describes how to scale an app using the Cloud Foundry Command Line Interface (cf CLI). For many apps, increasing the available disk space or memory can improve overall performance. Similarly, running additional instances of an app can allow the app to handle increases in user load and concurrent requests. These adjustments are called **scaling** an app.

Use cf scale to scale your app up or down to meet changes in traffic or demand.

Note: You can configure your app to scale automatically based on rules that you set. For more information, see Scaling an App Using Autoscaler and Using the App Autoscaler CLI.

Scaling horizontally

Horizontally scaling an app creates or destroys instances of your app.

Incoming requests to your app are automatically load balanced across all instances of your app, and each instance handles tasks in parallel with every other instance. Adding more instances allows your app to handle increased traffic and demand.

Use `cf scale APP -i INSTANCES` to horizontally scale your app. Cloud Foundry will increase or decrease the number of instances of your app to match `INSTANCES`.

```
$ cf scale myApp -i 5
```

Note: In cf CLI v7, you can also use `--process` with `cf scale` to scale specific processes of your app.

Scaling vertically

Vertically scaling an app changes the disk space limit or memory limit that Cloud Foundry applies to all instances of the app.

Use `cf scale APP -k DISK` to change the disk space limit applied to all instances of your app. `DISK` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -k 512M
```

Use `cf scale APP -m MEMORY` to change the memory limit applied to all instances of your app. `MEMORY` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -m 1G
```

Using Cloud Foundry health checks

You can configure a health check for an app using the Cloud Foundry Command Line Interface (cf CLI) or by specifying the `health-check-http-endpoint` and `health-check-type` fields in an app manifest.

To configure a health check using the cf CLI, see Configure health checks when creating or updating and Configure health checks for an existing app. For more information about using an app manifest to configure a health check, see health-check-http-endpoint and health-check-type in *Deploying with App Manifests*.

App health checks function as part of the app lifecycle managed by Diego architecture. For more information, see Diego components and architecture.

Configure health checks when creating or updating an app

To configure a health check while creating or updating an app, run:

```
cf push APP-NAME -u HEALTH-CHECK-TYPE -t HEALTH-CHECK-TIMEOUT
```

In cf CLI v7, this is also supported:

```
cf push APP-NAME -u HEALTH-CHECK-TYPE --app-start-timeout HEALTH-CHECK-TIMEOUT
```

Where:

- `APP-NAME` is the name of your app.
- `HEALTH-CHECK-TYPE` is the type of health check that you want to configure. Valid health check types are `port`, `process`, and `http`. For more information, see Health check types.
- `HEALTH-CHECK-TIMEOUT` is the amount of time allowed to elapse between starting an app and the first healthy response. For more information, see Health check timeouts.

For more information about the `cf push` command, see push in the Cloud Foundry CLI Reference Guide.

Note: The health check configuration that you provide with `cf push` overrides any configuration in the app manifest.

Configure health checks for an existing app

To configure a health check for an existing app or to add a custom HTTP endpoint, run:

```
cf set-health-check APP-NAME HEALTH-CHECK-TYPE --endpoint CUSTOM-HTTP-ENDPOINT
```

Where:

- `APP-NAME` is the name of your app.
- `HEALTH-CHECK-TYPE` is the type of health check that you want to configure. Valid health check types are `port`, `process`, and `http`. For more information, see Health check types .
- `CUSTOM-HTTP-ENDPOINT` is the custom HTTP endpoint that you want to add to the health check. An `http` health check defaults to using `/` as its endpoint unless you specify a custom

endpoint. For more information, see Health check HTTP endpoints.

For more information about the `cf set-health-check` command, see set-health-check in the Cloud Foundry CLI Reference Guide.

> **Note:** After you set the health check configuration of a deployed app with the `cf set-health-check` command, you must restart the app for the change to take effect.

> **Note:** You can also change the health check invocation timeout for an app. If you have installed cf CLI v6, use `cf v3-set-health-check`. If you have installed cf CLI v7, use `cf set-health-check`. This option also requires restarting the app. For more information, see Apps in the Cloud Foundry CLI Reference Guide.

Understanding health checks

Health check lifecycle

The following table describes how app health checks work.

Stage	Description
1	The app developer deploys an app to TAS for VMs.
2	When deploying the app, the developer specifies a health check type for the app and, optionally, a timeout. If the developer does not specify a health check type, then the monitoring process defaults to a `port` health check.
3	Cloud Controller stages, starts, and runs the app.
4	Based on the type specified for the app, Cloud Controller configures a health check that runs periodically for each app instance.
5	When Diego starts an app instance, the app health check runs every two seconds until a response indicates that the app instance is healthy or until the health check timeout elapses. The 2-second health check interval is not configurable.
6	When an app instance becomes healthy, its route is advertised, if applicable. Subsequent health checks are run every 30 seconds once the app becomes healthy. The 30-second health check interval is not configurable.
7	If a previously healthy app instance fails a health check, Diego considers that particular instance to be unhealthy. As a result, Diego stops and deletes the app instance, then reschedules a new app instance. This stoppage and deletion of the app instance is reported back to the Cloud Controller as a crash event.
8	When an app instance crashes, Diego immediately attempts to restart the app instance several times. After three failed restarts, TAS for VMs waits 30 seconds before attempting another restart. The wait time doubles each restart until the ninth restart, and remains at that duration until the 200th restart. After the 200th restart, TAS for VMs stops trying to restart the app instance.

Health check types

The following table describes the types of health checks available for apps and recommended circumstances in which to use them:

Health check type	Recommended use case	Explanation
`http`	The app can provide an `HTTP 200` response.	The `http` health check performs a GET request to the configured HTTP endpoint on the app's default port. When the health check receives an `HTTP 200` response, the app is declared healthy. VMware recommends that you use the `http` health check type whenever possible. A healthy HTTP response ensures that the web app is ready to serve HTTP requests. The configured endpoint must respond within one second to be considered healthy. ⚠ **Warning:** To prevent false negatives, use a dedicated endpoint for health checks where response time and result do not depend on business logic.
`port`	The app can receive TCP connections, including HTTP web apps.	A health check makes a TCP connection to the port or ports configured for the app. For apps with multiple ports, a health check monitors each port. If you do not specify a health check type for your app, then the monitoring process defaults to a `port` health check. The TCP connection must be established within one second to be considered healthy.
`process`	The app does not support TCP connections. An example of such an app is a worker.	For a `process` health check, Diego ensures that any process declared for the app stays running. If the process exits, Diego stops and deletes the app instance.

Health check timeouts

The value configured for the health check timeout is the amount of time allowed to elapse between starting an app and the first healthy response from the app. If the health check does not receive a healthy response within the configured timeout, then the app is declared unhealthy.

In Ops Manager, the default timeout is 60 seconds and the maximum configurable timeout is 600 seconds. You can modify the timeout in the **App Containers** pane of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.

Health check HTTP endpoints

Only used by `http` type, the `--endpoint` flag of the `cf set-health-check` command specifies the path portion of a URI that must be served by the app and return `HTTP 200` when the app is healthy.

This command only checks the health of the default port of the app.

Note: For HTTP apps, VMware recommends setting the health check type to `http` instead of a simple port check.

Cloud Foundry API app revisions

A revision represents code and configuration used by an app at a specific time. It is a Cloud Foundry API (CAPI) object that can contain references to a droplet, a custom start command, and environment variables. The most recent revision for a running app represents code and configuration currently running in Ops Manager.

For additional information, see Revisions in the Cloud Foundry API (CAPI) documentation.

> **Note:** CAPI v3 is the recommended API version for revisions. While revisions work with CAPI v2, there are several inconsistencies. For example, revision descriptions for apps with multiple processes may be inaccurate because CAPI v2 does not support apps with multiple processes. Additionally, pushing an app for the first time with revisions in CAPI v2 creates two revisions.

Caution: The app revisions API is experimental, and future releases might have breaking changes.

Revisions use cases

Some use cases for revisions include:

- **Viewing revisions for an app:** This can help you understand how your app has changed over time.
- **Rolling back to a previous revision:** This allows you to deploy a version of the app that you had running previously without needing to track that previous state yourself or have multiple apps running. When you create a deployment and reference a revision, the revision deploys as the current version of your app.

Events that trigger revisions

Revisions are generated automatically through these events:

- A new droplet is created for an app.
- An app's environment variables are changed.
- The custom start command for an app is added or changed.
- An app rolls back to a prior revision.

By default, CAPI retains a maximum of 100 revisions per app.

Revision descriptions

Each revision includes a description of what changed in your app at the time the revision was created. The description includes one or more of these descriptions:

- `Process type removed`
- `New process type added`
- `Rolled back to revision X`
- `Custom start command removed`
- `Custom start command updated`

- `Custom start command added`
- `New environment variables deployed`
- `New droplet deployed`

Droplet storage considerations

By default, Ops Manager retains the five most recent staged droplets in its droplets bucket. This means that you can roll back to revisions as long as they are using one of those five droplets. Not all revisions include a change in droplet.

Operators can configure Ops Manager to retain more droplets if necessary using the **Maximum staged droplets per app** field in the **File Storage** pane of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.

View revisions

This section describes how to use CAPI endpoints for viewing revisions.

List revisions for an app

To list revisions for an app:

1. Retrieve the GUID of the app by running:

   ```
   cf app APP-NAME --guid
   ```

   Where `APP-NAME` is the name of your app.

2. Run:

   ```
   cf curl /v3/apps/GUID/revisions
   ```

   Where `GUID` is the GUID you retrieved in the previous step.

List deployed revisions for an app

Deployed revisions are revisions linked to started processes in an app. To list deployed revisions:

1. Retrieve the GUID of the app by running:

   ```
   cf app APP-NAME --guid
   ```

   Where `APP-NAME` is the name of your app.

2. Run:

   ```
   cf curl /v3/apps/GUID/revisions/deployed
   ```

   Where `GUID` is the GUID you retrieved in the previous step.

Retrieve a revision

To retrieve a revision:

1. Run:

   ```
   cf curl /v3/revisions/GUID
   ```

   Where `GUID` is the GUID of the revision.

Roll back to a previous revision

To roll back to a previous revision:

1. Retrieve the GUID of the app by running:

   ```
   cf app APP-NAME --guid
   ```

   Where `APP-NAME` is the name of your app.

2. Retrieve the GUID of the revision. For more information, see Retrieve a Revision.

3. Create a deployment using CAPI by running:

   ```
   cf curl v3/deployments \
   -X POST \
   -d '{
     "revision": {
       "guid": "REVISION-GUID"
     },
     "relationships": {
       "app": {
         "data": {
           "guid": "APP-GUID"
         }
       }
     }
   }'
   ```

   Where:

   - `REVISION-GUID` is the GUID of the revision.
   - `APP-GUID` is the GUID of the app.

Add metadata to a revision

To add metadata to a revision, see Add Metadata to an Object.

Deactivate revisions for an app

CAPI enables app revisions by default. To disable revisions for an app, you must manually turn them off.

To disable revisions for an app:

1. Retrieve the GUID of the app by running:

   ```
   cf app APP-NAME --guid
   ```

Where `APP-NAME` is the name of your app.

2. Run:

```
cf curl /v3/apps/GUID/features/revisions -X PATCH -d '{ "enabled": false }'
```

Where `GUID` is the GUID you retrieved in the previous step.

Configuring container-to-container networking

- Configure the overlay network
- Create and manage networking policies
  - Prerequisites
  - Grant permissions
  - Add a network policy
  - List network policies
  - Remove a network policy
  - Disable network policy enforcement
  - Manage network policy quotas
- App service discovery
  - Enable app service discovery

The container-to-container networking feature allows direct network traffic between apps. For an overview of how Container-to-Container Networking works, see the Container-to-Container Networking topic.

> **Note:** Container-to-container networking is not available for apps hosted on Windows.

Container-to-container networking enables VMware Tanzu Application Service for VMs (TAS for VMs) to generate logs whenever containers communicate or attempt to communicate with each other. For more information about managing app traffic logging, see the App Traffic Logging section of the *Configuring Logging in TAS for VMs* topic.

Configure the Overlay Network

Container-to-container networking uses an overlay network to manage communication between app instances. By default, each Diego Cell in the overlay network is allocated a /24 range that supports 254 containers per Diego Cell, one container for each of the usable IP addresses, `.1` through `.254`. For more information about the overlay network, see Overlay Network in *Container-to-Container Networking*.

Configure the Number of Diego Cells

To modify the number of Diego Cells supported by the overlay network:

1. In Ops Manager, select the TAS for VMs tile.
2. Select **Networking**.
3. Under **Overlay Subnet**, enter an IP range for the overlay network. By default, Ops Manager uses `10.255.0.0/16`. Modifying the subnet range allocated to the overlay network changes the number of Diego Cells supported in your deployment. Use the table below as a reference.

Overlay subnet mask	Number of Diego Cells	Containers per Diego Cell
/20	15	254
/16	255	254
/12	4,095	254

> **Caution:** The overlay network IP address range must not conflict with any other IP addresses in the network. If a conflict exists, Diego Cells cannot reach any endpoint that has a conflicting IP address.

Create and manage networking policies

This section describes how to create and modify Container-to-Container Networking policies using the Cloud Foundry Command Line Interface (cf CLI).

> **Note:** You can also create and modify container-to-container networking policies using Apps Manager. For more information, see Create Container-to-Container Networking Policies in the *Managing Apps and Service Instances Using Apps Manager* topic.

> **Note:** With the NSX-T integration, container networking policies and ASGs continue to work as normal. Advanced ASG logging is not supported with NSX-T.

Prerequisites

Ensure that you are using cf CLI v6.42 or later:

```
$ cf version
```

For more information about updating the cf CLI, see the Installing the cf CLI topic.

Grant permissions

CF admins use the following UAA scopes to grant specific users or groups permissions to configure network policies:

UAA Scope	Suitable for...	Allows users to create policies...
`network.admin`	operators	for any apps in the CF deployment

`network.write`	space developers	for apps in spaces that they can access

If you are a CF admin, you already have the `network.admin` scope. An admin can also grant the `network.admin` scope to a space developer.

For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions.

To grant all Space Developers permissions to configure network policies, open the **App Developer Controls** pane in the TAS for VMs tile and enable the **Allow space developers to manage network policies** check box.

Add a network policy

To add a policy that allows direct network traffic from one app to another, run the following command:

```
cf add-network-policy SOURCE_APP DESTINATION_APP -s DESTINATION_SPACE_NAME -o DESTINAT
ION_ORG_NAME --protocol (tcp | udp) --port RANGE
```

Replace the placeholders in the above command as follows:

- `SOURCE_APP` is the name of the app that sends traffic.
- `DESTINATION_APP` is the name of the app that will receive traffic.
- `DESTINATION_SPACE_NAME` is the space of the destination app. The default is the targeted space.
- `DESTINATION_ORG_NAME` is the org of the destination app. The default is the targeted org.
- `PROTOCOL` is one of the following: `tcp` or `udp`.
- `RANGE` are the ports at which to connect to the destination app. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

The following example command allows access from the `frontend` app to the `backend` app over TCP at port 8080:

```
$ cf add-network-policy frontend backend --protocol tcp --port 8080
Adding network policy to app frontend in org my-org / space dev as admin...
OK
```

Space Developers can add up to 50 network policies per space. This limit does not apply to users with the `network.admin` scope.

List policies

You can list all the policies in your space, or just the policies for which a single app is the source:

- To list the all the policies in your space, run `cf network-policies`.

  ```
  $ cf network-policies
  ```

- To list the policies for an app, run `cf network-policies --source MY-APP`. Replace `MY-APP` with the name of your app.

```
$ cf network-policies --source example-app
```

The following example command lists policies for the app `frontend`:

```
$ cf network-policies --source frontend
Listing network policies in org my-org / space dev as admin...

source      destination   protocol   ports    destination space    dest
ination org
frontend    backend       tcp        8080     example-space        exam
ple-org
```

Remove a network policy

To remove a policy that allows direct network traffic from an app, run the following command:

```
cf remove-network-policy SOURCE_APP DESTINATION_APP -s DESTINATION_SPACE_NAME -o DESTI
NATION_ORG_NAME --protocol PROTOCOL --port RANGE
```

Replace the placeholders in the above command to match an existing policy, as follows:

- `SOURCE_APP` is the name of the app that sends traffic.
- `DESTINATION_APP` is the name of the app that receives traffic.
- `DESTINATION_SPACE_NAME` is the space of the destination app. The default is the targeted space.
- `DESTINATION_ORG_NAME` is the org of the destination app. The default is the targeted org.
- `PROTOCOL` is either `tcp` or `udp`.
- `PORTS` are the ports connecting the apps. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

The following command deletes the policy that allowed the `frontend` app to communicate with the `backend` app over TCP on port 8080:

```
$ cf remove-network-policy frontend backend --protocol tcp --port 8080
Removing network policy to app frontend in org my-org / space dev as admin...
OK
```

Disable network policy enforcement

You can disable Silk network policy enforcement between apps. Disabling network policy enforcement allows all apps to send network traffic to all other apps in the foundation despite no policy specifically allowing it.

To disable network policy enforcement between apps, do the following:

1. Go to the **Networking** pane of your TAS for VMs tile.
2. Disable the **Enable Silk Network Policy Enforcement** checkbox.
3. Click **Save**.

App service discovery

With app service discovery, apps pushed to TAS for VMs can establish container-to-container communications through a known route served by internal BOSH DNS. This allows front end apps to easily connect with back end apps.

> **Note**: The internal domain used for service discovery is `apps.internal` by default. Operators can modify this value in **App Developer Controls** pane of the TAS for VMs tile.

To establish container-to-container communications between a front end and back end app, a developer:

1. Launches a back end app that publishes a local endpoint.
2. Maps a named route to the endpoint.
3. Creates a network policy that allows direct traffic from the front end to the back end app.
4. Launches the front end app.

See Cats and Dogs with Service Discovery in GitHub for an example, written in Go, that demonstrates communication between front end and back end apps.

Managing Services

In this section:

- **Service Brokers**
  - Example Service Brokers
  - Binding Credentials
  - Enabling Service Instance Sharing
  - App Log Streaming
  - Route Services
  - Supporting Multiple TAS for VMs Instances
- Managing Service Instances with the cf CLI
- Sharing Service Instances
- Delivering Service Credentials to an App
- Managing Service Keys
- Managing App Requests with Route Services
- Configuring Play Framework Service Connections
- Using an External File System (Volume Services)
- User-Provided Service Instances

Service Brokers

In this section:

- Example Service Brokers
- Binding Credentials
- Enabling Service Instance Sharing
- App Log Streaming
- Route Services

Example Service Brokers

The following example service broker applications have been developed - these are a great starting point if you are developing your own service broker.

Ruby

- GitHub Repository service - this is designed to be an easy-to-read example of a service broker, with complete documentation, and comes with a demo app that uses the service. The broker can be deployed as an application to any Cloud Foundry instance or hosted elsewhere. The service broker uses GitHub as the service back end.
- MySQL database service - this broker and its accompanying MySQL server are designed to be deployed together as a BOSH release. BOSH is used to deploy or upgrade the release, monitors the health of running components, and restarts or recreates unhealthy VMs. The broker code alone can be found here.

Java

- Spring Cloud - Cloud Foundry Service Broker - This implements the REST contract for service brokers and the artifacts are published to the Spring Maven repository. This greatly simplifies development: include a single dependency in Gradle, implement interfaces, and configure. A sample implementation has been provided for MongoDB.
- MySQL Java Broker - a Java port of the Ruby-based MySQL broker above.

Go

- Asynchronous Service Broker for AWS EC2 - This broker implements support for Asynchronous Service Operations, and calls AWS APIs to provision EC2 VMs.

Binding Credentials

A bindable service returns credentials that an application can consume in response to the `cf bind` API call. Cloud Foundry writes these credentials to the `VCAP_SERVICES` environment variable. In some cases, buildpacks write a subset of these credentials to other environment variables that frameworks might need.

Choose from the following list of credential fields if possible, though you can provide additional fields as needed. Refer to the Using Bound Services section of the *Managing Service Instances with* *the CLI* topic for information about how these credentials are consumed.

> **Note**: If you provide a service that supports a connection string, provide the `uri` key for buildpacks and application libraries to use.

CREDENTIALS	DESCRIPTION
uri	Connection string of the form `DB-TYPE://USERNAME:PASSWORD@HOSTNAME:PORT/NAME`, where `DB-TYPE` is a type of database such as mysql, postgres, mongodb, or amqp.
hostname	FQDN of the server host
port	Port of the server host
name	Name of the service instance
vhost	Name of the messaging server virtual host - a replacement for a `name` specific to AMQP providers
username	Server user
password	Server password

The following is an example output of `ENV['VCAP_SERVICES']`.

> **Note**: Depending on the types of databases you are using, each database might return different credentials.

```
VCAP_SERVICES=
{
  cleardb: [
    {
      name: "cleardb-1",
      label: "cleardb",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4
446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:330
6/ad_c6f4446532610ab"
      }
    }
  ],
  cloudamqp: [
    {
      name: "cloudamqp-6",
      label: "cloudamqp",
      plan: "lemur",
      credentials: {
        uri: "amqp://ksvyjmiv:IwN6dCdZmeQD4O0ZPKpu1YOaLx1he8wo@lemur.cloudamqp.com/ksv
yjmiv"
      }
    }
    {
      name: "cloudamqp-9dbc6",
      label: "cloudamqp",
      plan: "lemur",
      credentials: {
        uri: "amqp://vhuklnxa:9lNFxpTuJsAdTts98vQIdKHW3MojyMyV@lemur.cloudamqp.com/vhu
klnxa"
      }
    }
  ],
  rediscloud: [
    {
      name: "rediscloud-1",
      label: "rediscloud",
      plan: "20mb",
      credentials: {
        port: "6379",
        host: "pub-redis-6379.us-east-1-2.3.ec2.redislabs.com",
        password: "1M5zd3QfWi9nUyya"
      }
    },
  ],
}
```

Enabling Service Instance Sharing

This topic provides information about enabling service instance sharing in managed services.

Overview

Service authors can allow instances of their services to be shared across spaces and orgs within Cloud Foundry. This enables apps running in different spaces and orgs to use the same service instance. Developers with Space Developer permissions in the space where the service instance was created are responsible for sharing, unsharing, updating, and deleting the service instance.

For more information about the developer and administrator tasks related to service instance sharing, see Sharing Service Instances.

Enabling Service Instance Sharing

Service brokers must explicitly enable service instance sharing by setting a flag in their service-level metadata object. This allows service instances, of any service plan, to be shared across orgs and spaces. The `"shareable"` flag must be set to `true` in the service-level metadata to enable service instance sharing. If the flag is set to `false` or is absent, sharing is disabled.

An example catalog is shown below:

```
{
   "services":[{
      "id":"766fa866-a950-4b12-adff-c11fa4cf8fdc",
      "name": "example-service",
      "metadata": {
         "shareable": true
```

```
        }
    }]
}
```

Binding Permissions Based on Instance Sharing

When a service instance is created in one space and shared into another, developers can bind their apps to the service instance from both spaces.

You may want to have the service broker return credentials with different permissions depending on which space an app is bound from. For example, a messaging service may permit writes from the originating space and only reads from any spaces that the service is shared into.

To determine whether the space of the app is the same as the originating space of the service instance, the service broker can compare the `context.space_guid` and `bind_resource.space_guid` fields in the binding request. The `context.space_guid` field represents the space where the service instance was created, and `bind_resource.space_guid` represents the space of the app involved in the binding.

Security Considerations

Service authors must consider the following before enabling service instance sharing:

- Service keys can only be generated by users who have access to the space where the service instance was created. This ensures that only developers with access to this space have visibility into where and how many times the service instance is used. It is not possible to distinguish between service keys created when targeting the originating space and those created when targeting the space where the instance has been shared to. Also, unsharing an instance from a space will not delete any service keys.
- Consider the impact of giving out excessive permissions for service bindings, especially bindings that originate from spaces that the service instance has been shared into. For example, a messaging service may permit writes from the originating space and only reads from any shared spaces. For more information, see Binding Permissions Based on Instance Sharing.
- You should generate unique credentials for each binding. This ensures that developers can unshare a service instance at any time. Unsharing an instance deletes any service bindings and revokes access for those credentials. Unsharing an instance prevents unauthorized future access from developers and apps that saved the credentials they were previously provided using the service binding.
- Consider the impact of a service instance dashboard being accessed by users of shared service instances. If authenticating through SSO, see Dashboard Single Sign-On.

App Log Streaming

By binding an application to an instance of an applicable service, Cloud Foundry will stream logs for the bound application to the service instance.

- Logs for all apps bound to a log-consuming service instance will be streamed to that instance

- Logs for an app bound to multiple log-consuming service instances will be streamed to all instances

To enable this functionality, a service broker must implement the following:

1. In the catalog endpoint, the broker must include `requires: syslog_drain`. This minor security measure validates that a service returning a `syslog_drain_url` in response to the bind operation has also declared that it expects log streaming. If the broker does not include `requires: syslog_drain`, and the bind request returns a value for `syslog_drain_url`, Cloud Foundry will return an error for the bind operation.
2. In response to a bind request, the broker should return a value for `syslog_drain_url`. The syslog URL has a scheme of syslog, syslog-tls, or https and can include a port number. For example:

   ```
   "syslog_drain_url": "syslog://logs.example.com:1234"
   ```

How does it work?

1. Service broker returns a value for `syslog_drain_url` in response to bind
2. Loggregator periodically polls an internal Cloud Controller endpoint for updates
3. Upon discovering a new `syslog_drain_url`, Loggregator identifies the associated app
4. Loggregator streams app logs for that app to the locations specified by the service instances' `syslog_drain_url`s

Users can manually configure app logs to be streamed to a location of their choice using User-provided Service Instances. For details, see Using Third-Party Log Management Services.

Route Services

This topic provides guidance for offering a service to a VMware Tanzu Application Service for VMs (TAS for VMs) services marketplace. For information about consuming these services, see Manage App Requests with Route Services.

Introduction

TAS for VMs app developers may wish to apply transformation or processing to requests before they reach an app. Common examples of use cases include authentication, rate limiting, and caching services. Route services are a kind of Marketplace service that developers can use to apply various transformations to app requests by binding an app's route to a service instance. Through integrations with service brokers and, optionally, with the TAS for VMs routing tier, providers can offer these services to developers with a familiar, automated, self-service, and on-demand user experience.

> **Note:** The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also manage route services using Apps Manager. For more information, see the Manage Route Services section of the *Managing Apps and Service Instances Using Apps Manager* topic.

Architecture

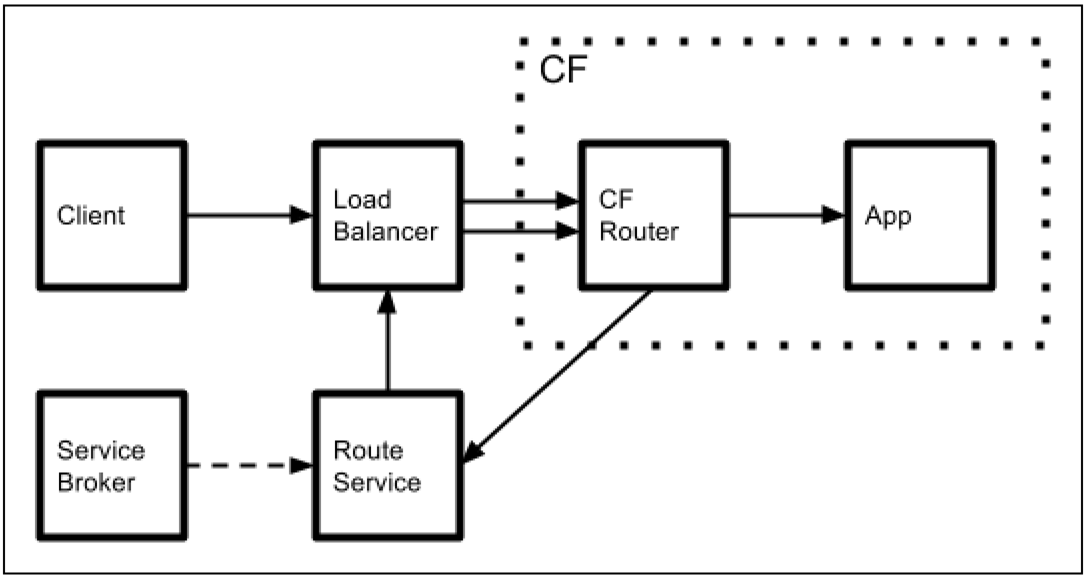
TAS for VMs supports the following three models for route services:

- Fully-brokered services
- Static, brokered services
- User-provided services

In each model, you configure a route service to process traffic addressed to an app.

Fully-Brokered Service

In the fully-brokered service model, the TAS for VMs router receives all traffic to apps in the deployment before any processing by the route service. Developers can bind a route service to any app, and if an app is bound to a route service, the TAS for VMs router sends its traffic to the service. After the route service processes requests, it sends them back to the load balancer in front of the TAS for VMs router. The second time through, the TAS for VMs router recognizes that the route service has already handled them, and forwards them directly to app instances.



The route service can run inside or outside of TAS for VMs, so long as it fulfills the Service Instance Responsibilities to integrate it with the TAS for VMs router. A service broker publishes the route service to the TAS for VMs marketplace, making it available to developers. Developers can then create an instance of the service and bind it to their apps with the following commands:

```
cf create-service BROKER-SERVICE-PLAN SERVICE-INSTANCE
```

```
cf bind-route-service YOUR-APP-DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME] [--path P
ATH]
```

Developers configure the service either through the service provider's web interface or by passing arbitrary parameters to their `cf create-service` call through the `-c` flag.

Advantages:

- Developers can use a Service Broker to dynamically configure how the route service processes traffic to specific apps.

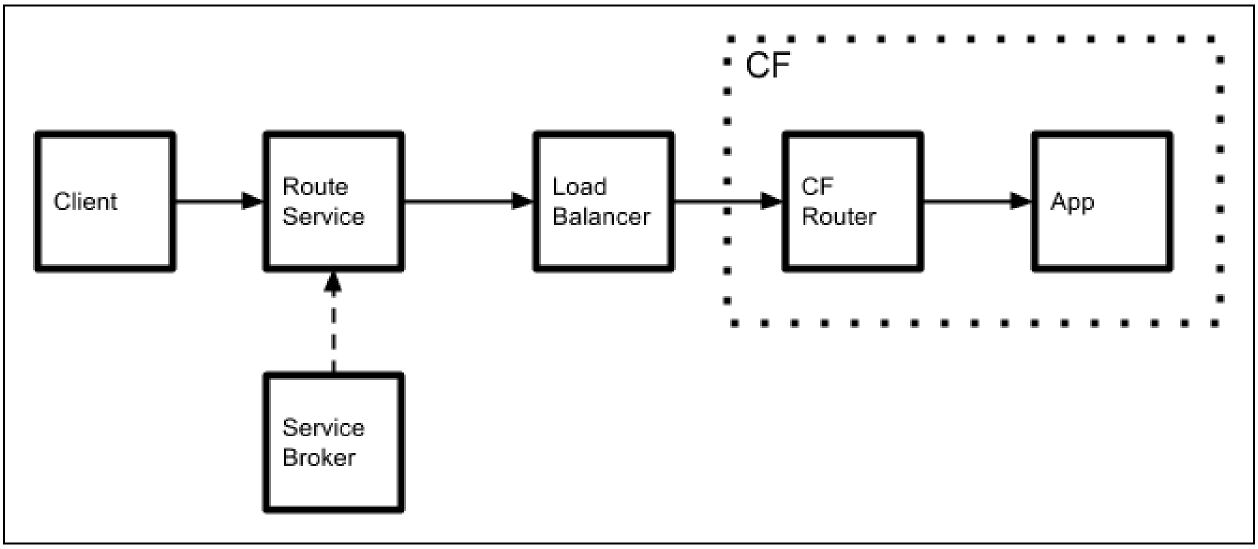
- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

Disadvantages:

- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Static, Brokered Service

In the static, brokered service model, an operator installs a static routing service, which might be a piece of hardware, in front of the load balancer. The routing service runs outside of TAS for VMs and receives traffic to all apps running in the TAS for VMs deployment. The service provider creates a service broker to publish the service to the TAS for VMs marketplace. As with a fully-brokered service, a developer can use the service by instantiating it with `cf create-service` and binding it to an app with `cf bind-route-service`.



In this model, you configure route services on an app-by-app basis. When you bind a service to an app, the service broker directs the routing service to process that app’s traffic rather than pass the requests through unchanged.

Advantages:

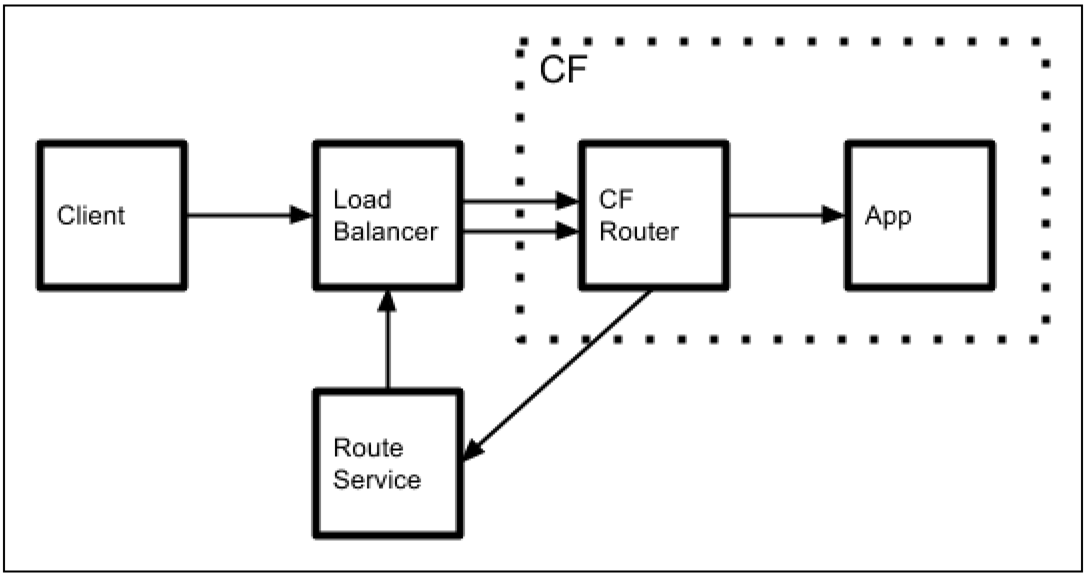
- Developers can use a service broker to dynamically configure how the route service processes traffic to specific apps.
- Traffic to apps that use the route service takes fewer network hops.

Disadvantages:

- Adding route services requires manual infrastructure configuration.
- Traffic to apps that do not use the route service make unnecessary network hops. Requests for all apps hosted by the deployment pass through the route service component.

User-Provided Service

If a route service is not listed in the TAS for VMs marketplace by a broker, a developer can still bind it to their app as a user-provided service. The service can run anywhere, either inside or outside of TAS for VMs, but it must fulfill the integration requirements described in Service Instance Responsibilities. The service also needs to be reachable by an outbound connection from the TAS for VMs router.



This model is identical to the fully-brokered service model, except without the broker. Developers configure the service manually, outside of TAS for VMs. They can then create a user-provided service instance and bind it to their app with the following commands, supplying the URL of their route service:

```
cf create-user-provided-service SERVICE-INSTANCE -r ROUTE-SERVICE-URL
```

```
cf bind-route-service DOMAIN SERVICE-INSTANCE [--hostname HOSTNAME]
```

Advantages:

- Adding route services requires no manual infrastructure configuration.
- Traffic to apps that do not use the service makes fewer network hops because requests for those apps do not pass through the route service.

Disadvantages:

- Developers must manually provision and configure route services out of the context of TAS for VMs because no service broker automates these operations.
- Traffic to apps that use the route service makes additional network hops, as compared to the static model.

Architecture Comparison

The models above require the broker and service instance responsibilities summarized in the following table:

Route Services Architecture	Fulfills TAS for VMs Service Instance Responsibilities	Fulfills TAS for VMs Broker Responsibilities
Fully-Brokered	Yes	Yes
Static Brokered	No	Yes
User-Provided	Yes	No

Enabling Route Services in TAS for VMs

You configure Route Services for your deployment in the TAS for VMs tile, in the **Networking** pane.

Service Instance

The following applies only when a broker returns `route_service_url` in the bind response.

How It Works

Binding a service instance to a route associates the `route_service_url` with the route in the TAS for VMs router. All requests for the route are proxied to the URL specified by `route_service_url`.

The TAS for VMs router includes the `X-CF-Forwarded-Url` header containing the originally requested URL, as well as the `X-CF-Proxy-Signature` and `X-CF-Proxy-Metadata` headers used by the router to validate that the route-service sent the request. These headers are described in the Headers section, below.

Service Instance Responsibilities

The route service must handle requests by either:

- Accepting the request by making a new request to the originally requested URL, or to another location, and then responding to the originating requestor.
- Rejecting the request by responding with a non `2xx` HTTP status code.

When forwarding a request to the originally requested URL, the route service must forward the `X-CF-Forwarded-Url`, `X-CF-Proxy-Signature` and `X-CF-Proxy-Metadata` headers on the request or it will be rejected. When forwarding a request to a location other than the originally requested URL, the route service should strip these headers.

Headers

The following HTTP headers are added by the Gorouter to requests forwarded to route services.

X-CF-Forwarded-Url

The `X-CF-Forwarded-Url` header contains the originally requested URL. The route service may choose to forward the request to this URL or to another.

X-CF-Proxy-Signature

When the Gorouter receives a request with this header, it will accept and forward the request to the app only if the URL of the request matches the one associated with the token, and if the request was received on time. Otherwise, the request is rejected.

X-CF-Proxy-Signature also signals to the Gorouter that a request has transited a route service. If this header is present, the Gorouter does not forward the request to a route service. The route-service needs to forward these headers in subsequent requests to the orignally requested URL, so that it knows not to send the request back to the route service but to the app. The headers should NOT be sent in the HTTP reponse to the GoRouter, only in the new HTTP request to the GoRouter.

The X-CF-Proxy-Signature header is an **access token**. Anyone possessing your X-CF-Proxy-Signature token can bypass the route service. Treat this token as a secret that cannot be shared with anyone.

CF-hosted Route Services cannot be chained: If the route service forwards the request to a URL which resolves to a route for a different app on TAS for VMs, the route must not have a bound route service. Otherwise, the request is rejected as the requested URL does not match the one in the forwarded X-CF-Proxy-Signature header.

X-CF-Proxy-Metadata

The X-CF-Proxy-Metadata header aids in the encryption and description of X-CF-Proxy-Signature.

SSL Certificates

When TAS for VMs is deployed in a development environment, certificates hosted by the load balancer are self-signed, and not signed by a trusted Certificate Authority. When the route service finishes processing an inbound request and makes a call to the value of X-CF-Forwarded-Url, be prepared to accept the self-signed certificate when integrating with a non-production deployment of TAS for VMs.

Timeouts

Route services must forward the request to the app route within 60 seconds.

In addition, all requests must respond in 900 seconds.

Broker Responsibilities

Catalog Endpoint

Brokers must include requires: ["route_forwarding"] for a service in the catalog endpoint. If this is not present, TAS for VMs does not permit users to bind an instance of the service to a route.

Binding Endpoint

When users bind a route to a service instance, TAS for VMs sends a bind request to the broker, including the route address with bind_resource.route. A route is an address used by clients to reach apps mapped to the route. The broker may return route_service_url, containing a URL where TAS for VMs should proxy requests for the route. This URL must have an https scheme, or the Cloud Controller rejects the binding. `route_service_url` is optional, and not returning this field enables a broker to dynamically configure a network component already in the request path for the route, requiring no change in the TAS for VMs router.

For more information about bind requests, see the Binding section of the *Open Service Broker API (v2.13)* spec on GitHub.

Example Route Services

- Logging Route Service: This route service can be pushed as an app to TAS for VMs. It fulfills the service instance responsibilities above and logs requests received and sent. It can be used to see the route service integration in action by tailing its logs.
- Rate Limiting Route Service: This example route service is a simple TAS for VMs app that provides rate limiting to control the rate of traffic to an app.
- [Spring Boot examples] (https://github.com/in28minutes/spring-boot-examples).

Tutorial

The following instructions show how to use the Logging Route Service described in Route Services examples to verify that when a route service is bound to a route, requests for that route are proxied to the route service.

For a video of this tutorial, see Route Services in Pivotal Cloud Foundry 1.7 on YouTube.

These commands requires the Cloud Foundry Command Line Interface (cf CLI) v6.16 or later.

To use the logging route service:

1. Push the logging route service as an app by running:

   ```
   cf push logger
   ```

2. Create a user-provided service instance, and include the route of the logging route service you pushed as `route_service_url`. Be sure to use `https` for the scheme. Run:

   ```
   cf create-user-provided-service mylogger -r https://logger.cf.example.com
   ```

3. Push a sample app like Spring Music. By default, this creates a route `spring-music.cf.example.com`. Run:

   ```
   cf push spring-music
   ```

4. Bind the user-provided service instance to the route of your sample app. The `bind-route-service` command takes a route and a service instance; the route is specified in the following example by domain `cf.example.com` and hostname `spring-music`. Run:

   ```
   cf bind-route-service cf.example.com mylogger --hostname spring-music
   ```

5. Tail the logs for your route service by running:

   ```
   cf logs logger
   ```

6. Send a request to the sample app and view in the route service logs that the request is forwarded to it by running:

   ```
   curl spring-music.cf.example.com
   ```

Supporting Multiple TAS for VMs Instances

This topic describes registering a service broker with multiple VMware Tanzu Application Service for VMs (TAS for VMs) instances.

Overview

It may be necessary for the broker to know which TAS for VMs instance is making a given request. For example, when using Dashboard Single Sign-On, the broker is expected to interact with the authorization and token endpoints for a given TAS for VMs instance.

There are two strategies that can be used to discover which TAS for VMs instance is making a given request.

Routing and Authentication

The broker can use unique credentials, a unique URL, or both for each TAS for VMs instance. When registering the broker, you can configure different TAS for VMs instances to use different base URLs that include a unique ID. For example:

- On TAS for VMs instance 1, the service broker is registered with the URL `broker.example.com/123`.
- On TAS for VMs instance 2, the service broker is registered with the URL `broker.example.com/456`.

X-Api-Info-Location Header

All calls to the broker from TAS for VMs include an `X-Api-Info-Location` header containing the `/v2/info` URL for that instance. The `/v2/info` endpoint returns further information, including the location of that TAS for VMs instance's UAA.

Support for this header was introduced in cf-release v212.

Managing service instances with the cf CLI

You can manage lifecycle operations for service instances using the Cloud Foundry CLI (cf CLI). This includes creating, updating, and deleting service instances. For documentation about other service management operations, see Services Overview. If you are interested in building services for Ops Manager and making them available to end users, see Services.

To run the commands in this topic, you must first install the Cloud Foundry Command Line Interface (cf CLI). See the Cloud Foundry Command Line Interface topics for more information.

List marketplace services

After targeting and logging into Cloud Foundry, to view the services available to your targeted organization, run the cf CLI command:

```
cf marketplace
```

Available services may differ between organizations and between Cloud Foundry marketplaces.

```
$ cf marketplace
Getting services from marketplace in org my-org / space test as user@example.
com...
OK

service             plans               description                       brok
er
p-mysql             100mb, 1gb          A DBaaS                           mysq
l-broker
p-riakcs            developer           An S3-compatible object store     obje
ct-store-broker
```

Creating service instances

You can create a service instance with the following command:

```
cf create-service SERVICE PLAN SERVICE_INSTANCE
```

Use the information in the list below to replace `SERVICE`, `PLAN`, and `SERVICE_INSTANCE` with appropriate values.

- `SERVICE`: The name of the service you want to create an instance of.
- `PLAN`: The name of a plan that meets your needs. Service providers use **plans** to offer varying levels of resources or features for the same service.
- `SERVICE_INSTANCE`: The name you provide for your service instance. You use this name to refer to your service instance with other commands. Service instance names can include alpha-numeric characters, hyphens, and underscores, and you can rename the service instance at any time.

```
$ cf create-service rabbitmq small-plan my-rabbitmq

Creating service my-rabbitmq in org console / space development as user@examp
le.com...
OK
```

User Provided Service Instances provide a way for developers to bind apps with services that are not available in their Cloud Foundry marketplace. For more information, see the User Provided Service Instances topic.

Note: When multiple brokers provide two or more services with the same name, you must specify the broker by including the `-b BROKER` flag in the `cf create-`

service command.

Arbitrary parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support providing additional configuration parameters with the provision request. Pass these parameters in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see the documentation for the particular service offering.

Example providing service-specific configuration parameters in-line:

```
$ cf create-service my-db-service small-plan my-db -c '{"storage_gb":4}'

Creating service my-db in org console / space development as user@example.co
m...
OK
```

Example providing service-specific configuration parameters in a file:

```
$ cf create-service my-db-service small-plan my-db -c /tmp/config.json

Creating service my-db in org console / space development as user@example.co
m...
OK
```

Instance tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the VCAP_SERVICES Environment Variable. These tags provide developers with a more generic way for apps to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including the `-t` flag followed by a comma-separated list of tags.

Example providing a comma-separated list of tags:

```
$ cf create-service my-db-service small-plan my-db -t "prod, workers"

Creating service my-db in org console / space development as user@example.co
m...
OK
```

List service instances

To list the service instances in your targeted space, run:

```
cf services
```

The output from running this command includes any bound apps and the state of the last requested operation for the service instance.

```
$ cf services
Getting services in org my-org / space test as user@example.com...
OK

name       service       plan        bound apps   last operation      broker
upgrade available
mybucket   p-riakcs      developer   myapp        create succeeded    object-
store-broker   no
mydb       p-mysql       100mb                    create succeeded    mysql-b
roker          yes
```

Get details for a particular service instance

Details include dashboard urls, if applicable, and operation start and last updated timestamps.

```
$ cf service mydb

service instance:       mydb
service:                p-mysql
plan:                   100mb
description:            mysql databases on demand
documentation url:
dashboard:              https://p-mysql.example.com/manage/instances/abcd-ef1
2-3456
service broker:         mysql-broker

This service is not currently shared.

Showing status of last operation from service mydb...

status:    create succeeded
message:
started:   2019-02-13T12:02:19Z
updated:   2019-02-13T12:02:19Z

There are no bound apps for this service.
```

Bind a service instance

Depending on the service, you can bind service instances to apps and/or routes.

Not all services support binding, as some services deliver value to users directly without integration with Cloud Foundry, such as SaaS apps.

Bind a service instance to an app

Depending on the service, binding a service instance to your app may deliver credentials for the service instance to the app. See the Delivering Service Credentials to an App topic for more information. Binding a service instance to an app may also trigger app logs to be streamed to the service instance. For more information, see Streaming App Logs to Log Management Services.

```
$ cf bind-service my-app mydb
Binding service mydb to my-app in org my-org / space test as user@example.com...
OK
TIP: Use 'cf push' to ensure your env variable changes take effect

$ cf restart my-app
```

Note: You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Binding with app manifest

As an alternative to binding a service instance to an app after pushing an app, you can use the app manifest to bind the service instance during push. As of cf CLI v6.12.1, Arbitrary Parameters are not supported in app manifests. Using the manifest to bind service instances to routes is also not supported.

The following excerpt from an app manifest binds a service instance called `test-mysql-01` to the app on push.

```
services:
 - test-mysql-01
```

The following excerpt from the `cf push` command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called `test-msg-app`.

```
$ cf push
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

...

Binding service test-mysql-01 to test-msg-app in org My-Org / space development as user@example.com
OK
```

For more information about app manifests, see Deploying with App Manifests.

Bind a service instance to a route

Binding a service instance to a route will cause app requests and responses to be proxied through the service instance, where it may be used to transform or intermediate requests. For more information, see Managing App Requests with Route Services.

```
$ cf bind-route-service shared-domain.example.com --hostname my-app my-service-instance
```

```
Binding route my-app.shared-domain.example.com to service instance my-service
-instance in org my-org / space test as user@example.com...
OK
```

Restaging your app is not required.

Arbitrary parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf bind-service rails-sample my-db -c '{"role":"read-only"}'

Binding service my-db to app rails-sample in org console / space development
as user@example.com...
OK
```

```
$ cf bind-service rails-sample my-db -c /tmp/config.json

Binding service my-db to app rails-sample in org console / space development
as user@example.com... OK
```

Unbind a service instance

Unbind a service instance from an app

Unbinding a service instance from an app removes the credentials created for your app from the VCAP_SERVICES environment variable.

```
$ cf unbind-service my-app mydb
Unbinding app my-app from service mydb in org my-org / space test as user@exa
mple.com...
OK
```

Note: You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Unbind a service instance from a route

Unbinding a service instance from a route will result in requests and responses no longer being proxied through the service instance. For more information, see Managing App Requests with Route Services.

> **Note**: If your bound service instance is providing security features, like authorization, unbinding the service instance may leave your app vulnerable.

```
$ cf unbind-route-service shared-domain.example.com --hostname my-app my-service-instance

Unbinding may leave apps mapped to route my-app.shared-domain.example.com vulnerable; e.g. if service instance my-service-instance provides authentication. Do you want to proceed?> y

Unbinding route my-app.shared-domain.example.com from service instance my-service-instance n org my-org / space test as user@example.com...
OK
```

Restaging your app is not required.

Rename a service instance

You can change the name given to a service instance. Keep in mind that upon restarting any bound apps, the name of the instance will change in the VCAP_SERVICES environment variable. If your app depends on the instance name for discovering credentials, changing the name could break your app's use of the service instance.

```
$ cf rename-service mydb mydb1
Renaming service mydb to mydb1 in org my-org / space test as user@example.com...
OK
```

Update a service instance

Changing a service plan

Changing a plan requires cf CLI v6.7+ and cf-release v192+. CLI v7 or v8 is recommended.

By updating the service plan for an instance, users can change their service instance to other service plans. Though the platform and CLI now support this feature, services must expressly implement support for it so not all services will. Further, a service might support updating between some plans but not others. For instance, a service might support updating a plan where only a logical change is required, but not where data migration is necessary. In either case, users can expect to see a meaningful error when plan update is not supported.

```
$ cf update-service mydb -p new-plan
Updating service instance mydb as user@example.com...
OK
```

Arbitrary parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the update request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf update-service mydb -c '{"storage_gb":4}'

Updating service instance mydb as me@example.com...
```

```
$ cf update-service mydb -c /tmp/config.json

Updating service instance mydb as user@example.com...
```

Instance tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the VCAP_SERVICES Environment Variable. These tags provide developers with a more generic way for apps to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including a comma-separated list of tags with the `-t` flag.

```
$ cf update-service my-db -t "staging, web"

Updating service my-db in org console / space development as user@example.co
m...
OK
```

Upgrade a service instance

Upgrading a Service Instance requires cf CLI v6.46.0+ and CAPI release 1.83.0+

Some service brokers support upgrading service instances to the latest version of a service plan. For example, a broker may want to provide a way for users of the service to upgrade the underlying operating system that their service instances run on.

To upgrade your service instances, do the following:

1. Confirm that an upgrade is available by running `cf services` and reviewing the `upgrade available` column. For example:

    ```
    $ cf services
    Getting services in org acceptance / space dev as admin...

    name      service   plan     bound apps   last operation     broker
    upgrade available
    mydb      p-mysql   small                 create succeeded   mysql-brok
    er   yes
    otherdb   p-mysql   medium                create succeeded   mysql-brok
    er   no
    ```

2. Upgrade the service instance using the `--upgrade` flag:

   ```
   cf update-service SERVICE-INSTANCE --upgrade
   ```

   For example:

   ```
   $ cf update-service mydb --upgrade
   You are about to update mydb.
   Warning: This operation may be long running and will block further operations on the service until complete.
   Really update service mydb? [yN]: y
   OK
   ```

Delete a service instance

Deleting a service instance deprovisions the service instance and deletes all data associated with the service instance.

```
$ cf delete-service mydb

Are you sure you want to delete the service mydb ? y
Deleting service mydb in org my-org / space test as user@example.com...
OK
```

Sharing Service Instances

This topic explains how to use service instance sharing.

About Service Instance Sharing

Sharing a service instance between spaces allows apps in different spaces to share databases, messaging queues, and other types of services. This eliminates the need for development teams to use service keys and user-provided services to bind their apps to the same service instance that was provisioned using the `cf create-service` command. Sharing service instances improves security, auditing, and provides a more intuitive user experience.

- Service instances can be shared into multiple spaces and across orgs.
- Developers and administrators can share service instances between spaces in which they have the Space Developer role.
- Developers who have a service instance shared with them can only bind and unbind apps to that service instance. They cannot update, rename, or delete it.
- Developers who have a service instance shared with them can view the values of any configuration parameters that were used to provision or update the service instance.

For example, if two development teams have apps in their own spaces, and both of those apps want to send messages to each other using a messaging queue, you can do the following:

1. The development team in space A can create a new instance of a messaging queue service, bind it to their app, and share that service instance into space B.

2. A developer in space B can then bind their app to the same service instance, and the two apps can begin publishing and receiving messages from one another.

Enabling Service Instance Sharing in Cloud Foundry

To enable service instance sharing, the platform operator must enable the `service_instance_sharing` flag in Cloud Foundry.

```
$ cf enable-feature-flag service_instance_sharing
```

Sharing a Service Instance

You can share a service instance from one space to another if you have the Space Developer role in both spaces.

To share a service instance to another space, run the following Cloud Foundry Command Line Interface (cf CLI) command:

```
$ cf share-service SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```

- You cannot share a service instance into a space where a service instance with the same name already exists.
- To share a service instance into a space, the space must have access to the service and service plan of the service instance that you are sharing. Run the `cf enable-service-access` command to set this access.
- If you no longer have access to the service or service plan used to create your service instance, you cannot share that service instance.

Get Information About Service Instance Sharing

To get information about service instance sharing on the originating space, run the following cf CLI command from that originating space:

```
cf service SERVICE-INSTANCE-NAME
```

For example:

```
$ cf service mydb

name:             mydb
service:          p-mysql
tags:
plan:             100mb
description:      mysql databases on demand been created by a platform.
documentation:
dashboard:
service broker:   mysql-broker

shared with spaces:
```

```
org         space         bindings
org-1       space-1       0
org-1       space-2       2

Showing status of last operation from service my-service-3...

status:     create succeeded
message:    Operation succeeded
started:    2021-03-01T15:10:15Z
updated:    2021-03-01T15:10:17Z

bound apps:
name                 binding name    status              message
my-music-app                         create succeeded    Operation succeeded

There is no upgrade available for this service.
```

- For each of the spaces, the service instance is shared into the output. The output shows the number of bindings to apps of the service instance in that space.
- When the service instance is not shared, `This service is not currently shared` appears instead of `shared with spaces`.

If you run the command while targeting the space where the service instance has been shared, you see the originating space and organization.

For example:

```
$ cf service mydb

name:                          mydb
shared from org/space:         acceptance / dev
service:                       p-mysql
tags:
plan:                          100mb
description:                   mysql databases on demand been created by a platfor
m.
documentation:
dashboard:
service broker:                mysql-broker

Showing status of last operation from service my-service-3...

status:     create succeeded
message:    Operation succeeded
started:    2021-03-01T15:10:15Z
updated:    2021-03-01T15:10:17Z

bound apps:
name                 binding name    status              message
my-library-app                       create succeeded    Operation succeeded
```

```
There is no upgrade available for this service.
```

In this case, no information about other spaces is exposed.

Unsharing a Service Instance

⚠ **Caution:** Unsharing a service instance automatically deletes all bindings to apps in the spaces it was shared into. This may cause apps to fail. Before unsharing a service instance, run the `cf service SERVICE-INSTANCE` command to see how many bindings exist in the spaces the service instance is shared into.

You can unshare a service instance if you have the Space Developer role in the space where this service instance was shared from.

Developers cannot delete or rename a service instance until it is unshared from all spaces.

To unshare a service instance, run the following cf CLI command:

```
$ cf unshare-service SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG] [-f]
```

The optional `-f` flag forces unsharing without confirmation.

Security Considerations

- Service keys cannot be created from a space that a service instance has been shared into. This ensures that developers in the space where a service instance has been shared from have visibility into where and how many times the service instance is used.
- Sharing service instances does not automatically update app security groups (ASGs). The network policies defined in your ASGs may need to be updated to ensure that apps using shared service instances can access the underlying service.
- Access to a service must be enabled using the `cf enable-service-access` command for a service instance to be shared into a space.
- Not all services are enabled for sharing instances functionality. Contact the service vendor directly if you are unable to share instances of their service. If you are a service author, see Enabling Service Instance Sharing.

Disabling Service Instance Sharing in Cloud Foundry

To disable service instance sharing, run the following command:

```
$ cf disable-feature-flag service_instance_sharing
```

This only prevents new shares from being created. To remove existing shares, see Deleting All Shares.

Deleting All Shares

The script below finds all service instances that are shared, and for each space that the service instance is shared into, all service bindings to that service instance are deleted, and all shares are deleted.

If a service binding is not successfully deleted, the script continues trying to unshare subsequent service instances.

To use this script, you must be logged in as an administrator and have jq installed.

> **Note**: This script has been tested on macOS Sierra 10.12.4 and Ubuntu 14.04.5. Use the script at your own risk.

```
#!/usr/bin/env bash

set -u
set -e

# refresh auth token
cf oauth-token >/dev/null

for instance_guid in $(cf curl /v3/service_instances | jq -r '.resources[].guid'); do
  for space_guid in $(cf curl /v2/service_instances/$instance_guid/shared_to | jq -r '.resources[].space_guid'); do
    echo "Unsharing service instance $instance_guid from space $space_guid"

    set +e
    cf curl -X DELETE "/v3/service_instances/$instance_guid/relationships/shared_spaces/$space_guid"
    set -e
  done
done
```

Delivering service credentials to an app

You will learn how to bind your apps to service instances for the purpose of generating credentials and delivering them to apps. For an overview of services, and documentation on other service management operations, see Using Services. If you are interested in building services for Ops Manager and making them available to end users, see Services.

Bind a Service Instance

Binding a service instance to your app triggers credentials to be provisioned for the service instance and delivered to the app runtime in the VCAP_SERVICES environment variable. For details on consuming these credentials with your app, see Using Bound Service Instances.

Not all services support binding, as some services deliver value to users directly without integration with an app. In many cases binding credentials are unique to an app, and another app bound to the same service instance would receive different credentials; however this depends on the service.

```
$ cf bind-service my-app mydb
Binding service mydb to my-app in org my-org / space test as me@example.com...
```

```
OK
TIP: Use 'cf push' to ensure your env variable changes take effect

$ cf restart my-app
```

Note: You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Arbitrary Parameters

Some services support additional configuration parameters with the bind request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

See the following examples:

```
$ cf bind-service rails-sample my-db -c '{"role":"read-only"}'

Binding service my-db to app rails-sample in org console / space development
as user@example.com...
OK
```

```
$ cf bind-service rails-sample my-db -c /tmp/config.json

Binding service my-db to app rails-sample in org console / space development
as user@example.com... OK
```

Binding with App Manifest

As an alternative to binding a service instance after pushing an app, you can use the app manifest to bind the service instance during `cf push`.

The following excerpt from an app manifest binds a service instance called `test-mysql-01` to the app on push.

```
services:
 - test-mysql-01
```

The following excerpt from an app manifest binds a service instance called `db-test` with arbitrary parameters to the app on push.

Note: Arbitrary parameters are supported in app manifests in cf CLI v7.0 and later.

```
services:
  - name: db-test
    parameters:
      schema: customschema
```

The following excerpt from the `cf push` command and response demonstrates that the cf CLI reads the manifest and binds the service instance to an app called `test-msg-app`.

```
$ cf push
Using manifest file /Users/Bob/test-apps/test-msg-app/manifest.yml

...

Binding service test-mysql-01 to test-msg-app in org My-Org / space developme
nt as Bob@shared-domain.example.com
OK
```

For more information about app manifests, see Deploying with App Manifests.

Named Service Bindings

Service offering and service instance names vary across environments. App authors can discover the service instances their apps require, without having to know environment-specific service names. Developers can specify a service binding name to be included in VCAP_SERVICES.

To specify the service binding name, provide the `--binding-name` flag when binding an app to a service instance:

```
$ cf bind-service my-app my-service --binding-name postgres-database
OK
```

The provided name will be available in the `name` and `binding_name` properties in the VCAP_SERVICES environment variable:

```
"VCAP_SERVICES": {
  "service-name": [
    {
      "name": "postgres-database",
      "binding_name": "postgres-database",
      ...
    }
  ]
}
```

Using Bound Service Instances

Once you have a service instance created and bound to your app, you need to configure the app to dynamically fetch the credentials for your service instance. The VCAP_SERVICES environment variable contains credentials and additional metadata for all bound service instances. There are two methods developers can leverage to have their apps consume binding credentials.

- **Parse the JSON yourself:** See the documentation for VCAP_SERVICES. Helper libraries are available for some frameworks.
- **Auto-configuration:** Some buildpacks create a service connection for you by creating additional environment variables, updating config files, or passing system parameters to the JVM.

For details on consuming credentials specific to your development framework, refer to the Service Binding section in the documentation for your framework's buildpack.

Update Service Credentials

This section describes how to update service instance credentials. When updating the credentials, you can restage the app immediately and experience app downtime. If your app uses blue-green deployment, you can update service instance credentials with no downtime.

For more information about user-provided services, see Update a User-provided Service Instance in *User-Provided Service Instances*.

With Downtime

To update your service credentials:

1. Unbind the service instance using the credentials you are updating with the following command:

   ```
   $ cf unbind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

2. Bind the service instance with the following command. This adds your credentials to the VCAP_SERVICES environment variable.

   ```
   $ cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

3. Restart or re-push the app bound to the service instance so that the app recognizes your environment variable updates.

Without Downtime

To update your service credentials without experiencing app downtime:

1. Start a blue-green update of the app. For more information, see Using Blue-Green Deployment to Reduce Downtime and Risk. Push the "Green" version of the app with the `--no-start` parameter to prevent the app from starting right away:

   ```
   $ cf push YOUR-APP --no-start
   ```

2. Bind the service instances to the newly-deployed "Green" app with the following command:

   ```
   $ cf bind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

3. Start the "Green" app with `cf start`:

   ```
   $ cf start YOUR-APP
   ```

4. Unbind the service instances from the `Blue` app:

   ```
   $ cf unbind-service YOUR-APP YOUR-SERVICE-INSTANCE
   ```

Unbind a Service Instance

Unbinding a service removes the credentials created for your app from the VCAP_SERVICES environment variable.

```
$ cf unbind-service my-app mydb
Unbinding app my-app from service mydb in org my-org / space test as me@examp
le.com...
OK
```

Note: You must restart or in some cases re-push your app for changes to be applied to the VCAP_SERVICES environment variable and for the app to recognize these changes.

Managing service keys

Here are instructions for managing service instance credentials with service keys.

Service keys generate credentials for manually configuring consumers of marketplace services. Once you configure them for your service, local clients, apps in other spaces, or entities outside your deployment can access your service with these keys.

Note: Some service brokers do not support service keys. If you want to build a service broker that supports service keys, see Services. If you want to use a service broker that does not support service keys, see Delivering Service Credentials to an App.

Create a service key

To generate credentials for a service instance, use the `cf create-service-key` command:

```
$ cf create-service-key MY-SERVICE MY-KEY
Creating service key MY-KEY for service instance MY-SERVICE as me@example.co
m...
OK
```

Use the `-c` flag to provide service-specific configuration parameters in a valid JSON object, either in-line or in a file.

To provide the JSON object in-line, use the following format:

```
$ cf create-service-key MY-SERVICE MY-KEY -c '{"read-only":true}'
Creating service key MY-KEY for service instance MY-SERVICE as me@example.co
m...
OK
```

To provide the JSON object as a file, give the absolute or relative path to your JSON file:

```
$ cf create-service-key MY-SERVICE MY-KEY -c PATH-TO-JSON-FILE
Creating service key MY-KEY for service instance MY-SERVICE as me@example.co
m...
OK
```

List service keys for a service instance

To list service keys for a service instance, use the `cf service-keys` command:

```
$ cf service-keys MY-SERVICE
Getting service keys for service instance MY-SERVICE as me@example.com...

name
mykey1
mykey2
```

Get credentials for a service key

To retrieve credentials for a service key, use the `cf service-key` command:

```
$ cf service-key MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

{
  uri: foo://user2:pass2@example.com/mydb,
  servicename: mydb
}
```

Use the `--guid` flag to display the API GUID for the service key:

```
$ cf service-key --guid MY-SERVICE MY-KEY
Getting key MY-KEY for service instance MY-SERVICE as me@example.com...

e3696fcb-7a8f-437f-8692-436558e45c7b

OK
```

Configure credentials for a service key

Once these credentials are obtained, you can use a local CLI or utility to connect to the service instance, configure an app running outside the platform to connect to the service instance, or create a user-provided service instance so that apps in another space can connect to the service instance. How you configure these credentials will depend on what local client, app, or entity is used to access your service instance.

For more information on configuring credentials with a user-provided service instance, see User-Provided Service Instances.

Delete a service key

To delete a service key, use the `cf delete-service-key` command:

```
$ cf delete-service-key MY-SERVICE MY-KEY

Are you sure you want to delete the service key MY-KEY ? y
Deleting service key MY-KEY for service instance MY-SERVICE as me@example.co
m...

OK
```

Add option `-f` to force deletion without confirmation.

```
$ cf delete-service-key -f MY-SERVICE MY-KEY

Deleting service key MY-KEY for service instance MY-SERVICE as me@example.co
m...

OK
```

Managing app requests with route services

Here are instructions for binding a service instance to a route for the purpose of adding preprocessing to your Cloud Foundry app requests.

> **Note**: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also manage route services using Apps Manager. For more information, see the Manage Route Services section of the *Managing Apps and Service Instances Using Apps Manager* topic.

Route services are a class of Marketplace Services that perform filtering or content transformation on app requests and responses. For more information about Marketplace Services, see the Managing Services and User-provided Service Instances topics.

Route services exist to help remove the burden on developers who would otherwise have to implement these functions themselves. Popular use cases for route services include rate limiting, authorization, and caching. A route service may reject requests or, after some transformation, pass requests to apps.

To use route services, developers must first create a service instance, choosing from compatible Marketplace services. Developers then bind this service instance to a route, and all requests for the route are preprocessed by the service instance. While some services may support instances being bound to both routes and apps, these operations have different effects. For app requests and responses to be routed through a route service, the service instance must be bound to the route.

To view an example demonstrating the use of a sample route service, see Route Services in Pivotal Cloud Foundry on YouTube.

Bind a route to a service instance

You must install the Cloud Foundry Command Line Interface (cf CLI) to bind a route to a service instance. For more information, see the Installing the cf CLI topic.

Note: Gorouter rejects WebSockets requests for routes that are bound to route services. These requests return a 503 error and a `X-Cf-Routererror route_service_unsupported header`.

Run the cf bind-route-service command to bind a route from an app to a service instance.

The following example binds the route from `my-app.shared-domain.example.com` to the service instance `my-route-service`.

```
$ cf bind-route-service shared-domain.example.com my-route-service --hostname
my-app

Binding route my-app.shared-domain.example.com to service instance my-route-s
ervice in org my-org / space my-space as developer...
OK
```

Note: When binding a service instance to a route, Cloud Foundry may proxy requests for the route to the service instance, or configure a network component already in the request path.

Bind with parameters

Some services support additional configuration parameters supplied with the bind request. You can pass service-specific configuration parameters in a valid JSON object, provided either in-line or in a file. For a list of supported configuration parameters, see the documentation for the particular service offering you bind to.

The following example binds the route from `my-app.shared-domain.example.com` to the service instance `my-route-service`, and passes configuration parameters in-line.

```
$ cf bind-route-service shared-domain.example.com my-route-service --hostname
my-app -c '{"rate_limit_threshold_rps":10000}'

Binding service my-db to app rails-sample in org console / space development
as user@example.com...
OK
```

The following example binds the route from `my-app.shared-domain.example.com` to the service instance `my-route-service`, and passes configuration parameters in a file, `/tmp/config.json`.

```
$ cf bind-route-service shared-domain.example.com my-route-service --hostname
my-app -c /tmp/config.json

Binding route my-app.shared-domain.example.com to service instance my-route-s
```

```
ervice in org my-org / space my-space as developer...
OK
```

Unbind a Route from a service instance

You must install the Cloud Foundry Command Line Interface (cf CLI) to bind a route to a service instance. For more information, see the Installing the cf CLI topic.

Run the cf unbind-route-service command to unbind a route from an app to a service instance.

The following example removes the route from `my-app.shared-domain.example.com` to the service instance `my-route-service`.

```
$ cf unbind-route-service shared-domain.example.com my-route-service --hostna
me my-app

Unbinding may leave apps mapped to route my-app.shared-domain.example.com vul
nerable e.g. if service instance my-route-service provides authentication. Do
you want to proceed?> y

Unbinding route my-app.shared-domain.example.com from service instance my-rou
te-service in org my-org / space my-space as developer...
OK
```

Configuring Play Framework service connections

Cloud Foundry provides support for connecting a Play Framework app to services such as MySQL and Postgres. In many cases, a Play Framework app running on Cloud Foundry can detect and configure connections to services.

Auto-configuration

By default, Cloud Foundry detects service connections in a Play Framework app and configure them to use the credentials that are provided in the Cloud Foundry environment. Auto-configuration only happens if it’s a single service for one of the supported types – MySQL or Postgres.

Using an external file system (volume services)

Here are instructions to help you read and write to a mounted file system from your apps. In VMware Tanzu Application Service for VMs (TAS for VMs), a volume service provides a volume so your app can read or write to a reliable, persistent file system.

Note: If you are using VMware Tanzu Application Service for VMs [Windows], see Using SMB Volumes in .NET Apps.
NFS is not available on Windows systems.

Prerequisite

Before you can use a volume service with your app, determine if any volume services are available for your app.

To determine if volume services are available:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI). Run:

   ```
   cf login
   ```

2. List available NFS volume services. Run:

   ```
   cf marketplace
   ```

   See the following example output of the NFS volume service:

   ```
   cf marketplace
   service    plans      description
   nfs        Existing   Service for connecting to NFS volumes
   ```

3. Do one of the following:

   - **If no NFS volume service exists**: If no volume service that fits your requirements exists, contact your Ops Manager admin. For more information, see Enabling Volume Services.
   - **If an NFS volume service exists**: Continue to Mount an External File System.

Mount an External Filesystem

The sections below describe how to mount an external filesystem to your app.

Create and Bind a Service Instance

To use a volume service deployed by your Ops Manager admin, you must first create an instance of the specific volume service that you need.

> **Note:** You can also bind volume services using an app manifest. However, app manifests do not support bind configuration. If you want to bind a volume service using an app manifest, you must specify bind configuration when you create the service instance. The releases that support this are `nfs-volume` v1.3.1 and later and `smb-volume` v1.0.0 and later. For more information, see Services in *Deploying with App Manifests*.

To create and bind an instance for the volume service:

1. Create a service instance:

   - **NFS**: See Create an NFS Volume Service below.
   - **SMB**: See Create an SMB Volume Service below.

2. Bind your service instance to an app:

   - **NFS-only**: Bind an NFS service instance to an app. Run:

```
cf bind-service YOUR-APP SERVICE-NAME -c '{"uid":"UID","gid":"GID","mount":"OPTIONAL-MOUNT-PATH","readonly":true}'
```

Where:

- `YOUR-APP` is the name of the app for which you want to use the volume service.
- `SERVICE-NAME` is the name of the volume service instance you created in the previous step.
- (Optional) `UID` and `GID` are the UID and GID to use when mounting the share to the app. The `GID` and `UID` must be positive integer values. Provide the UID and GID as a JSON string in-line or in a file. If you omit `uid` and `gid`, the driver skips `mapfs` mounting and performs just the normal kernel mount of the NFS file system without the overhead associated with FUSE mounts.

  > **Note:** For security reasons, nfs-volume v2.0.0 and later does not support UID and GID values of `0`.

  > **Note:** The user specified by `uid` must have access to the files on the share. When `uid` and `gid` are omitted, the app file operations use the UID of the running app process. For buildpack apps, this UID is always `2000`. For Docker apps, the effective UID is the same as the UID of the process inside the Docker container, except for `root`, which is mapped to `4294967294` outside the Docker container.

- (Optional) `OPTIONAL-MOUNT-PATH` is a JSON string that indicates the volume should be mounted to a particular path within your app rather than the default path. Choose a path with a root-level folder that already exists in the container, such as `/home`, `/usr`, or `/var`.

  > **Note:** Do not specify a `MOUNT-PATH` within the `/app` directory, which is where TAS for VMs unpacks the droplet. For more information, see Mount a Shared Volume in the /app Directory.

- (Optional) `"readonly":true` is an optional JSON string that creates a read-only mount. By default, Volume Services mounts a read-write file system. For read-only mounts, the driver enables attribute caching. This results in fewer attribute RPCs and better performance.

The following example shows binding `my-app` to the `nfs_service_instance` and specifying a read-only volume to be mounted to `/var/volume1`, passing an in-line JSON string:

```
cf bind-service my-app nfs_service_instance -c '{"uid":"1000","gid":"1000","mount":"/var/volume1","readonly":true}'
```

- **LDAP-only**: Bind an LDAP service to an app. Run:

```
cf bind-service YOUR-APP SERVICE-NAME -c '{"username":"USERNAME","password":"PASSWORD","mount":"OPTIONAL-MOUNT-PATH","readonly":true}'
```

Where:

- `YOUR-APP` is the name of the app for which you want to use the volume service.
- `SERVICE-NAME` is the name of the volume service instance you created in the previous step.
- `USERNAME` and `PASSWORD` are the username and password for the LDAP server. If you omit `username` and `password`, the driver skips `mapfs` mounting and performs just the normal kernel mount of the NFS file system without the overhead associated with FUSE mounts.
- (Optional) `OPTIONAL-MOUNT-PATH` is a JSON string that indicates the volume should be mounted to a particular path within your app rather than the default path. Choose a path with a root-level folder that already exists in the container, such as `/home`, `/usr`, or `/var`.

  Note: Do not specify a `MOUNT-PATH` within the `/app` directory, which is where TAS for VMs unpacks the droplet. For more information, see Mount a Shared Volume in the /app Directory.

- (Optional) `"readonly":true` is an optional JSON string that creates a read-only mount. By default, Volume Services mounts a read-write file system. For read-only mounts, the driver enables attribute caching. This results in fewer attribute RPCs and better performance.

3. Restage your app. Run:

```
cf restage YOUR-APP
```

Where `YOUR-APP` is the name of the app.

Access the Volume Service from Your App

To access the volume service from your app, you must know which file path to use in your code.

You can view the file path in the details of the service binding, which are available from the `VCAP_SERVICES` environment variable. See VCAP_SERVICES.

To access the volume service from your app:

1. View environment variables for your app. Run:

```
cf env YOUR-APP
```

Where `YOUR-APP` is the name of your app.

The following is example output of the `cf env` command:

```
$ cf env YOUR-APP
"VCAP_SERVICES": {
  "nfs": [
    {
      "credentials": {},
      "label": "nfs",
      "name": "nfs_service_instance",
      "plan": "Existing",
      "provider": null,
      "syslog_drain_url": null,
      "tags": [
        "nfs"
      ],
      "volume_mounts": [
        {
          "container_dir": "/var/vcap/data/153e3c4b-1151-4cf7-b311-948d
d77fce64",
          "device_type": "shared",
          "mode": "rw"
        }
      ]
    }
  ]
}
```

2. Use the properties under `volume_mounts` for any information your app needs. See the following table:

Property	Description
`container_dir`	String containing the path to the mounted volume that you bound to your app.
`device_type`	The NFS volume release. This currently only supports `shared` devices. A `shared` device represents a distributed file system that can mount on all app instances simultaneously.
`mode`	String that informs what type of access your app has to NFS, either read-only, `ro`, or read and write, `rw`.

Mount a Shared Volume in the /app Directory

If you specify a mount inside the `/app` directory, the app may fail to start and parts of the app droplet may be written to the remote file share. This is because TAS for VMs mounts the volume before moving your compiled app into the droplet.

If your app requires the shared volume to be placed within the /app directory, you can use a symbolic link at app startup.

To mount a volume in the /app directory:

1. Specify a mount volume in a location outside of the /app directory.
2. Create a symbolic link at app startup time, prior to launching the app. For example, run:

   ```
   cf push YOUR-APP -c "ln -s /var/volume1 /app/volume1 && \$HOME/boot.sh"
   ```

   Where YOUR-APP is the name of the app.

Create and Use NFS Volume Services

This section describes how to use the NFS volume service.

TAS for VMs offers two NFS volume services:

- nfs: This volume service provides support for NFS volumes using both v3 and v4.x protocols.
- nfs-legacy (deprecated): Although it is deprecated, this volume service is still available due to the difficulty of retiring services. If you use this service, it performs exactly the same mount as the nfs service. For information about migrating to nfs, see Migrate nfs-legacy Services to nfs.

Both services offer a single plan called Existing.

Note: NFS is not available on Windows systems.

Create an NFS Volume Service

To create an NFS volume service using the Existing plan of the nfs service:

1. Create an NFS volume service. Run:

   ```
   cf create-service nfs Existing SERVICE-INSTANCE-NAME -c '{"share":"SERVER/SHAR
   E", "version":"NFS-PROTOCOL"}'
   ```

   Where:

   - SERVICE-INSTANCE-NAME is a name you provide for this NFS volume service instance.
   - SERVER/SHARE is the NFS address of your server and share.

     Note: Ensure you omit the : that usually follows the server name in the address.

   - (Optional) NFS-PROTOCOL is the NFS protocol you want to use. For example, if you want to use NFSv4, set the version to 4.1. Valid values are 3.0, 4.0, 4.1 or 4.2. If you do not specify a version, the protocol version used is negotiated between client and server at mount time. This usually results in the latest available version being used.

2. Confirm that the NFS volume service appears in your list of services. Run:

   ```
   cf services
   ```

Migrate nfs-legacy services to nfs

With the release of NFS Volume Service v1.5.4, the original fuse-based NFS service is deprecated in favor of the newer kernel mount-based NFS service. Existing NFS volume service bindings are listed as `nfs-legacy`.

To migrate from `nfs-legacy` to the newer `nfs` service, VMware recommends that you re-create and re-bind your `nfs` service instances.

With the release of NFS Volume Service v2.0.0, the `nfs-legacy` service uses the `nfs` service. To avoid being affected when the `nfs-legacy` service is retired, re-create and re-bind your service instances using the `nfs` service.

Deploy and Bind a Sample App

This section describes how to deploy a sample app and bind it to the NFS volume service.

To deploy and bind a sample app:

1. Clone the GitHub repository for the sample app into your workspace by running these commands:

   ```
   cd ~/workspace
   ```

   ```
   git clone https://github.com/cloudfoundry/persi-acceptance-tests.git
   ```

2. Change into the `persi-acceptance-tests/assets/pora/` directory:

   ```
   cd ~/workspace/persi-acceptance-tests/assets/pora
   ```

3. Push the `pora` test app by running:

   ```
   cf push pora --no-start
   ```

4. Bind the service to your app. Run:

   ```
   cf bind-service pora SERVICE-INSTANCE-NAME -c '{"uid":"UID","gid":"GID"}'
   ```

   Where:

   - `SERVICE-INSTANCE-NAME`: The name of the volume service instance you created previously.
   - `UID` and `GID`: The `UID` and `GID` to use when mounting the share to the app. The NFS driver uses these values in the following ways:

        - When sending traffic to the NFS server, the NFS driver translates the app user ID and group ID to the `UID` and `GID` values.
        - When returning attributes from the NFS server, the NFS driver translates the `UID` and `GID` back to the running user UID and default GID. This allows you to interact with your NFS server as a specific user while allowing TAS for VMs to run your app as an arbitrary user. `UID` and `GID` must be positive integer values.

          > **Note:** In NFS v2.0.0 and later, `uid` and `gid` values of `0` are no longer permissible due to security reasons.

    - (Optional) `mount`: Use this option to specify the path at which volumes mount to the app container. The default is an arbitrarily-named folder in `/var/vcap/data`. You may need to modify this value if your app has specific requirements. For example:

      ```
      cf bind-service pora myVolume -c '{"mount":"/var/path"}'
      ```

    - (Optional) `readonly`: When you run the `cf bind-service` command, Volume Services mounts a read-write file system by default. You can specify a read-only mount by adding `"readonly":true` to the bind configuration JSON string.
    - (Optional) `cache`: When you run the `cf bind-service` command, Volume Services mounts the remote file system with attribute caching disabled by default. You can enable attribute caching using default values by adding `"cache":true` to the bind configuration JSON string.

5. Start the app. Run:

   ```
   cf start pora
   ```

6. Confirm the app is running. Run:

   ```
   curl http://pora.YOUR-CF-DOMAIN.com
   ```

   The command returns an instance index for your app.

7. Confirm the app can access the shared volume. Run:

   ```
   curl http://pora.YOUR-CF-DOMAIN.com/write
   ```

   The command writes a file to the share and then reads it back out again.

Use NFS Volume Services

This section describes using the NFS Volume Service.

Configure LDAP Credentials with Service Instance Creation

If your Ops Manager deployment has LDAP enabled, you can configure LDAP credentials for your NFS Volume Service instance.

To configure LDAP credentials while creating your NFS Volume Service instance:

1. Specify values for `username` and `password` in the JSON string for your `cf create-service` command:

   ```
   cf create-service nfs PLAN SERVICE-INSTANCE-NAME -c '{"share":"SERVER/SHARE",
   "username":"USERNAME", "password":"PASSWORD"}'
   ```

   Where:

   - `PLAN` is the name of the service plan.
   - `SERVICE-INSTANCE-NAME` is a name you provide for this NFS Volume Service instance.
   - `SERVER/SHARE` is the NFS address of your server and share.
   - `USERNAME` is a username you provide for this NFS Volume Service instance.
   - `PASSWORD` is a password you provide for this NFS Volume Service instance.

Specify Bind Parameters During Service Instance Creation

As of `nfs-volume-release` v1.3.1, you can specify bind parameters in advance, when you create a service instance. Use this option if you bind the service to your app in an app manifest, where bind configuration is not supported.

File Locking with flock() and lockf()/fcntl()

Apps that use file locking through unix system calls such as `flock()` and `fcntl()` or script commands such as `flock` may use the `nfs` service. The `nfs-legacy` service uses a fuse mounting process that does not enforce locks across Diego Cells.

Hard Links in the NFS Service

The mapfs UID mapping layer used by the NFS service does not support hard link operations. You get a `Function not implemented` error if you try to create a hard link in an NFS share when `uid` or `username` has been specified for the service.

Workarounds for this issue:

- Use symbolic links, `ln -s`, instead of hard links.
- Omit the `uid` and `gid` or the `username` and `password` parameters to mount the share without UID mapping. For this workaround, the app user must have access to the files on the share.

Create and Use SMB Volume Services

This section describes how to use a Server Message Block (SMB) volume service. For more information about SMB volume services, see Microsoft SMB Protocol and CIFS Protocol Overview in the Microsoft documentation.

> **Note:** If you are using VMware Tanzu Application Service for VMs [Windows], see Using SMB Volumes in .NET Apps.

Create an SMB Volume Service

TAS for VMs offers an `smb` volume service. This volume service provides support for existing SMB shares.

The service offers a single plan called `Existing`.

To create an SMB volume service:

1. Create the service by running:

   ```
   cf create-service smb Existing SERVICE-INSTANCE-NAME -c '{"share":"//SERVER/SHARE", "version":"SMB-VERSION"}'
   ```

   Where:

   - `SERVICE-INSTANCE-NAME` is a name you provide for this SMB volume service instance.
   - `//SERVER/SHARE` is the SMB address of your server and share.
   - (Optional) `SMB-VERSION` is the SMB protocol version you want to use. For example, to use SMB 2.1, set the version to `2.1`. Valid values are `1.0`, `2.0`, `2.1`, or `3.0`. If you do not specify a `version`, the client and server negotiate a protocol version at mount time. The client and server usually select the latest available version.
2. Confirm that the SMB volume service appears in your list of services. Run:

   ```
   cf services
   ```

Deploy and Bind a Sample App

This section describes how to deploy a sample app and bind it to the SMB volume service.

To deploy and bind a sample app:

1. Clone the GitHub repository for the sample app into your workspace. Run:

   ```
   cd ~/workspace
   ```

   ```
   git clone https://github.com/cloudfoundry/persi-acceptance-tests.git
   ```
2. Change into the `persi-acceptance-tests/assets/pora/` directory:

   ```
   cd ~/workspace/persi-acceptance-tests/assets/pora
   ```
3. Push the `pora` test app. Run:

   ```
   cf push pora --no-start
   ```
4. Bind the service to your app. Run:

   ```
   cf bind-service pora SERVICE-INSTANCE-NAME -c '{"username":"USERNAME","password":"PASSWORD"}'
   ```

   Where:

   - `SERVICE-INSTANCE-NAME`: The name of the volume service instance you created previously.
   - `USERNAME` and `PASSWORD`: The username and password to use when mounting the share to the app. This allows you to interact with your SMB server as a specific user while allowing TAS for VMs to run your app as an arbitrary user.
   - (Optional) `mount`: Use this option to specify the path at which volumes mount to the app container. The default is an arbitrarily-named folder in `/var/vcap/data`. You may need to modify this value if your app has specific requirements. For example, run:

     ```
     cf bind-service pora myVolume -c '{"username":"some-user","password":"some-password","mount":"/var/path"}'
     ```

   - (Optional) `readonly`: When you run the `cf bind-service` command, Volume Services mounts a read-write file system by default. You can specify a read-only mount by adding `"readonly":true` to the bind configuration JSON string.
   - (Optional) `domain`: If you use a Windows domain, you can specify a `domain` parameter.

5. Start the app. Run:

   ```
   cf start pora
   ```

6. Confirm the app is running. Run:

   ```
   curl http://pora.YOUR-CF-DOMAIN.com
   ```

   The command returns an instance index for your app.

7. Confirm the app can access the shared volume. Run:

   ```
   curl http://pora.YOUR-CF-DOMAIN.com/write
   ```

   The command writes a file to the share and then reads it back out again.

User-provided service instances

User-provided service instances enable you to use services that are not available in the marketplace with their apps running on Cloud Foundry.

User-provided service instances can be used to deliver service credentials to an app, and/or to trigger streaming of app logs to a syslog compatible consumer. These two functions can be used alone or at the same time.

Note: The procedures in this topic use the Cloud Foundry Command Line Interface (cf CLI). You can also create user-provided service instances in Apps Manager from the Marketplace. To update existing user-provided service instances, navigate to the service instance page and select the **Configuration** tab.

Once created, user-provided service instances behave like service instances created through the marketplace; see Managing Service Instances and App Binding for details on listing, renaming, deleting, binding, and unbinding.

Create a user-provided service instance

The alias for cf create-user-provided-service is `cf cups`.

Deliver Service Credentials to an app

Suppose a developer obtains a URL, port, username, and password for communicating with an Oracle database managed outside of Cloud Foundry. The developer could manually create custom environment variables to configure their app with these credentials (of course you would never hard code these credentials in your app!).

User-provided service instances enable developers to configure their apps with these using the familiar App Binding operation and the same app runtime environment variable used by Cloud Foundry to automatically deliver credentials for marketplace services (VCAP_SERVICES).

```
cf cups SERVICE_INSTANCE -p '{"username":"admin","password":"pa55woRD"}'
```

To create a service instance in interactive mode, use the `-p` option with a comma-separated list of parameter names. The Cloud Foundry Command Line Interface (cf CLI) prompts you for each parameter value.

```
$ cf cups my-user-provided-route-service -p "host, port"

host> rdb.local

port> 5432

Creating user provided service my-user-provided-route-service in org my-org /
space my-space as user@example.com...
OK
```

Once the user-provided service instance is created, to deliver the credentials to one or more apps see App Binding.

Stream app logs to a service

User-provided service instances enable developers to stream app logs to a syslog compatible aggregation or analytics service that isn't available in the marketplace. For more information about the syslog protocol see RFC 5424 and RFC 6587.

Create the user-provided service instance, specifying the URL of the service with the `-l` option.

```
cf cups SERVICE_INSTANCE -l syslog://example.log-aggregator.com
```

To stream app logs to the service, bind the user-provided service instance to your app.

Proxy app requests to a route service

User-provided service instances enable developers to proxy app requests to route services for preprocessing. To create a user-provided service instance for a route service, specify the url for the route service using the `-r` option.

```
$ cf create-user-provided-service my-user-provided-route-service -r https://m
y-route-service.example.com
Creating user provided service my-user-provided-route-service in org my-org /
space my-space as user@example.com...
OK
```

> **Note**: When creating the user-provided service, the route service url specified must be https.

To proxy requests to the user-provided route service, you must bind the service instance to the route. For more information, see Managing App Requests with Route Services.

Update a user-provided service instance

You can use cf update-user-provided-service to update the attributes of an instance of a user-provided service. New credentials overwrite old credentials, and parameters that are not provided are deleted.

The alias for `update-user-provided-service` is `uups`. Bound apps can access the new configuration after restart. You can use rolling restarts to avoid any app downtime. For more information, see Restart an App in *Rolling App Deployments*.

> **Note**: If you are rotating credentials, the old credentials must be active until the restart is finished.

Streaming App Logs

In this section:

- Streaming App Logs to Log Management Services
- Service-Specific Instructions for Streaming App Logs
- Streaming App Logs to Splunk
- Streaming App Logs with Fluentd
- Streaming App Logs to Azure OMS Log Analytics

Streaming app logs to log management services

Here are instructions for draining logs from Cloud Foundry to a third-party log management service.

Cloud Foundry aggregates logs for all instances of your apps and for requests made to your apps through internal components of Cloud Foundry. For example, when the Cloud Foundry router forwards a request to an app, the router records that event in the log stream for that app. Run the following command to access the log stream for an app in the terminal:

```
$ cf logs YOUR-APP-NAME
```

If you want to persist more than the limited amount of logging information that Cloud Foundry can buffer, drain these logs to a log management service.

For more information about the systems responsible for log aggregation and streaming in Cloud Foundry, see App Logging in Cloud Foundry.

Using Services from the Cloud Foundry Marketplace

Your Cloud Foundry marketplace may offer one or more log management services. To use one of these services, create an instance of the service and bind it to your app with the following commands:

```
$ cf create-service SERVICE PLAN SERVICE-INSTANCE
$ cf bind-service YOUR-APP YOUR-LOG-STORE
```

For more information about service instance lifecycle management, see Managing Service Instances.

Note: Not all marketplace services support syslog drains. Some services implement an integration with Cloud Foundry that enables automated streaming of app syslogs. If you are interested in building services for Ops Manager and making them available to end users, see Services.

Using Services Not Available in Your Marketplace

If a compatible log management service is not available in your Cloud Foundry marketplace, you can use user-provided service instances to stream app logs to a service of your choice. For more information, see the Stream App Logs to a Service section of the *User-Provided Service Instances* topic.

You may need to prepare your log management service to receive app logs from Cloud Foundry. For specific instructions for several popular services, see Service-Specific Instructions for Streaming App Logs. If you cannot find instructions for your service, follow the generic instructions below.

Step 1: Configure the Log Management Service

To set up a communication channel between the log management service and your Cloud Foundry deployment:

1. Obtain the external IP addresses that your Ops Manager admin assigns to outbound traffic.
2. Provide these IP addresses to the log management service. The specific steps to configure a third-party log management service depend on the service.
3. Add these IP addresses to your allow list to ensure unrestricted log routing to your log management service.

4. Record the syslog URL provided by the third-party service. Third-party services typically provide a syslog URL to use as an endpoint for incoming log data. You use this syslog URL in Step 2: Create a User-Provided Service Instance.

   Cloud Foundry uses the syslog URL to route messages to the service. The syslog URL has a scheme of `syslog`, `syslog-tls`, or `https`, and can include a port number. For example:

   `syslog://logs.example.com:1234`

> **Note**: TAS for VMs does not support using `syslog-tls` or `https` with self-signed certificates. If you are running your own syslog server and want to use `syslog-tls` or `https`, you must have an SSL certificate signed by a well-known certificate authority.

> **Note:** If the URL is an IP address, then it must not contain any leading zeros, for example, `10.0.01.14`. If you include the leading zeros, then URL parsing fails.

Step 2: Create and Bind a User-Provided Service Instance

You can create a syslog drain service and bind apps to it using Cloud Foundry Command Line Interface (cf CLI) commands.

1. To create the service instance, run `cf create-user-provided-service` (or `cf cups`) with the `-l` flag.

   ```
   cf create-user-provided-service DRAIN-NAME -l SYSLOG-URL
   ```

   Where:

   - `DRAIN-NAME` is a name to use for your syslog drain service instance.
   - `SYSLOG-DRAIN-URL` is the syslog URL from Step 1: Configure the Log Management Service.

   In case of the usage of the mTLS feature delivered in CAPI release 1.143.0, you can use `-p` flag to define the credentials, filling in values as follows.

   ```
   cf create-user-provided-service DRAIN-NAME -l SYSLOG-URL -p {"cert":"-----BEGIN
   CERTIFICATE-----\nMIIH...-----END CERTIFICATE-----","key":"-----BEGIN PRIVATE K
   EY-----\nMIIE...-----END PRIVATE KEY-----", "ca":"-----BEGIN CERTIFICATE-----\n
   MIIH...-----END CERTIFICATE-----"}
   ```

   For more information, see User-Provided Service Instances.

2. To bind an app to the service instance, do one of the following:

   - Run `cf push` with a manifest. The services block in the manifest must specify the service instance that you want to bind.
   - Run `cf bind-service`:

     ```
     cf bind-service YOUR-APP-NAME DRAIN-NAME
     ```

After a short delay, logs begin to flow automatically.

For more information, see Managing Service Instances with the CLI.

Step 3: Verify Logs Are Draining

To verify that logs are draining correctly to a third-party log management service:

1. Take actions that produce log messages, such as making requests of your app.
2. Compare the logs displayed in the CLI against those displayed by the log management service.

For example, if your app serves web pages, you can send HTTP requests to the app. In Cloud Foundry, these generate Router log messages, which you can view in the CLI. Your third-party log management service should display corresponding messages.

> **Note:** For security reasons, Cloud Foundry apps do not respond to `ping`. You cannot use `ping` to generate log entries.

Streaming app logs to third-party services

Here are instructions for configuring some third-party log management services for your Cloud Foundry apps.

Once you have configured a service, refer to the Third-Party Log Management Services topic for instructions on binding your app to the service.

Logit.io

From your Logit.io dashboard:

1. Identify the Logit ELK stack you want to use.
2. Click Logstash **Configuration**.
3. Note your Logstash **Endpoint**.
4. Note your TCP-SSL, TCP, or UDP **Port** (not the syslog port).
5. Create the log drain service in Cloud Foundry.

   ```
   $ cf cups logit-ssl-drain -l syslog-tls://ENDPOINT:PORT
   ```

   or

   ```
   $ cf cups logit-drain -l syslog://ENDPOINT:PORT
   ```

6. Bind the service to an app.

   ```
   $ cf bind-service YOUR-CF-APP-NAME logit-ssl-drain
   ```

   or

   ```
   $ cf bind-service YOUR-CF-APP-NAME logit-drain
   ```

7. Restage or push the app using one of the following commands:

```
$ cf restage YOUR-CF-APP-NAME
```

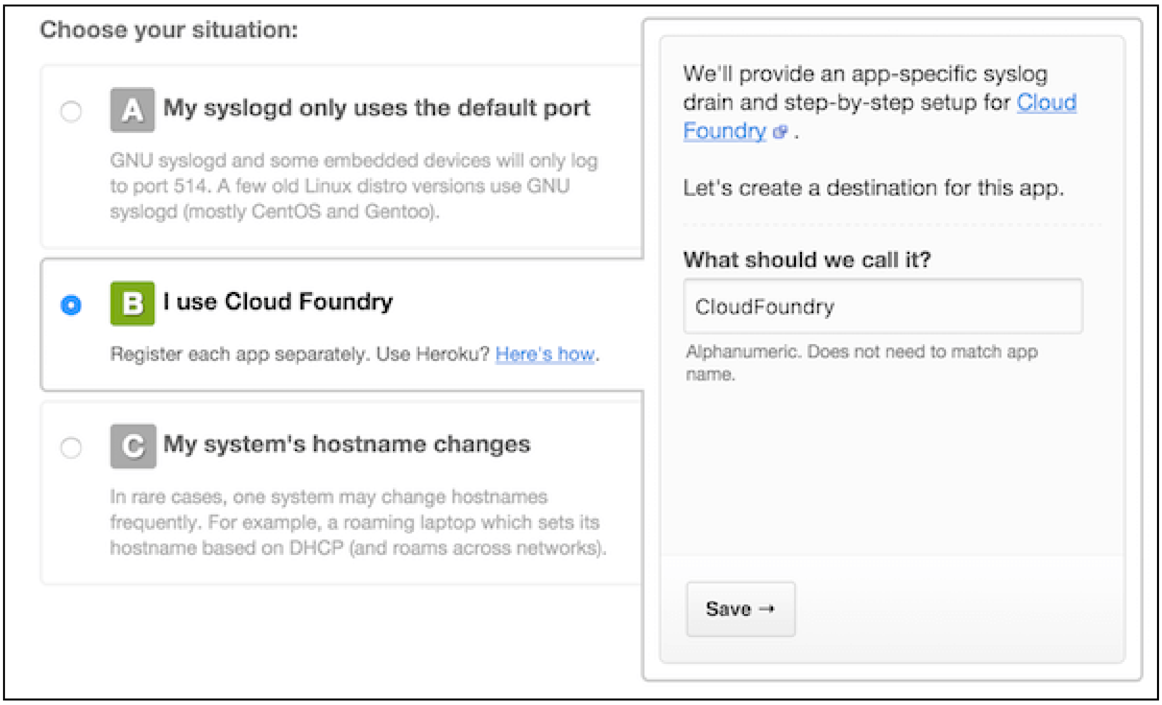
```
$ cf push YOUR-CF-APP-NAME
```

After a short delay, logs begin to appear in Kibana.

Papertrail

From your Papertrail account:

1. Click **Add System**. The Dashboard appears.
2. Click the **Other** link. The **Setup Systems** screen appears.
3. Select **I use Cloud Foundry**, enter a name, and click **Save**.



Choose your situation:

○ A My syslogd only uses the default port

GNU syslogd and some embedded devices will only log to port 514. A few old Linux distro versions use GNU syslogd (mostly CentOS and Gentoo).

● B I use Cloud Foundry

Register each app separately. Use Heroku? Here's how.

○ C My system's hostname changes

In rare cases, one system may change hostnames frequently. For example, a roaming laptop which sets its hostname based on DHCP (and roams across networks).

We'll provide an app-specific syslog drain and step-by-step setup for Cloud Foundry .

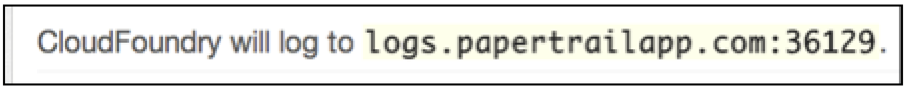
Let's create a destination for this app.

What should we call it?

CloudFoundry

Alphanumeric. Does not need to match app name.

4. Record the URL with port that is displayed after creating the system.



CloudFoundry will log to `logs.papertrailapp.com:36129`.

5. Create the log drain service in Cloud Foundry.

```
$ cf cups my-logs -l syslog-tls://logs.papertrailapp.com:PORT
```

6. Bind the service to an app.

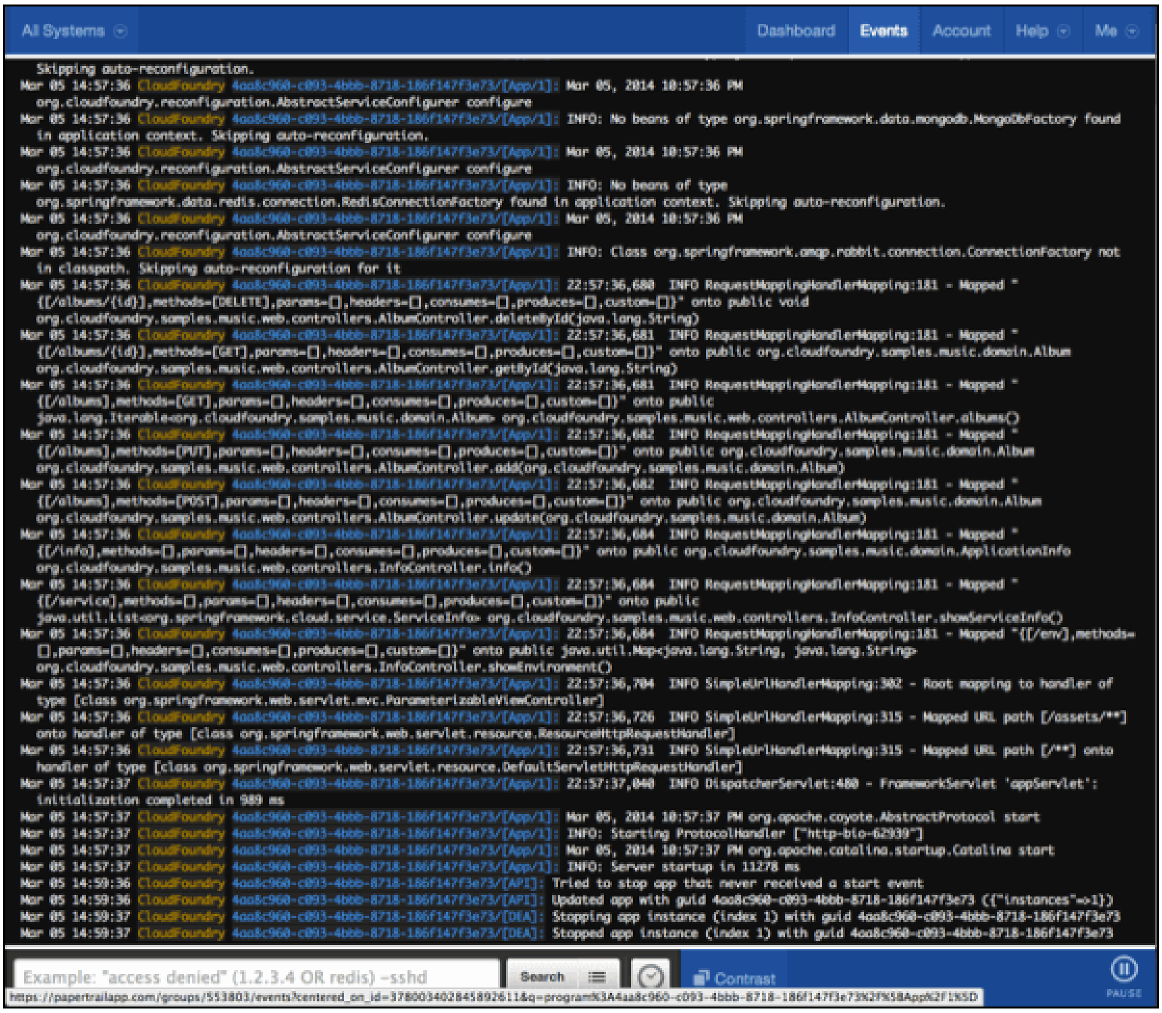
```
$ cf bind-service APPLICATION-NAME my-logs
```

7. Restage the app.

```
$ cf restage APPLICATION-NAME
```

After a short delay, logs begin to flow.

8. When Papertrail starts receiving log entries, the view changes to show the logs viewing page.



```
  Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: Mar 05, 2014 10:57:36 PM
  org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: INFO: No beans of type org.springframework.data.mongodb.MongoDbFactory found
  in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: Mar 05, 2014 10:57:36 PM
  org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: INFO: No beans of type
  org.springframework.data.redis.connection.RedisConnectionFactory found in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: Mar 05, 2014 10:57:36 PM
  org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: INFO: Class org.springframework.amqp.rabbit.connection.ConnectionFactory not
  in classpath. Skipping auto-reconfiguration for it
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,680  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/albums/{id}],methods=[DELETE],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public void
  org.cloudfoundry.samples.music.web.controllers.AlbumController.deleteById(java.lang.String)
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,681  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/albums/{id}],methods=[GET],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.Album
  org.cloudfoundry.samples.music.web.controllers.AlbumController.getById(java.lang.String)
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,681  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/albums],methods=[GET],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public
  java.lang.Iterable<org.cloudfoundry.samples.music.domain.Album> org.cloudfoundry.samples.music.web.controllers.AlbumController.albums()
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,682  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/albums],methods=[PUT],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.Album
  org.cloudfoundry.samples.music.web.controllers.AlbumController.add(org.cloudfoundry.samples.music.domain.Album)
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,682  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/albums],methods=[POST],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.Album
  org.cloudfoundry.samples.music.web.controllers.AlbumController.update(org.cloudfoundry.samples.music.domain.Album)
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,684  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/info],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.ApplicationInfo
  org.cloudfoundry.samples.music.web.controllers.InfoController.info()
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,684  INFO RequestMappingHandlerMapping:181 - Mapped "
  {[/service],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public
  java.util.List<org.springframework.cloud.service.ServiceInfo> org.cloudfoundry.samples.music.web.controllers.InfoController.showServiceInfo()
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,684  INFO RequestMappingHandlerMapping:181 - Mapped "{[/env],methods=
  [],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public java.util.Map<java.lang.String, java.lang.String>
  org.cloudfoundry.samples.music.web.controllers.InfoController.showEnvironment()
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,704  INFO SimpleUrlHandlerMapping:302 - Root mapping to handler of
  type [class org.springframework.web.servlet.mvc.ParameterizableViewController]
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,726  INFO SimpleUrlHandlerMapping:315 - Mapped URL path [/assets/**]
  onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
Mar 05 14:57:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:36,731  INFO SimpleUrlHandlerMapping:315 - Mapped URL path [/**] onto
  handler of type [class org.springframework.web.servlet.resource.DefaultServletHttpRequestHandler]
Mar 05 14:57:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: 22:57:37,040  INFO DispatcherServlet:480 - FrameworkServlet 'appServlet':
  initialization completed in 989 ms
Mar 05 14:57:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: Mar 05, 2014 10:57:37 PM org.apache.coyote.AbstractProtocol start
Mar 05 14:57:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: INFO: Starting ProtocolHandler ["http-bio-62939"]
Mar 05 14:57:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: Mar 05, 2014 10:57:37 PM org.apache.catalina.startup.Catalina start
Mar 05 14:57:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[App/1]: INFO: Server startup in 11278 ms
Mar 05 14:59:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[API]: Tried to stop app that never received a start event
Mar 05 14:59:36 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[API]: Updated app with guid 4aa8c960-c093-4bbb-8718-186f147f3e73 ({"instances"=>1})
Mar 05 14:59:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[DEA]: Stopping app instance (index 1) with guid 4aa8c960-c093-4bbb-8718-186f147f3e73
Mar 05 14:59:37 CloudFoundry 4aa8c960-c093-4bbb-8718-186f147f3e73/[DEA]: Stopped app instance (index 1) with guid 4aa8c960-c093-4bbb-8718-186f147f3e73
```

Example: "access denied" (1.2.3.4 OR redis) -sshd
Search
Contrast
PAUSE
https://papertrailapp.com/groups/553803/events?centered_on_id=378003402845892611&q=program%3A4aa8c960-c093-4bbb-8718-186f147f3e73%2F%5BApp%2F1%5D

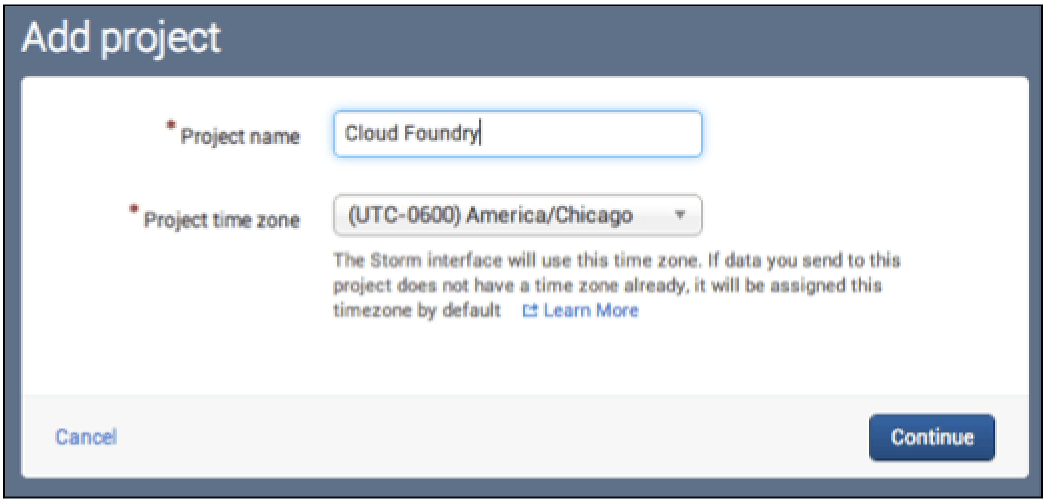
Splunk

See Streaming App Logs to Splunk for details.

Splunk Storm

From your Splunk Storm account:

1. Click **Add project**.

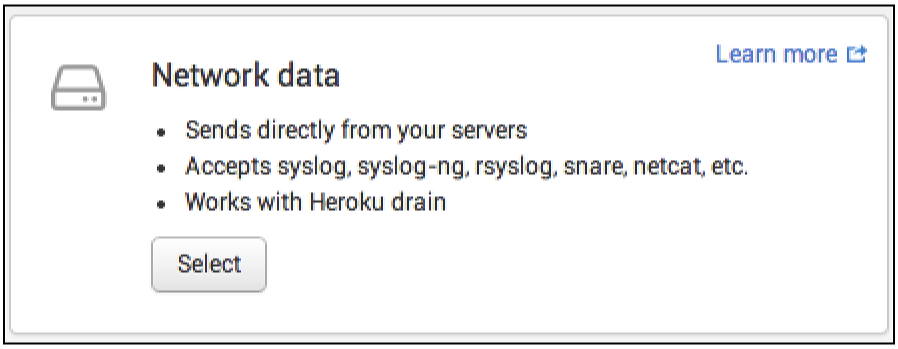


* Project name: Cloud Foundry

* Project time zone: (UTC-0600) America/Chicago

The Storm interface will use this time zone. If data you send to this project does not have a time zone already, it will be assigned this timezone by default

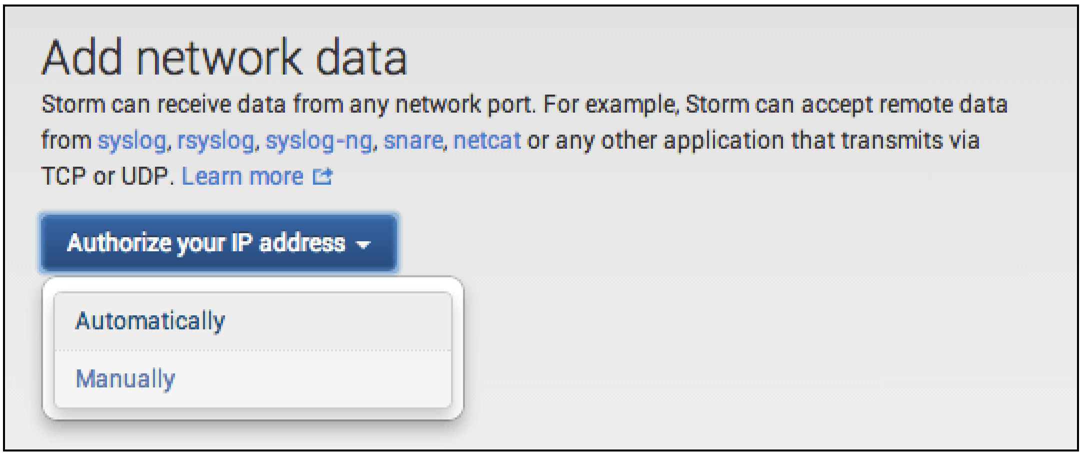
2. Enter the project details.



Network data

- Sends directly from your servers
- Accepts syslog, syslog-ng, rsyslog, snare, netcat, etc.
- Works with Heroku drain

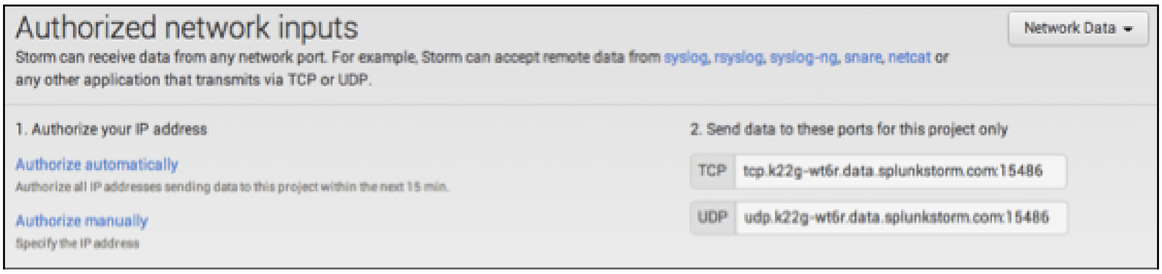
3. Create a new **input** for **Network data**.



Add network data

Storm can receive data from any network port. For example, Storm can accept remote data from syslog, rsyslog, syslog-ng, snare, netcat or any other application that transmits via TCP or UDP. Learn more

4. Manually enter the external IP addresses your Cloud Foundry administrator assigns to outbound traffic.



Authorized network inputs

Storm can receive data from any network port. For example, Storm can accept remote data from syslog, rsyslog, syslog-ng, snare, netcat or any other application that transmits via TCP or UDP.

1. Authorize your IP address

Authorize automatically
Authorize all IP addresses sending data to this project within the next 15 min.

Authorize manually
Specify the IP address

2. Send data to these ports for this project only

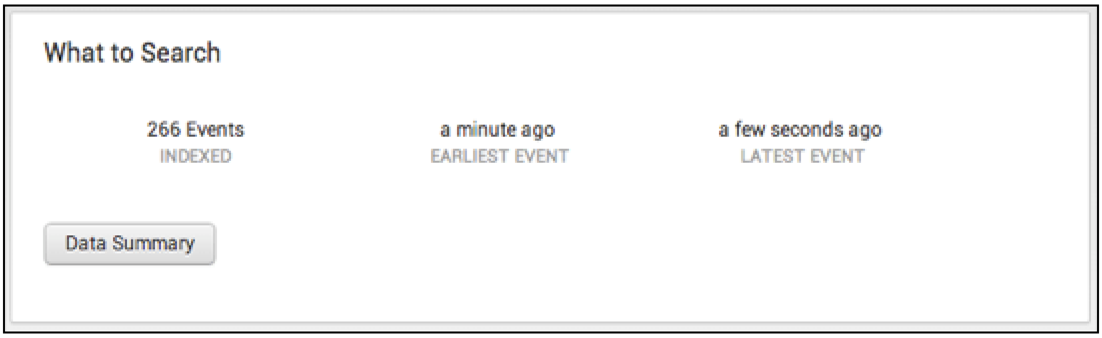
TCP: tcp.k22g-wt6r.data.splunkstorm.com:15486
UDP: udp.k22g-wt6r.data.splunkstorm.com:15486

5. Using the cf CLI, create the log drain service in Cloud Foundry using the TCP host and port you recorded. Then you bind teh service to an app and restage the app using the syntax

shown here. After a short delay, the logs begin to flow.

```
$ cf cups my-logs -l syslog://HOST:PORT
$ cf bind-service APPLICATION-NAME my-logs
$ cf restage APPLICATION-NAME
```

6. When events begin to appear, click **Data Summary**. The **Data Summary** button appears in the **What to Search** section.



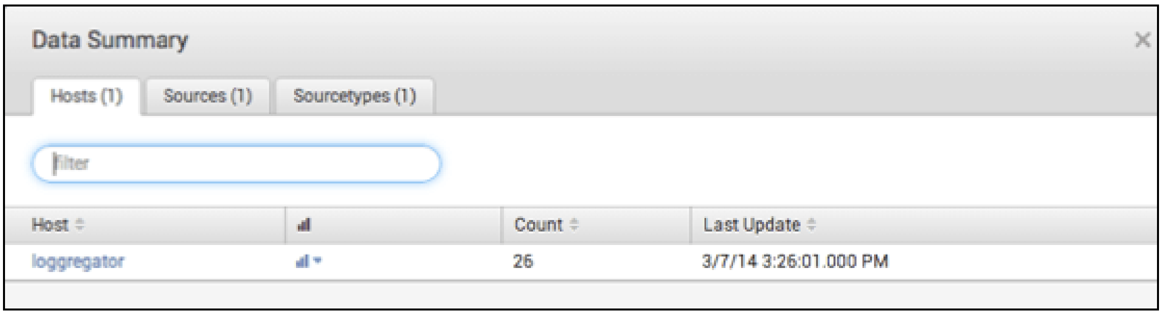
What to Search

266 Events
INDEXED

a minute ago
EARLIEST EVENT

a few seconds ago
LATEST EVENT

7. In the **Data Summary** table, click the **loggregator** link to view all incoming log entries from Cloud Foundry.



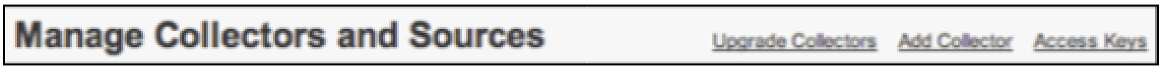
Host		Count	Last Update
loggregator		26	3/7/14 3:26:01.000 PM

SumoLogic

> **Note**: SumoLogic uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

From your SumoLogic account:

1. Beside **Manage Collectors and Sources**, click the **Add Collector** link.



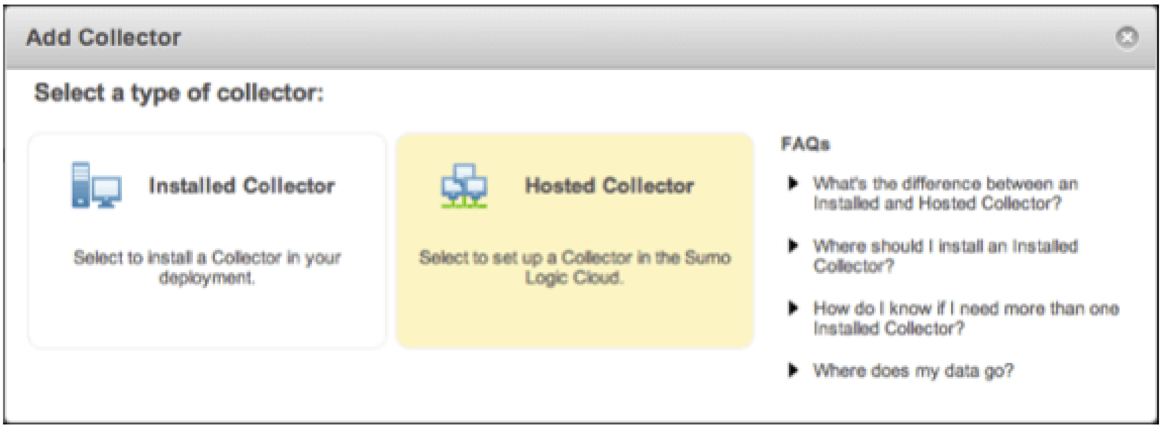
Manage Collectors and Sources

2. Under **Add Collector**, select **Hosted Collector** and fill in the details.

    1. In **Name**, enter `Cloud Foundry`.

    2. In **Description**, enter the purpose of the new collector.

    3. In **Category**, you can enter the source category, if you want. The collector sets the source category to this value unless it is overwritten by the source metadata.



Add Collector

Select a type of collector:

Installed Collector

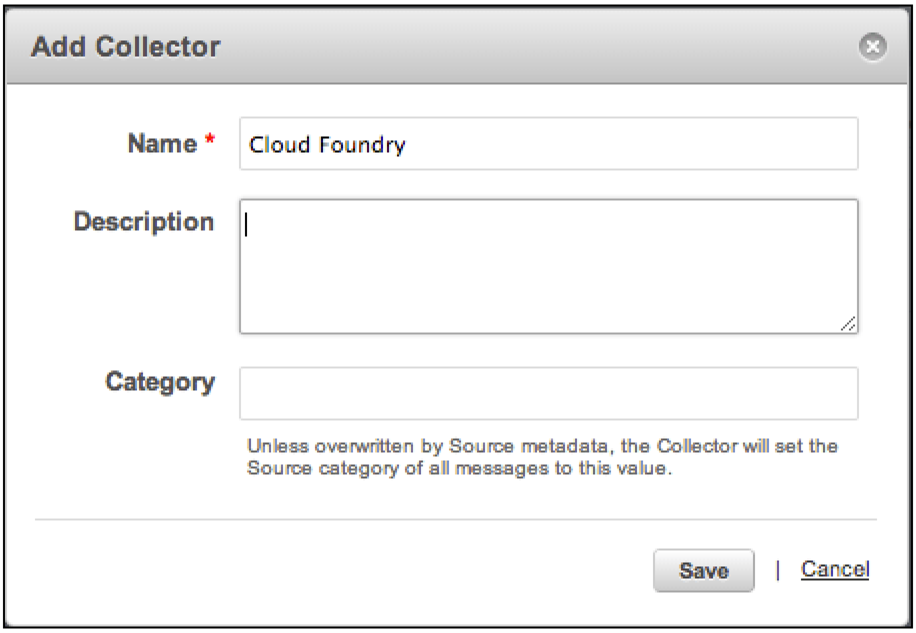
Select to install a Collector in your deployment.

Hosted Collector

Select to set up a Collector in the Sumo Logic Cloud.

FAQs

- What's the difference between an Installed and Hosted Collector?
- Where should I install an Installed Collector?
- How do I know if I need more than one Installed Collector?
- Where does my data go?



Add Collector

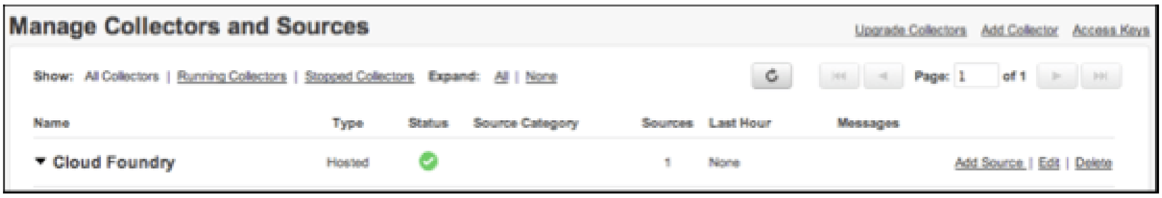
Name * Cloud Foundry

Description

Category

Unless overwritten by Source metadata, the Collector will set the Source category of all messages to this value.

3. In the **Manage Collectors and Sources** table, in the row for the new collector, click the **Add Source** link.



Manage Collectors and Sources

Name	Type	Status	Source Category	Sources	Last Hour	Messages	
▾ Cloud Foundry	Hosted	✓		1	None		Add Source | Edit | Delete

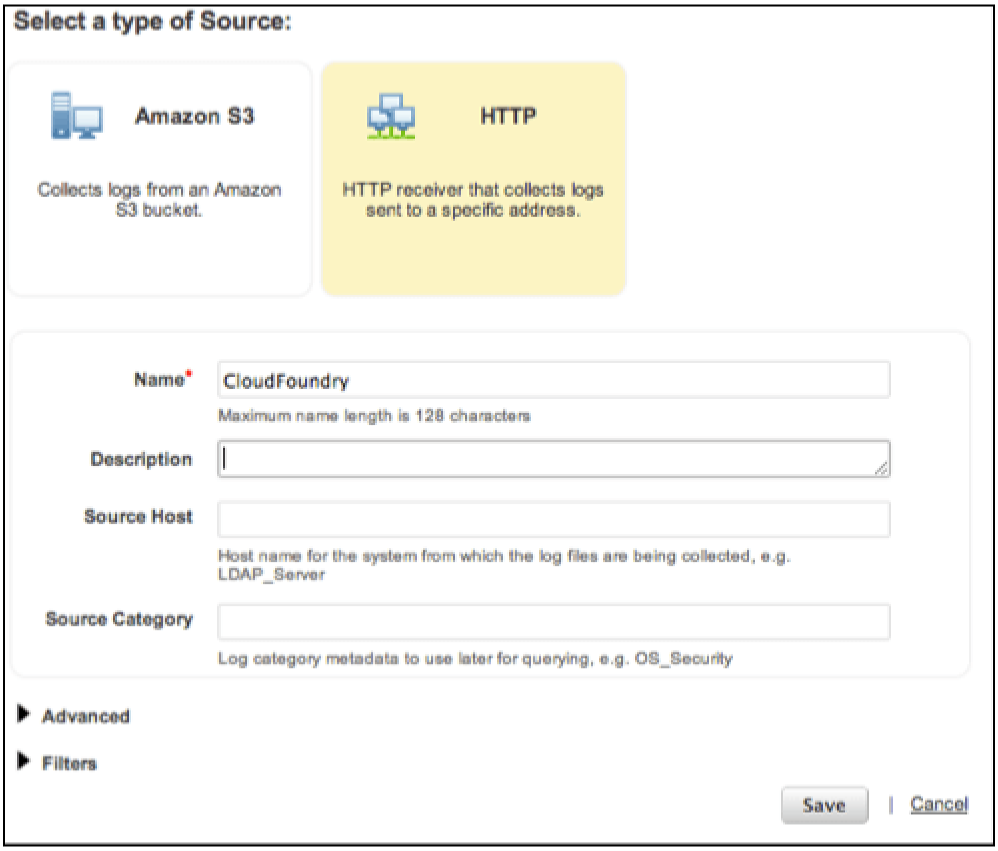
4. Under **Select a type of Source**, select **HTTP** and fill in the details. An HTTPS URL is provided.

    1. In **Name**, leave `CloudFoundry`.

    2. In **Description**, enter a description of the source.

    3. In **Source Host**, enter the host name for the system from which the log files are being collected.

    4. In **Source Category**, enter the log category metadata. You can use this later in queries.



Select a type of Source:

Amazon S3 Collects logs from an Amazon S3 bucket.	**HTTP** HTTP receiver that collects logs sent to a specific address.

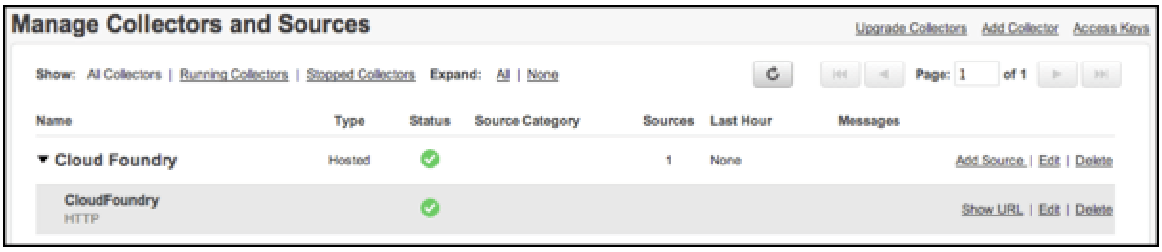
Name*	CloudFoundry Maximum name length is 128 characters
Description	
Source Host	 Host name for the system from which the log files are being collected, e.g. LDAP_Server
Source Category	 Log category metadata to use later for querying, e.g. OS_Security

▶ Advanced

▶ Filters

Save | Cancel

5. When the source has been created, a URL is displayed. You can also view the URL by clicking the **Show URL** link beside the newly created source in the **Manage Collectors and Sources** table. Record the URL for the next step.



Manage Collectors and Sources

Upgrade Collectors Add Collector Access Keys

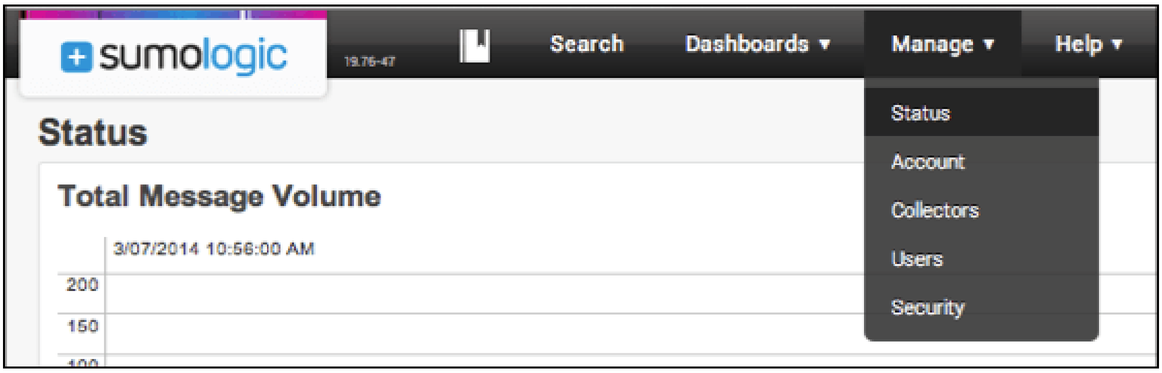
Show: All Collectors | Running Collectors | Stopped Collectors Expand: All | None Page: 1 of 1

Name	Type	Status	Source Category	Sources	Last Hour	Messages	
▾ Cloud Foundry	Hosted	✓		1	None		Add Source | Edit | Delete
CloudFoundry HTTP		✓					Show URL | Edit | Delete

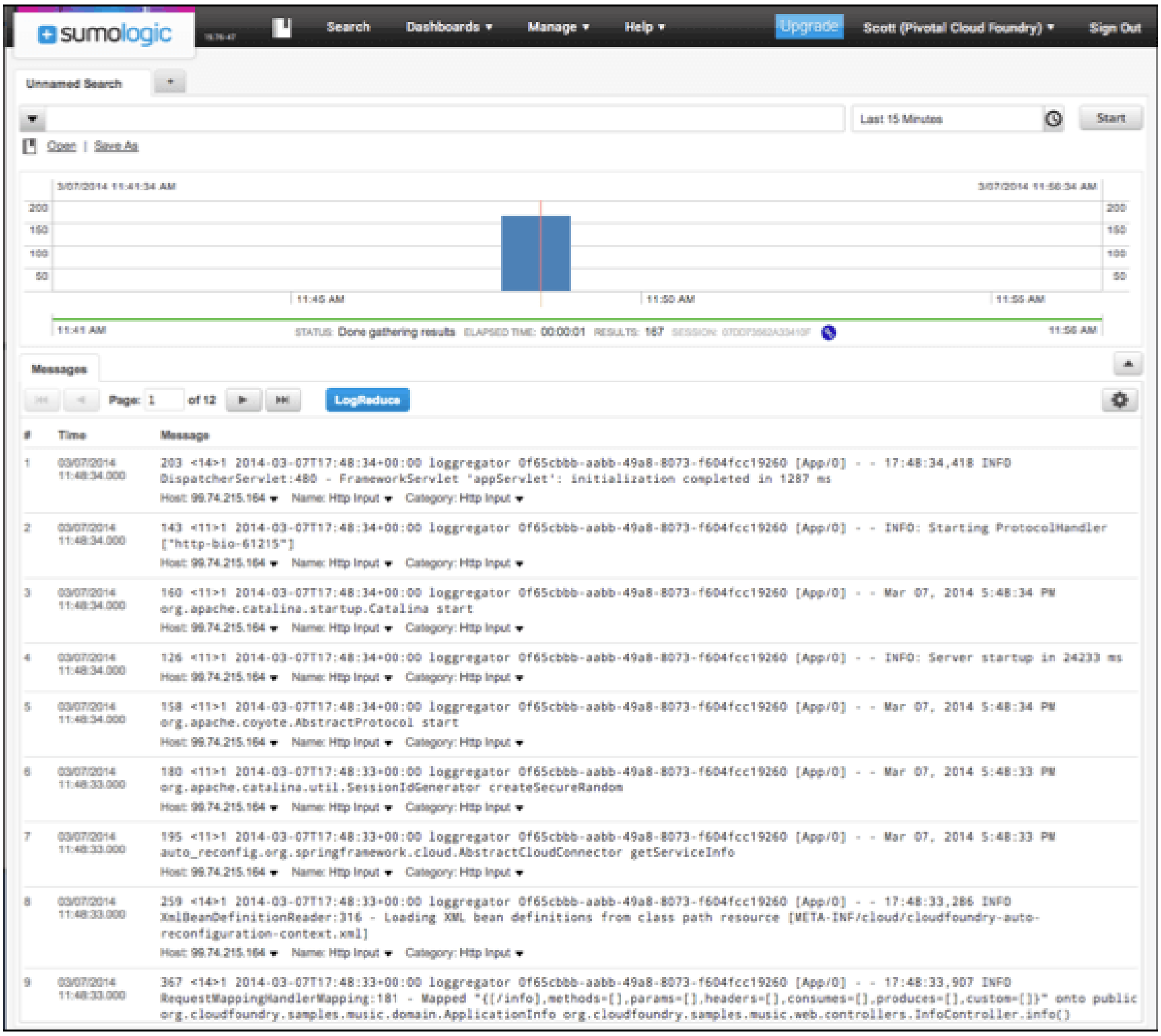
6. Using the cf CLI, create the log drain service in Cloud Foundry using the source URL you just recorded. Then you bind the service to an app and restage the app using the syntax shown here. After a short delay, the logs will begin to flow.

```
$ cf cups my-logs -l HTTPS-SOURCE-URL
$ cf bind-service APPLICATION-NAME my-logs
$ cf restage APPLICATION-NAME
```

7. In the SumoLogic dashboard, click **Manage**, then click **Status** to see a view of the log messages received.



8. Click **Search**. Place the cursor in the search box, then press **Enter** to submit an empty search query.



#	Time	Message
1	03/07/2014 11:48:34.000	203 <14>1 2014-03-07T17:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - 17:48:34,418 INFO DispatcherServlet:480 - FrameworkServlet 'appServlet': initialization completed in 1287 ms Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
2	03/07/2014 11:48:34.000	143 <11>1 2014-03-07T17:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - INFO: Starting ProtocolHandler ["http-bio-61215"] Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
3	03/07/2014 11:48:34.000	160 <11>1 2014-03-07T17:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - Mar 07, 2014 5:48:34 PM org.apache.catalina.startup.Catalina start Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
4	03/07/2014 11:48:34.000	126 <11>1 2014-03-07T17:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - INFO: Server startup in 24233 ms Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
5	03/07/2014 11:48:34.000	158 <11>1 2014-03-07T17:48:34+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - Mar 07, 2014 5:48:34 PM org.apache.coyote.AbstractProtocol start Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
6	03/07/2014 11:48:33.000	180 <11>1 2014-03-07T17:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - Mar 07, 2014 5:48:33 PM org.apache.catalina.util.SessionIdGenerator createSecureRandom Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
7	03/07/2014 11:48:33.000	195 <11>1 2014-03-07T17:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - Mar 07, 2014 5:48:33 PM auto_reconfig.org.springframework.cloud.AbstractCloudConnector getServiceInfo Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
8	03/07/2014 11:48:33.000	259 <14>1 2014-03-07T17:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - 17:48:33,286 INFO XmlBeanDefinitionReader:316 - Loading XML bean definitions from class path resource [META-INF/cloud/cloudfoundry-auto-reconfiguration-context.xml] Host: 99.74.215.164 ▾ Name: Http Input ▾ Category: Http Input ▾
9	03/07/2014 11:48:33.000	367 <14>1 2014-03-07T17:48:33+00:00 loggregator 0f65cbbb-aabb-49a8-8073-f604fcc19260 [App/0] - - 17:48:33,907 INFO RequestMappingHandlerMapping:181 - Mapped "{[/info],methods=[],params=[],headers=[],consumes=[],produces=[],custom=[]}" onto public org.cloudfoundry.samples.music.domain.ApplicationInfo org.cloudfoundry.samples.music.web.controllers.InfoController.info()

Logsene

> **Note**: Logsene uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and later.

From your Sematext account:

1. Click the Create App / Logsene App menu item. Enter a name and click **Add Application** to create the Logsene App.
2. Using the cf CLI, create the log drain service in Cloud Foundry using the source URL displayed. Then you bind the service to an app and restage the app using the commands shown here. After a short delay, the logs begin to flow. The logs appear in the Logsene UI.

   ```
   $ cf cups logsene-log-drain -l https://logsene-cf-receiver.sematext.co
   m/YOUR_LOGSENE_TOKEN
   $ cf restage APPLICATION-NAME
   $ cf bind-service YOUR-CF-APP-NAME logsene-log-drain
   ```

Logentries is not supported

Using Logentries is not recommended because it does not support multiple syslog sources. Cloud Foundry distributes log messages over multiple servers to handle the load.

Streaming app logs to Splunk

Use the process here to integrate Cloud Foundry with Splunk Enterprise for logging.

1. Create a Cloud Foundry syslog drain for Splunk

In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.

Choose one or more apps whose logs you want to drain to Splunk through the service.

Bind each app to the service instance and restart the app.

Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service. Locate the port number in the syslog URL. For example:

`syslog://logs.example.com:1234`

2. Prepare Splunk for Cloud Foundry

For detailed information about the following tasks, see the Splunk documentation.

Install the RFC5424 Syslog Technology add-on

The Cloud Foundry Loggregator component formats logs according to the Syslog Protocol defined in RFC 5424. Splunk does not parse log fields according to this protocol. To allow Splunk to correctly parse RFC 5424 log fields, install the Splunk RFC5424 Syslog Technical Add-On.

Patch the RFC5424 Syslog Technology add-on

1. SSH into the Splunk VM
2. Replace `/opt/splunk/etc/apps/rfc5424/default/transforms.conf` with a new `transforms.conf` file that consists of the following text:

```
[rfc5424_host]
DEST_KEY = MetaData:Host
REGEX = <\d+>\d{1}\s{1}\S+\s{1}(\S+)
FORMAT = host::$1

[rfc5424_header]
REGEX = <(\d+)>\d{1}\s{1}\S+\s{1}\S+\s{1}(\S+)\s{1}(\S+)\s{1}(\S+)
FORMAT = prival::$1 appname::$2 procid::$3 msgid::$4
MV_ADD = true
```

3. Restart Splunk

Create a TCP syslog data input

Create a TCP Syslog Data Input in Splunk, with the following settings:

- **TCP port** is the port number you assigned to your log drain service
- **Set sourcetype** is `Manual`
- **Source type** is `rfc5424_syslog` (type this value into text field)
- **Index** is the index you created for your log drain service

Your Cloud Foundry syslog drain service is now integrated with Splunk.

3. Verify that the integration was successful

Use Splunk to execute a query of the form:

```
sourcetype=rfc5424_syslog index=-THE-INDEX-YOU-CREATED appname=APP-GUID
```

To view logs from all apps at once, you can omit the `appname` field.

Verify that results rows contain the three Cloud Foundry-specific fields:

- **appname**: The GUID for the Cloud Foundry app
- **host**: The IP address of the Loggregator host
- **procid**: The Cloud Foundry component emitting the log

If the Cloud Foundry-specific fields appear in the log search results, integration is successful.

If logs from an app are missing, make sure that the following are true:

- The app is bound to the service and was restarted after binding
- The service port number matches the TCP port number in Splunk

Streaming app logs with Fluentd

Fluentd is an open source log collector that allows you to implement unified logging layers. With Fluentd, you can stream app logs to different backends or services like Elasticsearch, HDFS and Amazon S3. This topic explains how to integrate Fluentd with Cloud Foundry apps.

Step 1: Create a Cloud Foundry syslog drain for Fluentd

1. In Cloud Foundry, create a syslog drain user-provided service instance as described in Using Third-Party Log Management Services.
2. Choose one or more apps whose logs you want to drain to Fluentd through the service.
3. Bind each app to the service instance, and restart the app.
4. Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service.
5. Locate the port number in the syslog URL. For example:

   ```
   syslog://logs.example.com:5140
   ```

Step 2: Set up Fluentd for Cloud Foundry

This section assumes you have an active Fluentd instance running. If you do not have an active Fluentd instance, refer to the Fluentd Documentation/Install steps for more details. If you use cf to deploy your fluentd instances, you will have to use tcp routing.

Fluentd comes with native support for syslog protocol. To set up Fluentd for Cloud Foundry, configure the syslog input of Fluentd as follows.

1. In your main Fluentd configuration file, add the following `source` entry:

   ```
   <source>
     @type syslog
     port 8080
     bind 0.0.0.0
     tag cf.app
     message_length_limit 99990
     frame_type octet_count
     <transport tcp>
     </transport>
     <parse>
           message_format rfc5424
     </parse>
   </source>
   ```

2. Restart the Fluentd service.

> **Note**: The Fluentd syslog input plugin supports `tls` and `tcp` options. Make sure to use the same transport that Cloud Foundry is using.

Fluentd will start listening for Syslog message on port 8080 and tagging the messages with `cf.app`, which can be used later for data routing. For more details about the full setup for the service, refer to the Config File topic.

If you want to use an Elasticsearch or Amazon S3 backend, see the Fluentd documentation.

Streaming app logs to Azure OMS Log Analytics

Here are instructions for integrating your VMware Tanzu Application Service for VMs (TAS for VMs) apps with OMS Log Analytics.

Operations Management Suite (OMS) Log Analytics is a monitoring service for Microsoft Azure. The OMS Log Analytics Firehose Nozzle is a TAS for VMs component that forwards metrics from the Loggregator Firehose to OMS Log Analytics.

This topic assumes you are using the latest version of the Cloud Foundry Command Line Interface (cf CLI) and a working TAS for VMs deployment on Azure.

Step 1: Create an OMS workspace in Azure

To create an OMS workspace, see Get started with Log Analytics in the Microsoft Azure documentation.

Step 2: Deploy the nozzle to TAS for VMs

To deploy the OMS Log Analytics Firehose nozzle to TAS for VMs:

1. Authenticate to your TAS for VMs instance. For more information, see Creating and Managing Users with the UAA CLI (UAAC) and Orgs, Spaces, Roles, and Permissions. Run:

   ```
   cf login -a https://api.YOUR-DOMAIN -u YOUR-USERNAME --skip-ssl-validation
   ```

   Where:

   - `YOUR-DOMAIN` is your domain.
   - `YOUR-USERNAME` is your TAS for VMs username.

2. To create a new TAS for VMs user and grant it access to the Loggregator Firehose using the UAA CLI (UAAC):

   1. Target your UAA server by running:

      ```
      uaac target uaa.YOUR-DOMAIN --skip-ssl-validation
      ```

      Where `YOUR-DOMAIN` is your domain.

   2. Obtain an access token for the admin client by running:

      ```
      uaac token client get admin
      ```

   3. Create a new user by running:

      ```
      uaac user add USERNAME -p PASSWORD  --email EMAIL
      ```

      Where:

      - `USERNAME` is a new username.
      - `PASSWORD` is a password.
      - `EMAIL` is an email address.

   4. Grant the new user admin permissions by running:

```
uaac member add cloud_controller.admin USERNAME
```

Where USERNAME is the username you set in the previous step.

5. Grant the new user permission to read logs from the Loggregator Firehose endpoint by running:

```
uaac member add doppler.firehose USERNAME
```

Where USERNAME is the username you set.

3. Download the OMS Log Analytics Firehose Nozzle BOSH release from Github. Clone the repository and navigate to the oms-log-analytics-firehose-nozzle directory by running:

```
git clone https://github.com/Azure/oms-log-analytics-firehose-nozzle.git
cd oms-log-analytics-firehose-nozzle
```

4. Set the following environment variables in the OMS Log Analytics Firehose Nozzle manifest:

Environment Variable	Description

`applications:`
`- name: oms_nozzle`
`...`
`env:`
`  OMS_WORKSPACE: YOUR-WORKSPACE-ID`
`  OMS_KEY: YOUR-OMS-KEY`
	Enter the ID and key value for your OMS workspace.

`OMS_POST_TIMEOUT: 10s`	(Optional) Set the HTTP post timeout for sending events to OMS LogmAnalytics. The default value is 10 seconds.
`OMS_BATCH_TIME: 10s`	(Optional) Set the interval for posting a batch to OMS. The default value is 10 seconds. For more information, see Configure Additional Logging.
`OMS_MAX_MSG_NUM_PER_BATCH: 1000`	(Optional) Set the maximum number of messages to include in an OMS batch. The default amount is 1000. For more information, see Configure Additional Logging.

`FIREHOSE_USER: YOUR-FIREHOSE-USER`
`FIREHOSE_USER_PASSWORD: YOUR-FIREHOSE-PASSWORD`
	Enter the username and password for the Firehose user you created in Step 2c.

`API_ADDR: https://api.YOUR-DOMAIN`	Enter the URL of your API endpoint.
`DOPPLER_ADDR: wss://doppler.YOUR-DOMAIN:443`	Enter the URL of your Loggregator Traffic Controller endpoint.
`EVENT_FILTER: YOUR-LIST`	(Optional) Enter the event types you want to filter out in a comma-separated list. The valid event types are `METRIC`, `LOG`, and `HTTP`.
`IDLE_TIMEOUT: 60s`	(Optional) Set the duration for the Firehose keep-alive connection. The default time is 60 seconds.
`SKIP_SSL_VALIDATION: TRUE-OR-FALSE`	Set this value to `TRUE` to allow insecure connections to the UAA and the Traffic Controller. To block insecure connections to the UAA and Traffic Controller, set this value to `FALSE`.
`LOG_LEVEL: INFO`	(Optional) Change this value to increase or decrease the amount of logs. Valid log levels in increasing order include `INFO`, `ERROR`, and `DEBUG`. The default value is `INFO`.
`LOG_EVENT_COUNT: TRUE-OR-FALSE`	Set this value to `TRUE` to log the total count of events that the nozzle has sent, received, and lost. OMS logs this value as `CounterEvents`. For more information, see Configure Additional Logging.
`LOG_EVENT_COUNT_INTERVAL: 60s`	(Optional) Set the time interval for logging the event count to OMS. The default interval is 60 seconds. For more information, see Configure Additional Logging.

5. Push the app by running:

   ```
   cf push
   ```

Step 3: View logs in OMS Portal

Import the TAS for VMs OMS view to your OMS Portal to view visualized logs and metrics. You can also create alert rules for specific events.

> **Note:** The OMS view of TAS for VMs is not yet available in the OMS Solutions Gallery. You can add it manually to view your logs in OMS Portal.

Import the OMS view

To import the OMS view:

1. From the main OMS Overview page, go to **View Designer**.
2. Click **Import**.
3. Click **Browse**.
4. Select the **Cloud Foundry (Preview).omsview** file.
5. Save the view. The main OMS Overview page displays the **Tile**.
6. Click the **Tile** to view visualized metrics.

For more information, see Create custom views by using View Designer in Azure Monitor in the Azure documentation.

Create alert rules

For more information about OMS Log Analytics alerts, see Overview of alerts in Microsoft Azure in the Azure documentation.

Set alert queries

This section includes example queries that operators can set in the OMS Portal.

- The following query alerts the operator when the nozzle sends a `slowConsumerAlert` to OMS:

  ```
  Type=CF_ValueMetric_CL Name_s=slowConsumerAlert
  ```

- The following query alerts the operator when Loggregator sends an `LGR` to indicate problems with the logging process:

  ```
  Type=CF_LogMessage_CL SourceType_s=LGR MessageType_s=ERR
  ```

- The following query alerts the operator when the number of lost events reaches a certain threshold, specified in the OMS Portal:

  ```
  Type=CF_CounterEvent_CL Job_s=nozzle Name_s=eventsLost
  ```

- The following query alerts the operator when the nozzle receives the `TruncatingBuffer.DroppedMessages` **CounterEvent**:

  ```
  Type=CF_CounterEvent_CL Name_s="TruncatingBuffer.DroppedMessages"
  ```

Step 4: Configure additional logging (optional)

OMS Log Analytics Firehose Nozzle forwards metrics from the Loggregator Firehose to OMS with minimal processing, but the nozzle can push additional metrics to OMS.

Logs sent and received, and events lost

If you set the `LOG_EVENT_COUNT` environment variable to `TRUE` in the manifest, the nozzle periodically sends the count of sent, received, and lost events to OMS. The value you set for the `LOG_EVENT_COUNT_INTERVAL` determines how frequently the nozzle sends the count.

> **Note:** The nozzle does not count **CounterEvents** themselves in the sent, received, or lost event count.

The nozzle sends the count as a **CounterEvent** with a **CounterKey** of one of the following:

CounterEvent	CounterKey
nozzle.stats.eventsReceived	The number of events the Firehose has received during the interval
nozzle.stats.eventsSent	The number of events the nozzle has successfully sent to OMS during the interval
nozzle.stats.eventsLost	The number of events the nozzle has tried to send to OMS during the interval, but failed to send after 4 attempts

In most cases, the total count of `eventsSent` plus `eventsLost` is less than the total `eventsReceived` at the same time. The nozzle buffers some messages and posts them in a batch to OMS. Operators can adjust the buffer size by adjusting the `OMS_BATCH_TIME` and `OMS_MAX_MSG_NUM_PER_BATCH` environment variables in the manifest).

Log slow consumer alerts

> **Note:** The nozzle does not count **ValueMetrics** in the sent, received, or lost event count.

Loggregator sends the nozzle a `slowConsumerAlert` in the following situations:

- WebSocket sends the error code `ClosePolicyViolation (1008)`.
- The nozzle receives a **CounterEvent** with the value `TruncatingBuffer.DroppedMessages`.

In either case, the nozzle sends the `slowConsumerAlert` event to OMS as the following **ValueMetric**:

ValueMetric	MetricKey
nozzle.alert.slowConsumerAlert	1

For more information, see the Slow Nozzle Alerts section of the *Loggregator Guide for TAS for VMs Operators* topic.

Step 5: Scale the deployment (optional)

Scale the nozzle

If the nozzle is unable to keep up with processing logs from the Firehose, Loggregator alerts the nozzle. When the nozzle receives the alert, it sends a `slowConsumerAlert` to OMS. If this happens, scaling up the nozzle minimizes data loss.

If an operator chooses to scale up their deployment, the Firehose evenly distributes events across all instances of the nozzle. For more information, see the Scaling Nozzles section of the *Loggregator Guide for TAS for VMs Operators* topic.

Operators can create an alert rule for the `slowConsumerAlert` message. For more information, see Create Alert Rules.

Scale Loggregator

Loggregator sends `LGR` log messages to indicate problems with the logging process. For more information, see Scaling Loggregator in *Loggregator Guide for TAS for VMs Operators*.

Operators can create an alert rule for the `LGR` message. For more information, see Create Alert Rules.

SSH for Apps and Services

In this section:

- App SSH Overview
- Accessing Apps with SSH
- Accessing Services with SSH

Configuring SSH access for your deployment

If you need to troubleshoot an instance of an app, you can gain SSH access to the app using the SSH proxy and daemon. For example, one of your app instances might be unresponsive, or the log output from the app is inconsistent or incomplete. You can SSH into the individual VM to troubleshoot the problem instance.

About SSH access

The SSH system components include the SSH proxy and daemon, and the system also supports authentication and load balancing of incoming SSH traffic. For a conceptual overview, see App SSH Components and Processes.

SSH access control hierarchy

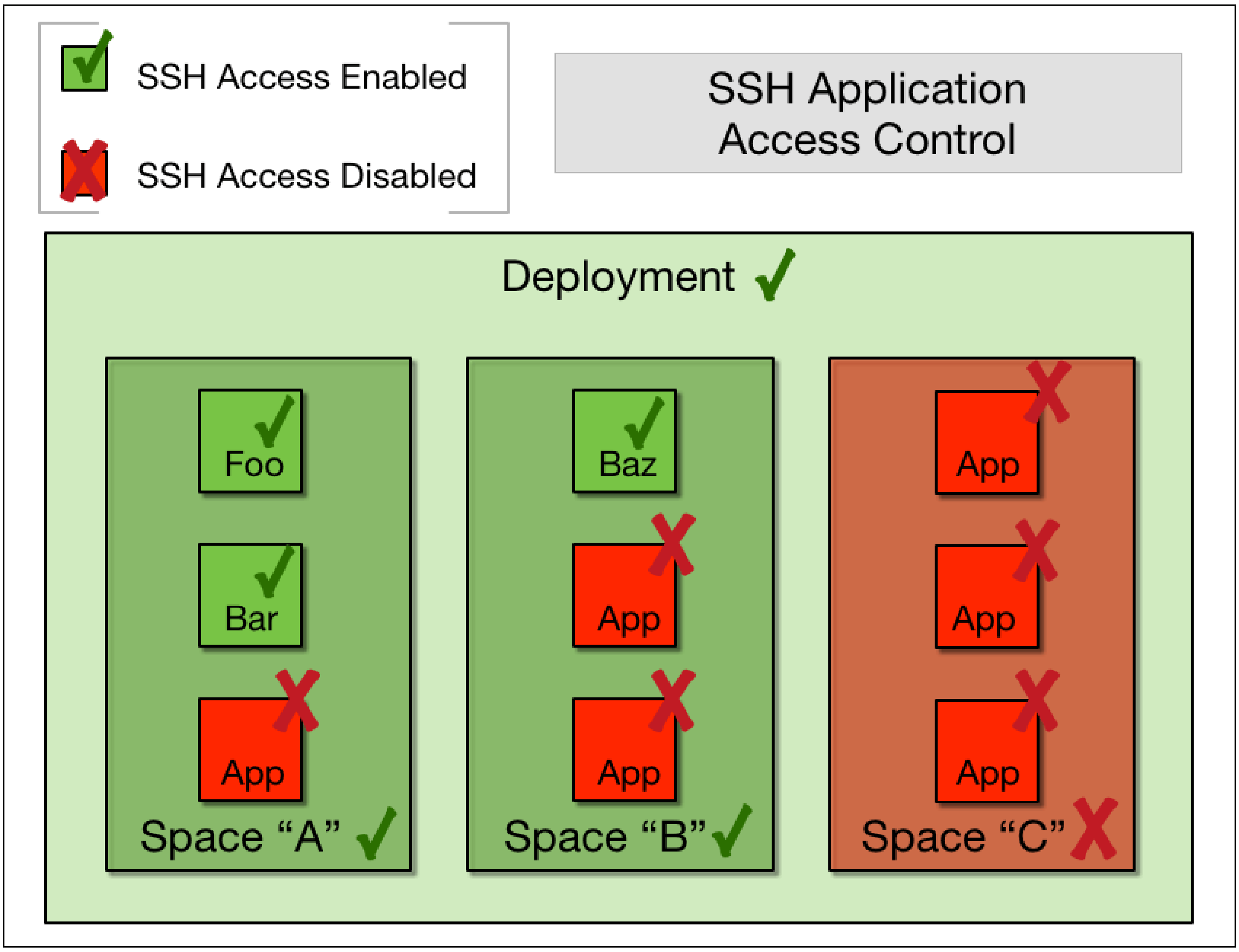
Operators, space managers, and space developers can configure SSH access for TAS for VMs, for spaces, and for apps as described in this table:

User Role	Scope of SSH Permissions Control	How They Define SSH Permissions
Operator	Entire deployment	Configure the deployment to allow or prohibit SSH access (one-time). For more information, see Configuring SSH Access for TAS for VMs.
Space Manager	Space	cf CLI allow-space-ssh and disallow-space-ssh commands
Space Developer	App	cf CLI enable-ssh and disable-ssh commands

An app is SSH-accessible only if operators, space managers, and space developers all grant SSH access at their respective levels. For example, the image below shows a deployment where:

- An operator allowed SSH access at the deployment level.
- A space manager allowed SSH access for apps running in spaces “A” and “B” but not “C”.
- A space developer enabled SSH access for apps that include “Foo”, “Bar”, and “Baz”.

As a result, apps “Foo”, “Bar”, and “Baz” accept SSH requests.



Space A has SSH Access Enabled, indicated by a green checkmark, for apps “Foo” and “Bar”. Space A does not have SSH Access Enabled for the third app, indicated by a red X.

Space B has has SSH Access Enabled, indicated by a green checkmark, for app “Baz”. Space B does not have SSH Access Enabled for the other two apps, indicated by a red X.

Space C does not have SSH Access Enabled for all three apps, indicated by a red X.

SSH access for apps and spaces

Space managers and space developers can configure SSH access from the command line. The Cloud Foundry Command Line Interface (cf CLI) also includes commands to return the value of the SSH access setting. To use and configure SSH at both the app level and the space level, see Accessing Apps with Diego SSH.

Configuring SSH access for VMware Tanzu Application Service for VMs

Ops Manager deployments control SSH access to apps at the TAS for VMs level. Additionally, TAS for VMs supports load balancing of SSH sessions with your load balancer. For information about setting SSH access for your deployment, see Configuring SSH Access.

Accessing your apps with SSH

The Cloud Foundry Command Line Interface (cf CLI) lets you securely log into remote host virtual machines (VMs) running VMware Tanzu Application Service for VMs (TAS for VMs) app instances. The commands that enable SSH access to apps, and enable, deactivate, and check permissions for such access are described here.

This page assumes you are using the Latest version of the cf CLI.

> **Note:** The `cf ssh` command in cf CLI v7 and the `cf v3-ssh` command in cf CLI v6 include the `all_proxy` environment variable, which allows you to specify a proxy server to activate proxying for all requests. For more information, see ssh in the Cloud Foundry CLI Reference Guide and Use SOCKS5 with cf v3-ssh in *Using the cf CLI with a Proxy Server*.

The cf CLI looks up the `app_ssh_oauth_client` identifier in the Cloud Controller `/v2/info` endpoint, and uses this identifier to query the UAA server for an SSH authorization code. On the target VM side, the SSH proxy contacts the Cloud Controller through the `app_ssh_endpoint` listed in `/v2/info` to confirm permission for SSH access.

App SSH Commands

cf CLI Command	Purpose

`cf enable-ssh`
`cf disable-ssh`
`cf allow-space-ssh`
`cf disallow-space-ssh`
	Enable and Disable SSH Access

`cf ssh-enabled`
`cf space-ssh-allowed`
	Check SSH Access Permissions

`cf ssh`	Log Into an App Container with cf SSH
`cf ssh-code`	App SSH Access without cf CLI using non-`cf SSH` tools like `ssh`, `scp`, and `sftp`

Enable and Disable SSH Access

A cloud operator can deploy TAS for VMs to either allow or prohibit app SSH across the entire deployment. For more information, see Configuring SSH Access for TAS for VMs.

Within a deployment that permits SSH access to apps, Space Developers can enable or disable SSH access to individual apps, and Space Managers can enable or disable SSH access to all apps running within a space.

You must restart your app after enabling SSH access.

Configuring SSH Access at the App Level

cf enable-ssh enables SSH access to all instances of an app:

```
$ cf enable-ssh MY-AWESOME-APP
```

cf disable-ssh disables SSH access to all instances of an app:

```
$ cf disable-ssh MY-AWESOME-APP
```

Configuring SSH Access at the Space Level

cf allow-space-ssh allows SSH access into all apps in a space:

```
$ cf allow-space-ssh SPACE-NAME
```

cf disallow-space-ssh disallows SSH access into all apps in a space:

```
$ cf disallow-space-ssh SPACE-NAME
```

Check SSH Permissions

cf ssh-enabled checks whether an app is accessible with SSH:

```
$ cf ssh-enabled MY-AWESOME-APP
ssh support is disabled for 'MY-AWESOME-APP'
```

cf space-ssh-allowed checks whether all apps running within a space are accessible with SSH:

```
$ cf space-ssh-allowed SPACE-NAME
ssh support is enabled in space 'SPACE-NAME'
```

Log Into an App Container with cf SSH

If SSH access is allowed at the deployment, space, and app level, you can run the `cf ssh APP-NAME` command to start an interactive SSH session with a VM hosting an app. By default, the command accesses the container running the first instance of the app, the instance with index **0**.

```
$ cf ssh MY-AWESOME-APP
```

When logged into a VM hosting an app, you can use tools like the Cloud Foundry Diego Operator Toolkit (CFDOT) to run app status diagnostics. For more information, see How to use Cloud Foundry Diego Operator Toolkit (CFDOT) in the Knowledge Base.

Common cf SSH Flags

You can tailor cf ssh commands with the following flags, most of which mimic flags for the Unix or Linux `ssh` command. Run the `cf ssh --help` command for more details.

- The `-i` flag targets a specific instance of an app. To log into the VM container hosting the third instance, `index=2`, of MY-AWESOME-APP, run:

  ```
  $ cf ssh MY-AWESOME-APP -i 2
  ```

- The `-L` flag enables local port forwarding, binding an output port on your machine to an input port on the app VM. Pass in a local port, and your app VM port and port number, all colon delimited. You can prepend your local network interface, or use the default `localhost`.

```
$ cf ssh MY-AWESOME-APP -L [LOCAL-NETWORK-INTERFACE:]LOCAL-PORT:REMOTE-HOST-NAME:REMOTE-HOST-PORT
```

- The `-N` flag skips returning a command prompt on the remote machine. This sets up local port forwarding if you do not need to execute commands on the host VM.
- The `--process` flag in cf CLI v7 allows you to SSH into the container for a specific process running as part of your app.
- The `--request-pseudo-tty` and `--force-pseudo-tty` flags let you run an SSH session in pseudo-tty mode rather than generate terminal line output.

SSH Session Environment

If you want the environment of your interactive SSH session to match the environment of your buildpack-based app, with the same environment variables and working directory, run the following commands after starting the session:

```
/tmp/lifecycle/shell
```

After running the above command, the value of the `VCAP_APPLICATION` environment variable differs slightly from its value in the environment of the app process, as it does not have the `host`, `instance_id`, `instance_index`, or `port` fields set. These fields are available in other environment variables, as described in VCAP_APPLICATION in *TAS for VMs Environment Variables*.

App SSH Access without cf CLI

In addition to `cf ssh`, you can use other SSH clients such as `ssh`, `scp`, or `sftp` to access your app, if you have SSH permissions.

Follow one of the processes below to securely connect to an app instance by logging in with a specially-formed username that passes information to the SSH proxy running on the host VM. For the password, use a one-time SSH authorization code generated by cf ssh-code.

- Access App SSH Using Process GUID
- Access App SSH Using App GUID

Access App SSH Using Process GUID

1. Query the `/v2/info` endpoint of the Cloud Controller in your deployment. Record the domain name and port number of the `app_ssh_endpoint` field, and the `app_ssh_host_key_fingerprint` field. You will compare the `app_ssh_host_key_fingerprint` with the fingerprint returned by the SSH proxy on your target VM. For example:

```
$ cf curl /v2/info
{
     ...
     "app_ssh_endpoint": "ssh.example.com:2222",
    "app_ssh_host_key_fingerprint": "a6:14:c0:ea:42:07:b2:f7:53:2c:0b:
60:e0:00:21:6c",
    ...
}
```

In this example:

- The domain name is `ssh.example.com`.
- The port number is `2222`.
- The fingerprint is `a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c`.

2. Run the following command:

```
ssh -p PORT-NUMBER cf:$(cf curl /v3/apps/$(cf app APP-NAME --guid)/processes |
jq -r '.resources[] | select(.type=="web") | .guid')/0@SSH-ENDPOINT
```

Where:

- `PORT-NUMBER` is the port number of the `app_ssh_endpoint` field that you recorded in the previous step.
- `APP-NAME` is the name of your target app.
- `SSH-ENDPOINT` is the domain name of the `app_ssh_endpoint` field that you recorded in theprevious step.

For example:

```
ssh -p 2222 cf:$(cf curl /v3/apps/$(cf app my-app --guid)/processes | j
q -r '.resources[] | select(.type=="web") | .guid')/0@ssh.example.com
```

3. Run `cf ssh-code` to obtain a one-time authorization code that substitutes for an SSH password. You can run `cf ssh-code | pbcopy` to automatically copy the code to the clipboard. For example:

```
$ cf ssh-code
E1x89n
```

4. When the SSH proxy reports its RSA fingerprint, confirm that it matches the `app_ssh_host_key_fingerprint` recorded above. When prompted for a password, paste in the authorization code returned by `cf ssh-code`. For example:

```
$ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.c
om
The authenticity of host '[ssh.example.com]:2222 ([203.0.113.5]:2222)'
can't be established.
RSA key fingerprint is a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[ssh.example.com]:2222 [203.0.113.5]:2222'
(RSA) to the list of known hosts.
cf:d0a2e11d-e6ca-4120-b32d-140@ssh.ketchup.cf-app.com's password:
vcap@ce4l5164kws:~$
```

You have now securely connected to the app instance.

Access App SSH Using App GUID

1. Display the GUID of your target app by running:

   ```
   cf app APP-NAME --guid`
   ```

   Where `APP-NAME` is the name of the app.

   For example:

   ```
   $ cf app my-app --guid
   abcdefab-1234-5678-abcd-1234abcd1234
   ```

2. Query the `/v2/info` endpoint of the Cloud Controller in your deployment. Record the domain name and port number of the `app_ssh_endpoint` field, and the `app_ssh_host_key_fingerprint` field. You will compare the `app_ssh_host_key_fingerprint` with the fingerprint returned by the SSH proxy on your target VM. For example:

   ```
   $ cf curl /v2/info
   {
        ...
        "app_ssh_endpoint": "ssh.example.com:2222",
        "app_ssh_host_key_fingerprint": "a6:14:c0:ea:42:07:b2:f7:53:2c:0b:
   60:e0:00:21:6c",
        ...
   }
   ```

   In this example:

   - The domain name is `ssh.example.com`.
   - The port number is `2222`.
   - The fingerprint is `a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c`.

3. Run `cf ssh-code` to obtain a one-time authorization code that substitutes for an SSH password. You can run `cf ssh-code | pbcopy` to automatically copy the code to the clipboard. For example:

   ```
   $ cf ssh-code
   E1x89n
   ```

4. Run your `ssh` or other command to connect to the app instance.

   - SSH into the container hosting the first instance of your app by running the following command. For the username, use a string of the form `cf:APP-GUID/APP-INSTANCE-INDEX@SSH-ENDPOINT`:

     ```
     ssh -p `SSH-PORT` cf:APP-GUID/APP-INSTANCE-INDEX@SSH-ENDPOINT
     ```

     Where:

- `SSH-PORT` is the port number recorded in earlier steps.
- `APP-GUID` comes from the previous steps.
- `APP-INSTANCE-INDEX` is the index of the instance that you want to access.
- `SSH-ENDPOINT` comes from the previous steps and is in the form `ssh.MY-DOMAIN.com`.

For example:

```
$ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.examp
le.com
```

- Or you can use `scp` to transfer files by running one of the following commands:

```
scp -P `SSH-PORT` -o User=cf:APP-GUID/APP-INSTANCE-INDEX ssh.MY-DOMAIN.co
m:REMOTE-FILE-TO-RETRIEVE LOCAL-FILE-DESTINATION
```

```
scp -P `SSH-PORT` -o User=cf:APP-GUID/APP-INSTANCE-INDEX LOCAL-FILE-TO-CO
PY ssh.MY-DOMAIN.com:REMOTE-FILE-DESTINATION
```

- Or you can use `ssh` piped with `cat` command to transfer file by running:

```
cat local_file_path | cf ssh MY-AWESOME-APP -c "cat > remote_file_path"
```

5. When the SSH proxy reports its RSA fingerprint, confirm that it matches the `app_ssh_host_key_fingerprint` recorded above. When prompted for a password, paste in the authorization code returned by `cf ssh-code`, for example:

```
$ ssh -p 2222 cf:abcdefab-1234-5678-abcd-1234abcd1234/0@ssh.MY-DOMAIN.c
om
The authenticity of host '[ssh.MY-DOMAIN.com]:2222 ([203.0.113.5]:222
2)' can't be established.
RSA key fingerprint is a6:14:c0:ea:42:07:b2:f7:53:2c:0b:60:e0:00:21:6c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[ssh.MY-DOMAIN.com]:2222 [203.0.113.5]:222
2' (RSA) to the list of known hosts.
cf:d0a2e11d-e6ca-4120-b32d-140@ssh.ketchup.cf-app.com's password:
vcap@ce4l5164kws:~$
```

You have now securely connected to the app instance.

SSH Proxy Security Configuration

The SSH proxy has the following SSH security configuration by default:

Security Parameter	Values

Ciphers	`chacha20-poly1305@openssh.com`
	`aes128-gcm@openssh.com`
	`aes256-ctr`
	`aes192-ctr`
	`aes128-ctr`

MACs	hmac-sha2-256-etm@openssh.com hmac-sha2-256
Key Exchanges	curve25519-sha256@libssh.org

The `cf ssh` command is compatible with this security configuration. If you use a different SSH client to access apps over SSH, you should ensure that you configure your client to be compatible with these ciphers, MACs, and key exchanges. For more information about other SSH clients, see App SSH Access without cf CLI above.

Proxy to Container Authentication

A second layer of SSH security runs within each container. When the SSH proxy attempts to handshake with the SSH daemon inside the target container, it uses the following fields associated with the `diego-ssh` key in its route to the app instance. This inner layer works invisibly and requires no user action, but is described here to complete the SSH security picture.

CONTAINER_PORT (required)

`container_port` indicates which port inside the container the SSH daemon is listening on. The proxy attempts to connect to host side mapping of this port after authenticating the client.

HOST_FINGERPRINT (optional)

When present, `host_fingerprint` declares the expected fingerprint of the SSH daemon's host public key. When the fingerprint of the actual target's host key does not match the expected fingerprint, the connection is terminated. The fingerprint should only contain the hex string generated by `ssh-keygen -l`.

USER (optional)

`user` declares the user ID to use during authentication with the container's SSH daemon. While this is not a required part of the routing data, it is required for password authentication and may be required for public key authentication.

PASSWORD (optional)

`password` declares the password to use during password authentication with the container's ssh daemon.

PRIVATE_KEY (optional)

`private_key` declares the private key to use when authenticating with the container's SSH daemon. If present, the key must be a PEM encoded RSA or DSA public key.

Example App Process

```
{
  "process_guid": "ssh-process-guid",
  "domain": "ssh-experiments",
  "rootfs": "preloaded:cflinuxfs3",
  "instances": 1,
  "start_timeout": 30,
  "setup": {
    "download": {
      "artifact": "diego-sshd",
      "from": "http://file-server.service.cf.internal.example.com:8080/v1/static/diego
-sshd/diego-sshd.tgz",
      "to": "/tmp",
      "cache_key": "diego-sshd"
    }
  },
  "action": {
    "run": {
      "path": "/tmp/diego-sshd",
      "args": [
          "-address=0.0.0.0:2222",
          "-authorizedKey=ssh-rsa ..."
      ],
      "env": [],
      "resource_limits": {}
    }
  },
  "ports": [ 2222 ],
  "routes": {
    "diego-ssh": {
      "container_port": 2222,
      "private_key": "PEM encoded PKCS#1 private key"
    }
  }
}
```

Daemon Discovery

To be accessible through the SSH proxy, containers must host an SSH daemon, expose it through a mapped port, and advertise the port in a `diego-ssh` route. If a proxy cannot find the target process or a route, user authentication fails.

```
  "routes": {
    "diego-ssh": { "container_port": 2222 }
  }
```

The Diego system generates the appropriate process definitions for TAS for VMs apps which reflect the policies that are in effect.

Accessing services with SSH

You can gain direct command line access to your deployed service instance using SSH. This allows you to, for example, access your database to run raw SQL commands to edit the schema, import and export data, or debug app data issues.

To establish direct command line access to a service, you deploy a host app and use its SSH and port forwarding features to communicate with the service instance through the app container. The technique described here works with TCP services such as MySQL or Redis.

This topic assumes you are using Cloud Foundry Command Line Interface (cf CLI) v6.15.0 or later, but at least v7 is recommended.

Note: The procedure in this topic requires use of a service key, and not all services support service keys. Some services support credentials through app binding only.

Create a Service Instance

1. In your terminal window, log in to your deployment with `cf login`.
2. List the marketplace services installed as product tiles in your Ops Manager deployment. If you need to add the service as a tile, see Adding and Deleting Products. In this example, you create a p-mysql service instance.

   ```
   $ cf marketplace
   p-mysql  100mb MySQL databases on demand
   ```

3. Create your service instance. As part of the create-service command, indicate the service name, the service plan, and the name you choose for your service instance.

   ```
   $ cf create-service p-mysql 100mb MY-DB
   ```

Push Your Host App

To push an app that will act as the host for the SSH tunnel, push any app that will successfully deploy to VMware Tanzu Application Service for VMs.

Note: Your app must be prepared before you push it. See the Pushing an App topic for details on preparing apps for pushing.

1. Push your app:

   ```
   cf push YOUR-HOST-APP
   ```

2. Enable SSH for your app:

   ```
   cf enable-ssh YOUR-HOST-APP
   ```

Note: To enable SSH access to your app, SSH access must also be enabled for both the space that contains the app and VMware Tanzu Application Service for VMs. For more information, see App SSH Overview.

Create Your Service Key

To establish SSH access to your service instance, you must create a service key that contains critical information for configuring your SSH tunnel.

1. Create a service key for your service instance using the cf create-service-key command.

   ```
   cf create-service-key MY-DB EXTERNAL-ACCESS-KEY
   ```

2. Retrieve your new service key using the cf service-key command.

   ```
   cf service-key MY-DB EXTERNAL-ACCESS-KEY
   ```

   For example:

   ```
   $ cf service-key MY-DB EXTERNAL-ACCESS-KEY
   Getting key EXTERNAL-ACCESS-KEY for service instance MY-DB as user@exam
   ple.com

   {
   "hostname": "us-cdbr-iron-east-01.p-mysql.net",
   "jdbcUrl": "jdbc:mysql://us-cdbr-iron-east-03.p-mysql.net/ad\_b2fca6t49
   704585d?user=b5136e448be920\u0026password=231f435o05",
   "name": "ad\_b2fca6t49704585d",
   "password": "231f435o05",
   "port": "3306",
   "uri": "mysql://b5136e448be920:231f435o05@us-cdbr-iron-east-03.p-mysql.
   net:3306/ad\_b2fca6t49704585d?reconnect=true",
   "username": "b5136e448be920"
   }
   ```

Configure Your SSH Tunnel

Configure an SSH tunnel to your service instance using cf ssh. Tailor the example command below with information from your service key.

```
$ cf ssh -L 63306:us-cdbr-iron-east-01.p-mysql.net:3306 YOUR-HOST-APP
```

- Use any available local port for port forwarding. For example, `63306`.
- Replace `us-cdbr-iron-east-01.p-mysql.net` with the address provided under `hostname` in the service key retrieved above.
- Replace `3306` with the port provided under `port` above.
- Replace `YOUR-HOST-APP` with the name of your host app.

After you enter the command, open another terminal window and perform the steps below in Access Your Service Instance.

Access Your Service Instance

To establish direct command-line access to your service instance, use the relevant command line tool for that service. This example uses the MySQL command line client to access the p-mysql service instance.

```
$ mysql -u b5136e448be920 -h 0 -p -D ad_b2fca6t49704585d -P 63306
```

- Replace `b5136e448be920` with the username provided under `username` in your service key.
- `-h 0` instructs `mysql` to connect to your local machine (use `-h 127.0.0.1` for Windows).
- `-p` instructs `mysql` to prompt for a password. When prompted, use the password provided under `password` in your service key.
- Replace `ad_b2fca6t49704585d` with the database name provided under `name` in your service key.
- `-P 63306` instructs `mysql` to connect on port `63306`.

Buildpacks

In this section:

System Buildpacks

In this section:

System buildpacks in Cloud Foundry

This topic tells you about the existing system buildpacks that Cloud Foundry supports.

Cloud Foundry includes a set of system buildpacks for common languages and frameworks.

The following table lists the system buildpacks:

Name	Supported Languages, Frameworks, and Technologies	GitHub Repository
Binary	*n/a*	Binary source
Go	Go	Go source
HWC	HWC	HWC source
Java	Grails, Play, Spring, or any other JVM-based language or framework	Java source
.NET Core	.NET Core	.NET Core source
NGINX	NGINX	NGINX source
Node.js	Node or JavaScript	Node.js source
PHP	Cake, Symfony, Zend, NGINX, or HTTPD	PHP source
Python	Django or Flask	Python source
R	R	R source
Ruby	Ruby, JRuby, Rack, Rails, or Sinatra	Ruby source
Staticfile	HTML, CSS, JavaScript, or NGINX	Staticfile source

Binary buildpack

Use the binary buildpack to run arbitrary binary web servers.

Push an app

Specify the binary buildpack to stage an app as a binary file. On a command line, use `cf push MY-AWESOME-APP` with the `-b` option to specify the buildpack.

For example:

```
$ cf push MY-AWESOME-APP -b https://github.com/cloudfoundry/binary-buildpack.git
```

You can provide Cloud Foundry with the shell command to run your binary in the following two ways:

- **Procfile**: In the root directory of your app, add a `Procfile` that specifies a `web` task:

  ```
  web: ./app
  ```

- **Command line**: Use `cf push MY-AWESOME-APP` with the `-c` option:

  ```
  $ cf push MY-AWESOME-APP -c './app' -b binary_buildpack
  ```

Compile your binary

Cloud Foundry expects your binary to bind to the port specified by the `PORT` environment variable.

The following example in Go binds a binary to the PORT environment variable:

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %s", "world!")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":"+os.Getenv("PORT"), nil)
}
```

Your binary should run without any additional runtime dependencies on the cflinuxfs3 or lucid64 root filesystem (rootfs). Any such dependencies should be statically linked to the binary.

Important

> To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

To boot a Docker container running the cflinuxfs3 filesystem, run the following command:

```
$ docker run -it cloudfoundry/cflinuxfs3 bash
```

To boot a Docker container running the lucid64 filesystem, run the following command:

```
$ docker run -it cloudfoundry/lucid64 bash
```

To compile the above Go application on the rootfs, golang must be installed. `apt-get install golang` and `go build app.go` will produce an `app` binary.

When deploying your binary to Cloud Foundry, use `cf push` with the `-s` option to specify the root filesystem it should run against.

```
$ cf push MY-AWESOME-APP -s (cflinuxfs3|lucid64)
```

BOSH configured custom trusted certificate support

Your platform operator can configure the platform to add the custom certificates into the application container. The custom trusted certificates are added to the `/etc/ssl/certs` directory and can be used by binary applications.

For more information, see Configuring Trusted System Certificates for Applications.

.NET apps

.NET Core on Windows

To run .NET Core apps on the Windows stack, you must use the Binary buildpack. Follow the steps below to configure your Cloud Foundry manifest appropriately.

1. In the app manifest file, specify the `windows` stack and a custom start command. This file should be in the same folder as your `.csproj` file and be marked **copy always**.

   ```
   ---
   applications:
   - name: MY-AWESOME-APP
     stack: windows
     command: cmd /c .\MY-AWESOME-APP --urls=http://0.0.0.0:%PORT%
   ```

2. Publish the project using the dotnet CLI or Visual Studio.

   ```
   $ dotnet publish -o ./publish -r win10-x64 -f netcoreapp2.1
   ```

3. A `publish` folder should now exist. Navigate to this `publish` folder and verify the `manifest.yml` file exists. Run `cf push` to deploy your app.

```
$ cd publish
$ cf push MY-AWESOME-APP
```

.NET Framework

To run a self hosted application using OWIN or another kind of .NET Framework app that requires a TCP port, compile the app to `myapp.exe` and use the Binary buildpack. For an example, see the following manifest:

```
applications:
- name: MY-AWESOME-APP
  stack: windows
  command: cmd /c .\MY-AWESOME-APP --server.urls=http://0.0.0.0:%PORT%
```

Console Apps

Console apps are not recommended to run on Cloud Foundry because they do not normally run as a single instance and are not highly available. If you need to run a console app on Cloud Foundry, see Console Applications in the .NET Cookbook.

For information about deploying different types of .NET apps, follow the links in the table below.

Type of .NET App	Buildpack
ASP.NET MVC ASP.NET Web Forms ASP.NET WebAPI Apps Windows Communication Foundation (WCF)	HWC
.NET Core pushed to Linux stack	.NET Core

Help and support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the binary buildpack in Cloud Foundry, see the binary-buildpack GitHub repository.

You can find current information about this buildpack on the binary buildpack release page in GitHub.

Go buildpack

Supported versions

You can find supported Go versions in the Cloud Foundry release notes.

Push an app

The Go buildpack is automatically detected in the following circumstances:

- Your app has been packaged with godep using `godep save`.
- Your app has a `vendor/` directory and has any files ending with `.go`.

- Your app has a `GOPACKAGENAME` environment variable specified and has any files ending with `.go`.
- Your app has a `glide.yml` file and is using glide, starting in buildpack version 1.7.9.
- Your app has a `Gopkg.toml` file and is using dep, starting in buildpack version 1.8.9.

If your Cloud Foundry deployment does not have the Go Buildpack installed, or the installed version is out of date, you can use the latest version with the command:

```
$ cf push my_app -b https://github.com/cloudfoundry/go-buildpack.git
```

When you specify versions, specify only majoror minor versions, such as Go 1.6, rather than Go 1.6.0. This ensures you receive the most recent patches.

Start command

When pushing Go apps, you can specify a start command for the app. You can place the start command in the `Procfile` file in root directory of your app. For example, if the binary generated by your Go project is `my-go-server`, your `Procfile` could contain the following:

```
web: my-go-server
```

For more information about Procfiles, see the Configuring a Production Server topic.

You can also specify the start command for your app in the `manifest.yml` file in the root directory. For example, your `manifest.yml` could contain the following:

```
---
applications:
  - name: my-app-name
    command: my-go-server
```

If you do not specify a start command in a `Procfile`, in the manifest, or with the `-c` flag for `cf push`, the generated binary is used as the start command. Example: `my-go-server`

Push an app with Go modules

As of Go 1.11 you can use the `go mod` vendoring tool that comes with Go. If you are using go mod to package your dependencies, make sure that you have created a valid `go.mod` file in the root directory of your app by running `go mod init`.

Go sample app

- Sample Go app

An example `go.mod` file:

```
module go-online

require (
        github.com/BurntSushi/toml v0.3.1
        github.com/satori/go.uuid v1.2.0
)
```

When your main package is not in the project root

A common project pattern is to place main packages in a subdirectory called `cmd` like in the example below:

```
$ tree app-package-name
app-package-name
├── cmd
│   ├── cli
│   │   └── main.go
│   └── server
│       └── main.go
├── go.mod
├── go.sum
├── shared.go
├── shared_test.go
└── manifest.yml
```

When you configure your project like this, set the environment variable `GO_INSTALL_PACKAGE_SPEC` to `$MODULE_NAME/$MAIN_PACKAGE_PATH`.

If the module name for the above `app-package-name` app is `example.com/user/app-package-name`, the value of `GO_INSTALL_PACKAGE_SPEC` must be `example.com/user/app-package-name/cmd/server`.

If you want to put this in your `manifest.yml`, see the following example:

```
---
applications:
- name: app-package-name
  env:
    GO_INSTALL_PACKAGE_SPEC: example.com/user/app-package-name/cmd/server
```

Push an app with godep

If you are using godep to package your dependencies, make sure that you have created a valid `Godeps/Godeps.json` file in the root directory of your app by running `godep save`.

When using godep, you can fix your Go version in `GoVersion` key of the `Godeps/Godeps.json` file.

Go sample app

- Sample Go app

An example `Godeps/Godeps.json`:

```
{
    "ImportPath": "go_app",
    "GoVersion": "go1.6",
    "Deps": []
}
```

Push an app with Glide

If you use glide to specify or package your dependencies, make sure that you have created a valid `glide.yml` file in the root directory of your app by running `glide init`.

To vendor your dependencies before pushing, run `glide install`. This generates a `vendor` directory and a `glide.lock` file specifying the latest compatible versions of your dependencies. You must have a `glide.lock` file when pushing a vendored app. You do not need a `glide.lock` file when deploying a non-vendored app.

Glide

- Sample Go app with Glide

An example `glide.yml` file:

```
package: go_app_with_glide
import:
- package: github.com/ZiCog/shiny-thing
  subpackages:
  - foo
```

You can specify Go version in the `manifest.yml` file:

```
---
applications:
  - name: my-app-name
    env:
      GOVERSION: go1.8
```

Push an app with dep

If you use dep to specify or package your dependencies, make sure that you have created a valid `Gopkg.toml` file in the root directory of your app by running `dep init`.

To vendor your dependencies before pushing, run `dep ensure`. This generates a `vendor` directory and a `Gopkg.lock` file specifying the latest compatible versions of your dependencies. You must have a `Gopkg.lock` file when pushing a vendored app. You do not need a `Gopkg.lock` file when deploying a non-vendored app.

dep

- Sample Go app with dep

An example `Gopkg.toml` file:

```
[[constraint]]
  branch = "main"
  name = "github.com/ZiCog/shiny-thing"
```

You can specify Go version in the `manifest.yml` file:

```
---
applications:
  - name: my-app-name
    env:
      GOVERSION: go1.8
```

Push an app without a vendoring tool

You can use Go without a vendoring tool by packaging all local dependencies in the `vendor/` directory, and by specifying the app package name in the `GOPACKAGENAME` environment variable.

An example `manifest.yml`:

```
---
applications:
  - name: my-app-name
    command: go-online
    env:
      GOPACKAGENAME: example.com/user/app-package-name
```

Pass a symbol and string to the Go Linker

The Go buildpack supports the Go linker's ability, `-X symbol value`, to set the value of a string at link time. Set the `GO_LINKER_SYMBOL` and `GO_LINKER_VALUE` in the application's configuration before pushing code.

This can be used to embed the commit SHA or other build-specific data directly into the compiled executable.

For a sample Go app, see the go-buildpack repository on GitHub.

C dependencies

The Go buildpack supports building with C dependencies using cgo. You can set config vars to specify cgo flags to, for example, specify paths for vendored dependencies. As an example, to build gopgsqldriver, add the config var `CGO_CFLAGS` with the value `-I/app/code/vendor/include/postgresql` and include the relevant Postgres header files in `vendor/include/postgresql/` in your app.

Proxy support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH configured custom trusted certificate support

Go uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Go buildpack in Cloud Foundry, see the go-buildpack GitHub repository.

You can find current information about this buildpack on the Go buildpack release page in GitHub.

HWC (.NET Framework)

In this section:

- HWC Buildpack
- Creating an Extension Buildpack for .NET Apps
- Tips for .NET Framework Developers

Hosted Web Core buildpack

You can configure .NET Framework apps to use with the Hosted Web Core (HWC) buildpack and push your .NET Framework apps to VMware Tanzu Application Service for VMs [Windows].

Prerequisites

Using the HWC buildpack requires deploying Microsoft Windows cells with TAS for VMs [Windows].

The HWC buildpack supports the following common app types by default:

- ASP.NET MVC
- ASP.NET Web Forms
- ASP.NET WebAPI Apps
- Windows Communication Foundation (WCF)

For information about deploying different types of .NET apps, follow the links in the following table:

Type of .NET App	Buildpack
.NET Console	Binary
.NET Core pushed to Linux stack	.NET Core
.NET Core pushed to Windows stack	Binary

HWC buildpack overview

The HWC buildpack provides a runtime server that uses the Hosted Web Core API for running .NET Framework applications in Windows Server containers. For more information, see Hosted Web Core API Reference in the Microsoft documentation.

The HWC buildpack provides access to .NET Framework 4.5.1 and later, made available by the Windows root file system (rootfs).

Before you push your first app using the HWC buildpack, see the Getting Started guide in the *.NET Cookbook*.

Step 1. Configure HWC

HWC relies on `Web.config` and `applicationHost.config` configuration files for configuring the .NET applications.

Most `Web.config` files work immediately with TAS for VMs [Windows], but with the following constraints:

- Integrated Windows Authentication (IWA) is not yet supported on TAS for VMs [Windows].
- SQL server connection strings must use fully qualified domain names.
- Place connection string values in environment variables or user-provided service instances.

In addition, the Shadow Copy Setting, and Dynamic and Static HTTP Compression `Web.config` settings can be customized as needed.

The HWC buildpack includes a default configuration for the `applicationHost.config`, similar to IIS.

Step 2. Add a global error handler

Before you push your app for the first time, add a global error handler to receive log information from your app if it crashes on startup.

To configure a global error handler that logs to `stdout`, see Application Error Handling in the *.NET Cookbook*.

Step 3. Pushing an app

Follow these steps to push your application:

1. Build your HWC app in Visual Studio.
2. On the command line, go to the directory containing the app files.
3. To push your HWC app, run the following `cf push` command:

   ```
   cf push APP-NAME -s windows -b hwc_buildpack
   ```

   Where `APP-NAME` is the name you want to give your app.
   For example:

   ```
   $ cf push my-app -s windows -b hwc_buildpack
   Creating app my-app in org sample-org / space sample-space as username@
   example.com...
   OK
   ...
   requested state: started
   instances: 1/1
   usage: 1 GB x 1 instances
   urls: my-app.example.com
   ```

4. Confirm your application is running by going to your app's URL in the push command output. In the previous example, `my-app.example.com` is the URL of your app.

Features

You can use the following features with HWC buildpack:

- Context Path Routing

- Shadow Copy Setting
- Dynamic and Static HTTP Compression
- URL Rewrite
- Profile Scripts

Context path routing

With context path routing you can implement multiple apps to share the same route hostname. For example, `app1.example.com/app2`. The context path routing feature is analogous to IIS virtual directories.

Making an application accessible under another app’s URL requires pushing both apps and applying a map-route correlation between them. To define a context path route, for example, `app1.example.com/app2`, run the following commands:

1. To push the primary app, run the following command:

   ```
   cf push TOP-LEVEL-APP-NAME -s windows -b hwc_buildpack
   ```

   Where `TOP-LEVEL-APP-NAME` is your top-level app’s name.

2. To push the secondary app and deactivate the app’s starting and default routing, run the following command:

   ```
   cf push LOWER-LEVEL-APP-NAME --no-start --no-route -s windows -b hwc_buildpack
   ```

   Where `LOWER-LEVEL-APP-NAME` is the name of the lower-level app.

3. To map routes between the primary and secondary apps, run the following command:

   ```
   cf map-route LOWER-LEVEL-APP-NAME APP-DOMAIN --hostname TOP-LEVEL-APP-NAME --path LOWER-LEVEL-APP-NAME
   ```

   Where:

   - `TOP-LEVEL-APP-NAME` is your top-level app's name.
   - `LOWER-LEVEL-APP-NAME` is your lower-level app's name.
   - `APP-DOMAIN` is your site's public domain name.

4. To start the secondary app, run the following command:

   ```
   cf start LOWER-LEVEL-APP-NAME
   ```

   Where `LOWER-LEVEL-APP-NAME` is your lower-level app’s name.

For example, the following commands define context path routing for two HWC apps, `app1` and `app2`, where `app2` is made accessible under `app1` as `app1.example.com/app2`:

```
$ cf push app1 -s windows -b hwc_buildpack
$ cf push app2 --no-start --no-route -s windows -b hwc_buildpack
$ cf map-route app2 example.com --hostname app1 --path app2
$ cf start app2
```

HWC-hosted apps use the `VCAP_APPLICATION` environment variable to read out the bound app URIs. Any context path that exists underneath the root in the app's bound route corresponds to the `applicationHost.config`.

Shadow Copy setting

Shadow Copy is a hosting option that copies assemblies for an app in the `bin` directory to the app's temporary files directory. This feature is turned off and is unnecessary for apps running under Cloud Foundry. An app can override this setting in its `Web.config` file.

Dynamic and static HTTP compression

The HWC buildpack implements dynamic and static HTTP compression by default. You can deactivate HTTP compression in your app's `Web.config` file.

Dynamic HTTP compression is hardcoded at level 4. Static HTTP compression is hardcoded at level 9.

URL rewrite

The HWC buildpack supports the URL Rewrite module. It's preinstalled in the Windows file system.

Profile scripts

With the HWC buildpack you can provide `.profile.bat` scripts with your applications. You can use a `.profile.bat` script to perform app-specific initialization tasks. For example, setting custom environment variables.

For information about configuring `.profile.bat` scripts, see Configure Pre-Runtime Hooks section of *Pushing an App*.

Buildpack support

A number of channels exist to assist you with using the HWC buildpack, or developing your own HWC buildpack.

- **HWC Buildpack Repository in GitHub**: For more information about using and extending the HWC buildpack in the [HWC buildpack repository] see, (https://github.com/cloudfoundry/hwc-buildpack) in GitHub.
- **Release Notes**: For more information about this buildpack, see HWC buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in the Cloud Foundry Slack community.

Creating an extension buildpack for .NET apps

You can write extension buildpacks for your .NET apps.

Most .NET Framework apps are pushed with the Hosted Web Core (HWC) buildpack.

The HWC buildpack contains the commonly required .NET dependencies used by .NET apps. This set of HWC buildpack dependencies is a subset entire gamut of possible .NET dependencies and does not contain all the DLLs that every app could need.

An extension buildpack supplies a custom set of .NET dependencies to an app's container.

You cannot add modules or extensions directly to the HWC buildpack. If the HWC buildpack lacks dependencies your app requires, VMware recommends that you create an extension buildpack containing the missing dependencies required by your app.

The HWC buildpack contains a number of built in HWC features. For example, the URL Rewrite and HTTP compression modules. Extensions should only provide additional functionality to your app. For more information about existing HWC features, see Features in *HWC Buildpack*.

For additional extension buildpack configuration and creation commands, see the Creating custom buildpacks.

Prerequisites

- Linux or MacOS development machine or VM
- golang v1.10 or later programming language
- direnv environment variable manager for your shell

Step 1: Initializing an extension buildpack

The buildpack-packager provides boilerplate code that you can use to start writing your buildpack.

1. To install the `buildpack-packager` CLI tool, run:

   ```
   go get github.com/cloudfoundry/libbuildpack/
   go install github.com/cloudfoundry/libbuildpack/tree/master/packager
   ```

2. To create your buildpack boilerplate, run:

   ```
   buildpack-packager init --name BUILDPACK-NAME --path=BUILDPACK-DIRECTORY
   ```

   Where:

   - `BUILDPACK-NAME` is the name of the buildpack you are creating.
   - `BUILDPACK-DIRECTORY` is the directory where the boilerplate code is written. If the directory does not exist, it is created by the `buildpack-packager init` process.

3. To activate direnv in your buildpack directory, run:

   ```
   cd BUILDPACK-DIRECTORY
   direnv allow
   ```

   Where `BUILDPACK-DIRECTORY` is the directory created by the buildpack-packager command, containing the boilerplate code.

Step 2: Creating an extension buildpack to supply additional DLLs to a .NET Framework app

An extension buildpack provides the dependencies required by your app that are missing in HWC buildpack. The extension buildpack references and supplies needed dependencies to the app's container when you push the app.

To create an extension buildpack containing additional DLLs:

1. Create a list of the dependencies that your published app requires.
2. Copy the DLLs for the dependencies into a ZIP file.
3. On the command line, browse to the directory containing the new ZIP file.
4. To generate a SHA for your ZIP file, run the following command:

   ```
   shasum -a256 DLL-ZIP-FILE
   ```

   Where `DLL-ZIP-FILE` is the name of your DLL ZIP file.
5. Upload the ZIP file containing your dependencies to a private web server that is accessible by Cloud Foundry. You can also use an S3 bucket or Azure Storage.
6. On the command line, browse to the directory created by the `buildpack-packager init` command above.
7. Edit the `manifest.yml` file by adding a `dependency` section that references the DLL ZIP file as follows:

   ```
   dependencies:
   - name: BINARY-NAME
     uri: http://s3.amazon.com/BUCKET-NAME/DLL-ZIP-FILE
     version: 0.0.1
     sha256: GENERATED-SHA-256
     cf_stacks:
     - windows
   ```

   Where:

   - `BINARY-NAME` is the name of your binary application.
   - `BUCKET-NAME` is your S3 bucket name.
   - `DLL-ZIP-FILE` is the name of your DLL ZIP file.
   - `GENERATED-SHA-256` is the generated SHA.
8. Ensure the `include_files` section of the manifest contains `bin/supply.exe`.
9. To navigate to the `BUILDPACK-DIRECTORY/src/BUILDPACK-NAME/` directory, use either Windows Explorer, or your text editor's 'File - Open' option.
10. Edit the `supply.go` script.
11. Add dependencies to the container's `PATH` as follows:
    1. Add a `stager.InstallDependency` for the `dependency` and `version` you have in your manifest.yml file.
    2. Add a `stager.AddBinDependency` for every DLL file you want your app to be able to access. For example:

```
dep := libbuildpack.Dependency{Name: "my-binary", Version: "0.0.1"}
if err := s.Installer.InstallDependency(dep, s.Stager.DepDir()); err != nil {
  return err
}
if err := s.Stager.AddBinDependencyLink(filepath.Join(s.Stager.DepDir(),
"MyBinary.dll"), "MyBinary.dll"); err != nil {
  return err
}
```

12. To configure the build.sh script to cross-compile, edit `scripts/build.sh` to include the following:

    ```
    GOOS=windows go build -ldflags="-s -w" -o bin/supply.exe mysql/supply/cli
    ```

Step 3: Building the extension buildpack

To build a cached extension buildpack artifact:

1. Run:

   ```
   buildpack-packager build -cached -stack STACK-NAME
   ```

   Where `STACK-NAME` is the pre-built operating system that can run apps.

   If the buildpack's `supply.go` script contains a mistake, `buildpack-packager` throws an error. Don't push your app with the new extension `buildpack` until after all local errors have been corrected. You must push your app to determine whether the new extension buildpack functions as intended.

Step 4: Validating the extension buildpack

To push an app with the cached extension buildpack artifact:

1. Upload the buildpack to a private web server accessible to Cloud Foundry. You can upload the finished buildpack to the same web server as your dependencies. You can also use an S3 bucket or Azure Storage.
2. To deploy your app with the extension buildpack and determine whether the new buildpack functions as intended, run:

   ```
   cf push APP-NAME -s windows -b EXT-BUILDPACK-URL -b hwc_buildpack
   ```

   Where:

   - `APP-NAME` is your app's name.
   - `EXT-BUILDPACK-URL` is the URL of the uploaded extension buildpack ZIP file.

If your extension buildpack does not include the correct dependencies, you receive an error message.

Step 5: Combine an extension buildpack with other features

If your extension buildpack has executables or scripts that need to be run, you can reference them either in the start command or in profile scripts.

- For more information about the start command, see command in *Deploying with Application Manifests*.
- For more information about profile scripts, see Configure Pre-Runtime Hooks in *Deploying an Application*.

Tips for .NET Framework Developers

This topic describes how to push .NET Framework apps to VMware Tanzu Application Service for VMs [Windows] Diego Cells.

For information about how to develop microservices for .NET Framework and .NET Core using Steeltoe, see the Steeltoe documentation.

Prerequisites

The TAS for VMs [Windows] tile must be installed and configured. To install TAS for VMs [Windows], see Installing and Configuring TAS for VMs [Windows].

Operators must also install the Cloud Foundry Command Line Interface (cf CLI) to run the commands on this topic. For information about installing the cf CLI, see Installing the cf CLI.

Overview

After operators install and configure the TAS for VMs [Windows] tile, developers can push .NET Framework apps to the Windows Diego Cell. Developers can push both OWIN and non-OWIN apps, and can push apps that are served by Hostable Web Core or self-hosted.

If you have upgraded to TAS for VMs [Windows] and have apps that you want to migrate, see Upgrading Windows Diego Cells.

Develop .NET Framework Apps

.NET on Ops Manager Cookbook

For useful tips and recipes for migrating and developing .NET Framework apps to run on TAS for VMs [Windows], see the .NET Cookbook.

Push a .NET Framework App

By default, Ops Manager serves .NET Framework apps with Hostable Web Core (HWC). HWC is a lighter version of the Internet Information Services (IIS) server that contains the core IIS functionality.

To push a .NET Framework app to a Windows Diego Cell:

1. Target the Cloud Controller of your Ops Manager deployment by running:

   ```
   cf api api.APP-DOMAIN
   ```

   Where `APP-DOMAIN` is your app's public domain name. For example, `example.com`.

2. Run one of the following commands to deploy your .NET app:

   - To deploy an app with `.bat` or `.exe` files, run:

     ```
     cf push -s windows -b binary_buildpack
     ```

   - To deploy a .NET Framework app, run:

     ```
     cf push APP-NAME -s windows -b hwc_buildpack
     ```

     Where `APP-NAME` is the name of your app.

     Note: The `-s windows` option instructs Ops Manager to run the app in the Windows Diego Cell.

     Note: If you are not pushing your app from its directory, use the `-p` option and specify the path to the directory that contains the app.

3. Wait for your app to stage and start. If you see an error message, see Troubleshoot App Errors below.

Context Path Routing Support for ASP.NET Apps

Context path routing enables multiple apps to share the same route hostname, such as `app1.example.com/app2`. ASP.NET developers can host apps under a route path. Within Windows Diego Cells, you can have multiple routes to an app, but those routes cannot have different context paths.

Making an app accessible under another app's URL requires a pair of commands. To define a context path route, such as `app1.example.com/app2`:

1. Push the top-level app by running:

   ```
   cf push TOP-LEVEL-APP-NAME
   ```

   Where `TOP-LEVEL-APP-NAME` is the name of your top-level app.

2. Push the lower-level app by running:

   ```
   cf push LOWER-LEVEL-APP-NAME -d APP-DOMAIN --hostname TOP-LEVEL-APP-NAME --route-path LOWER-LEVEL-APP-NAME
   ```

   Where:

   - `TOP-LEVEL-APP-NAME` is the name of your top-level app.
   - `LOWER-LEVEL-APP-NAME` is the name of your lower-level app.

- `APP-DOMAIN` is your app's public domain name. For example, `example.com`.

> **Note:** The `-d` parameter is only needed when pushing an app to a non-default domain.

Enable Graceful Shutdown for a .NET App

Developers can configure .NET apps to shut down gracefully after running `cf stop`. When you run `cf stop`, the .NET app receives a `CTRL_SHUTDOWN_EVENT` and is allowed ten seconds to shut down. To enable graceful shutdown, you must include a control handler in the app.

For more information, see Graceful Shutdown in the .NET Cookbook.

Push a Self-Hosted App

Developers can choose to push a self-hosted app instead of using Hostable Web Core. Self-hosted apps combine server code with the app code.

To push a self-hosted app:

1. Target the Cloud Controller of your Ops Manager deployment by running:

   ```
   cf api api.APP-DOMAIN
   ```

   Where `APP-DOMAIN` is your app's public domain name. For example, `example.com`.

2. Push your .NET app from the app root by running:

   ```
   cf push APP-NAME -s windows -b binary_buildpack -c PATH-TO-BINARY
   ```

   Where:

   - `APP-NAME` is the name of your app.
   - `PATH-TO-BINARY` is the path to your executable.

3. Wait for your app to stage and start. If you see an error message, see Troubleshoot App Errors below.

Push a SOAP Service

Developers can push Simple Object Access Protocol (SOAP) web services to their Ops Manager deployment by following the procedures in the sections below.

Step 1: Deploy Your Web Service

To deploy a SOAP web service:

1. Develop the service as an ASMX web service in Microsoft Visual Studio.
2. Publish the service to your local file system.
3. Open a command line to the directory containing the published web service.
4. Push your service by running:

```
cf push SOAP-SERVICE-NAME -s windows -b hwc_buildpack -p WEB-SERVICE-DIRECTORY
-u none
```

Where:

- `SOAP-SERVICE-NAME` is the name of your service.
- `WEB-SERVICE-DIRECTORY` is the path to the directory containing the published web service.

For example:

```
$ cf push webservice -s windows -b hwc_buildpack -u none

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: webservice.example.com
```

Note: The push command must include the `-s` flag to specify the stack, which instructs Ops Manager to run the app in the Windows Diego Cell.

Note: The `-p` and `-u` parameters are optional parameters. The `-p` parameter is needed only when pushing your service from a directory that does not contain the published web service. The `-u` parameter is needed only when disabling the health check when you do not have a route serving `/`.

5. Confirm your service is running by finding your service's URL in the push command's output and browsing to it. In the example above, the URL for the service would be `http://webservice.example.com`.

Step 2: Modify the WSDL File

Your SOAP web service is now deployed on Ops Manager, but the service's WSDL file contains incorrect port information. The WSDL file must be modified to enable an app to consume your web service. Either you or the service developer can perform the needed modification.

See the following portion of an example WSDL file:

```
- <wsdl:service name="WebService1">
  - <wsdl:port name="WebService1Soap" binding="tns:WebService1Soap">
      <soap:address location="http://webservice.example.com:62492/WebService1.asmx"/>
    </wsdl:port>
  - <wsdl:port name="WebService1Soap12" binding="tns:WebService1Soap12">
      <soap12:address location="http://webservice.example.com:62492/WebService1.asmx"/
>
    </wsdl:port>
- </wsdl:service>
```

The WSDL file provides the port number for the SOAP web service as `62492`. This is the port that the web service listens on in the Garden container, but external apps cannot access the service on this port. Instead, external apps must use port 80, and the Gorouter routes requests to the web service in the container. For more information about the Garden container, see Container Mechanics in *Container Security*. For more information about the Gorouter, see TAS for VMs Routing Architecture.

The URL of the web service in the WSDL file must be modified to remove `62492`. With no port number, the URL defaults to port 80. In the example above, the modified URL would be `http://webservice.example.com/WebService1.asmx`.

SOAP web service developers can resolve this problem in one of two ways:

- Modify the WSDL file by following the instructions in Modify a Web Service's WSDL Using a SoapExtensionReflector from the Microsoft Developers Network.
- Instruct the developers of external apps that consume the web service to follow the procedure in Consume the SOAP Web Service.

Consume the SOAP Web Service

Developers of external apps that consume the SOAP web service can use a modified version of the WSDL file.

To use a modified version of the WSDL file:

1. In a browser, navigate to the WSDL file of the web service, using the following URL:

   ```
   https://SOAP-SERVICE-NAME.APP-DOMAIN/ASMX-FILE?wsdl
   ```

   Where:

   - `SOAP-SERVICE-NAME` is the name of your service.
   - `APP-DOMAIN` is your site's public domain name.
   - `ASMX-FILE` is the filename of your asmx file.

   For example:

   ```
   https://webservice.example.com/WebService1.asmx?wsdl
   ```

2. Download the WSDL file to your local machine.
3. Edit the WSDL file to eliminate the container port, as described in Modify the WSDL File.
4. In Microsoft Visual Studio, right-click on your app in the **Solution Explorer** and select **Add** > **Service Reference**.
5. Under **Address**, enter the local path to the modified WSDL file. For example:

   ```
   C:\Users\example\wsdl.xml
   ```

6. Click **OK**. Microsoft Visual Studio generates a client from the WSDL file that you can use in your codebase.

Context Path Routing Support for SOAP Web Services

Developers can push SOAP web services to their Ops Manager deployment with context path routing. For more information, see Context Path Routing Support for ASP.NET Apps.

Troubleshoot App Errors

If a .NET app fails to start, see the following list of errors and their possible solutions:

- `NoCompatibleCell`: Your Ops Manager deployment cannot connect to your Windows Diego Cell. For more information about troubleshooting your Windows Diego Cell configuration, see Troubleshooting Windows Diego Cells.
- `Start unsuccessful`: Your app may may be misconfigured or lacks the required DLL files and dependencies.
  - Ensure that your app directory contains either a valid `.exe` binary or a valid `Web.config` file.
  - Ensure that you are pushing from a directory containing your app dependencies. If it does not, specify the app dependency directory with the `-p` flag.

Java

In this section:

- Java Buildpack
- Tips for Java Developers
- Cloud Foundry Java Client Library
- Using Java Native Image
- Configuring Service Connections

Java buildpack

You can use the Java buildpack with apps written in Grails, Play, Spring, or any other JVM-based language or framework.

See the following topics for more information:

- Tips for Java Developers
- Getting Started Deploying Apps
- Configuring Service Connections
- Cloud Foundry Java Client Library
- Using Java Native Image

For information about specific versions, see Java Buildpack Release Notes. You can find the source for the Java buildpack in the Java buildpack repository on GitHub.

Buildpack and application logging

The Java buildpack only runs during the staging process, and only logs staging information such as the downloaded components, configuration data, and work performed on your application by the buildpack.

The Java buildpack source documentation states the following:

- The Java buildpack logs all messages, regardless of severity, to `APP-DIRECTORY/.java-buildpack.log`. The buildpack also logs messages to `$stderr`, filtered by a configured severity level.
- If the buildpack fails with an exception, the exception message is logged with a log level of `ERROR`. The exception stack trace is logged with a log level of `DEBUG`. This prevents users from seeing stack traces by default.

Once staging completes, the buildpack stops logging then the Loggregator handles application logging.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. For more information about logging, see App Logging in TAS for VMs.

BOSH Custom Trusted Certificate support

Versions 3.7 and later of the Java buildpack support BOSH configured custom trusted certificates. For more information, see Configuring Trusted Certificates in the BOSH documentation.

The Java buildpack pulls the contents of `/etc/ssl/certs/ca-certificates.crt` and `$CF_INSTANCE_CERT/$CF_INSTANCE_KEY` by default.

The log output for Diego Instance Identity-based `KeyStore` appears as follows:

```
Adding System Key Manager
Adding Key Manager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.crt
Start watching /etc/cf-instance-credentials/instance.key
Initialized KeyManager for /etc/cf-instance-credentials/instance.key and /etc/cf-instance-credentials/instance.crt
```

The log output for Diego Trusted Certificate-based `TrustStore` appears as follows:

```
Adding System Trust Manager
Adding TrustManager for /etc/ssl/certs/ca-certificates.crt
Start watching /etc/ssl/certs/ca-certificates.crt
Initialized TrustManager for /etc/ssl/certs/ca-certificates.crt
```

Memory constraints in Java Buildpack v4.0

The memory calculator in Java buildpack v4.0 accounts for the following memory regions:

- `-Xmx`: Heap
- `-XX:MaxMetaspaceSize`: Metaspace
- `-Xss`: Thread Stacks

- `-XX:MaxDirectMemorySize`: Direct Memory
- `-XX:ReservedCodeCacheSize`: Code Cache
- `-XX:CompressedClassSpaceSize`: Compressed Class Space

Applications, which previously ran in 512 MB or smaller containers, might no longer be able to. Most applications run if they use the Cloud Foundry default container size of 1 G without any modifications. However, you can configure those memory regions directly as needed.

The Java buildpack optimizes for all non-heap memory regions first and leaves the remainder for the heap.

The Java buildpack prints a histogram of the heap to the logs when the JVM encounters a terminal failure.

The Cloud Foundry default Java buildpack is currently v3.x to allows time for apps to be upgrade to v4.x.

For more information, see Java buildpack 4.0.

Using Cloud Foundry Java buildpack

You can use the Java buildpack with Cloud Foundry.

You can deploy a number of different JVM-based artifact types. For a more detailed explanation of what the Cloud Foundry Java Builpack supports, see Additional Documentation in the repository on GitHub.

Java buildpack

For information about using, configuring, and extending the Cloud Foundry Java buildpack, see the Cloud Foundry Java Buildpack repository on GitHub.

Java buildpack design

The Java buildpack can convert artifacts that run on the JVM into executable apps. It does this by identifying one of the supported artifact types (Grails, Groovy, Java, Play Framework, Spring Boot, and Servlet) and downloading all additional dependencies needed to run. It also analyzes the collection of services bound to the app and downloads any dependencies related to those services.

For example, pushing a WAR file that is bound to a PostgreSQL database and New Relic for performance monitoring shows output like this:

```
-----> Java Buildpack v4.50 | https://github.com/cloudfoundry/java-buildpack#
e1d6dbc
-----> Downloading Jvmkill Agent 1.16.0_RELEASE from https://java-buildpack.c
loudfoundry.org/jvmkill/bionic/x86_64/jvmkill-1.16.0-RELEASE.so (0.1s)
-----> Downloading Open Jdk JRE 1.8.0_342 from https://java-buildpack.cloudfo
undry.org/openjdk/bionic/x86_64/bellsoft-jre8u342%2B7-linux-amd64.tar.gz (0.6
s)
Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.3s)
JVM DNS caching deactivated in lieu of BOSH DNS caching
-----> Downloading Open JDK Like Memory Calculator 3.13.0_RELEASE from http
```

```
s://java-buildpack.cloudfoundry.org/memory-calculator/bionic/x86_64/memory-ca
lculator-3.13.0-RELEASE.tar.gz (0.1s)
Loaded Classes: 21014, Threads: 250
-----> Downloading Client Certificate Mapper 1.11.0_RELEASE from https://java
-buildpack.cloudfoundry.org/client-certificate-mapper/client-certificate-mapp
er-1.11.0-RELEASE.jar (0.0s)
-----> Downloading Container Security Provider 1.19.0_RELEASE from https://ja
va-buildpack.cloudfoundry.org/container-security-provider/container-security-
provider-1.19.0-RELEASE.jar (0.1s)
Exit status 0
```

Configuration

In most cases, the buildpack can work without any configuration. If you are new to Cloud Foundry, VMware recommends that you make your first attempts without modifying the buildpack configuration. The buildpack is flexible, though, and you can configure it through environment variables. For more information, see Configuration and Extension in the Cloud Foundry Java Buildpack repository on GitHub.

Java Client Library

The Cloud Foundry Client Library provides a Java API for interacting with an instance. This library, `cloudfoundry-client`, can be used by Java based tools to interact with the platform.

For information about using this library, see Java Cloud Foundry Library.

Grails

Grails packages apps into WAR files for deployment into a Servlet container. To build the WAR file and deploy it, run:

```
grails prod war
cf push YOUR-APP -p target/YOUR-APP-VERSION.war
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the WAR file you want to build and deploy.

Groovy

TAS for VMs supports Groovy apps based on both Ratpack and a simple collection of files.

Ratpack

Ratpack packages apps into two different styles. TAS for VMs supports the `distZip` style. To build the ZIP file and deploy it, run:

```
gradle distZip
```

```
cf push YOUR-APP -p build/distributions/YOUR-ZIP-FILE.zip
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-ZIP-FILE` is the name of the ZIP file you want to build and deploy.

For more information, see the Ratpack website.

Raw Groovy

You can run Groovy apps that are made up of a single entry point and any supporting files without any other work. To deploy them, run:

```
cf push YOUR-APP
```

Where `YOUR-APP` is the name of your app.

For more information, see Groovy Container in the Cloud Foundry Java Buildpack repository on GitHub.

HTTP/2

To deploy Java apps that use HTTP/2, you must have:

1. A Cloud Floundry foundation that has HTTP/2 support enabled. For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.
2. Cloud Foundry Command-Line Interface (cf CLI) v8 or later. See Upgrading to cf CLI v8.

You can deploy any Java app and get automatic support for the HTTP/2 protocol without making any changes to your app. When a client connects through a route mapped to your Java apps over HTTP/2, the foundation transparently downgrades the protocol and communicates with your app over HTTP/1.1.

If you require end-to-end HTTP/2, for example, because of gRPC, do the following:

1. Enable HTTP/2 support in the app:
    - See Configure HTTP/2 in the Spring Boot documentation.
    - If you are deploying a standard non-executable WAR file, you only need to make sure that you are using Java buildpack v4.43 or later. Starting with v4.43, the Java buildpack configures Apache Tomcat to accept HTTP/2 connections.
    - Other frameworks, including Play, Ratpack, and apps that use the distZip format, embed an HTTP server. You must configure these apps to enable HTTP/2, specifically H2C, clear-text. See your framework's documentation for enabling HTTP/2 and H2C.

    H2C is required because TAS for VMs uses Envoy to secure communications into the app container.
2. Configure the route to use the HTTP/2 protocol using either the cf CLI or the app manifest:
    - Using the cf CLI: `cf map-route MY-APP EXAMPLE.COM --hostname host --destination-protocol http2`.

  - Using a `manifest.yml` file:

    ```
    ---
    applications:
    - name: MY-APP
      routes:
        - route: MY-APP.EXAMPLE.COM
          protocol: http2
    ```

HTTP/2 example

For example:

1. Fetch the Spring Music app:

   ```
   git clone https://github.com/cloudfoundry-samples/spring-music.git
   ```

2. Build the app:

   ```
   ./gradle assemble
   ```

3. Push the app without assigning a route:

   ```
   cf push --no-route
   ```

4. Map a route:

   ```
   cf map-route spring-music EXAMPLE.COM --hostname spring-music --destination-protocol http2
   ```

5. Validate:

   ```
   curl https://spring-music.EXAMPLE.COM/request --http2 -s | jq .
   ```

   A successful response looks like the following:

   ```
   {
     "protocol": "HTTP/2.0",
     "remote-addr": "198.51.100.100",
     "method": "GET",
     "scheme": "https",
     "session-id": "1A891C614AB6B6CB59A56D7F520BBCBD"
   }
   ```

Java main

Java apps with a `main()` method can be run provided that they are packaged as self-executable JARs. For more information, see Java Main Container in the Cloud Foundry Java Buildpack repository on GitHub.

If your app is not web enabled, you must suppress route creation to avoid a `failed to start accepting connections` error. To suppress route creation, add `no-route: true` to the app manifest or use the `--no-route` flag with the `cf push` command.

For more information about the `no-route` attribute, see Deploying with App Manifests.

Maven

A Maven build can create a self-executable JAR. To build and deploy the JAR, run:

```
mvn package
cf push YOUR-APP -p target/YOUR-APP-VERSION.jar
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the JAR you want to build and deploy.

Gradle

A Gradle build can create a self-executable JAR. To build and deploy the JAR, run:

```
gradle build
cf push YOUR-APP -p build/libs/YOUR-APP-VERSION.jar
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the JAR you want to build and deploy.

Play Framework

The Play Framework packages apps into two different styles. TAS for VMs supports both the `staged` and `dist` styles. To build the `dist` style and deploy it, run:

```
play dist
cf push YOUR-APP -p target/universal/YOUR-APP-VERSION.zip
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the `dist` style ZIP you want to build and deploy.

For more information, see the Play Framework website.

Spring Boot CLI

Spring Boot can run apps comprised entirely of POGOs. To deploy them, run:

```
spring grab *.groovy
cf push YOUR-APP
```

Where `YOUR-APP` is the name of your app.

For more information, see Spring Boot on the Spring website and Spring Boot CLI Container in the Cloud Foundry Java Buildpack repository on GitHub.

Servlet

Java apps can be packaged as Servlet apps.

Maven

A Maven build can create a Servlet WAR. To build and deploy the WAR, run:

```
mvn package
cf push YOUR-APP -p target/YOUR-APP-VERSION.war
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the WAR you want to build and deploy.

Gradle

A Gradle build can create a Servlet WAR. To build and deploy the WAR, run:

```
gradle build
cf push YOUR-APP -p build/libs/YOUR-APP-VERSION.war
```

Where:

- `YOUR-APP` is the name of your app.
- `YOUR-APP-VERSION` is the name of the WAR you want to build and deploy.

Binding to services

For more information about binding apps to services, see Configuring Service Connections.

Java and Grails best practices

Provide a JDBC driver

The Java buildpack does not bundle a JDBC driver with your app. If you want your app to access a SQL RDBMS, include the appropriate driver in your app.

Allocate sufficient memory

If you do not allocate sufficient memory to a Java app when you deploy it, it may fail to start, or TAS for VMs may terminate it. You must allocate enough memory to allow for:

- Java heap
- Metaspace
- Stack size per Thread

- JVM overhead

The `config/open_jdk_jre.yml` file of the Java buildpack contains default memory size and weighting settings for the JRE. For an explanation of JRE memory sizes and weightings and how the Java buildpack calculates and allocates memory to the JRE for your app, see Open JDK JRE in the Cloud Foundry Java Buildpack on GitHub.

To configure memory-related JRE options for your app, you can override the default memory settings of your buildpack as described in Configuration and Extension with the properties listed in the Open JDK JRE README in the Cloud Foundry Java Buildpack on GitHub. For more information about configuring manifests, see Deploying with App Manifests.

To see memory utilization when your app is running, run:

```
cf app YOUR-APP
```

Where `YOUR-APP` is the name of your app.

Troubleshoot out of memory

A Java app may crash because of insufficient memory on the Garden container or the JVM on which it runs. The sections below provide guidance for help diagnosing and resolving such issues.

JVM

- **Error**: `java.lang.OutOfMemoryError`. For example:

  ```
  $ cf logs YOUR-APP --recent
  2016-06-20T09:18:51.00+0100 [APP/0] OUT java.lang.OutOfMemoryError: Met
  aspace
  ```

  Where `YOUR-APP` is the name of your app.

- **Cause**: If the JVM cannot garbage-collect enough space to ensure the allocation of a data-structure, it fails with java.lang.OutOfMemoryError. In the example above, JVM has an under-sized `metaspace`. You may see failures in other memory pools, such as `heap`.
- **Solution**: Configure the JVM correctly for your app. For more information, see Allocate Sufficient Memory.

Garden container

The solutions in this section require configuring the memory calculator, which is a sub project of the Java buildpack that calculates suitable memory settings for Java apps when you push them. For more information, see the java-buildpack-memory-calculator repository on GitHub. If you have questions about the memory calculator, you can ask them in the **#java-buildpack** channel of the Cloud Foundry Slack organization.

- **Error**: The Garden container terminates the Java process with the out of memory event. For example:

  ```
  $ cf events YOUR-APP
  time                          event         actor         description
  ```

```
2016-06-20T09:18:51.00+0100   app.crash     app-name      index: 0, rea
son: CRASHED, exit_description: out of memory, exit_status: 255
```

Where `YOUR-APP` is the name of your app.

This error appears when the JVM allocates more OS-level memory than the quota requested by the app, such as through the manifest.

- **Cause 1 - Insufficient native memory**: This error commonly means that the JVM requires more native memory. In the scope of the Java buildpack and the memory calculator, the term “native” means the memory required for the JVM to work, along with forms of memory not covered in the other classifications of the memory calculator. This includes the memory footprint of OS-level threads, program counters, when an app forks and runs subprocesses, or when an app uses JNI to allocate memory.
- **Solution 1**: Determine how much native memory a Java app needs by measuring it with realistic workloads and fine-tuning it accordingly. You can then configure the Java buildpack using the native setting of the memory calculator, as in the example below:

  ```
  ---
  applications:
  - name: YOUR-APP
    memory: 1G
    env:
      JBP_CONFIG_OPEN_JDK_JRE: '[memory_calculator: {headroom: 10}}]'
  ```

  Where `YOUR-APP` is the name of your app.

  This example shows that 10% of the overall available `1G` is reserved for anything that is not `heap`, `metaspace`, `direct`, `code cache` or `threads`. In less common cases, this may come from companion processes started by the JVM, such as the Process API.

  For more information about measuring how much native memory a Java app needs, see Native Memory Tracking in the Java documentation. For more information about configuring the Java buildpack using the native setting, see OpenJDK JRE in the Cloud Foundry Java Buildpack on GitHub. For more information about the Process API, see Class Process in the Java documentation.
- **Cause 2 - High thread count**: Java threads in the JVM can cause memory errors at the Garden level. When an app is under heavy load, it uses a high number of threads. Each thread consumes some memory, and, if there are enough threads, they consume a significant amount of memory.
- **Solution 2**: Set the reserved memory for stack traces to the correct value for your app.

  You can use the `-Xss` setting of the JVM to configure the amount of space the JVM reserves for each Java thread. You must multiply this value by the number of threads your app requires. Specify the number of threads in the `stack_threads` setting of the memory calculator. For example, if you estimate the max thread count for an app at `800` and the amount of memory needed to represent the deepest stacktrace of a Java thread is `512KB`, configure the memory calculator as follows:

```
---
applications:
- name: YOUR-APP
  memory: 1G
  env:
    JBP_CONFIG_OPEN_JDK_JRE: '[memory_calculator: {stack_threads: 800}]'
    JAVA_OPTS: '-Xss512k'
```

Where `YOUR-APP` is the name of your app.

In this example, the overall memory amount reserved by the JVM for representing the stacks of Java threads is `800 * 512k = 400m`.

The correct settings for `-Xss` and `stack_threads` depend on your app code, including the libraries it uses. Your app may technically have no upper limit, such as in the case of cavalier usage of `CachedThreadPool` executors. However, you still must calculate the depth of the thread stacks, and the amount of space the JVM reserves for each of them. For more information, see Executors.newCachedThreadPool() considered harmful on the Bizo website and the newCachedThreadPool section of the *Class Executors* topic in the Java documentation.

Troubleshoot failed upload

If your app fails to upload when you push it to TAS for VMs, it may be for one of the following reasons:

- **WAR is too large**: An upload may fail due to the size of the WAR file. TAS for VMs testing indicates WAR files as large as 250 MB upload successfully. If a WAR file larger than that fails to upload, it may be a result of the file size.
- **Connection issues**: App uploads can fail if you have a slow Internet connection, or if you upload from a location that is very remote from the target TAS for VMs instance. If an app upload takes a long time, your authorization token can expire before the upload completes. A workaround is to copy the WAR to a server that is closer to the TAS for VMs instance, and then push it from there.
- **Out-of-date cf CLI client**: Upload of a large WAR is faster and therefore less likely to fail if you are using a recent version of the cf CLI. If you are using an older version of the cf CLI client to upload a large WAR, and having problems, try updating to the latest version of the cf CLI.
- **Incorrect WAR targeting**: By default, `cf push` uploads everything in the current directory. For a Java app, `cf push` with no option flags uploads source code and other unnecessary files, in addition to the WAR. When you push a Java app, specify the path to the WAR by running:

  ```
  cf push YOUR-APP -p PATH/TO/WAR-FILE
  ```

  Where:

  - `YOUR-APP` is the name of your app.

- `PATH/TO/WAR-FILE` is the path to the WAR. You can determine whether or not the path was specified for a previously pushed app by examining the app deployment manifest, `manifest.yml`. If the `path` attribute specifies the current directory, the manifest includes a line like:

  ```
  path: .
  ```

  To re-push just the WAR, do one of the following:

  - Delete `manifest.yml` and run `cf push` again, specifying the location of the WAR using the `-p` flag.
  - Edit the `path` argument in `manifest.yml` to point to the WAR, and re-push the app.

Debuging Java apps

Because of the way TAS for VMs deploys your apps and isolates them, it is not possible to connect to your app with the remote Java debugger. Instead, instruct the app to connect to the Java debugger on your local machine.

To set up remote debugging when using BOSH Lite or a TAS for VMs installation:

1. Open your project in Eclipse.
2. Right-click your project, go to **Debug as** and pick **Debug Configurations**.
3. Create a new **Remote Java Application**.
4. Make sure your project is selected, pick **Standard (Socket Listen)** from the **Connection Type** drop down and set a port. Make sure this port is open if you are running a firewall.
5. Click **Debug**.

The debugger is running. If you switch to the Debug perspective, you see your app listed in the Debug panel, and it says `Waiting for vm to connect at port`.

Next, to push your app to TAS for VMs and instruct TAS for VMs to connect to the debugger running on your local machine:

1. Edit your `manifest.yml` file. Set the instances count to 1. If you set this greater than one, multiple apps try to connect to your debugger.
2. Also in `manifest.yml`, add an `env` block and create a variable named `JAVA_OPTS`. For more information about the `env` block, see Deploying with App Manifests.
3. Add the remote debugger configuration to the `JAVA_OPTS` variable: `-agentlib:jdwp=transport=dt_socket,address=YOUR-IP-ADDRESS:YOUR-PORT`.
4. Save the `manifest.yml` file.
5. Run:

   ```
   cf push
   ```

Upon completion, you can see that your app has started and is now connected to the debugger running in your IDE. You can now add breakpoints and interrogate the app just as you would if it were running locally.

Slow starting Java or Grails apps

Some Java and Grails apps do not start quickly, and the health check for an app can fail if an app starts too slowly.

The current Java buildpack implementation sets the Tomcat `bindOnInit` property to `false`. This prevents Tomcat from listening for HTTP requests until an app has fully deployed.

If your app does not start quickly, the health check might fail because it checks the health of the app before the app can accept requests. By default, the health check fails after a timeout threshold of 60 seconds.

To resolve this issue, run `cf push` with the `-t TIMEOUT-THRESHOLD` option to increase the timeout threshold. Run:

```
$ cf push YOUR-APP -t TIMEOUT-THRESHOLD
```

or

```
$ cf push YOUR-APP --app-start-timeout TIMEOUT-THRESHOLD
```

Where:

- `YOUR-APP` is the name of your app.
- `TIMEOUT-THRESHOLD` is the number of seconds to which you want to increase the timeout threshold.

The timeout threshold cannot exceed 180 seconds. Specifying a timeout threshold greater than 180 seconds causes the following error:

```
Server error, status code: 400, error code: 100001, message: The app is invalid: healt
h_check_timeout maximum_exceeded</code>
```

Extension

The Java buildpack can also be easily extended. It creates abstractions for three types of components (containers, frameworks, and JREs) to allow users to easily add functionality. In addition to these abstractions, there are a number of utility classes for simplifying typical buildpack behaviors.

As an example, the New Relic framework looks like this:

```
class NewRelicAgent < JavaBuildpack::Component::VersionedDependencyComponent

  # @macro base_component_compile
  def compile
    FileUtils.mkdir_p logs_dir

    download_jar
    @droplet.copy_resources
  end

  # @macro base_component_release
  def release
```

```
    @droplet.java_opts
    .add_javaagent(@droplet.sandbox + jar_name)
    .add_system_property('newrelic.home', @droplet.sandbox)
    .add_system_property('newrelic.config.license_key', license_key)
    .add_system_property('newrelic.config.app_name', "'#{application_name}'")
    .add_system_property('newrelic.config.log_file_path', logs_dir)
  end

  protected

  # @macro versioned_dependency_component_supports
  def supports?
    @application.services.one_service? FILTER, 'licenseKey'
  end

  private

  FILTER = /newrelic/.freeze

  def application_name
    @application.details['application_name']
  end

  def license_key
    @application.services.find_service(FILTER)['credentials']['licenseKey']
  end

  def logs_dir
    @droplet.sandbox + 'logs'
  end

end
```

For more information, see Design, Extending, and Configuration and Extension in the Cloud Foundry Java Buildpack repository on GitHub.

Environment variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` by running:

```
System.getenv("VCAP_SERVICES");
```

For more information, see TAS for VMs Environment Variables.

Cloud Foundry Java Client Library

You can use the Cloud Foundry Java Client Library to manage an account on a Cloud Foundry instance.

Cloud Foundry Java Client Library v1.1.x works with apps using Spring v4.x, and Cloud Foundry Java Client Library v1.0.x work with apps using Spring v3.x. Both versions are available in the Cloud Foundry Java Client Library repository on GitHub.

Adding the Java Client Library

To obtain the correct components, see the Cloud Foundry Java Client Library repository on GitHub.

Most projects need two dependencies: the Operations API and an implementation of the Client API. For more information about how to add the Cloud Foundry Java Client Library as dependencies to a Maven or Gradle project, see the sections below.

Maven

Add the `cloudfoundry-client-reactor` dependency (formerly known as `cloudfoundry-client-spring`) to your `pom.xml` as follows:

```
<dependencies>
    <dependency>
        <groupId>org.cloudfoundry</groupId>
        <artifactId>cloudfoundry-client-reactor</artifactId>
        <version>2.0.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>org.cloudfoundry</groupId>
        <artifactId>cloudfoundry-operations</artifactId>
        <version>2.0.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>io.projectreactor</groupId>
        <artifactId>reactor-core</artifactId>
        <version>2.5.0.BUILD-SNAPSHOT</version>
    </dependency>
    <dependency>
        <groupId>io.projectreactor</groupId>
        <artifactId>reactor-netty</artifactId>
        <version>2.5.0.BUILD-SNAPSHOT</version>
    </dependency>
    ...
</dependencies>
```

The artifacts can be found in the Spring release and snapshot repositories:

```
<repositories>
    <repository>
        <id>spring-releases</id>
        <name>Spring Releases</name>
        <url>http://repo.spring.io/release</url>
    </repository>
    ...
</repositories>
```

```
<repositories>
    <repository>
        <id>spring-snapshots</id>
        <name>Spring Snapshots</name>
        <url>http://repo.spring.io/snapshot</url>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
    ...
</repositories>
```

Gradle

Add the `cloudfoundry-client-reactor` dependency to your `build.gradle` file as follows:

```
dependencies {
    compile 'org.cloudfoundry:cloudfoundry-client-reactor:2.0.0.BUILD-SNAPSHOT'
    compile 'org.cloudfoundry:cloudfoundry-operations:2.0.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-core:2.5.0.BUILD-SNAPSHOT'
    compile 'io.projectreactor:reactor-netty:2.5.0.BUILD-SNAPSHOT'
    ...
}
```

The artifacts can be found in the Spring release and snapshot repositories:

```
repositories {
    maven { url 'http://repo.spring.io/release' }
    ...
}
```

```
repositories {
    maven { url 'http://repo.spring.io/snapshot' }
    ...
}
```

Sample code

The following is a very simple sample app that connects to a Cloud Foundry instance, logs in, and displays some information about the Cloud Foundry account. When running the program, provide the Cloud Foundry target API endpoint, along with a valid user name and password as command-line parameters.

```
import org.cloudfoundry.client.lib.CloudCredentials;
import org.cloudfoundry.client.lib.CloudFoundryClient;
import org.cloudfoundry.client.lib.domain.CloudApplication;
import org.cloudfoundry.client.lib.domain.CloudService;
import org.cloudfoundry.client.lib.domain.CloudSpace;

import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;

public final class JavaSample {

    public static void main(String[] args) {
        String target = args[0];
        String user = args[1];
        String password = args[2];

        CloudCredentials credentials = new CloudCredentials(user, password);
        CloudFoundryClient client = new CloudFoundryClient(credentials, getTargetURL(t
arget));
        client.login();

        System.out.printf("%nSpaces:%n");
        for (CloudSpace space : client.getSpaces()) {
            System.out.printf("  %s\t(%s)%n", space.getName(), space.getOrganization
().getName());
        }

        System.out.printf("%nApplications:%n");
        for (CloudApplication application : client.getApplications()) {
            System.out.printf("  %s%n", application.getName());
        }

        System.out.printf("%nServices%n");
        for (CloudService service : client.getServices()) {
            System.out.printf("  %s\t(%s)%n", service.getName(), service.getLabel());
        }
    }

    private static URL getTargetURL(String target) {
        try {
            return URI.create(target).toURL();
        } catch (MalformedURLException e) {
            throw new RuntimeException("The target URL is not valid: " + e.getMessage
());
        }
    }

}
```

For more details about the Cloud Foundry Java Client Library, see the Cloud Foundry Java Client Library repository on GitHub.

To view the objects that you can query and inspect, see domain package in the cloudfoundry/cf-java-client repository on GitHub.

Using Java Native Image in Cloud Foundry

You can build your Java apps with native image support and deploy those apps to Cloud Foundry.

A growing number of Java users are building Java apps using support for native images. For more information, see Native Image in the GraalVM documentation.

Java users deploying to Cloud Foundry can deploy apps compiled using native image support. However, the Cloud Foundry Java buildpack does not provide support to build, compile, and turn apps into a native image. You must perform those steps before deploying to Cloud Foundry.

To build your app that is suitable for running on Cloud Foundry you can do one of the following:

- Build Using Cloud Native Buildpacks
- Build Using Native Build Tools

Building using Cloud Native buildpacks

Cloud Native Buildpacks include support for building native image apps. You can pass in your source code or a compiled JAR and Cloud Native Buildpacks installs the required tools and builds a compatible image.

Using Cloud Native buildpacks example

1. Clone the example repository from GitHub:

   ```
   git clone https://github.com/paketo-buildpacks/samples
   cd samples/java/native-image/java-native-image-sample
   ```

2. Build the example image:

   ```
   ./mvnw package
   pack build apps/native-image -p target/demo-0.0.1-SNAPSHOT.jar -e BP_NATIVE_IMAGE=true -B paketobuildpacks/builder:tiny
   ```

For more information about building with Cloud Native Buildpacks, see Getting Started in the Spring Native documentation.

Deploying using Cloud Native buildpacks

This section describes how to deploy a Java app with native image support using Cloud Native Buildpacks.

To deploy an app compiled using the steps from the previous section to TAS for VMs, you can deploy the image directly using TAS for VMs's Docker support. If Docker support is deactivated, you can extract the native image binary from the container image and deploy the app using the binary buildpack.

Deploying from an image

To deploy the app from an image:

1. Publish your image. You can do this in a number of ways. For example, you can use the `--publish` flag to `pack build` with `docker tag` and `docker push`, or through any other means of publishing a container image to a registry.

2. Validate that your foundation supports deploying Docker images by running the following command and confirming that `diego_docker` is set to `enabled`. If it is set to `disabled`, you cannot use this deployment option.

   ```
   cf feature-flags
   ```

   For example:

   ```
   $ cf feature-flags
   Retrieving status of all flagged features as user@example.com...

   features                                      state
   user_org_creation                             disabled
   private_domain_creation                       enabled
   app_bits_upload                               enabled
   ```

```
app_scaling                              enabled
route_creation                           enabled
service_instance_creation                enabled
diego_docker                             enabled
set_roles_by_username                    enabled
unset_roles_by_username                  enabled
task_creation                            enabled
env_var_visibility                       enabled
space_scoped_private_broker_creation     enabled
space_developer_env_var_visibility       enabled
service_instance_sharing                 enabled
hide_marketplace_from_unauthenticated_users   disabled
resource_matching                        enabled
```

3. Push your app by running:

```
cf push -o registry.example.com/apps/native-image native-image-app
```

Extracting the image and deploy the Binary buildpack

To extract and deploy your app using the binary buildpack:

1. Create a script to extract the files:

```
#!/bin/bash

if [ -z "$1" ] || [ -z "$2" ]; then
    echo "USAGE: extract.sh image-name full-main-class"
    echo
    exit 1
fi

CONTAINER_ID=$(docker create "$1")
mkdir -p ./out
docker cp "$CONTAINER_ID:/workspace/$2" "./out/$2"
docker rm "$CONTAINER_ID" > /dev/null
```

This script is optional, but illustrates the process of extracting the binary file. It runs `docker create` and `docker cp` to extract the file from the image, followed by `docker rm` to remove the containerd. You can do this manually, or you can use any other tools for interacting with a container image.

2. Run the extract script:

```
$ ./extract.sh apps/native-image io.paketo.demo.Demoapp
$ file ./out/io.paketo.demo.Demoapp
io.paketo.demo.Demoapp: ELF 64-bit LSB pie executable, x86-64, version 1 (SYS
V), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux
3.2.0, BuildID[sha1]=05cb81992f3859c9653a9ef6a691e798a9c48b9b, with debug_info,
not stripped
```

The extract script places the binary in a directory called `out` under the current working directory. You can run `file` to examine it and confirm that it is a 64-bit Linux executable.

3. Push and run the compiled binary by running the following command. You can adjust other properties or use a `manifest.yml` file to deploy as well.

    ```
    cf push -b binary_buildpack -m 256M -p ./out -c ./io.paketo.demo.Demoapp native
    -image
    ```

Building using Native Build Tools

This section describes how to deploy a Java app with native image support using Native Build Tools. If you do not want to build using Cloud Native Buildpacks, you can build and compile directly using the Native Build Tools.

To build a Java app using Native Build Tools:

1. Obtain an Ubuntu Bionic computer, VM, or container. Ubuntu Bionic is recommended for best compatibility with the Cloud Foundry `cflinuxfs3` root filesystem. Ubuntu 22.04 LTS (Jammy Jellyfish) is recommended for use with `cflinuxfs4`.
2. Install GraalVM. See Get Started with GraalVM.
3. Install the Java Native Image tools. See Install Native Image.
4. To add the Native Build Tools to your Maven or Gradle project, follow the instructions in Add the native build tools plugin in the Spring documentation. This only needs to be done once.
5. Clone the example repository from GitHub:

    ```
    git clone https://github.com/paketo-buildpacks/samples
    cd samples/java/native-image/java-native-image-sample
    ```

6. Build the example image:

    ```
    mvn -Pnative -DskipTests package
    ```

For more information about building with Native Build Tools, see Getting started with Native Build Tools in the Spring documentation.

Deploying with direct build

This section describes how to deploy a Java app with native image support using a direct build.

To deploy a Java app with native image support using the binary buildpack:

1. Zip the executable created by your build tool, either Maven or Gradle. For example, with Maven, `zip demo.zip target/demo`. Alternatively, you can copy the executable into a directory by itself. For example, with Maven, `mkdir -p ./out && cp target/demo ./out/`. This is the root for `cf push`.
2. Push the root ZIP file or directory you created in the previous step. If you use a directory instead of a ZIP archive, adjust the `-p` argument. This is important so that it only uploads the compiled binary, not your entire project. You can adjust other properties or use a `manifest.yml` file to deploy as well.

```
cf push -b binary_buildpack -p demo.zip -c ./target/demo native-image
```

Spring Boot and Spring Native

Regardless of how you build your app, with Spring Boot v2.5 and Spring Native v0.10.1, there is a bug that causes `cf push` to fail. The problem is caused by conditional behavior in Spring Boot Actuator when run on TAS for VMs. It is fixed in Spring Native v0.10.2.

For more information, see the spring-projects-experimental repository on GitHub.

You can also work around this issue by adding the following hints above your `@SpringBootapp` annotation:

For example:

```
@NativeHint(trigger = ReactiveTAS for VMsActuatorAutoConfiguration.class, types = {
        @TypeHint(types = EndpointTAS for VMsExtension.class, access = AccessBits.ANNO
TATION),
        @TypeHint(typeNames = "org.springframework.boot.actuate.autoconfigure.TAS for
VMs.TAS for VMsEndpointFilter"),
        @TypeHint(typeNames = "org.springframework.boot.actuate.autoconfigure.TAS for
VMs.reactive.TAS for VMsWebFluxEndpointHandlerMapping$TAS for VMsLinksHandler", access
= AccessBits.LOAD_AND_CONSTRUCT
                | AccessBits.DECLARED_METHODS) })
@NativeHint(trigger = TAS for VMsActuatorAutoConfiguration.class, types = {
        @TypeHint(types = EndpointTAS for VMsExtension.class, access = AccessBits.ANNO
TATION),
        @TypeHint(typeNames = "org.springframework.boot.actuate.autoconfigure.TAS for
VMs.TAS for VMsEndpointFilter"),
        @TypeHint(typeNames = "org.springframework.boot.actuate.autoconfigure.TAS for
VMs.servlet.TAS for VMsWebEndpointServletHandlerMapping$TAS for VMsLinksHandler", acce
ss = AccessBits.LOAD_AND_CONSTRUCT
                | AccessBits.DECLARED_METHODS) })
@SpringBootapp
public class Demoapp {

    public static void main(String[] args) {
        Springapp.run(Demoapp.class, args);
    }

}
```

Configuring service connections

The path for configuring service access in your Java based applications is the java-cfenv library. This library can read and parse `VCAP_SERVICES` and help you extract the information for use in your application.

There are a number of ways to implement this and all Java applications can use the library; it is not limited to specific frameworks. To get started, you must first add a dependency in your project for the library.

Dependencies

The following examples show the dependency syntax for Maven and Gradle.

For Maven:

```
<dependency>
  <groupId>io.pivotal.cfenv</groupId>
  <artifactId>java-cfenv</artifactId>
  <version>2.4.0</version>
</dependency>
```

For Gradle:

```
implementation "io.pivotal.cfenv:java-cfenv:2.4.0"
```

Java-only / No framework

The entry point for the library is the class `CfEnv`, which parses Cloud Foundry environment variables. For example, `VCAP_SERVICES`. `VCAP_SERVICES` which contains a JSON string that includes credential information used to access bound services, for example, databases.

Create a `CfEnv` instance and use its `findCredentialsBy*` methods. There are methods for finding by label, name, and tag. Multiple strings can be passed to match against more than one tag, and the finder method supports passing a regex string for pattern matching.

For example:

```
CfEnv cfEnv = new CfEnv();

String redisHost = cfEnv.findCredentialsByTag("redis").getHost();
String redisPort = cfEnv.findCredentialsByTag("redis").getPort();
String redisPassword = cfEnv.findCredentialsByTag("redis").getPassword();

List<CfService> cfService = cfEnv.findAllServices();

CfService redisService = cfEnv.findServiceByTag("redis");
List<String> redisServiceTags = redisService.getTags();
String redisPlan = redisService.getPlan();
redisPlan = redisService.get("plan")

CfCredentials redisCredentials = cfEnv.findCredentialsByTag("redis");
String redisPort = redisCredentials.getPort();
Integer redisPort = redisCredentials.getMap().get("port");

cfService = cfEnv.findServiceByName("redis");
cfService = cfEnv.findServiceByLabel("p-redis");
cfService = cfEnv.findServiceByLabel(".*-redis");
```

JDBC support

There is additional support for getting a JDBC URL from a service binding. This support is contained in the module `java-cfenv-jdbc`. To enable this module, add the appropriate dependency.

For Maven:

```
<dependency>
  <groupId>io.pivotal.cfenv</groupId>
  <artifactId>java-cfenv-jdbc</artifactId>
  <version>2.4.0</version>
</dependency>
```

For Gradle:

```
implementation "io.pivotal.cfenv:java-cfenv-jdbc:2.4.0"
```

The entry point for this feature is the class `CfJdbcEnv`, which is a subclass of `CfEnv` and adds a few methods. The method `findJdbcService` heuristically looks at all services for known tags, labels, and names of common database services to create the URL.

For example:

```
CfJdbcEnv cfJdbcEnv = new CfJdbcEnv()
CfJdbcService cfJdbcService = cfJdbcEnv.findJdbcService();

String jdbcUrl = cfJdbcService.getJdbcUrl();
String username = cfJdbcService.getUsername();
String password = cfJdbcService.getPassword();
String driverClassName = cfJdbcService.getDriverClassName();
```

Spring Framework

The Spring Framework provides additional support for application developers.

Spring Expression Language

If you register the `CfJdbcEnv` class as a bean, then you can use the Spring Expression Language to set properties.

```
@Bean
public CfEnv cfEnv() {
  return new CfEnv();
}
```

Then, in the properties file imported by Spring, refer to the `CfEnv` bean using the following syntax:

```
cassandra.contact-points=#{ cfEnv.findCredentialsByTag('cassandra').get('node_ips') }
cassandra.username=#{ cfEnv.findCredentialsByTag('cassandra').getUserName() }
cassandra.password=#{ cfEnv.findCredentialsByTag('cassandra').getPassword() }
cassandra.port=#{ cfEnv.findCredentialsByTag('cassandra').get('cqlsh_port') }
```

To specifically target JDBC databases, register this instead:

```
@Bean
public CfJdbcEnv cfJdbcEnv() {
  return new CfJdbcEnv();
}
```

Then in a property file imported by Spring, refer to the `CfJdbcEnv` bean using the following syntax:

```
myDatasourceUrl=#{ cfJdbcEnv.findJdbcService().getUrl() }
```

Spring Boot

The module `java-cfenv-boot` provides several `EnvironmentPostProcessor` implementations that set well-known Spring Boot properties so that Spring Boot's auto-configuration is active. For example, the `CfDataSourceEnvironmentPostProcessor` sets the Spring Boot property, `spring.datasource.url`.

To use these, add a dependency on `java-cfenv-boot`.

For Maven:

```
<dependency>
  <groupId>io.pivotal.cfenv</groupId>
  <artifactId>java-cfenv-boot</artifactId>
  <version>2.4.0</version>
</dependency>
```

For Gradle:

```
implementation "io.pivotal.cfenv:java-cfenv-boot:2.4.0"
```

The list of supported services are:

- Databases - DB2, MySQL, Oracle, PostgreSQL, SQL Server
- RabbitMQ
- Cassandra
- MongoDB
- Redis
- CredHub
- HashiCorp Vault

If you need to prevent processing of a specific service instance, set the flag in your application properties to:

```
cfenv.service.{serviceName}.enabled=false
```

Migrating from Spring AutoReconfiguration and Spring Cloud Connectors

The `java-cfenv` library replaces the older Spring AutoReconfiguration and Spring Cloud Connectors libraries. Use the information in the following sections to migrate to `java-cfenv`.

Change dependencies

Remove references to any of these libraries from the application build files.

```
org.springframework.boot:spring-boot-starter-cloud-connectors
```

or

```
org.springframework.cloud:spring-cloud-core
org.springframework.cloud:spring-cloud-connectors-core
org.springframework.cloud:spring-cloud-cloudfoundry-connector
org.springframework.cloud:spring-cloud-spring-service-connector
```

Then add a reference to the `java-cfenv` library.

Code changes

Remove any of the `@ServiceScan` or `@CloudScan` annotations from Spring Java configuration classes (provided by Spring Cloud Connectors). Replace them with the Spring SPeL or Spring Boot configuration options listed above.

Migration considerations

Review these additional considerations before you migrate.

- **Non-Spring Boot applications:**

  If you have a Spring Application that is a non-Spring Boot application, you can still migrate to `java-cfenv`. You must use either the no framework options or the Spring SPeL option. With SPeL, you might need to manually process the expressions, depending on where you are configuring them. See the Spring documentation for places where SPeL expressions are processed by default.

- **Multiple service instances:**

  Spring Cloud Connectors support connections to multiple service instances.

  If you need to configure connections to multiple instances of a given service type, or do anything more than setting application properties for Spring Boot to pick up and use in auto-configuration, then you must follow the manual configuration approaches laid out in the sections above to access the binding credentials. Either with direct Java code or with SPeL. Then follow the same procedure that is used to connect to the services in any other non-Cloud Foundry deployment environment.

- **Code modifications:**

  The Java Buildpack injects the Spring Auto Reconfiguration module code into your application and overwrites your service configuration. This works well in some cases, but sometimes it causes problems.

  With `java-cfenv`, there is no auto-reconfiguration magic. You can explicitly configure your services or you can use the Spring Boot mappers. The Spring Boot mappers are the option most similar to previous operation. Note, that when things don’t work, it’s generally clearer what happened, and it's easier to debug the problem.

- **Cloud property placeholders:**

  The Spring Auto Reconfiguration module exposes a set of property placeholder values that you can use to access values from `VCAP_SERVICES`. If you are using these placeholders, then you must switch from using `cloud.<property>`. Use `vcap.<property>` instead.

  Spring Boot exposes the same information, just under the `vcap.` prefix instead of the `cloud.` prefix.

- **Spring Cloud Profile:**

  The Spring Auto Reconfiguration module enables a Spring Profile called `cloud`, by default. Users have come to expect this behavior when deploying to Cloud Foundry. Without the Spring Auto Reconfiguration module, you do not get this behavior. Fortunately, you can enable it using one of these methods:

  - Run `cf set-env <APP> SPRING_PROFILES_ACTIVE cloud`
  - Add `SPRING_PROFILES_ACTIVE: cloud` to the `env:` block in your `manifest.yml` file. This supplies the list of profiles for Spring to use.

  If you need to set additional profiles, you can use `SPRING_PROFILES_INCLUDE` instead. This appends to the existing set of profiles.

- **Spring Cloud Connector extensions:**

  If you have created any custom Spring Cloud Connector extensions, you must migrate them to `java-cfenv`. This requires two steps:

  1. Write a Spring Boot Auto Configuration library that creates connections to your service from Spring configuration properties. This makes it easy to also use it in a non-cloud Spring Boot app. When this is done correctly, you can to use the library, and set properties in `application.properties` (or through other means), and you can have a connection to your service.
  2. Write a java-cfenv extension. This takes values from `VCAP_SERVICES` and maps them to the properties that you exposed with your Spring Boot Auto Configuration library from the previous step.

Java Buildpack warnings

The Java Buildpack generates warnings to help with migrating from Spring Cloud Connectors and Spring Auto Reconfiguration.

- **Spring Auto Reconfiguration installed:**

  This is the message that is generated when the buildpack installs the Spring Auto Reconfiguration JAR. This happens by default, and notifies you that it is happening.

```
[SpringAutoReconfiguration]      WARN  ATTENTION: The Spring Auto Reconfigurati
on and shaded Spring Cloud Connectors libraries are being installed. These proj
ects have been deprecated, are no longer receiving updates, and should not be u
sed going forward.
[SpringAutoReconfiguration]      WARN  If you are not using these libraries, se
t `JBP_CONFIG_SPRING_AUTO_RECONFIGURATION='{enabled: false}'` to their installa
tion and clear this warning message. The buildpack switches the default to deac
tivate by default after Aug 2022. Spring Auto Reconfiguration and its shaded Sp
ring Cloud Connectors are removed from the buildpack after Dec 2022.
[SpringAutoReconfiguration]      WARN  If you are using these libraries, please
migrate to java-cfenv immediately. See https://via.vmw.com/EhzD for migration i
nstructions
```

How you resolve this depends on whether you are depending on the Auto Reconfiguration to occur.

  - If your application depends on Auto Reconfiguration behavior, then you must make code changes in your application to use the `java-cfenv` library. See the instructions above for how to include the dependencies and how to access service information using this library. After you have added `java-cfenv` to your classpath, the Java buildpack no longer installs the Auto Reconfiguration JAR and you no longer see this message.

  - If your application is not depending on Auto Reconfiguration behavior, and has already been updated to `java-cfenv`, then this message is not generated. If your application does not use Auto Reconfiguration or `java-cfenv`, then you can either:

  - Run `cf set-env <APP> JBP_CONFIG_SPRING_AUTO_RECONFIGURATION '{enabled: false}'`

  - Add `JBP_CONFIG_SPRING_AUTO_RECONFIGURATION: '{enabled: false}'` to the `env:` block in your `manifest.yml` file.

    Alternatively, you can use a buildpack released after Aug 2022, after which this feature is no longer enabled by default.

- **Spring Cloud Connectors present:**

  The following message is generated when the buildpack detects that the Spring Cloud Connectors library is present on the classpath.

```
[SpringAutoReconfiguration]      WARN  ATTENTION: The Spring Cloud Conn
ectors library is present in your application. This library has been in
maintenance mode since July 2019 and stops receiving all updates after
Dec 2022.
[SpringAutoReconfiguration]      WARN  Please migrate to java-cfenv imm
ediately. See https://via.vmw.com/EhzD for migration instructions.
```

  When this message appears, it means that your application or one of its dependencies is including the Spring Cloud Connectors library. You must remove it and migrate to `java-cfenv`.

  After you have migrated to `java-cfenv`, and the Spring Cloud Connectors libraries are no longer on your classpath, this error no longer appears. There is no way to manually suppress this message.

.NET Core buildpack

You can push Cloud Foundry apps to Linux Diego Cells using the .NET Core buildpack. To find supported ASP.NET Core versions, see .NET Core buildpack release notes.

The .NET Core buildpack can only be used to deploy apps to Linux Diego Cells. To deploy .NET Core apps to Windows Diego Cells, use the Binary buildpack. For more information about the Binary buildpack, see Binary Buildpack.

Buildpacks provide needed dependencies to Cloud Foundry apps. Cloud Foundry automatically uses the .NET Core buildpack when one or more of the following conditions are met:

- The pushed app contains one or more `*.csproj` or `*.fsproj` files.
- The app is pushed from the output directory of the `dotnet publish` command.

For information about deploying different types of .NET apps, follow the links in the following table:

Type of .NET App	**Buildpack**
ASP.NET MVC ASP.NET Web Forms ASP.NET WebAPI Apps Windows Communication Foundation (WCF)	HWC
.NET Console	Binary
.NET Core pushed to Windows stack	Binary

Push a .NET Core app

To push a .NET Core app to a Linux Diego Cell:

1. Run:

   ```
   cf push APP-NAME
   ```

   Where `APP-NAME` is the name you want to give your app.

   If your Cloud Foundry deployment does not have the .NET Core buildpack installed or the installed version is out of date, run the same command with the `-b` option to specify the buildpack:

   ```
   cf push APP-NAME -b https://github.com/cloudfoundry/dotnet-core-buildpack.git
   ```

   Where `APP-NAME` is the name you want to give your app.

2. Find the URL of your app in the output of the `cf push` command.
3. Open a browser and navigate to the URL to see your app running.

For a basic sample app, see ASP.NET Core Getting Started App in GitHub.

Source-based, non-published deployments

For a source-based, non-published deployment, you push your app's source code, not the output directory of the `dotnet publish` command.

The source-based, non-published workflow ensures the buildpack can keep all your dependencies in sync and up to date. For additional information about using source-based, non-published deployments, see the following sections:

- Deploy Apps with Multiple Projects
- Use Non-Default Package Sources
- Deactivate and Clear Your NuGet Package Cache

The source-based deployment workflow also uses the `cf push` command to push source-based apps to Cloud Foundry.

Deploying apps with multiple projects

If you are deploying an app containing multiple projects, you must specify which of the app's projects is the main project.

To specify the main project in a multi-project deployment:

1. Create a `.deployment` file in your app's root folder and open the new file in a text editor.
2. Designate the main project's path by configuring the file, using the following format:

   ```
   [config]
   project = PATH-TO-YOUR-MAIN-PROJECT
   ```

   Where `PATH-TO-YOUR-MAIN-PROJECT` is the location of your main project's `*.csproj` or `*.fsproj` file.
   For example:

   ```
   [config]
   project = src/MyApp.Web/MyApp.Web.csproj
   ```

   In this example, by pointing to the `MyApp.Web` `*.csproj` file, `MyApp.Web` is configured as the main project.
3. Save the revised `.deployment` file.

When deployed, the buildpack attempts to run the main project using the `dotnet run` command, `dotnet run -p PATH-TO-YOUR-MAIN-PROJECT`, and automatically compiles all projects listed as dependencies in the main project's `*.csproj` or `*.fsproj` file.

For example: Suppose your `.deployment` file is configured as previously noted and you have an app `src` folder containing three projects: `MyApp.Web`, `MyApp.DAL` and `MyApp.Services`. If your `MyApp.Web.csproj` file lists the `MyApp.DAL` and `MyApp.Services` projects as dependencies, the two additional projects are compiled by the buildpack.

Use non-Default package sources

If you want to deploy an app that uses non-default package sources, you must specify those package sources in the `NuGet.Config` file. For information about `NuGet.Config`, see nuget.config reference in the Microsoft documentation.

Deactivating and clearing your NuGet package cache

You might need to deactivate NuGet package caching or clear NuGet packages cached in the staging environment in one of the following scenarios:

- Your app fails to stage because it runs out of space, exceeding the maximum allowable disk quota.
- You have added pre-release packages to test a new feature and then decided to revert back to the main NuGet feed. You might need to remove the packages you changed from the cache to avoid conflicts.

Deactivating NuGet caching clears any existing NuGet dependencies from the staging cache and prevents the buildpack from adding NuGet dependencies to the staging cache.

NuGet package caching is deactivated by default. If the default is not explicitly overridden, no additional NuGet caching configuration is required.

To deactivate NuGet package caching:

1. Confirm the `CACHE_NUGET_PACKAGES` environment variable is not set to `true` in your app manifest by locating the `manifest.yml` file and confirming `CACHE_NUGET_PACKAGES` is not set to `true`.
2. If needed, set `CACHE_NUGET_PACKAGES` to `false` in the `manifest.yml` file by setting the `CACHE_NUGET_PACKAGES` environment variable to `false`.
   For example:

   ```
   ---
   applications:
   - name: sample-aspnetcore-app
     memory: 512M
     env:
       CACHE_NUGET_PACKAGES: false
   ```

3. To alternatively configure the setting to `false` in the environment variables settings, run the following command:

   ```
   cf set-env APP-NAME CACHE_NUGET_PACKAGES false
   ```

   Where `APP-NAME` is the name of your app.

For more information, see Environment Variables in *Deploying with App Manifests*.

Framework dependent deployments

For a framework-dependent deployment (FDD), you deploy only your app and third-party dependencies. VMware recommends using this workflow if you deploy an app in an offline setting.

For information about deploying FDDs, see Framework-dependent deployments (FDD) in the Microsoft documentation.

Deploy a FDD using the buildpack:

1. Publish the app by running:

   ```
   dotnet publish [-f FRAMEWORK-NAME] [-c Release]
   ```

   Where `FRAMEWORK-NAME` is your target framework.

2. Prepare to push your app with one of these methods:

   - If your app uses a `manifest.yml`, specify a path to the output folder of `dotnet publish`. This allows you to push your app from any directory.
   - If not, go to `bin/Debug|Release/FRAMEWORK-NAME/RUNTIME-NAME/publish` directory, where `FRAMEWORK-NAME` is your target framework, and `RUNTIME-NAME` is the runtime you are using to push your app.

3. Push your app.

Self contained deployments

For a self-contained deployment (SCD), you deploy your app, third-party dependencies, and the version of .NET Core that you used to build your app.

For information about SCDs, see Self-contained deployments (SCD) in the Microsoft documentation.

> **Important**
>
> VMware does not recommend using the SCD workflow. The buildpack is unable to keep dependencies in sync and up to date for workflows that deploy a pre-published binary.

When using the SCD workflow for deploying your app, you must:

- Specify a runtime in the `dotnet publish` command. For example:

  ```
  dotnet publish -r ubuntu14.04-x64
  ```

- Include the specified runtime in the `RuntimeIdentifiers` section of the project file.

Specify .NET Core SDKs

To pin the .NET Core SDK to a specific version or version line, create a `buildpack.yml` file at the app root and add your SDK version in one of the following formats:

```
dotnet-core:
  sdk: 2.1.201
```

```
dotnet-core:
  sdk: 2.1.x
```

```
dotnet-core:
  sdk: 2.x
```

The buildpack chooses what SDK to install based on the files present at the app root in the following order of precedence:

1. `buildpack.yml`
2. `global.json`
3. `*.fsproj`

The app respects the SDK version specified in `global.json` at runtime. If you provide versions in both `global.json` and `buildpack.yml` files, make sure you specify the same versions in both files.

Specify .NET Runtime versions

This section explains how to specify a .NET Runtime version for source-based and framework-dependent apps.

Source-based apps

If you want to lock the .NET Runtime version, configure your `.csproj` or `.fsproj` file to lock your app to the desired version.

For example, the following configuration locks the runtime to 2.1:

```
<PropertyGroup>
  <TargetFramework>netcoreapp2.1</TargetFramework>
  <RuntimeFrameworkVersion>2.1.*</RuntimeFrameworkVersion>
</PropertyGroup>
```

For source-based apps, specify a minor version of the .NET Runtime. Do not specify a patch version, because buildpacks contain only the two most recent patch versions of each minor version.

Framework-dependent apps

Your app is configured to use the latest .NET Runtime patch version by default.

If you want your app to maintain a specific .NET Runtime version, you must modify your app's `.runtimeconfig.json` file to include the `applyPatches` property and set the property to `false`.

For example:

```
{
  "runtimeOptions": {
    "tfm": "netcoreapp2.0",
    "framework": {
      "name": "Microsoft.NETCore.App",
      "version": "2.0.0"
    },
    "applyPatches": false
  }
}
```

Set `applyPatches: false` in `*.runtimeconfig.json` only if you want to pin your .NET Framework to a specific version. This prevents your app from receiving updates to the runtime version and assemblies.

Push an app in a disconnected environment

For offline environments, VMware recommends using the Framework-Dependent Deployment workflow. This workflow enables the deployed app to use the latest runtime provided by the offline buildpack. For more information, see Framework-Dependent Deployments above.

Maintain ASP.NET Core assemblies

This section applies only to source-based and framework-dependent deployments.

For maintaining ASP.NET Core assemblies, VMware recommends one of these methods:

- Configure your app as a fully vendored app requiring fewer buildpack updates. Add the following to your `.csproj` file:

```
<PropertyGroup>
  <PublishWithAspNetCoreTargetManifest>false</PublishWithAspNetCoreTargetManife
st>
</PropertyGroup>
```

- Keep your SDK up to date, by setting `buildpack.yml` to the .NET SDK line you want to use. For example:

```
---
dotnet-core:
        sdk: 2.0.x
```

`2.0.x` ASP.NET Core assemblies are released in the `2.1.200`-`2.1.299` SDK versions, and `2.1.x` assemblies are released in the `2.1.300` and higher SDK versions.

Configuring the listen port

Cloud Foundry sets the `$PORT` environment variable automatically. For your .NET Core app to work on Cloud Foundry, you must configure the app to listen on the environment's specified port.

For C# apps, the following modifications activate the buildpack to pass the correct port from `$PORT` environment variable to the app when you run the initial startup command:

1. Open the file that contains your `Main` method.
2. Add a `using` statement to the top of the file:

```
using Microsoft.Extensions.Configuration;
```

3. Add the following lines before the line `var host = new WebHostBuilder()`:

```
var config = new ConfigurationBuilder()
    .AddCommandLine(args)
    .Build();
```

4. Add the following line after `.UseKestrel()`:

   ```
   .UseConfiguration(config)
   ```

   The `Main` method resembles the following example:

   ```
   public static void Main(string[] args)
   {
       var config = new ConfigurationBuilder()
           .AddCommandLine(args)
           .Build();
       var host = new WebHostBuilder()
           .UseKestrel()
           .UseConfiguration(config)
           .UseContentRoot(Directory.GetCurrentDirectory())
           .UseStartup<Startup>()
           .Build();
       host.Run();
   }
   ```

5. Save your changes.

6. Add `Microsoft.Extensions.Configuration.CommandLine` as a dependency in `*.csproj`:

   ```
   <PackageReference Include="Microsoft.Extensions.Configuration.CommandLine">
     <Version>VERSION</Version>
   </PackageReference>
   ```

   Where `VERSION` is the version of the package to use. For a list of valid versions, navigate to https://www.nuget.org.

7. If your app requires any other files at runtime, such as JSON configuration files, add them to the `include` section of `copyToOutput`.

8. Save your changes.

With these changes, the `dotnet run` command copies your app `Views` to the build output where the .NET CLI can find them.

Add custom libraries

If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `ld_library_path` directory at the app root.

You must keep these libraries up to date. They do not update automatically.

The .NET Core buildpack adds the directory `APP-ROOT/ld_library_path` to `LD_LIBRARY_PATH`, where `APP-ROOT` is the app root, so your app can access these libraries at runtime.

NGINX buildpack

You can push your NGINX app to Cloud Foundry and configure your NGINX app to use the NGINX buildpack.

Push an app

If your app contains an `nginx.conf` file, Cloud Foundry automatically uses the NGINX buildpack when you run `cf push` to deploy your app.

If your Cloud Foundry deployment does not have the NGINX buildpack installed or the installed version is outdated, deploy your app with the current buildpack by running:

```
cf push YOUR-APP -b https://github.com/cloudfoundry/nginx-buildpack.git
```

Where `YOUR-APP` is the name of your app.

For example:

```
$ cf push my-app -b https://github.com/cloudfoundry/nginx-buildpack.git
```

Configure NGINX

VMware recommends that you use the default NGINX directory structure for your NGINX web server. You can view this directory structure in the nginx-buildpack repository in GitHub.

Configure the NGINX web server. You need these elements.

- A root directory for all static web content
- A MIME type configuration file
- An NGINX configuration file
- A `buildpack.yml` YAML file that defines the version of NGINX to use. For example, see buildpack.yml Buildpack repository in GitHub.

Make any custom configuration changes based on these default files to verify compatibility with the buildpack.

Create the nginx.conf file

Use the templating syntax when you create an `nginx.conf` file. This templating syntax loads modules and binds to ports based on values known at start time.

Port

Use `{{port}}` to set the port on which to listen. At start time, `{{port}}` interpolates in the value of `$PORT`.

You must use `{{port}}` in your `nginx.conf` file.

For example, to set an NGINX server to listen on `$PORT`, include the following in your `nginx.conf` file:

```
server {
  listen {{port}};
}
```

Name resolution

The NGINX buildpack does not resolve internal routes by default. To resolve internal routes, use `{{nameservers}}` to set the resolver IP address. At start time, `{{nameservers}}` interpolates the address of a platform-provided DNS service that includes information about internal routes.

Connections to internal routes do not go through the Cloud Foundry routing tier. As a result, you might see errors if you proxy an app on an internal route while it is restarting. There are some workarounds you might need to consider.

For more information, see Using DNS for Service Discovery with NGINX and NGINX Plus on the NGINX blog.

Environment variables

To use an environment variable, include `{{env "YOUR-VARIABLE"}}`, where `YOUR-VARIABLE` is the name of an environment variable. At staging and at startup, the current value of the environment variable is retrieved.

For example, include the following in your `nginx.conf` file to activate or deactivate GZipping based on the value of `GZIP_DOWNLOADS`:

```
gzip {{env "GZIP_DOWNLOADS"}};
```

- If you set `GZIP_DOWNLOADS` to `off`, NGINX does not GZip files.
- If you set `GZIP_DOWNLOADS` to `on`, NGINX GZips files.

Unescaped environment variables

To use unescaped environment variables, add an array of environment variable names to the `buildpack.yml`. See the following example:

```
---
nginx:
  version: stable
  plaintext_env_vars:
    - "OVERRIDE"
```

In this example, the `OVERRIDE` environment variable can contain `.json` content without being `html` escaped. You must properly quote such variables to appear as strings in the `nginx.conf` file.

Loading dynamic modules

NGINX can dynamically load modules at runtime. These modules are shared-object files that can be dynamically loaded using the `load_module` directive. In addition to loading modules dynamically, the NGINX version provided by the buildpack has statically compiled the following modules into the NGINX binary:

- `ngx_http_ssl_module`
- `ngx_http_realip_module`
- `ngx_http_gunzip_module`

- ngx_http_gzip_static_module
- ngx_http_auth_request_module
- ngx_http_random_index_module
- ngx_http_secure_link_module
- ngx_http_stub_status_module
- ngx_http_sub_module

These statically compiled modules do not need to be loaded at runtime and are already available for use.

To load a dynamic NGINX module, use the following syntax in the app `nginx.conf` file for your app:

```
{{module "MODULE-NAME"}}
```

If you have provided a module in a `modules` directory located at the root of your app, the buildpack instructs NGINX to load that module. If you have not provided a module, the buildpack instructs NGINX to search for a matching built in dynamic module.

As of v0.0.5 of the buildpack, the `ngx_stream_module` is available as a dynamic module that is built into the buildpack.

For example, to load a custom module named `ngx_hello_module`, provide a `modules/ngx_hello_module.so` file in your app directory and add the line below to the top of your `nginx.conf` file:

```
{{module "ngx_hello_module"}}
```

To load a built in module like `ngx_stream_module`, add the following line to the top of your `nginx.conf` file. You do not need to provide an `ngx_stream_module.so` file:

```
{{module "ngx_stream_module"}}
```

To name your modules directory something other than `modules`, use the NGINX `load_module` directive, providing a path to the module relative to the location of your `nginx.conf` file. For example: `load_module some_module_dir/my_module.so`

Enable logging

By default, logging is deactivated in the NGINX buildpack. This helps optimize performance.

If you configure NGINX to log to stdout or stderr, the logs are captured by the Cloud Foundry logging subsystem.

Logging access

Use the `access_log` directive in the appropriate location in the `nginx.conf` to enable access logging. Use the following syntax:

```
access_log &lt;file&gt; [format]
```

Where:

- `file` is the name of the file where the log is to be stored. The special value `/dev/stdout` selects the standard output.
- `format` is the logging format to use. Refer to the NGINX documentation for valid customization options.

Example:

```
access_log /dev/stdout;
```

Logging errors

Set the debug level with the `error_log` directive in the `nginx.conf` to enable debug logging. Add the `error_log` entry to your `nginx.conf` file using the syntax below:

```
error_log &lt;file&gt; [level];
```

Where:

- `file` is the name of the file where the log is to be stored. The special value `stderr` selects the standard error file. Logging to `syslog` can be configured by specifying the “`syslog:`” prefix.
- `level` specifies the level of logging, and can be one of the following:
  - `debug`
  - `info`
  - `notice`
  - `warn`
  - `error` (default)
  - `crit`
  - `alert`
  - `emerg`

The log levels above are listed in the order of increasing severity. Setting a certain log level causes all messages of the specified and more severe log levels to be logged. For example, the default level `error` causes error, crit, alert, and emerg messages to be logged. If this parameter is omitted, then `error` is used.

Example:

```
error_log stderr debug;
```

Debug logs for selected clients

To enable the debugging log for selected client addresses only, use the syntax below.

```
error_log stderr;

events {
debug_connection 192.168.1.1;
debug_connection 192.168.10.0/24;
}
```

Logging to a cyclic memory buffer

The debugging log can be written to a cyclic memory buffer. Use the syntax shown below.

```
error_log memory:32m debug;
```

Logging to the memory buffer on the debug level does not have significant impact on performance, even under high load. In this case, the log can be extracted using a gdb script like the one in the following example:

```
set $log = ngx_cycle->log

while $log->writer != ngx_log_memory_writer
set $log = $log->next
end

set $buf = (ngx_log_memory_buf_t *) $log->wdata
dump binary memory debug_log.txt $buf->start $buf->end
```

NGINX buildpack support

The resources listed in this section can assist you when using the NGINX buildpack or when developing your own NGINX buildpack.

- **NGINX Buildpack Repository in GitHub**: Find more information about using and extending the NGINX buildpack in the NGINX buildpack GitHub repository.
- **Release Notes**: Find current information about this buildpack on the NGINX buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in the Cloud Foundry Slack community.

Node.js

In this section:

- Node.js Buildpack
- Tips for Node.js Developers
- Environment Variables Defined by the Node Buildpack
- Configuring Service Connections for Node.js

Node.js buildpack

You can configure and use the Node.js buildpack in Cloud Foundry.

You can use the Node.js buildpack with Node or JavaScript apps. The follow sections describe how to push apps with the Node.js buildpack, along with features of the buildpack.

You must install the Cloud Foundry Command Line Interface tool (cf CLI) to run some of the commands listed in this topic.

Push Node.js apps

Cloud Foundry automatically uses the Node.js buildpack if it detects a `package.json` file in the root directory of your project.

The `-b` option lets you specify a buildpack to use with the `cf push` command. If your Cloud Foundry deployment does not have the Node.js buildpack installed or the installed version is out of date, run the following command to push your app with the latest version of the buildpack:

```
cf push APP-NAME -b https://github.com/cloudfoundry/nodejs-buildpack
```

Where `APP-NAME` is the name of your app.

For example:

```
$ cf push my-nodejs-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

For more detailed information about deploying Node.js apps, see the following topics:

- Tips for Node.js Developers
- Environment Variables Defined by the Node Buildpack
- Configuring Service Connections for Node
- Node.js Buildpack Source Code on GitHub

Supported versions

For a list of supported Node versions, see the Node.js Buildpack release notes on GitHub.

Specify a Node.js version

To specify a Node.js version, set the `engines.node` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of Node you are using.

Example showing a semver range:

```
"engines": {
  "node": "6.9.x"
}
```

Example showing a specific version:

```
"engines": {
  "node": "6.9.0"
}
```

If you try to use a version that is not currently supported, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:...
!
!     exit
!
Staging failed: Buildpack compilation step failed
```

Specify an npm version

To specify an npm version, set `engines.npm` in the `package.json` file to the semantic versioning specification (semver) range or the specific version of npm you are using:

Example showing a semver range:

```
"engines": {
  "node": "6.9.x",
  "npm": "2.15.x"
}
```

Example showing a specific version:

```
"engines": {
  "node": "6.9.0",
  "npm": "2.15.1"
}
```

If you do not specify an npm version, your app uses the default npm packaged with the Node.js version used by your app, as specified on the Node.js releases page.

If your environment cannot connect to the Internet and you specified a non-supported version of npm, the buildpack fails to download npm and you see the following error message:

```
We're unable to download the version of npm you've provided (...).
Please remove the npm version specification in package.json (...)
Staging failed: Buildpack compilation step failed
```

Using npm or Yarn

By default, the Node.js buildpack assumes you are using npm. If you want to use Yarn instead, you must provide a `yarn.lock` in your top-level app directory. For more information on the Yarn lock file, see yarn.lock in the Yarn documentation.

Vendoring app dependencies

To vendor dependencies for an app using the Node.js buildpack, run `npm install` (or `yarn install`, if you are using Yarn) from your app directory. This command vendors dependencies into the `node_modules` directory of your app directory.

For example, the following example vendors dependencies into the `my-nodejs-app/node_modules` directory:

```
$ cd my-nodejs-app
$ npm install
```

The `cf push` command uploads the vendored dependencies with the app.

For an app to run in a disconnected environment, it must vendor its dependencies and provide a lock file.

Use in offline environments

When vendoring apps for usage in offline environments, you must supply a lock file. For information on npm lockfiles, see npm-package-locks in the NPM documentation.

For information on `npm-package-lock.json` files, see package-lock.json in the NPM documentation.

This lock file informs the package manager of the exact versions of dependencies and transitive dependencies to look for when running `npm install` or `yarn install`. For this reason, you must keep `package.json`, the vendored `node_modules` directory, and your lock file synchronized to avoid network calls.

> **Important**
>
> The `package-lock.json` file is only supported by NPM 5.x and later. For earlier versions of NPM, provide a npm-shrinkwrap.json file instead.

Versions 1.5.28 and later of the Node.js buildpack include the ability to use Yarn in offline mode. To do so, you must mirror the Yarn registry locally by providing an `npm-packages-offline-cache` directory:

```
$ cd APP-DIR
$ yarn config set yarn-offline-mirror ./npm-packages-offline-cache
$ yarn config set yarn-offline-mirror-pruning true
$ cp ~/.yarnrc .
$ rm -rf node_modules/ yarn.lock # if they were previously generated
$ yarn install
```

When you push the app, the buildpack looks for an `npm-packages-offline-cache` directory at the top level of the app directory. If this directory exists, the buildpack runs Yarn in offline mode. Otherwise, it runs Yarn normally, which may require an Internet connection. You do not have to provide a `node_modules` directory when running Yarn in offline mode, as the offline cache provides the dependencies.

For more information about using an offline mirror with Yarn, see the Yarn Blog.

Integrity check

By default, the Node.js buildpack uses npm to download dependencies. Note, that npm does not perform integrity checks of the downloaded packages, which is a security risk.

If missing dependencies are detected, the buildpack runs `npm install` for non-vendored dependencies or `npm rebuild` for dependencies that are already vendored. This might result in code being downloaded and run without verification.

You can use Yarn as an alternative that verifies dependencies.

OpenSSL support

The nodejs-buildpack packages binaries of Node.js with OpenSSL that are statically linked. The Node.js buildpack supports Node.js 4.x and later, which relies on the Node.js release cycle to provide OpenSSL updates. The binary-builder enables static OpenSSL compilation.

Proxy support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH configured custom trusted certificate support

Node.js hardcodes root CA certs in its source code. To use BOSH configured custom trusted certificates, a developer must pass the specified CAs to the tls.connect function as extra arguments.

Help and support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Node.js buildpack in Cloud Foundry, see the Node.js GitHub repository.

You can find current information about this buildpack in the Node.js buildpack release page in GitHub.

Node.js buildpack additional information

You can use node-specific information to supplement the general guidelines in the Pushing an app topic.

For information about using and extending the Node.js buildpack in Cloud Foundry, see the nodejs-buildpack repository in GitHub.

You can find current information about this buildpack on the Node.js buildpack release page in GitHub.

The buildpack uses a default Node.js version. To specify the versions of Node.js and npm an app requires, edit the app’s `package.json`, as described in “node.js and npm versions” in the nodejs-buildpack repository.

Application package file

Cloud Foundry expects a `package.json` in your Node.js app. You can specify the version of Node.js you want to use in the `engine` node of your `package.json` file.

In general, Cloud Foundry supports the two most recent versions of Node.js. See the GitHub Node.js buildpack page for current information.

Example `package.json` file:

```
{
  "name": "first",
  "version": "0.0.1",
  "author": "Demo",
  "dependencies": {
    "express": "3.4.8",
    "consolidate": "0.10.0",
    "swig": "1.3.2"
  },
  "engines": {
    "node": "0.12.7",
    "npm": "2.7.4"
  }
}
```

Application port

You must use the PORT environment variable to determine which port your app listens on. To also run your app locally, set the default port as `3000`.

```
app.listen(process.env.PORT || 3000);
```

Low Memory environments

When running node apps, you might notice that instances are occasionally restarted due to memory constraints. Node does not know how much memory it is allowed to use, and thus sometimes allows the garbage collector to wait past the allowed amount of memory. To resolve this issue, set the `OPTIMIZE_MEMORY` environment variable to `true` (requires node v6.12.0 or greater). This sets `max_old_space_size` based on the available memory in the instance.

```
$ cf set-env my-app OPTIMIZE_MEMORY true
```

Application start command

Node.js apps require a start command. You can specify the web start command for a Node.js app in a Procfile or in the app deployment manifest. For more information about Procfiles, see the Configuring a Production Server topic.

The first time you deploy, you are asked if you want to save your configuration. This saves a `manifest.yml` in your app with the settings you entered during the initial push. Edit the `manifest.yml` file and create a start command as follows:

```
---
applications:
- name: my-app
  command: node my-app.js
... the rest of your settings ...
```

Alternately, specify the start command with `cf push -c`.

```
$ cf push my-app -c "node my-app.js"
```

Application bundling

You do not need to run `npm install` before deploying your app. Cloud Foundry runs it for you when your app is pushed. You can, if you prefer, run `npm install` and create a `node_modules` folder inside of your app.

Solve discovery problems

If Cloud Foundry does not automatically detect that your app is a Node.js app, you can override auto-detection by specifying the Node.js buildpack.

Add the buildpack into your `manifest.yml` and re-run `cf push` with your manifest:

```
---
applications:
- name: my-app
  buildpacks: https://github.com/cloudfoundry/nodejs-buildpack
... the rest of your settings ...
```

Alternately, specify the buildpack on the command line with `cf push -b`:

```
$ cf push my-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

Bind services

Refer to Configure Service Connections for Node.js.

Environment variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` as follows:

```
process.env.VCAP_SERVICES
```

Environment variables available to you include both those defined by the system and those defined by the Node.js buildpack, as described below.

BUILD_DIR directory

Directory into which Node.js is copied each time a Node.js app is run.

CACHE_DIR directory

Directory that Node.js uses for caching.

PATH

The system path used by Node.js.

`PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin`

Environment variables in Node buildpack

Cloud Foundry provides configuration information to apps through environment variables. You can also use the additional environment variables provided by the Node buildpack.

For more information about the standard environment variables provided, see TAS for VMs Environment Variables.

Node buildpack environment Variables

The following table describes the environment variables provided by the Node buildpack:

Environment Variable	Description
`BUILD_DIR`	The directory where Node.js is copied each time a Node.js app runs.
`CACHE_DIR`	The directory Node.js uses for caching.
`PATH`	The system path used by Node.js:`PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin`

Configuring service connections for Node.js apps

You can bind a data source to a Node.js application that is deployed and running on Cloud Foundry.

Parse VCAP_SERVICES for credentials

You must parse the `VCAP_SERVICES` environment variable in your code to get the required connection details such as host address, port, user name, and password.

For example, if you are using PostgreSQL, your `VCAP_SERVICES` environment variable might look something like this:

```
{
    "mypostgres": [{
        "name": "myinstance",
        "credentials": {
            "uri": "postgres://myusername:mypassword@host.example.com:5432/serviceinst
ance"
        }
    }]
}
```

This example JSON is simplified; yours might contain additional properties.

Parse with cfenv

The `cfenv` package provides access to Cloud Foundry application environment settings by parsing all the relevant environment. The settings are returned as JavaScript objects. `cfenv` provides reasonable defaults when running locally, as well as when running as a Cloud Foundry application.

For more information, see the npm website.

Manual parsing

First, parse the `VCAP_SERVICES` environment variable.

For example:

```
var vcap_services = JSON.parse(process.env.VCAP_SERVICES)
```

Then pull out the credential information required to connect to your service. Each service packages requires different information. If you are working with Postgres, for example, you need a `uri` to connect. You can assign the value of the `uri` to a variable as follows:

```
var uri = vcap_services.mypostgres[0].credentials.uri
```

Once assigned, you can use your credentials as you would normally in your program to connect to your database.

Connecting to a service

You must include the appropriate package for the type of services your application uses.

For example:

- Rabbit MQ through the amqp module
- mongoose modules
- MySQL through the mysql module
- Postgres through the pg module
- Redis through the redis module

Add dependency to package.json

Edit `package.json` and add the intended module to the `dependencies` section. Normally, only one is necessary, but for the sake of the example, add all of them:

```
{
  "name": "hello-node",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "mongodb": "*",
    "mongoose": "*",
    "mysql": "*",
    "pg": "*",
    "redis": "*",
    "amqp": "*"
  },
  "engines": {
    "node": "0.8.x"
  }
}
```

You must run `npm shrinkwrap` to regenerate your `npm-shrinkwrap.json` file after you edit `package.json` file.

PHP

In this section:

- PHP Buildpack
- Tips for PHP Developers
- Getting Started Deploying PHP Apps
- PHP Buildpack Configuration
- Composer
- Sessions
- New Relic

Using PHP buildpack with runtimes

You can use the PHP buildpack with PHP or HHVM runtimes.

Supported software and versions

The release notes page has a list of currently supported modules and packages.

- **PHP Runtimes**
  - php-cli
  - php-cgi
  - php-fpm
- **Third-Party Modules**
  - New Relic, in connected environments only.

Push an app

30-second tutorial

With the Cloud Foundry Command Line Interface installed, open a shell, change directories to the root of your PHP files and push your application using the argument `-b https://github.com/cloudfoundry/php-buildpack.git`.

Example:

```
$ mkdir my-php-app
$ cd my-php-app
$ cat << EOF > index.php
<?php
  phpinfo();
?>
EOF
$ cf push -m 128M -b https://github.com/cloudfoundry/php-buildpack.git my-php
-app
```

Change **my-php-app** in the above example to a unique name on your target Cloud Foundry instance to prevent a hostname conflict error and failed push.

The previous example creates and pushes a test application, **my-php-app**, to Cloud Foundry. The `-b` argument instructs CF to use this buildpack. The remainder of the options and arguments are not specific to the buildpack, for questions on those consult the output of `cf help push`.

Here's a breakdown of what happens when you run the example:

- On your PC:
  - It creates a new directory and one PHP file, which just invokes `phpinfo()`
  - Run `cf` to push your application. This creates a new application with a memory limit of 128M (more than enough here) and upload our test file.
- Within Cloud Foundry:
  - The buildpack is run.
  - Application files are copied to the `htdocs` folder.
  - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults and run.
  - Your application is accessible at the URL `http://my-php-app.example.com` (Replacing `example.com` with the domain of your public CF provider or private instance).

More information about deployment

While the *30 Second Tutorial* shows how quick and easy it is to get started using the buildpack, it skips over quite a bit of what you can do to adjust, configure and extend the buildpack. The following sections and links provide a more in-depth look at the buildpack.

Features

Here are some special features of the buildpack.

- Supports running commands or migration scripts prior to application startup.
- Supports an extension mechanism that allows the buildpack to provide additional functionality.
- Allows for application developers to provide custom extensions.
- Easy troubleshooting with the `BP_DEBUG` environment variable.

- Download location is configurable, allowing users to host binaries on the same network (i.e. run without an Internet connection)
- Smart session storage, defaults to file w/sticky sessions but can also use redis for storage.

Examples

Here are some example applications that can be used with this buildpack.

- php-info This app has a basic index page and shows the output of `phpinfo()`.
- PHPMyAdmin A deployment of PHPMyAdmin that uses bound MySQL services.
- PHPPgAdmin A deployment of PHPPgAdmin that uses bound PostgreSQL services.
- Drupal A deployment of Drupal that uses bound MySQL service.
- CodeIgniter CodeIgniter tutorial application running on CF.
- Stand Alone An example which runs a standalone PHP script.
- pgbouncer An example which runs the PgBouncer process in the container to pool database connections.
- phalcon An example which runs a Phalcon based application.
- composer An example which uses Composer.

Advanced topics

See the following topics:

- Tips for PHP Developers
- Getting Started Deploying PHP Apps
- PHP Buildpack Configuration
- Composer
- Sessions
- New Relic

You can find the source for the buildpack on GitHub: https://github.com/cloudfoundry/php-buildpack

Proxy support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and/or `https_proxy` environment variables. For more information, see the Proxy Usage Docs.

BOSH configured custom trusted certificate support

For versions of PHP 5.6.0 and later, the default certificate location is `/usr/lib/ssl/certs`, which symlinks to `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the PHP buildpack in Cloud Foundry, see the php-buildpack GitHub repository.

You can find current information about this buildpack on the PHP buildpack release page in GitHub.

License

The Cloud Foundry PHP Buildpack is released under v2.0 of the Apache License.

PHP buildpacks in Cloud Foundry

For information about using and extending the PHP buildpack in Cloud Foundry, see the php-buildpack GitHub repository.

You can find current information about this buildpack on the PHP buildpack release page in GitHub.

The buildpack uses a default PHP version specified in .defaults/options.json under the `PHP_VERSION` key.

To change the default version, specify the `PHP_VERSION` key in your app's `.bp-config/options.json` file.

Deploying PHP apps to Cloud Foundry

Learn about deploying your PHP apps to Cloud Foundry. If you experience a problem deploying PHP apps, review the Troubleshooting section.

Getting started

Prerequisites

- Basic PHP knowledge
- Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

A first PHP application

```
$ mkdir my-php-app
$ cd my-php-app
$ cat << EOF > index.php
<?php
  phpinfo();
?>
EOF
$ cf push my-php-app -m 128M
```

Change "my-php-app" to a unique name or you may see an error and a failed push.

The previous example creates and pushes a test application to Cloud Foundry.

Here is a breakdown of what happens when you run the example:

- On your workstation…
    - It creates a new directory and one PHP file, which calls `phpinfo()`
    - Run `cf push` to push your application. This creates a new application with a memory limit of 128M and uploads our test file.
- On Cloud Foundry…
    - The buildpack detects that your app is a php app
    - The buildpack is run.
        - Application files are copied to the `htdocs` folder.
    - Apache HTTPD & PHP are downloaded, configured with the buildpack defaults, and run.
    - Your application is accessible at the default route. Run `cf app my-php-app` to view the url of your new app.

Folder structure

The easiest way to use the buildpack is to put your assets and PHP files into a directory and push it to Cloud Foundry. This way, the buildpack takes your files and moves them into the `WEBDIR` (defaults to `htdocs`) folder, which is the directory where your chosen web server looks for the files.

URL rewriting

If you select Apache as your web server, you can include `.htaccess` files with your application.

Also, you can provide your own Apache or Nginx configurations.

Prevent access To PHP files

The buildpack puts all of your files into a publicly accessible directory. In some cases, you might want to have PHP files that are not publicly accessible but are on the include_path. To do that, create a `lib` directory in your project folder and place your protected files there.

For example:

```
$ ls -lRh
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff     0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff   102B Feb 27 21:40 lib

./lib:
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:40 my.class.php  <-- not public,
http://app.cfapps.io/lib/my.class.php == 404
```

This comes with a catch. If your project legitimately has a `lib` directory, these files will not be publicly available because the buildpack does not copy a top-level `lib` directory into the `WEBDIR` folder. If your project has a `lib` directory that needs to be publicly available, then you have two options as follows:

Option #1

In your project folder, create an `htdocs` folder (or whatever you've set for `WEBDIR`). Then move any files that can be publicly accessible into this directory. In the following example, the `lib/controller.php` file is publicly accessible.

Example:

```
$ ls -lRh
total 0
drwxr-xr-x  7 daniel  staff   238B Feb 27 21:48 htdocs

./htdocs:  <--  create the htdocs directory and put your files there
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff     0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff   102B Feb 27 21:48 lib

./htdocs/lib:   <--  anything under htdocs is public, including a lib directo
ry
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:48 controller.php
```

Given this setup, it is possible to have both a public `lib` directory and a protected `lib` directory. The following example demonstrates this setup:

Example:

```
$ ls -lRh
total 0
drwxr-xr-x  7 daniel  staff   238B Feb 27 21:48 htdocs
drwxr-xr-x  3 daniel  staff   102B Feb 27 21:51 lib

./htdocs:
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:40 images
-rw-r--r--  1 daniel  staff     0B Feb 27 21:39 index.php
drwxr-xr-x  3 daniel  staff   102B Feb 27 21:48 lib

./htdocs/lib:  <-- public lib directory
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:48 controller.php

./lib: <-- protected lib directory
total 0
-rw-r--r--  1 daniel  staff     0B Feb 27 21:51 my.class.php
```

Option #2

The second option is to pick a different name for the `LIBDIR`. This is a configuration option that you can set (it defaults to `lib`). Thus if you set it to something else such as `include`, your application's `lib` directory would no longer be treated as a special directory and it would be placed into `WEBDIR` (i.e. become public).

Other folders

Beyond the `WEBDIR` and `LIBDIR` directories, the buildpack also supports a `.bp-config` directory and a `.extensions` directory.

The `.bp-config` directory exists at the root of your project directory and it is the location of application-specific configuration files. Application-specific configuration files override the default settings used by the buildpack. This link explains application configuration files in depth.

The `.extensions` directory also exists at the root of your project directory and it is the location of application-specific custom extensions. Application-specific custom extensions allow you, the developer, to override or enhance the behavior of the buildpack. See the php-buildpack README in GitHub for more information about extensions.

Troubleshooting

To troubleshoot problems using the buildpack, you can do one of the following:

1. Review the output from the buildpack. The buildpack writes basic information to stdout, for example the files that it downloads. The buildpack also writes information in the form of stack traces when a process fails.
2. Review the logs from the buildpack. The buildpack writes logs to disk. Follow these steps to access these logs:
    1. SSH into the app container by running:

        ```
        cf ssh APP-NAME
        ```

        Where `APP-NAME` is the name of your app.
    2. To view the logs, run:

        ```
        cat app/.bp/logs/bp.log
        ```

By default, log files contain more detail than output to stdout. Set the `BP_DEBUG` environment variable to `true` for more verbose logging. This instructs the buildpack to set its log level to `DEBUG`, and to output logs to stdout. Follow Environment Variables documentation to set `BP_DEBUG`.

Increasing log output with fpm.d

If you use fpm.d, follow these steps to configure `fpm` to redirect worker stdout and stderr into the logs:

1. Create a file in the `.bp-config/php/fpm.d/` directory of your app.
2. Add `catch_workers_output=yes` to the file you created.

3. Push your app.

For more information about allowed configuration settings in the `.bp-config/php/fpm.d` directory, see the List of global php-fpm.conf directives.

You can see an example fmp fixture and configuration file in the php-buildpack GitHub repository.

Configuring the PHP buildpack

You can modify the configuration file for your PHP buildpack.

The PHP buildpack stores all default configuration settings in the defaults directory of the Cloud Foundry PHP Buildpack repository on GitHub.

options.json file

The `options.json` file is the configuration file for the buildpack itself. It instructs the buildpack what to download, where to download it from, and how to install it. It allows you to configure:

- Package names and versions, such as PHP, HTTPD, or Nginx versions.
- The web server to use, such as HTTPD, Nginx, or no server.
- The PHP extensions that are enabled.

The buildpack overrides the default `options.json` file with any configuration it finds in the `.bp-config/options.json` file of your app.

Here are explanations of the common options you might need to change:

Variable	Explanation
WEB_SERVER	Sets the web server to use. Must be one of `httpd`, `nginx`, or `none`. This value defaults to `httpd`.
HTTPD_VERSION	Sets the version of Apache HTTPD to use. Currently the build pack supports the latest stable version. This value defaults to the latest release that is supported by the build pack.
ADMIN_EMAIL	The value used in HTTPD's configuration for ServerAdmin
NGINX_VERSION	Sets the version of Nginx to use. By default, the buildpack uses the latest stable version.
PHP_VERSION	Sets the version of PHP to use. Set to a minor instead of a patch version, such as `"{PHP_70_LATEST}"`. See options.json.
PHP_EXTENSIONS	(DEPRECATED) A list of the extensions to enable. `bz2`, `zlib`, `curl`, and `mcrypt` are enabled by default.
ZEND_EXTENSIONS	A list of the Zend extensions to enable. Nothing is enabled by default.
APP_START_CMD	When the `WEB_SERVER` option is set to `none`, this command is used to start your app. If `WEB_SERVER` and `APP_START_CMD` are not set, then the buildpack searches, in order, for `app.php`, `main.php`, `run.php`, or `start.php`. This option accepts arguments.
WEBDIR	The root directory of the files served by the web server specified in `WEB_SERVER`. Defaults to `htdocs`. Other common settings are `public`, `static`, or `html`. The path is relative to the root of your app.

Variable	Explanation
LIBDIR	This path is added to PHP's `include_path`. Defaults to `lib`. The path is relative to the root of your app.
HTTP_PROXY	The buildpack downloads uncached dependencies using HTTP. If you are using a proxy for HTTP access, set its URL here.
HTTPS_PROXY	The buildpack downloads uncached dependencies using HTTPS. If you are using a proxy for HTTPS access, set its URL here.
ADDITIONAL_PREPROCESS_CMDS	A list of additional commands that run prior to the app starting. For example, you might use this command to run migration scripts or static caching tools before the app launches.

For details about supported versions, see the release notes for your buildpack version on the Releases page of the Cloud Foundry PHP Buildpack repository on GitHub.

HTTPD, Nginx, and PHP configuration

The buildpack automatically configures HTTPD, Nginx, and PHP for your app. This section explains how to modify the configuration.

The `.bp-config` directory in your app can contain configuration overrides for these components. Name the directories `httpd`, `nginx`, and `php`. VMware recommends that you use php.ini.d or fpm.d.

If you override the `php.ini` or `php-fpm.conf` files, many other forms of configuration do not work.

For example:

```
.bp-config
  httpd
  nginx
  php
```

Each directory can contain configuration files that the component understands.

For example, to change HTTPD logging configuration, run:

```
ls -l .bp-config/httpd/extra/
total 8
-rw-r--r--  1 daniel  staff  396 Jan  3 08:31 httpd-logging.conf
```

In this example, the `httpd-logging.conf` file overrides the one provided by the buildpack. VMware recommends that you copy the default from the buildpack and modify it.

You can find the default configuration files in the PHP Buildpack `/defaults/config` directory.

You must be careful when modifying configurations, as doing so can cause your app to fail, or cause Cloud Foundry to fail to stage your app.

You can add your own configuration files. The components do not reference them, so you must ensure that they are included. For example, you can add an include directive to the httpd configuration to include your file:

```
ServerRoot "${HOME}/httpd"
Listen ${PORT}
ServerAdmin "${HTTPD_SERVER_ADMIN}"
```

```
ServerName "0.0.0.0"
DocumentRoot "${HOME}/#{WEBDIR}"
Include conf/extra/httpd-modules.conf
Include conf/extra/httpd-directories.conf
Include conf/extra/httpd-mime.conf
Include conf/extra/httpd-logging.conf
Include conf/extra/httpd-mpm.conf
Include conf/extra/httpd-default.conf
Include conf/extra/httpd-remoteip.conf
Include conf/extra/httpd-php.conf
Include conf/extra/httpd-my-special-config.conf # This line includes your additional f
ile.
```

.bp-config/php/php.ini.d/

The buildpack adds any `.bp-config/php/php.ini.d/FILE-NAME.ini` files it finds in the app to the PHP configuration. You can use this to change any value acceptable to `php.ini`. For a list of directives, see http://php.net/manual/en/ini.list.php.

For example, adding a file `.bp-config/php/php.ini.d/something.ini` to your app, with the following contents, overrides both the default charset and mimetype:

```
default_charset="UTF-8"
default_mimetype="text/xhtml"
```

Precedence

In order of highest precedence, PHP configuration values come from the following sources:

- PHP scripts using `ini_set()` to manually override config files
- `user.ini` files for local values
- `.bp-config/php/php.ini.d` to override main value, but not local values from user.ini files

.bp-config/php/fpm.d/

The buildpack adds any files it finds in the app under `.bp-config/php/fpm.d` that end with `.conf` (i.e `my-config.conf`) to the PHP-FPM configuration. You can use this to change any value acceptable to `php-fpm.conf`. For a list of directives, see http://php.net/manual/en/install.fpm.configuration.php.

PHP FPM config snippets are included by the buildpack into the global section of the configuration file. If you need to apply configuration settings for a PHP FPM worker, you must indicate this in your configuration file.

For example:

```
; This option is specific to the `www` pool
[www]
catch_workers_output = yes
```

PHP Extensions

The buildpack adds any `.bp-config/php/php.ini.d/FILE-NAME.ini` files it finds in the app to the PHP configuration. You can use this to enable PHP or ZEND extensions. For example:

```
extension=redis.so
extension=gd.so
zend_extension=opcache.so
```

If an extension is already present and enabled in the compiled PHP, such as `intl`, you do not need to explicitly enable it to use that extension.

PHP_EXTENSIONS vs. ZEND_EXTENSIONS

PHP has two kinds of extensions, *PHP extensions* and *Zend extensions*. These hook into the PHP executable in different ways. For more information about the way extensions work internally in the engine, see https://wiki.php.net/internals/extensions.

Because they hook into the PHP executable in different ways, they are specified differently in ini files. Apps fail if a Zend extension is specified as a PHP extension, or a PHP extension is specified as a Zend extension.

If you see the following error:

```
php-fpm | [warn-ioncube] The example Loader is a Zend-Engine extension and not a modul
e (pid 40)
php-fpm | [warn-ioncube] Please specify the Loader using 'zend_extension' in php.ini
(pid 40)
php-fpm | NOTICE: PHP message: PHP Fatal error:  Unable to start example Loader module
in Unknown on line 0
```

Then move the `example` extension from `extension` to `zend_extension` and re-push your app.

If you see the following error:

```
NOTICE: PHP message: PHP Warning: example MUST be loaded as a Zend extension in Unknow
n on line 0
```

Then move the `example` extension from `zend_extension` to `extension` and re-push your app.

Buildpack extensions

The buildpack comes with extensions for its default behavior. These are the HTTPD, Nginx, PHP, and NewRelic extensions.

The buildpack is designed with an extension mechanism, allowing app developers to add behavior to the buildpack without modifying the buildpack code.

When you push an app, the buildpack runs any extensions found in the `.extensions` directory of your app.

For more information about writing extension, see the Cloud Foundry PHP Buildpack repository on GitHub.

Composer

Composer is activated when you supply a `composer.json` or `composer.lock` file. A `composer.lock` is not required, but is strongly recommended for consistent deployments.

You can require dependencies for packages and extensions. Extensions must be prefixed with the standard `ext-`. If you reference an extension that is available to the buildpack, it automatically is installed. See the main README for a list of supported extensions.

The buildpack uses the version of PHP specified in your `composer.json` or `composer.lock` file. Composer settings override the version set in the `options.json` file.

The PHP buildpack supports a subset of the version formats supported by Composer. The buildpack supported formats are:

Example	Expected Version
5.3.*	latest 5.4.x release (5.3 is not supported)
>=5.3	latest 5.4.x release (5.3 is not supported)
5.4.*	latest 5.4.x release
>=5.4	latest 5.4.x release
5.5.*	latest 5.5.x release
>=5.5	latest 5.5.x release
5.4.x	specific 5.4.x release that is listed
5.5.x	specific 5.5.x release that is listed

Configuration

The buildpack runs with a set of default values for Composer. You can adjust these values by adding a `.bp-config/options.json` file to your application and setting any of the following values in it.

Variable	Explanation
COMPOSER_VERSION	The version of Composer to use. It defaults to the latest bundled with the buildpack.
COMPOSER_INSTALL_OPTIONS	A list of options that should be passed to `composer install`. This defaults to `["--no-interaction", "--no-dev", "--no-progress"]`. The `--no-progress` option must be used due to the way the buildpack calls Composer.
COMPOSER_VENDOR_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to create the `vendor` directory. Defaults to `{BUILD_DIR}/{LIBDIR}/vendor`.
COMPOSER_BIN_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place executables from packages. Defaults to `{BUILD_DIR}/php/bin`.
COMPOSER_CACHE_DIR	Allows you to override the default value used by the buildpack. This is passed through to Composer and instructs it where to place its cache files. Generally you should not change this value. The default is `{CACHE_DIR}/composer` which is a subdirectory of the cache folder passed in to the buildpack. Composer cache files are restored on subsequent application pushes.

By default, the PHP buildpack uses the `composer.json` and `composer.lock` files that reside inside the root directory, or in the directory specified as `WEBDIR` in your `options.json`. If you have composer files inside your app, but not in the default directories, use a `COMPOSER_PATH` environment variable for your app to specify this custom location, relative to the app root directory. Note, that the `composer.json` and `composer.lock` files must be in the same directory.

GitHub API request limits

Composer uses GitHub’s API to retrieve zip files for installation into the application folder. If you do not vendor dependencies before pushing an app, Composer can fetch dependencies during staging using the GitHub API.

GitHub’s API is request-limited. If you reach your daily allowance of API requests (typically 60), GitHub’s API returns a `403` error and staging fails.

There are two ways to avoid the request limit:

- Vendor dependencies before pushing your application.
- Supply a GitHub OAuth API token.

Vendor dependencies

For vendor dependencies, you must run `composer install` before you push your application. You might also need to configure `COMPOSER_VENDOR_DIR` to “vendor”.

Supply a GitHub token

Composer can use GitHub API OAuth tokens, which increase your request limit, typically to 5000 per day.

During staging, the buildpack looks for this token in the environment variable `COMPOSER_GITHUB_OAUTH_TOKEN`. If you supply a valid token, Composer uses it. This mechanism does not work if the token is invalid.

To supply the token, use one of the following methods:

- Run:

  ```
  cf set-env YOUR_APP_NAME COMPOSER_GITHUB_OAUTH_TOKEN "OAUTH_TOKEN_VALUE"
  ```

- Add the token to the `env` block of your application manifest.

Buildpack staging environment

Composer runs in the buildpack staging environment. Variables set with `cf set-env` or with a manifest.yml ‘env’ block are visible to Composer.

For example:

```
$ cf push a_symfony_app --no-start
$ cf set-env a_symfony_app SYMFONY_ENV "prod"
$ cf start a_symfony_app
```

In this example, `a_symfony_app` is supplied with an environment variable, `SYMFONY_ENV`, which is visible to Composer and any scripts started by Composer.

Non-configurable environment variables

User-assigned environment variables are applied to staging and runtime. Unfortunately, `LD_LIBRARY_PATH` and `PHPRC` must be different for staging and runtime. The buildpack takes care of setting these variables, which means user values for these variables are ignored.

App session data

When your application has one instance, it's generally safe to use the default session storage, which is the local file system. You can only see problems if your single instance crashes, the local file system goes away, and you lose your sessions. For many applications, this works just fine, but you need to consider how this affects your application.

If you have multiple application instances or you need a more robust solution for your application, use Redis or Memcached as a backup store for your session data. The buildpack supports both backups and when one is bound to your application, it detects it and configures PHP to use it for session storage.

By default, there is no configuration necessary. To create a Redis or Memcached service, ensure that the service name contains `redis-sessions` or `memcached-sessions` and then bind the service to the application.

Example:

```
$ cf create-service redis some-plan app-redis-sessions
$ cf bind-service app app-redis-sessions
$ cf restage app
```

If you want to use a specific service instance or change the search key, you can set either `REDIS_SESSION_STORE_SERVICE_NAME` or `MEMCACHED_SESSION_STORE_SERVICE_NAME` in `.bp-config/options.json` to the new search key. The session configuration extension searches the bound services by name for the new session key.

Configuration changes

When detected, the following changes are made:

Redis

- the `redis` PHP extension is installed, which provides the session save handler
- `session.name` is set to `PHPSESSIONID` which deactivates sticky sessions
- `session.save_handler` is configured to `redis`
- `session.save_path` is configured based on the bound credentials, for example `tcp://host:port?auth=pass`

Memcached

- the `memcached` PHP extension is installed, which provides the session save handler
- `session.name` is set to `PHPSESSIONID` which deactivates sticky sessions

- `session.save_handler` is configured to `memcached`
- `session.save_path` is configured based on the bound credentials (i.e. `PERSISTENT=app_sessions host:port`)
- `memcached.sess_binary` is set to `On`
- `memcached.use_sasl` is set to `On`, which enables authentication
- `memcached.sess_sasl_username` and `memcached.sess_sasl_password` are set with the service credentials

Configuring New Relic forthe PHP buildpack

New Relic collects analytics about your application and client side performance.

Configuration

You can configure New Relic for the PHP buildpack in one of two ways:

- License key
- Cloud Foundry service

With a license key

Use this method if you already have a New Relic account. Use these steps:

1. In a web browser. go to the New Relic website to find your license key.
2. Set the value of the environment variable `NEWRELIC_LICENSE` to your New Relic license key, either through the manifest.yml file or by running the `cf set-env` command.

For more information, see https://github.com/cloudfoundry/php-buildpack#supported-software

With a Cloud Foundry service

To configure New Relic for the PHP buildpack with a Cloud Foundry service, bind a New Relic service to the app. The buildpack automatically detects and configures New Relic.

Your `VCAP_SERVICES` environment variable must contain a service named `newrelic`, the `newrelic` service must contain a key named `credentials`, and the `credentials` key must contain named `licenseKey`.

Important

You cannot configure New Relic for the PHP buildpack with user provided services.

Python buildpack

You can push your Python app to Cloud Foundry and configure your Python app to use the Python buildpack.

Push an app

Cloud Foundry automatically uses this buildpack if it detects a `requirements.txt` or `setup.py` file in the root directory of your project.

If your Cloud Foundry deployment does not have the Python Buildpack installed, or the installed version is out of date, you can use the latest version by specifying it with the `-b` option when you push your app. For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/python-buildpack.git
```

Supported versions

You can find the list of supported Python versions in the Python buildpack release notes.

Specify a Python version

You can specify a version of the Python runtime by including it within a `runtime.txt` file. For example:

```
$ cat runtime.txt
python-3.5.2
```

The buildpack only supports the stable Python versions, which are listed in the `manifest.yml` and Python buildpack release notes.

To request the latest Python version in a patch line, replace the patch version with `x`: `3.6.x`. To request the latest version in a minor line, replace the minor version: `3.x`.

If you try to use a binary that is not currently supported, staging your app fails and you see the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:
...
 !
 !     exit
 !
Staging failed: Buildpack compilation step failed
```

Specify a PIP version

The Python buildpack supports dependency installation using PIP when a `requirements.txt` file is included at the top level of your app's directory. By default, the PIP module built into Python is used when staging your app. The version of PIP used in this case depends on which version of Python is being used.

To use the latest version of PIP, set the `BP_PIP_VERSION` environment variable to `latest` before staging your app by doing either of the following:

- Use `cf set-env`. See set-env in the Cloud Foundry CLI Reference Guide.

- Set the PIP version in the app manifest. See *Environment Variables* in App Manifest Attribute Reference.

Currently, only `latest` is supported when setting `BP_PIP_VERSION`. The buildpack does not provide multiple versions of its independent PIP dependency. You can inspect the buildpack's releases to determine which version of PIP is currently provided. See Python buildpack releases on GitHub.

Specify a start command

The Python buildpack does not generate a default start command for your apps.

To stage with the Python buildpack and start an app, do one of the following:

- Supply a Procfile. For more information about Procfiles, see the Configuring a Production Server topic. The following example Procfile specifies `gunicorn` as the start command for a web app running on Gunicorn:

  ```
  web: gunicorn SERVER-NAME:APP-NAME
  ```

- Specify a start command with `-c`. The following example specifies `waitress-serve` as the start command for a web app running on Waitress:

  ```
  $ cf push python-app -c "waitress-serve --port=$PORT DJANGO-WEB-APP.wsg
  i:MY-APP"
  ```

- Specify a start command in the application manifest by setting the `command` attribute. For more information, see the Deploying with App Manifests topic.

Run the web server

The Python buildpack expects the Python app to listen to port 8080. You can use the exposed `PORT` variable to start the web server on the port that Cloud Foundry expects on all network interfaces.

For example, you can do the following using Flask:

```
if __name__ == "__main__":
  port = int(os.getenv("PORT", 8080))
  app.run(host='0.0.0.0', port=port)
```

The `PORT` variable is not visible in the GUI or the `cf env MY-APP` command.

Vendor app dependencies

If you are deploying in an environment that is disconnected from the Internet, your application must vendor its dependencies. For more information, see Disconnected environments in the `cloudfoundry/buildpack-packager` GitHub repository.

For the Python buildpack, use `pip`:

```
$ cd YOUR-APP-DIR
$ mkdir -p vendor
```

```
# vendors pip *.whl into vendor/
$ pip download -r requirements.txt --no-binary=:none: -d vendor
```

`cf push` uploads your vendored dependencies. The buildpack installs them directly from the `vendor/` directory.

To ensure proper installation of dependencies, VMware recommends binary vendored dependencies (wheels). The preceding `pip install` command achieves this.

Private dependency repository

To deploy apps in an environment that needs to use a private dependency repository, you have the following options:

- PIP
- Conda

PIP

To install dependencies using PIP, add the URL of the repository to the `requirements.txt` file in the following format:

```
--index-url=https://example.com/api/pypi/ext_pypi/simple
fixtures==2.0.0
```

If the private repository uses a custom SSL certificate that is installed on the platform, you may see an error similar to the following:

```
Could not fetch URL https://example.com/api/pypi/ext_pypi/simple/fixtures/:
There was a problem confirming the ssl certificate:
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:777) - ski
pping
```

This error occurs because `pip` does not use system certificates by default. To resolve this issue, set the `PIP_CERT` environment variable in the `manifest.yml` file to point to the system certificate store.

For example:

```
---
env:
  PIP_CERT: /etc/ssl/certs/ca-certificates.crt
```

Conda

To install dependencies using Conda, add a `channels` block to the `environment.yml` file.

In the `channels` block, list custom channels and add `nodefaults`. Specifying `nodefaults` tells Conda to only use the channels in the channels block.

For example:

```
channels:
  - https://conda.example.com/repo
  - nodefaults
```

If the private repository uses a custom SSL certificate that is installed on the platform, you may see an error similar to the following:

```
Error: Connection error: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify
failed (_ssl.c:581):
```

This error occurs because `conda` does not use system certificates by default. To resolve this issue, set the `CONDA_SSL_VERIFY` environment variable in the `manifest.yml` file to point to the system certificate store.

For example:

```
---
env:
  CONDA_SSL_VERIFY: /etc/ssl/certs/ca-certificates.crt
```

Parse environment variables

The `cfenv` package provides access to Cloud Foundry application environment settings by parsing all the relevant environment variables. The settings are returned as a class instance. See https://github.com/jmcarp/py-cfenv for more information.

Miniconda support (starting in buildpack version 1.5.6)

To use miniconda instead of pip for installing dependencies, place an `environment.yml` file in the root directory.

Pipenv support (starting in buildpack version 1.5.19)

To use Pipenv instead of pip (directly) for installing dependencies, place a `Pipfile` in the root directory. Easiest to let Pipenv generate this for you.

NLTK support

To use NLTK corpora in your app, you can include an `nltk.txt` file in the root of your application. Each line in the file specifies one dataset to download. The full list of data sets available this way can be found on the NLTK website. The `id` listed for the corpora on that page is the string you must include in your app's `nltk.txt`.

Example `nltk.txt`:

```
brown
wordnet
```

Having an `nltk.txt` file only causes the buildpack to download the corpora. You still must specify NLTK as a dependency of your app if you want to use it to process the corpora files.

Proxy support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH configured custom trusted certificate support

Versions of Python 2.7.9 and later use certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

To configure your Python applications to make HTTP requests with this custom certificate, set the environment variable `REQUESTS_CA_BUNDLE="/etc/ssl/certs/ca-certificates.crt"`.

Help and support

Join the #buildpacks channel in our Slack community if you need any further assistance.

For more information about using and extending the Python buildpack in Cloud Foundry, see Python buildpack.

You can find current information about this buildpack on the Python buildpack release page in GitHub.

R buildpack

You can push your R app to Cloud Foundry and configure your R app to use the R buildpack.

Cloud Foundry automatically uses the R buildpack if it detects a `r.yml` file in the root directory of your project.

If your Cloud Foundry deployment does not have the R buildpack installed, or the installed version is out of date, you can use the latest version by specifying it with the `-b` option when you push your app.

For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/r-buildpack.git
```

Supported versions

You can find the list of supported R versions in the R buildpack release notes.

Start command

The R buildpack does not generate a default start command for your applications. Instead, you must specify a start command for your app.

To stage an app with the R buildpack and start the app, do one of the following:

- **Option 1:** Supply a Procfile. For more information about Procfiles, see Production Server Configuration. The following example Procfile specifies `R -f` as the start command for a web app with the entrypoint `simple.r`:

```
web: R -f simple.r
```

- **Option 2:** Specify a start command with `-c`:

  ```
  $ cf push r-app -c "R -f simple.r"
  ```

- **Option 3:** Specify a start command in the application manifest by setting the `command` attribute. For more information, see the Deploying with Application Manifests topic.

For more information about starting apps, see Starting, Restarting, and Restaging Applications.

Specifying app dependencies

As of v1.1.21, the following packages are provided by the R buildpack:

- Rserve
- forecast
- plumber
- shiny

To specify additional packages needed by your app, provide the CRAN mirror and names of the packages inside your `r.yml` file. You can also specify the number of threads to use for parallel installation. For example:

```
---
packages:
  - cran_mirror: https://cran.r-project.org
    num_threads: 2
    packages:
      - name: stringr
      - name: jsonlite
```

Vendoring app dependencies

If you are deploying in an environment that is disconnected from the Internet, you must *vendor* your app’s dependencies, which means you must make these packages available for offline use. You can vendor dependencies by using a package manager.

To set up your own custom, local CRAN-like repository to vendor your packages, create the `src/contrib` directories and populate them with your package source tarballs. For more information, see How to Set Up a Custom CRAN-like Repository published on the *Packrat Documentation* site.

Add the `src/contrib` directories containing your package tarballs to a `vendor_r` directory at the root of your app. This directory is named `vendor_r` so as not to conflict with vendor directories of other languages, such as python.

Then, inside your `r.yml`, provide the names of your vendored packages in the `packages` list:

```
---
packages:
```

```
  - packages:
      - name: stringr
      - name: jsonlite
```

`cf push` uploads your vendored dependencies. The buildpack installs them directly from the `vendor_r/` directory.

Example app directory tree:

```
├── r.yml
├── simple-app.r
└── vendor_r
    └── src
        └── contrib
            ├── PACKAGES
            ├── PACKAGES.gz
            ├── jsonlite_1.5.tar.gz
            └── stringr_1.3.1.tar.gz
```

For more information about using buildpacks in disconnected environments, see Disconnected environments.

Proxy support

If you need to use a proxy to download dependencies during staging, you can set the `http_proxy` and `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH configured custom trusted certificate support

R uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container. For more information, see Configuring Trusted System Certificates for Applications.

Help and support

Join the **#buildpacks** channel in our Slack community if you need any further assistance.

For more information about using and extending the R buildpack in Cloud Foundry, see the R-buildpack GitHub repository.

You can also find current information about this buildpack on the R buildpack release page in GitHub.

Ruby

In this section:

- Ruby Buildpack
- Tips for Ruby Developers
- **Getting Started Deploying Ruby Apps**
  - Getting Started Deploying Ruby Apps

    - Getting Started Deploying Ruby on Rails Apps
- Configuring Rake Tasks for Deployed Apps
- Environment Variables Defined by the Ruby Buildpack
- Configuring Service Connections for Ruby
- Support for Windows Gemfiles

Ruby buildpack

Cloud Foundry uses the Ruby buildpack if your app has a `Gemfile` and `Gemfile.lock` in the root directory. The Ruby buildpack uses Bundler to install your dependencies.

If your Cloud Foundry deployment does not have the Ruby buildpack installed or the installed version is out of date, push your app with the `-b` option to specify the buildpack:

```
$ cf push MY-APP -b https://github.com/cloudfoundry/ruby-buildpack.git
```

For more detailed information about deploying Ruby applications see the following topics:

- Tips for Ruby Developers
- Getting Started Deploying Apps
- Configuring Rake Tasks for Deployed Apps
- Environment Variables Defined by the Ruby Buildpack
- Configuring Service Connections for Ruby
- Support for Windows Gemfiles

See the source for the Ruby buildpack in the Ruby buildpack repository on GitHub.

Supported versions

See the supported Ruby versions in the release notes for the Ruby buildpack on GitHub.

Specify a Ruby version

You can specify specific versions of the Ruby runtime in the `Gemfile` for your app as described in the following sections.

If you don't specify a version of the Ruby runtime, the default Ruby version listed in the [buildpack manifest.yml file](https://github.com/cloudfoundry/ruby-buildpack/blob/950b2337ff8a7ac0a68f7420f48ddc4ae4764977/manifest.yml#L3-L5) is used.

MRI

For MRI, specify the version of Ruby in your `Gemfile` as follows:

```
ruby '~> 2.2.3'
```

With this example declaration in the `Gemfile`, if Ruby versions `2.2.4`, `2.2.5`, and `2.3.0` are present in the Ruby buildpack, the app uses Ruby version `2.2.5`.

For more information about the `ruby` directive for Bundler Gemfiles, see the Bundler documentation.

If you use v1.6.18 or earlier, you must specify an exact version, such as `ruby '2.2.3'`. In Ruby Buildpack v1.6.18 and earlier, Rubygems do not support version operators for the `ruby` directive.

JRuby

For JRuby, specify the version of Ruby in your `Gemfile` based on the version of JRuby your app uses.

JRuby version `1.7.x` supports either `1.9` mode or `2.0` mode.

- For `1.9` mode, use:

  ```
  ruby '1.9.3', :engine => 'jruby', :engine_version => '1.7.25'
  ```

- For `2.0` mode, use:

  ```
  ruby '2.0.0', :engine => 'jruby', :engine_version => '1.7.25'
  ```

For Jruby version `>= 9.0`, use:

```
ruby '2.2.3', :engine => 'jruby', :engine_version => '9.0.5.0'
```

The Ruby buildpack only supports the stable Ruby versions listed in the `manifest.yml` and release notes for the Ruby buildpack on GitHub.

If you try to use a binary that is not supported, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST: ...
 !
 !     exit
 !
Staging failed: Buildpack compilation step failed
```

The Ruby buildpack does not support the pessimistic version operator `~>` on the Gemfile `ruby` directive for JRuby.

Vendor app dependencies

As stated in the Disconnected Environments documentation, your application must ‘vendor’ its dependencies.

For the Ruby buildpack, use the `bundle package --all` command in Bundler to vendor the dependencies.

Example:

```
$ cd my-app-directory
```

```
bundle package --all
```

The `cf push` command uploads your vendored dependencies. The Ruby buildpack compiles any dependencies requiring compilation while staging your app.

Buildpack logging and application logging

The Ruby buildpack only runs during staging, and only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

The buildpack stops logging when the staging is complete. The Loggregator handles application logging.

Your application must write to stdout or stderr for its logs to be included in the Loggregator stream. For more information, see Application Logging in Cloud Foundry.

If you are deploying a Rails application, the buildpack might not automatically install the necessary plug-in or gem for logging, depending on the Rails version of the application:

- Rails v2.x: The buildpack automatically installs the `rails_log_stdout` plugin into the application. For more information about the `rails_log_stdout` plugin, see the GitHub README.
- Rails v3.x: The buildpack automatically installs the `rails_12factor` gem if it is not present and issues a warning message. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, see the GitHub README.
- Rails v4.x: The buildpack issues a warning message that the `rails_12factor` gem is not present, but does not install the gem. You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message. For more information about the `rails_12factor` gem, see the GitHub README.

For more information about the `rails_12factor` gem, see GitHub README.

Proxy support

If you need to use a proxy to download dependencies during staging, set the `http_proxy` and `https_proxy` environment variables. For more information, see Using a Proxy.

BOSH configured custom trusted certificate support

Ruby uses certificates stored in `/etc/ssl/certs`. Your platform operator can configure the platform to add the custom certificates into the application container.

Help and support

Join the #buildpacks channel in our Slack community if you need help.

For more information about using and extending the Ruby buildpack in Cloud Foundry, see the Ruby buildpack repository on GitHub.

See the current information about the Ruby buildpack in the release notes for the Ruby buildpack on GitHub.

About Ruby buildpacks in Cloud Foundry

You can deploy Rack, Rails, or Sinatra apps with Cloud Foundry.

App bundling

You must run Bundler to create a `Gemfile` and a `Gemfile.lock`. These files must be in your app before you push to Cloud Foundry.

To download Bundler, see Gem Bundler.

Rack config file

For Rack and Sinatra, you must have a `config.ru` file. For example:

```
require './hello_world'
run HelloWorld.new
```

Asset pre compilation

Cloud Foundry supports the Rails asset pipeline. If you do not precompile assets before deploying your app, Cloud Foundry precompiles them when staging the app. Precompiling before deploying reduces the time it takes to stage an app.

To precompile assets before deployment, run:

```
rake assets:precompile
```

The Rake precompile task re-initializes the Rails app. This could pose a problem if initialization requires service connections or environment checks that are unavailable during staging. To prevent re-initialization during pre-compilation, add the following line to `application.rb`:

```
config.assets.initialize_on_precompile = false
```

If the `assets:precompile` task fails, Cloud Foundry uses live compilation mode, the alternative to asset precompilation. In this mode, assets are compiled when they are loaded for the first time. You can force live compilation by adding the following line to `application.rb`.

```
Rails.application.config.assets.compile = true
```

Running Rake tasks

Cloud Foundry does not provide a mechanism for running a Rake task on a deployed app. If you need to run a Rake task that must be performed in the Cloud Foundry environment, rather than locally before deploying or redeploying, you can configure the command that Cloud Foundry uses to start the app to invoke the Rake task.

An app start command is configured in the app manifest file, `manifest.yml`, using the `command` attribute.

For more information about app manifests and supported attributes, see Deploying with App Manifests.

Example: Invoking a Rake database migration task at App startup

See the following example of migrating a database schema using a Rake task.

For more information about migrating database schemas, see the Migrate a Database Schema section of the *Services Overview* topic.

To migrate a database schema using a Rake task:

1. If a Rakefile does not exist, create one and add it to your app directory.
2. In your Rakefile, add a Rake task to limit an idempotent command to the first instance of a deployed app:

    ```
    namespace :cf do
      desc "Only run on the first application instance"
      task :on_first_instance do
        instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"] resc
    ue nil
        exit(0) unless instance_index == 0
      end
    end
    ```

3. Add the task to the `manifest.yml` file, referencing the idempotent command `rake db:migrate` with the `command` attribute:

    ```
    ---
    applications:
    - name: my-rails-app
      command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rail
    s s -p $PORT -e $RAILS_ENV
    ```

4. Update the app by running:

    ```
    cf push
    ```

Rails 3 worker tasks

Learn how to create and deploy an example Rails app that uses a worker library to defer a task that a separate app runs. It also describes how to scale the resources available to the worker app.

Most worker tasks do not serve external requests. Use the `--no-route` flag with the `cf push` command, or `no-route: true` in the app manifest, to suppress route creation. If you are using cf CLI v7 or later, be aware that the `--no-route` flag no longer unbinds all existing routes associated with the app.

Choose a worker task library

You must choose a worker task library.

The following table summarizes the three main libraries available for Ruby/Rails:

Library	Description
Delayed::Job	A direct extraction from Shopify where the job table is responsible for a multitude of core tasks.
Resque	A Redis-backed library for creating background jobs, placing those jobs on multiple queues, and processing them later.
Sidekiq	Uses threads to handle many messages at the same time in the same process. It does not require Rails, but integrates tightly with Rails 3 to simplify background message processing. This library is Redis-backed and semi-compatible with Resque messaging.

For other alternatives, see Background Jobs on The Ruby Toolbox.

Creating an example app

The example app described in this section uses Sidekiq.

To create an example app:

1. Create a Rails app with an arbitrary model named “Things” by running:

   ```
   rails create rails-sidekiq
   cd rails-sidekiq
   rails g model Thing title:string description:string
   ```

2. Add `sidekiq` and `uuidtools` to the Gemfile:

   ```
   source 'https://rubygems.org'

   gem 'rails', '3.2.9'
   gem 'mysql2'

   group :assets do
     gem 'sass-rails',   '~> 3.2.3'
     gem 'coffee-rails', '~> 3.2.1'
     gem 'uglifier', '>= 1.0.3'
   end

   gem 'jquery-rails'
   gem 'sidekiq'
   gem 'uuidtools'
   ```

3. Install the bundle by running:

   ```
   bundle install
   ```

4. In `app/workers`, create a worker for Sidekiq to carry out its tasks by running:

   ```
   touch app/workers/thing_worker.rb
   ```

   Create the worker as follows:

```
class ThingWorker

  include Sidekiq::Worker

  def perform(count)

    count.times do

      thing_uuid = UUIDTools::UUID.random_create.to_s
      Thing.create :title =>"New Thing (#{thing_uuid})", :description =>
"Description for thing #{thing_uuid}"
    end

  end

end
```

This worker create `n` number of things, where `n` is the value passed to the worker.

5. Create a controller for “Things” by running:

```
rails g controller Thing
```

Create the controller as follows:

```
class ThingController < ApplicationController

  def new
    ThingWorker.perform_async(2)
    redirect_to '/thing'
  end

  def index
    @things = Thing.all
  end

end
```

6. Add a view to inspect our collection of “Things” by running:

```
mkdir app/views/things
touch app/views/things/index.html.erb
```

Create the view as follows:

```
<%= @things.inspect %>
```

Deploying the app

You must deploy your example app twice for it to work, once as a Rails web app and once as a standalone Ruby app. One way to do this is to keep separate Cloud Foundry manifests for each app type.

To create separate manifests for each app type:

1. Create a web manifest and save it as `web-manifest.yml`:

```
---
applications:
- name: sidekiq
  memory: 256M
  instances: 1
  host: sidekiq
  domain: ${target-base}
  path: .
  services:
  - sidekiq-mysql:
  - sidekiq-redis:
```

2. Create a worker manifest and save it as `worker-manifest.yml`:

   ```
   ---
   applications:
   - name: sidekiq-worker
     memory: 256M
     instances: 1
     path: .
     command: bundle exec sidekiq
     no-route: true
     services:
     - sidekiq-redis:
     - sidekiq-mysql:
   ```

3. Since the URL `sidekiq.cloudfoundry.com` is likely already taken, change it in `web-manifest.yml` first, then push the app with both manifest files by running:

   ```
   cf push -f web-manifest.yml
   cf push -f worker-manifest.yml
   ```

   If the cf CLI asks for a URL for the worker app, select **none**.

Testing the app

To test the app:

1. Visit the new action on the “Thing” controller at the assigned URL. In this example, the URL would be `http://sidekiq.cloudfoundry.com/thing/new`.

This creates a new Sidekiq job which is queued in Redis, then picked up by the worker app. The browser is then redirected to `/thing`, which shows the collection of “Things”.

Scale workers

To change the number of Sidekiq workers:

1. Run:

   ```
   cf scale sidekiq-worker -i WORKER-NUMBER
   ```

   Where `WORKER-NUMBER` is the number of Sidekiq workers you want.

Use rails_serve_static_assets on Rails 4

By default, Rails 4 returns a 404 error if an asset is not handled via an external proxy such as Nginx. The `rails_serve_static_assets` gem enables your Rails server to deliver static assets directly, instead of returning a 404 error.

You can use this capability to populate an edge cache CDN or serve files directly from your web app. The gem enables this behavior by setting the `config.serve_static_assets` option to `true`, so you do not need to configure it manually.

Add custom libraries

If your app requires external shared libraries that are not provided by the rootfs or the buildpack, you must place the libraries in an `ld_library_path` directory at the app root.

You must keep these libraries up-to-date. They do not update automatically.

The Ruby buildpack automatically adds the directory `<app-root>/ld_library_path` to `LD_LIBRARY_PATH` so that your app can access these libraries at runtime.

Environment variables

You can access environments variable programmatically. For example, you can obtain `VCAP_SERVICES` by running:

```
ENV['VCAP_SERVICES']
```

Environment variables available to you include both those defined by the system and those defined by the Ruby buildpack, as described in the sections below. For more information about system environment variables, see the App-Specific System Variables section of the *TAS for VMs Environment Variables* topic.

BUNDLE_BIN_PATH

This variable specifies the location where Bundler installs binaries.

For example: `BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`

BUNDLE_GEMFILE

This variable specifies the path to the app Gemfile.

For example: `BUNDLE_GEMFILE:/home/vcap/app/Gemfile`

BUNDLE_WITHOUT

The `BUNDLE_WITHOUT` environment variable instructs Cloud Foundry to skip gem installation in excluded groups.

Use this with Rails apps, where “assets” and “development” gem groups typically contain gems that are not needed when the app runs in production.

For example: `BUNDLE_WITHOUT=assets`

DATABASE_URL

Cloud Foundry examines the `database_uri` for bound services to see if they match known database types. If known relational database services are bound to the app, the `DATABASE_URL` environment variable is set using the first match in the list.

If your app depends on `DATABASE_URL` to be set to the connection string for your service and Cloud Foundry does not set it, use the `cf set-env` command to can set this variable manually.

For example:

```
cf set-env my-app-name DATABASE_URL mysql://example-database-connection-string
```

GEM_HOME

This variable specifies the location where gems are installed.

For example: `GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1`

GEM_PATH

This variable specifies the location where gems can be found.

For example: `GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:`

RACK_ENV

This variable specifies the Rack deployment environment. Valid values are `development`, `deployment`, and `none`. This governs which middleware is loaded to run the app.

For example: `RACK_ENV=development`

RAILS_ENV

This variable specifies the Rails deployment environment. Valid values are `development`, `test`, and `production`. This controls which of the environment-specific configuration files governs how the app is run.

For example: `RAILS_ENV=production`

RUBYOPT

This Ruby environment variable defines command-line options passed to Ruby interpreter.

For example: `RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`

Getting Started Deploying Ruby Apps

In this section:

- Getting Started Deploying Ruby Apps
- Getting Started Deploying Ruby on Rails Apps

Deploying Ruby apps

Learn how to deploy a Ruby app to Cloud Foundry. If you experience a problem following the steps, check the Known Issues topic, or refer to the Troubleshooting application deployment and health topic.

If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_ruby.git` to clone the `pong_matcher_ruby` app from GitHub, and follow the instructions in the Sample app step sections.

Ensure that your Ruby app runs locally before continuing with this procedure.

Deploy a Ruby app

You can deploy a Ruby application to Cloud Foundry, and use the output from a sample app to show specific steps of the deployment process.

Prerequisites

- A Ruby 2.x application that runs locally on your workstation
- Bundler configured on your workstation
- Basic to intermediate Ruby knowledge
- The Cloud Foundry Command Line Interface (cf CLI) installed on your workstation

Step 1: Create and bind a service instance for a Ruby app

Use the cf CLI to configure a Redis Cloud managed service instance for an app. You can use either the CLI or Apps Manager to perform this task. For more information about using Apps Manager, see About Apps Manager.

Cloud Foundry supports the following types of service instances:

- Managed services integrate with Cloud Foundry through service brokers that offer services and plans and manage the service calls between Cloud Foundry and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Creating a service instance

To create a service instance:

1. View managed and user provided services and plans that are available to you by running:

   ```
   cf marketplace
   ```

   The example shows three of the available managed database-as-a-service providers and the plans that they offer: `cleardb` MySQL and `postgresql-10-odb` PostgreSQL as a Service.

```
$  cf marketplace
Getting services from marketplace in org Cloud-Apps / space development
as clouduser@example.com...
OK

service             plans                                     descripti
on
...
cleardb             spark, boost, amp, shock                  Highly av
ailable MySQL for your Apps
...
postgresql-10-odb   standalone, standalone-replica, general   PostgreSQ
L as a Service
...
```

2. Create a service instance for your app.

```
cf create-service SERVICE PLAN SERVICE-INSTANCE
```

Choose a `SERVICE` and `PLAN` from the list, and provide a unique name for the `SERVICE-INSTANCE`.

Sample app step
Run `cf create-service rediscloud 30mb redis`. This creates a service instance named `redis` that uses the `rediscloud` service and the `30mb` plan, as the example below shows.

```
$ cf create-service rediscloud 30mb redis
Creating service redis in org Cloud-Apps / space development as cloudus
er@example.com....
OK
```

Bind a service instance

When you bind an app to a service instance, Cloud Foundry writes information about the service instance to the `VCAP_SERVICES` app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application by running:

```
cf bind-service APPLICATION SERVICE-INSTANCE
```

Alternately, you can configure the deployment manifest file by adding a `services` block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample app step.

You can also bind a service using Apps Manager. For more information about using Apps Manager, see Adding and Binding services using Apps Manager.

Sample app step

You can skip this step. The `manifest.yml` for the sample app contains a `services` sub-block in the `applications` block, as the example below shows. This binds the `redis` service instance that you created in the previous step.

```
services:
      - redis
```

Step 2: Configure deployment options

Configure the deployment manifest file

You can specify app deployment options in a manifest that the `cf push` command uses. For more information about application manifests and supported attributes, refer to the Deploying with Application Manifests topic.

Configure a production server

Cloud Foundry uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, Cloud Foundry can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the Configuring a Production Server topic for help configuring a server other than WEBrick.

Sample app step

You can skip this step. The `manifest.yml` file for `pong_matcher_ruby` does not require any additional configuration to deploy the app.

Step 3: Log in and target the API endpoint

Enter your login credentials, and select a space and org.

```
cf login -a API-ENDPOINT
```

The API endpoint is the URL of the Cloud Controller in your TAS for VMs instance.

Sample app step

You must do this step to run the sample app.

Step 4: Deploy an app

You must use the cf CLI to deploy apps.

Deploy your application by running the following command from the root directory of your application:

```
cf push APP-NAME
```

This command creates a URL route to your application in the form `HOST.DOMAIN`, where `HOST` is your `APP-NAME` and `DOMAIN` is specified by your administrator. Your `DOMAIN` is`shared-domain.example.com`.

For example, `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that Cloud Foundry hosts or the push fails. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words.
- `cf help push` to view other options for this command.

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Run `cf push pong_matcher_ruby -n HOST_NAME`.

Example: `cf push pong_matcher_ruby -n pongmatch-ex12`

The following example shows the terminal output of deploying the `pong_matcher_ruby` app. The n`cf push` command uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `redis` service and follows the instructions in the manifest to start one instance of the app with 256M. After the app starts, the output displays the health and status of the app.

These examples work for cf CLI v6. The `-n` flag is not supported for cf CLI v7/v8. Hostname must be set using the `routes` property in the manifest.

The `pong_matcher_ruby` app does not include a web interface. To interact with the `pong_matcher_ruby` app, see the interaction instructions on GitHub: https://github.com/cloudfoundry-samples/pong_matcher_ruby.

```
$ cf push pong_matcher_ruby -n pongmatch-ex12
Using manifest file /Users/clouduser/workspace/pong_matcher_ruby/manifest.yml

Creating app pong_matcher_ruby in org Cloud-Apps / space development as cloud
user@example.com...
OK

Creating route pongmatch-ex12.shared-domain.example.com
        Binding pongmatch-ex12.shared-domain.example.com to pong_matcher_rub
y...
OK

Uploading pong_matcher_ruby...
Uploading app files from: /Users/clouduser/workspace/pong_matcher_ruby
Uploading 8.8K, 12 files
OK
Binding service redis to app pong_matcher_ruby in org Cloud-Apps / space deve
lopment as clouduser@example.com...
OK

Starting app pong_matcher_ruby in org Cloud-Apps / space development as cloud
user@example.com...
```

```
OK
...

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_ruby in org Cloud-Apps / space
development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: pongmatch-ex12.cfapps.io

     state     since                    cpu    memory          disk
#0   running   2014-12-09 10:04:40 AM   0.0%   35.2M of 256M   45.8M of 1G
```

Step 5: Test a deployed app

You’ve deployed an app to Cloud Foundry!

Use the cf CLI or About Apps Manager to review information and administer your app and your account. For example, you could edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. For more information about using Apps Manager, see Using Apps Manager.

Manage your application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

You cannot perform certain tasks in the CLI or About Apps Manager because these are commands that only an administrator can run. If you are not an administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with Apps Manager, depending on your user role, see Getting Started with Apps Manager.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App deploy fails

Even when deploying an app fails, the app might exist on Cloud Foundry. Run `cf apps` to review the apps in the targeted org and space. You might be able to correct the issue using the CLI or About Apps Manager, or you might have to delete the app and redeploy it.

Common reasons deploying an app fails include:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL service instance. Refer to Step 2: Create and Bind a Service Instance for a Ruby Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip **App Requires Unique URL**.

App requires unique URL

Cloud Foundry requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and Cloud Foundry cannot successfully deploy the app. You can fix this issue by running `cf push` with the `--random-route` flag to create a unique URL. Using `--random-route` to create a URL that includes the app name and random words might create a long URL, depending on the number of words that the app name includes.

Deploying Ruby on Rails apps

You can use this information to deploy a Ruby on Rails app to Cloud Foundry.

Prerequisites

In order to deploy a sample Ruby on Rails app, you must have the following:

- Cloud Foundry deployment
- Cloud Foundry Command Line Interface
- Cloud Foundry username and password with **Space Developer** permissions. See your Org Manager if you require permissions.

Step 1: Clone the app

Run the following command to create a local copy of the cf-sample-app-rails:

```
git clone https://github.com/cloudfoundry-samples/cf-sample-app-rails.git
```

The newly created directory contains a `manifest.yml` file, which assists CF with deploying the app. See Deploying with Application Manifests for more information.

Step 2: Log in and target the API endpoint

1. Run the following terminal command to log in and target the API endpoint of your deployment. For more information, see Identifying the API Endpoint for Your TAS for VMs Instance.

```
cf login -a YOUR-API-ENDPOINT
```

2. Use your credentials to log in, and to select a Space and Org.

Step 3: Create a service instance

Run the following terminal command to create a PostgreSQL service instance for the sample app.

```
cf create-service postgresql-10-odb standalone rails-postgres
```

For example:

```
$ cf create-service postgresql-10-odb standalone rails-postgres
Creating service rails-postgres in org YOUR-ORG / space development as cloudu
ser@example.com....
OK
```

The service instance is `rails-postgres`. It uses the `postgresql-10-odb` service and the `standalone` plan. For more information about the `postgresql-10-odb` service, see Crunchy PostgreSQL.

Step 4: Deploy the app

Make sure you are in the `cf-sample-app-rails` directory. Run the following command to deploy the app:

```
cf push cf-sample-app-rails
```

This command creates a URL route to your application in the form `HOST.DOMAIN`. In this example, `HOST` is `cf-sample-app-rails`. Administrators specify the `DOMAIN`.

For example, for the `DOMAIN` `shared-domain.example.com`, running the previous command creates the URL `cf-sample-app-rails.shared-domain.example.com`.

The following example shows the terminal output when deploying the `cf-sample-app-rails`. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then follows the information in the manifest to start one instance of the app with 256M of RAM. After the app starts, the output displays the health and status of the app.

```
$ cf push cf-sample-app-rails
Using manifest file ~/workspace/cf-sample-app-rails/manifest.yml

Creating app cf-sample-app-rails in org my-rog / space dev as clouduser@examp
le.com...
OK

Creating route cf-sample-app-rails.cfapps.io...
OK

Binding cf-sample-app-rails.cfapps.io to cf-sample-app-rails...
OK
```

```
Uploading cf-sample-app-rails...
Uploading app files from: ~/workspace/cf-sample-app-rails
Uploading 746.6K, 136 files
Done uploading
OK

Starting app cf-sample-app-rails in org my-org  / space dev as clouduser@exam
ple.com...
. . .
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

OK

App cf-sample-app-rails was started using this command `bundle exec rails ser
ver -p $PORT`

Showing health and status for app cf-sample-app-rails in org my-org / space d
ev as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: cf-sample-app-rails.cfapps.io
last uploaded: Fri Dec 22 18:08:32 UTC 2017
stack: cflinuxfs3
buildpack: ruby

     state     since                    cpu    memory          disk
details
#0   running   2018-8-17 10:09:57 AM   0.0%   20.7M of 512M   186.8M of 1G
```

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs cf-sample-app-rails`.

To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

Step 5: Bind the service instance

1. Run the following command to bind the service instance to the sample app. Once bound, environment variables are stored that allow the app to connect to the service after a `cf push`, `cf restage`, or `cf restart` command.

   ```
   $ cf bind-service cf-sample-app-rails rails-postgres
   Binding service rails-postgres to app cf-sample-app-rails in org my-org
   / space dev
   ```

```
OK
TIP: Use 'cf restage cf-sample-app-rails' to ensure your env variable c
hanges take effect
```

2. Run the following command to restage the sample app.

   ```
   $ cf restage cf-sample-app-rails
   ```

3. Run the following command to verify the service instance is bound to the sample app.

   ```
   $ cf services
   Getting services in org my-org / space dev
   OK
   name              service             plan          bound apps
   last operation
   rails-postgres    postgresql-10-odb   standalone    cf-sample-app-rails
   create succeeded
   ```

Step 6: Verify the app

Verify that your app is running by browsing to the URL generated in the output of the previous step. In this example, go to `cf-sample-app-rails.shared-domain.example.com` and verify that the app is running.

For more information, see the Pushing an App topic.

Test a deployed app

Use the cf CLI or About Apps Manager to review information and administer your app and your account. For example, you could edit the `manifest.yml` file to increase the number of app instances from 1 to 3 or redeploy the app with a new app name.

Manage your app with the cf CLI

Run `cf help` to view a complete list of commands and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf CLI topics, especially the Getting Started with cf CLI topic.

Troubleshooting

If your app fails to start, verify that the app starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

Configuring Rake Tasks for deployed apps

For Cloud Foundry to automatically invoke a Rake task while a Ruby or Ruby on Rails app is deployed, you must do the following:

- Include the Rake task in your app.
- Configure the application start command using the `command` attribute in the application manifest.

Use the following example to invoke a Rake database migration task at application startup:

1. Create a file with the Rake task name and the extension `.rake`, and store it in the `lib/tasks` directory of your application.
2. Add the following code to your rake file:

   ```
   namespace :cf do
       desc "Only run on the first application instance"
       task :on_first_instance do
           instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"]
   rescue nil
       exit(0) unless instance_index == 0
       end
   end
   ```

   This Rake task limits an idempotent command to the first instance of a deployed application.
3. Add the task to the `manifest.yml` file with the `command` attribute, referencing the idempotent command `rake db:migrate` chained with a start command.

   ```
   applications:
   - name: my-rails-app
     command: bundle exec rake cf:on_first_instance db:migrate && rails s -p $PORT
   ```

Environment variables in Ruby buildpack

Cloud Foundry provides configuration information to apps through environment variables. You can use additional environment variables that are provided by the Ruby buildpack.

For more information about the standard environment variables, see TAS for VMs Environment Variables.

Ruby buildpack environment variables

The following table describes the environment variables provided by the Ruby buildpack:

Environment Variable	Description
`BUNDLE_BIN_PATH`	The directory where Bundler installs binaries. For example: `BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`
`BUNDLE_GEMFILE`	The path to the Gemfile for the app. For example: `BUNDLE_GEMFILE:/home/vcap/app/Gemfile`
`BUNDLE_WITHOUT`	Instructs Cloud Foundry to skip gem installation in excluded groups. Use this with Rails apps, where “assets” and “development” gem groups typically contain gems that are not needed when the app runs in production. For example: `BUNDLE_WITHOUT=assets`
`DATABASE_URL`	Cloud Foundry examines the `database_uri` for bound services to see if they match known database types. If known relational database services are bound to the app, then the `DATABASE_URL` environment variable is set to the first services in the list. If your app requires that `DATABASE_URL` is set to the connection string for your service, and Cloud Foundry does not set it, use the Cloud Foundry Command Line Interface (cf CLI) `cf set-env` command to set this variable manually. For example: `cf set-env my-app DATABASE_URL mysql://example-database-connection-string`

Environment Variable	Description
`GEM_HOME`	The directory where gems are installed. For example: `GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1`
`GEM_PATH`	The directory where gems can be found. For example: `GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:`
`RACK_ENV`	The Rack deployment environment, which governs the middleware loaded to run the app. Valid value are `development`, `deployment`, and `none`. For example: `RACK_ENV=none`
`RAILS_ENV`	The Rails deployment environment, which controls which environment-specific configuration file governs how the app is run. Valid values are `development`, `test`, and `production`. For example: `RAILS_ENV=production`
`RUBYOPT`	Defines command-line options passed to Ruby interpreter. For example: `RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`

Configuring service connections for Ruby

After you create a service instance and bind it to an application, you must configure the application to connect to the service.

Query VCAP_SERVICES with cf-app-utils

The `cf-app-utils` gem allows your application to search for credentials from the `VCAP_SERVICES` environment variable by name, tag, or label.

- cf-app-utils-ruby

VCAP_SERVICES defines DATABASE_URL

At runtime, Cloud Foundry creates a `DATABASE_URL` environment variable for every application based on the VCAP_SERVICES environment variable.

Example `VCAP_SERVICES`:

```
VCAP_SERVICES =
{
  "postgresql-10-odb": [
    {
      "name": "postgresql-10-odb-c6c60",
      "label": "postgresql-10-odb",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.postgresql-10-odb.com:5432/exampledb"
      }
    }
  ]
}
```

Based on this `VCAP_SERVICES`, Cloud Foundry creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.postgresql-10-odb.com:5432/exa
mpledb
```

Cloud Foundry uses the structure of the `VCAP_SERVICES` environment variable to populate `DATABASE_URL`. Any service containing a JSON object with the following form is recognized by Cloud Foundry as a candidate for `DATABASE_URL`:

```
{
  "some-service": [
    {
      "credentials": {
        "uri": "<some database URL>"
      }
    }
  ]
}
```

Cloud Foundry uses the first candidate found to populate `DATABASE_URL`.

Configure non Rails apps

Non Rails applications can also access the `DATABASE_URL` variable.

If you have more than one service with credentials, only the first is populated into `DATABASE_URL`. To access other credentials, you can inspect the `VCAP_SERVICES` environment variable.

```
vcap_services = JSON.parse(ENV['VCAP_SERVICES'])
```

Use the hash key for the service to obtain the connection credentials from `VCAP_SERVICES`.

- For services that use the v2 format, the hash key is the name of the service.
- For services that use the v1 format, the hash key is formed by combining the service provider and version, in the format PROVIDER-VERSION.

  For example, the service provider “p-mysql” with version “n/a” forms the hash key `p-mysql-n/a`.

Seed or migrate database

Before you can use your database the first time, you must create and populate, or migrate it.

For more information about migrating database schemas, see Services Overview.

Troubleshooting

To aid in troubleshooting issues connecting to your service, you can examine the environment variables and log messages Cloud Foundry records for your application.

View environment variables

Use the `cf env` command to view the Cloud Foundry environment variables for your application. `cf env` displays the following environment variables:

- The `VCAP_SERVICES` variables existing in the container environment
- The user provided variables set using the `cf set-env` command

```
$ cf env my-app
Getting env variables for app my-app in org My-Org / space development as adm
in...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "p-mysql-n/a": [
      {
        "credentials": {
            "uri":"postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lra
a
        },
        "label": "p-mysql-n/a",
        "name": "p-mysql",
        "syslog_drain_url": "",
        "tags": ["postgres","postgresql","relational"]
      }
    ]
  }
}

User-Provided:
my-env-var: 100
my-drain: http://drain.example.com
```

View logs

Use the `cf logs` command to view the Cloud Foundry log messages for your application. You can direct current logging to standard output, or you can dump the most recent logs to standard output.

Run `cf logs APPNAME` to direct current logging to standard output:

```
$ cf logs my-app
Connected, tailing logs for app my-app in org My-Org / space development as a
dmin...
1:27:19.72 [App/0]  OUT [CONTAINER]  org.apache.coyote.http11.Http11Protocol
INFO  Starting ProtocolHandler ["http-bio-61013"]
1:27:19.77 [App/0]  OUT [CONTAINER]   org.apache.catalina.startup.Catalina
INFO  Server startup in 10427 ms
```

Run `cf logs APPNAME --recent` to dump the most recent logs to standard output:

```
$ cf logs my-app --recent
Connected, dumping recent logs for app my-app in org My-Org / space developme
nt as admin...
```

```
1:27:15.93 [App/0]  OUT 15,935  INFO EmbeddedDatabaseFactory:124 - Creating embedded database 'SkyNet'
1:27:16.31 [App/0]  OUT 16,318  INFO LocalEntityManagerFactory:287 - Building TM container EntityManagerFactory for unit 'default'
1:27:16.50 [App/0]  OUT 16,505  INFO Version:37 - HCANN001: Hibernate Commons Annotations {4.0.1.Final}
1:27:16.51 [App/0]  OUT 16,517  INFO Version:41 - HHH412: Hibernate Core {4.1.9.Final}
1:27:16.95 [App/0]  OUT 16,957  INFO SkyNet-Online:73 - HHH268: Transaction strategy: org.hibernate.internal.TransactionFactory
1:27:16.96 [App/0]  OUT 16,963  INFO InintiateTerminatorT1000Deployment:48 - HHH000397: Using TranslatorFactory
1:27:17.02 [App/0]  OUT 17,026  INFO Version:27 - HV001: Hibernate Validator 4.3.0.Final
```

If you encounter the error, “A fatal error has occurred. Please see the Bundler troubleshooting documentation,” update your version of bundler and run `bundle install`.

```
$ gem update bundler
$ gem update --system
$ bundle install
```

Windows Gemfile support

Learn how you can handle Ruby buildpack dependencies on Microsoft Windows machines.

Windows Gemfiles

When a `Gemfile.lock` is generated on a Windows machine, it often contains gems with Windows specific versions. This results in versions of gems such as `mysql2`, `thin`, and `pg` containing “-x86-mingw32.” For example, the `Gemfile` might contain the following:

```
gem 'sinatra'
gem 'mysql2'
gem 'json'
```

When you run `bundle install` with `Gemfile` on a Windows machine, it results in the following `Gemfile.lock`:

```
GEM remote: http://rubygems.org/
specs:
  json (1.7.3)
  mysql2 (0.3.11-x86-mingw32)
  rack (1.4.1)
  rack-protection (1.2.0)
  rack sinatra (1.3.2)
  rack (~> 1.3, >= 1.3.6)
  rack-protection (~> 1.2)
  tilt (~> 1.3, >= 1.3.3)
  tilt (1.3.3)
PLATFORMS x86-mingw32
DEPENDENCIES
```

```
  json
  mysql2
  sinatra
```

Notice the “-x86-mingw32” in the version number of `mysql2`. Since Cloud Foundry runs on Linux machines, this fails to install. To mitigate this, the Ruby Buildpack removes the `Gemfile.lock` and uses Bundler to resolve dependencies from the `Gemfile`.

Important

Removing the `Gemfile.lock` causes dependency versions to be resolved from the `Gemfile`. This could result in different versions being installed than those listed in the `Gemfile.lock`.

Staticfile buildpack

You can configure and use the Staticfile buildpack in Cloud Foundry.

Staticfile app: An app or content that requires no backend code other than the NGINX webserver, which the buildpack provides. Examples of staticfile apps are front-end JavaScript apps, static HTML content, and HTML/JavaScript forms.

Staticfile buildpack: The buildpack that provides runtime support for staticfile apps and apps with back ends hosted elsewhere. To find which version of NGINX the current Staticfile buildpack uses, see Staticfile buildpack release notes.

Staticfile detection

If you create a file named `Staticfile` and locate it in the build directory of your app, Cloud Foundry automatically uses the Staticfile buildpack when you push your app.

The `Staticfile` file can be an empty file, or it can contain configuration settings for your app. For more information, see Configuring the buildpack and Pushing an App.

Memory usage

NGINX requires 20 MB of RAM to serve static assets. When using the Staticfile buildpack, VMware push the apps with the `-m 64M` option to reduce RAM allocation from the default 1 GB that are allocated to containers by default.

Configure the buildpack

This section describes configuration options that are available for the Staticfile buildpack. If you need to make configuration changes to NGINX that are not listed in the table, use the NGINX buildpack instead of the Staticfile buildpack. For more information, see NGINX buildpack.

Staticfile file configuration

To configure these options, add the configuration property as a new line in your `Staticfile` file as described in the following table:

Staticfile Configuration Property	Description

`root: YOUR-DIRECTORY-NAME`
 	
Example:
`root: public`
	Alternative root: Specifies a root directory other than the default. Use this option to serve `index.html` and other assets, such as HTML, CSS, or JavaScript files, from a location other than the root directory.
	
	For example, you can specify an alternate folder, `dist` or `public`.

`directory: visible`	**Directory list**: Displays an HTML page that shows a directory index for your site. If your site is missing an `index.html` file, your app displays a directory list instead of the standard 404 error page.
`ssi: enabled`	**SSI**: Activates Server Side Includes (SSI). You can embed the contents of one or more files into a web page on a web server. For general information about SSI, see Server Side Includes entry on Wikipedia.
`pushstate: enabled`	**Pushstate routing**: Keeps browser-visible URLs clean for client-side JavaScript apps that serve multiple routes. For example, pushState routing allows a single JavaScript file to route to multiple anchor tagged URLs that look like `/some/path1` instead of `/some#path1`.
`gzip: off`	**GZip file serving and compression**: Deactivate gzip_static and gunzip modules are activated by default. With these modules you can use NGINX to serve files stored in compressed GZ format and to decompress them for clients that do not support compressed content or responses. You might want to deactivate compression under certain circumstances. For example,when you serve content to very old browser clients.

`http_proxy: HTTP-URL`
`https_proxy: HTTPS-URL`
 	
Example:
`http_proxy:`
`http://www.example.com/`
`https_proxy:`
`https://www.example.com/`
	Proxy support: You can use a proxy when downloading dependencies during the staging of your app.

`enable_http2: true` Alternatively, set the `ENABLE_HTTP2` environment variable to `true`.	**Enable HTTP/2**: Serve requests over HTTP/2 instead of HTTP/1.1. This deactivates serving HTTP/1.1 traffic.
`force_https: true` Alternatively, you can set the `FORCE_HTTPS` environment variable to `true`.	**Force HTTPS**: Forces all requests to be sent through HTTPS. This redirects non-HTTPS requests as HTTPS requests. **Note** Do not activate `FORCE_HTTPS` if you have a proxy server or load balancer loops. For example, if you use Flexible SSL with CloudFlare.
`host_dot_files: true`	**Dot files**: By default, hidden files, which start with a `.`, are not served by this buildpack.

`status_codes:`
`  404: /404.html`
`  500: /500.html`
	Status Codes: You can define custom pages for HTTP STATUS codes instead of the default NGINX pages.

Staticfile Configuration Property	Description
`http_strict_transport_security: true`	**HTTP Strict Transport Security (HSTS)**: Causes NGINX to respond to all requests with the header `Strict-Transport-Security: max-age=31536000`. This forces receiving browsers to make all subsequent requests over HTTPS. This setting defaults to a `max-age` of one year. **Important** This setting persists in browsers for a long time, so you must enable this setting only *after* you ensure that you have completed your app configuration.
`http_strict_transport_security_include_subdomains: true`	**HSTS includes subdomains**: Causes NGINX to respond to all requests with the following header: `Strict-Transport-Security: max-age=31536000; includeSubDomains` This forces browsers to make all subsequent requests over HTTPS including subdomains. This setting defaults to a `max-age` of one year. **Important** Setting this property to `true` also makes `http_strict_transport_security` default to true.
`http_strict_transport_security_preload: true`	**HSTS preload**: Causes NGINX to respond to all requests with the following header: `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload` This forces browsers to make all subsequent requests over HTTPS including subdomains and requests inclusion in browser-managed HSTS preload lists. For more information, see https://hstspreload.org. This setting defaults to a `max-age` of one year. **Important** Setting this property to `true` also makes `http_strict_transport_security` and `http_strict_transport_security_include_subdomains` default to true.

Other configuration changes

Use the following instructions to make other configuration changes.

Basic authentication

You can enable basic authentication for your app or website. You can create a hashed username and password pair for each user by using a site like Htpasswd Generator.

Configuration

Add a new `Staticfile.auth` file to the root or alternative root directory. In the new file, add one or more username and password entries using the following format: `USERNAME:HASHED_PASSWORD`

Example:

```
pat:$example1$DuUQEQp8$ZccZCHQElNSjrgerwSFC0
stevie:$example1$22derfaecZSJJRw4rKE$KKEWKSK
```

Custom location

You can specify custom location definitions with additional directives. For information about NGINX directives, see Creating NGINX Plus and NGINX Configuration Files and Alphabetical index of directives in the *NGINX documentation*.

Configuration

To customize the `location` block of the NGINX configuration file:

1. Set an alternative `root` directory. The `location_include` property only works in conjunction with an alternative `root`. See the following example that includes a static content file:
   - **Directory**: `public`
   - **File**: `public/index.html`
2. Create a file with location-scoped NGINX directives. See the following example, which causes visitors of your site to receive the `X-MySiteName` HTTP header:
   - **File**: `nginx/conf/includes/custom_header.conf`
   - **Content**: `add_header X-MySiteName BestSiteEver;`
3. Set the `location_include` variable in your **Staticfile** to the path of the file from the previous step. This path is relative to `nginx/conf`.

   Example:

   ```
   ...
   root: public
   location_include: includes/*.conf
   ...
   ```

Additional MIME type support

You can configure additional MIME types on your NGINX server.

Configuration

To add MIME types, add a `mime.types` file to your root folder, or to the alternate root folder, if you specified one.

For more information about the `mime.types` file, see mime.types in the *NGINX documentation*.

Example MIME types file:

```

types {
    text/html      html htm shtml;
    text/css       css;
    text/xml       xml rss;
```

```
    image/gif      gif;
    ...
    }
```

Push an app

To push your app, you can use either the system Staticfile buildpack or specify a Staticfile buildpack.

Option 1: Use the system Staticfile buildpack

To use the Staticfile buildpack installed in your deployment, run the following command in the root directory of the app. The root directory of the app must contain a file named `Staticfile`.

```
cf push APP-NAME
```

Where `APP-NAME` is the name you want to give your app.

Option 2: Specify a Staticfile buildpack

To explicitly specify a Staticfile buildpack, run the following command in the root directory of the app. You might want to specify a buildpack if your deployment does not have the Staticfile buildpack installed or the installed version is outdated.

```
cf push APP-NAME -b BUILDPACK-NAME-OR-PATH
```

Where:

- `APP-NAME` is the name you want to give your app.
- `BUILDPACK-NAME-OR-PATH` is either the name of a buildpack that is installed in your deployment or the path to a buildpack. You can find the Cloud Foundry Staticfile buildpack in the Staticfile repository on GitHub.

Verifying the push

After you push the app, follow these steps to verify that the push was successful:

1. Find the URL of your app in the output of the `cf push` command. For example, the URL in the terminal output above is `my-app.example.com`.
2. In a browser, go to the URL to view your app.

Example Staticfile buildpack apps

For different examples of apps that use the Staticfile buildpack, see the fixtures directory in the Staticfile buildpack GitHub repo.

Help and support

A number of channels exist where you can get more help when using the Staticfile buildpack, or with developing your own Staticfile buildpack.

- **Staticfile Buildpack Repository in GitHub**: Find more information about using and extending the Staticfile buildpack in GitHub repository.
- **Release Notes**: Find current information about this buildpack on the Staticfile buildpack release page in GitHub.
- **Slack**: Join the #buildpacks channel in the Cloud Foundry Slack community.

Using Buildpacks

In this section:

- How Buildpacks Work
- Stack Association
- Pushing an App with Multiple Buildpacks
- Using a Proxy Server
- Supported Binary Dependencies
- Production Server Configuration
- Sidecar Buildpacks

Working with buildpacks in Cloud Foundry

This topic tells you about how buildpacks are structured and detected in Cloud Foundry.

Buildpack scripts

A buildpack repository might contain the following scripts in the `bin` directory:

- `bin/detect` determines whether or not to apply the buildpack to an app.
- `bin/supply` provides dependencies for an app.
- `bin/finalize` prepares the app for launch.
- `bin/release` provides feedback metadata to Cloud Foundry that indicates how you can run the app.
- `bin/compile` is a deprecated alternative to `bin/supply` and `bin/finalize`.

The `bin/supply` and `bin/finalize` scripts replace the deprecated `bin/compile` script. Older buildpacks might still use `bin/compile` with the latest version of Cloud Foundry. In this case, applying multiple buildpacks to a single app is not supported. Newer buildpacks might still provide `bin/compile` for compatibility with Heroku and older versions of Cloud Foundry.

The `bin/supply` script is required for non final buildpacks. The `bin/finalize` (or `bin/compile`) script is required for final buildpacks.

In this topic, the terms *non final buildpack* and *final buildpack*, or *last buildpack*, are used to describe the process of applying multiple buildpacks to an app. For example: `cf push APP-NAME -b FIRST-BUILDPACK -b SECOND-BUILDPACK -b FINAL-BUILDPACK`.

If you use only one buildpack for your app, this buildpack behaves as a final, or last, buildpack.

When using multi buildpack support, the last buildpack in the order is the final buildpack, and is able to make changes to the app and determine a start command. All other specified buildpacks are non final and supply only the dependencies.

bin/detect

The `detect` script determines whether or not to apply the buildpack to an app. The script is called with one argument, the `build` directory for the app. The `build` directory contains the app files uploaded when you run `cf push`.

The `detect` script returns an exit code of `0` if the buildpack is compatible with the app. In the case of system buildpacks, the script prints the buildpack name, version, and other information to `STDOUT`.

The following is an example `detect` script that checks for a Ruby app that is based on the existence of a `Gemfile`:

```
#!/usr/bin/env ruby

gemfile_path = File.join ARGV[0], "Gemfile"

if File.exist?(gemfile_path)
  puts "Ruby"
  exit 0
else
  exit 1
end
```

The buildpack `detect` script output provides additional details by the buildpack developer. This includes buildpack version information and a list of configured frameworks and their associated versions.

The following is an example of the detailed information returned by the Java buildpack:

```
java-buildpack=v3.0-https://github.com/cloudfoundry/java-buildpack.git#3bd15e1 open-jd
k-jre=1.8.0_45 spring-auto-reconfiguration=1.7.0_RELEASE tomcat-access-logging-support
=2.4.0_RELEASE tomcat-instance=8.0.21 tomcat-lifecycle-support=2.4.0_RELEASE ...
```

By default, Cloud Foundry detects only one buildpack. When you want to detect multiple buildpacks, you must explicitly specify them.

For more information, see Buildpack detection.

bin/supply

The `supply` script provides dependencies for the app and runs for all buildpacks. All output is sent to `STDOUT` and is relayed to the user through the Cloud Foundry Command Line Interface (cf CLI).

The script is run with four arguments:

- The `build` directory for the app.
- The `cache` directory, which is a location the buildpack can use to store assets during the build process.
- The `deps` directory, which is where dependencies provided by all buildpacks are installed.
- The `index`, which is a number that represents the ordinal position of the buildpack.

The `supply` script stores dependencies in the `deps`/`index` directory. It might also look in other directories in `deps` to find dependencies supplied by other buildpacks.

The `supply` script must not modify anything outside of the `deps`/`index` directory. Staging might fail if such modification is detected.

The `cache` directory is provided to the `supply` script for the final buildpack and is preserved even when the buildpack is upgraded or changes. The `finalize` script also has access to this cache directory.

The `cache` directories provided to the `supply` scripts of non final buildpacks are cleared if those buildpacks are upgraded or changed.

The following is an example of a simple `supply` script:

```
#!/usr/bin/env ruby

#sync output

$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

install_ruby

private

def install_ruby
  puts "Installing Ruby"

  # !!! build tasks go here !!!
  # download ruby
  # install ruby
end
```

bin/finalize

The `finalize` script prepares the app for launch and runs only for the last buildpack. All output is sent to `STDOUT` and is relayed to the user through the cf CLI.

The script is run with four arguments:

- The `build` directory for the app.
- The `cache` directory, which is a location the buildpack can use to store assets during the build process.

- The `deps` directory, which is where dependencies provided by all buildpacks are installed.
- The `index`, which is a number that represents the ordinal position of the buildpack.

The `finalize` script might find dependencies installed by the `supply` script of the same buildpack in `deps`/`index` directory. It might also look in other directories in `deps` to find dependencies that are supplied by other buildpacks.

The `cache` directory provided to the `finalize` script is preserved even when the buildpack is upgraded or otherwise changes. The `supply` script of the same buildpack also has access to this cache directory.

The following example shows a simple `finalize` script:

```
#!/usr/bin/env ruby

#sync output

$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]
deps_path = ARGV[2]
index = ARGV[3]

setup_ruby

private

def setup_ruby
  puts "Configuring your app to use Ruby"

  # !!! build tasks go here !!!
  # setup ruby
end
```

Deprecated bin/compile

The `compile` script is deprecated. It encompasses the behavior of the `supply` and `finalize` scripts for single buildpack apps by using the `build` directory to store dependencies.

The script is run with two arguments:

- The `build` directory for the app.
- The `cache` directory, which is a location the buildpack can use to store assets during the build process.

During the running of the `compile` script, all output is sent to `STDOUT` and relayed to the user through the cf CLI.

bin/release

The `release` script provides feedback metadata to Cloud Foundry that indicates how you run the app. The script is run with one argument, the `build` directory. The script must generate a YAML file in the following format:

```
default_process_types:
  web: start_command.filetype
```

The `default_process_types` line indicates the type of app being run and the command used to start it. This start command is used if a start command is not specified in the `cf push` or in a Procfile.

At this time, only the `web` type of apps is supported.

To define environment variables for your buildpack, add a Bash script to the `.profile.d` directory in the root folder of your app.

The following example shows what a `release` script for a Rack app might return:

```
default_process_types:
  web: bundle exec rackup config.ru -p $PORT
```

The `web` command runs as `bash -c COMMAND` when Cloud Foundry starts your app.

Droplet filesystem

The buildpack staging process extracts the droplet into the `/home/vcap` directory inside the instance container and creates the following filesystem tree:

```
app/
deps/
logs/
tmp/
staging_info.yml
```

The `app` directory includes the contents of the `build` directory, and the `staging_info.yml` file contains the staging metadata saved in the droplet.

Buildpack detection

When you push an app, Cloud Foundry uses a detection process to determine a single buildpack to use. For general information about this process, see How Diego Stages Buildpack Apps section of the *How Apps Are Staged* topic.

During staging, each buildpack has a position in a priority list. You can retrieve this position by running the `cf buildpacks`.

Cloud Foundry checks if the buildpack in position 1 is a compatible buildpack. If the position 1 buildpack is not compatible, Cloud Foundry moves on to the buildpack in position 2. Cloud Foundry continues this process until the correct buildpack is found.

If no buildpack is compatible, the `cf push` command fails with the following error:

```
None of the buildpacks detected a compatible application
Exit status 222
Staging failed: Exited with status 222

FAILED
NoAppDetectedError
```

For a more detailed account of how Cloud Foundry interacts with the buildpack, see Sequence of interactions.

Sequence of interactions

This section describes the sequence of interactions between the Cloud Foundry platform and the buildpack. The sequence of interactions differs depending on whether the platform skips or performs buildpack detection.

No buildpack detection

Cloud Foundry skips buildpack detection if the developer specifies one or more buildpacks in the app manifest or in the `cf push APP-NAME -b BUILDPACK-NAME` cf CLI command.

If you explicitly specify buildpacks, Cloud Foundry performs the following interactions:

1. For each buildpack except the last buildpack, the platform:
2. Creates the `deps`/`index` directory.
3. Runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`.
4. Accepts any modification of the `deps`/`index` directory.
5. Accepts any modification of the `cache` directory.
6. Might disallow modification of any other directories.
7. For the last buildpack, the platform:
8. If `/bin/finalize` is present:
9. Creates the `deps`/`index` directory if it does not exist.
10. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`.
11. Accepts any modification of the `deps`/`index` directory.
12. Might disallow modification of the `build` directory.
13. Runs `/bin/finalize` with the `build`, `cache`, and `deps` directories and the buildpack `index`.
14. Accepts any modification of the `build` directory.
15. If `/bin/finalize` is not present:
16. Runs `/bin/compile` with the `build` and `cache` directories.
17. Accepts any modification of the `build` directory.
18. Runs `/bin/release` to determine staging information.

At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.

Buildpack detection

Cloud Foundry does buildpack detection if you do not specify one or more buildpacks in the app manifest or in the `cf push APP-NAME -b BUILDPACK-NAME` cf CLI command.

Cloud Foundry detects only one buildpack to use with the app.

If the platform performs detection, it:

1. Runs `/bin/detect` for each buildpack.
2. Selects the first buildpack with a `/bin/detect` script that returns a zero exit status.
3. If `/bin/finalize` is present:
4. Creates the `deps`/`index` directory if it does not exist.
5. If `/bin/supply` is present, runs `/bin/supply` with the `build`, `cache`, and `deps` directories and the buildpack `index`.
6. Accepts any modification of the `deps`/`index` directory.
7. Might disallow modification of the `build` directory.
8. Runs `/bin/finalize` on the `build`, `cache`, and `deps` directories.
9. Accepts any modification of the `build` directory.
10. If `/bin/finalize` is not present:
11. Runs `/bin/compile` on the `build` and `cache` directories.
12. Accepts any modification of the `build` directory.
13. Runs `/bin/release` to determine staging information.

At the end of this process, the `deps` directory is included at the root of the droplet, adjacent to the `app` directory.

Stack association in Cloud Foundry buildpacks

Learn about the stack association feature for your Cloud Foundry buildpacks.

To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

Stack association

Each buildpack in your deployment is associated with a stack. When you run `cf buildpacks`, you can see this in the `stack` column for each buildpack.

For example:

```
$ cf buildpacks
Getting buildpacks...

position   name                            stack        enabled   locked
filename
1          staticfile_buildpack            cflinuxfs3   true      false
staticfile_buildpack-cached-cflinuxfs3-v1.5.36.zip
2          java_buildpack                  cflinuxfs3   true      false
java_buildpack-cached-cflinuxfs3-v4.53.zip
```

```
3          ruby_buildpack                  cflinuxfs3   true      false
ruby_buildpack-cached-cflinuxfs3-v1.9.0.zip
. . .
12         ruby_buildpack                  cflinuxfs4   true      false
ruby_buildpack-cached-cflinuxfs4-v1.9.0.zip
13         dotnet_core_buildpack           cflinuxfs4   true      false
dotnet-core_buildpack-cached-cflinuxfs4-v2.4.5.zip
. . .
```

Because of the stack association, buildpacks do not have to be uniquely named. This helps in managing similar buildpacks that are compatible with different stacks.

The buildpack packager includes a `-stack` option. If you use this option and upload a buildpack, the Cloud Controller detects the stack association, and creates a stack record for the buildpack.

Missing stack record in buildpacks

Some buildpacks might have a missing stack record, if for example, you uploaded a custom buildpack before Cloud Foundry introduced stack association. The output of `cf buildpacks` shows a blank `stack` column if the buildpack does not have a stack record.

In this case, you must manually assign a stack to the buildpack. To do this, run:

```
cf update-buildpack BUILDPACK-NAME --assign-stack stack
```

Buildpacks with a missing stack record continue to work, but are more manageable when the stack record is present.

If you push apps to a deployment that has buildpacks with a missing stack record, the following might occur:

- If you push an app and specify a stack with `cf push app-name -s stack`, Cloud Foundry uses that stack. Otherwise, it uses the system default, `cflinuxfs4`.
- You might see additional logging in the buildpack detection output of the `cf push` command when Cloud Foundry detects buildpacks without a stack record.

Managing stack association with the cf CLI

The cf CLI commands for managing buildpacks include functionality to support association between buildpacks and stacks. The `update-buildpack`, `rename-buildpack`, and `delete-buildpack` commands all include a `-s` flag for specifying a stack.

The cf CLI v7 command removes the `cf rename-buildpack` command in favor of a `--rename` option for `cf update-buildpack`.

When using buildpacks with the cf CLI, consider the following:

- You cannot upload a buildpack with `cf create-buildpack` if a buildpack of the same name already exists and has a missing stack record.
- When using `cf create-buildpack`, you might inadvertently create a duplicate buildpack with a `nil` stack. `cf create-buildpack` does not prevent creation of buildpacks with no stack association.

- The `-s` flag is required when there are buildpacks with the same name. If you are working on a uniquely named buildpack, you do not need to specify its stack.
- If you have buildpacks of the same name, one with a stack record and one without, run cf CLI commands without the `-s` flag on the buildpack with the missing stack record.

Scenario examples

See the following examples for managing buildpacks with the cf CLI. These examples are applicable when running `cf update-buildpack` or `cf delete-buildpack`:

- **Updating or deleting a uniquely-named buildpack:**
    - You have a single buildpack named `my-buildpack`, and it is associated with `stack_a`. To delete the buildpack, run `cf delete-buildpack my-buildpack`. You can also provide `-s stack_a`, but the option is not required if you have a uniquely-named buildpack.
- **Updating or deleting a uniquely-named buildpack that has a `nil` stack:**
    - You have a single buildpack named `my-buildpack`, and it is not associated with a stack. To delete the buildpack, run `cf delete-buildpack my-buildpack`.
- **Updating or deleting a buildpack when another buildpack exists with the same name.** Both buildpacks have stack associations:
    - You have two buildpacks named `my-buildpack`, one that is associated with `stack_a` and the other associated with `stack_b`. To delete the buildpack that uses `stack_a`, run `cf delete-buildpack my-buildpack -s stack_a`.
- **Updating or deleting a buildpack when another buildpack exists with the same name.** One buildpack has a stack association, and the other buildpack has a `nil` stack:
    - You have two buildpacks named `my-buildpack`, one associated with `stack_a` and the other with no (`nil`) stack association:
        - To delete the buildpack that uses `stack_a`, run `cf delete-buildpack my-buildpack -s stack_a`.
        - To delete the buildpack that is associated with the `nil` stack, run `cf delete-buildpack my-buildpack`.

Pushing an app with multiple buildpacks

You can push an app with multiple buildpacks using the Cloud Foundry Command Line Interface (cf CLI).

As an alternative to the cf CLI procedure, you can specify multiple buildpacks in your app manifest. This is not compatible with deprecated app manifest features. For more information, see Deploying with App Manifests.

For more information about pushing apps, see Pushing an App.

Specifying buildpacks with the cf CLI

To push an app with multiple buildpacks using the cf CLI:

Note

You must use cf CLI v6.38 or later.

1. Ensure that you are using the cf CLI v6.38 or later by running:

   ```
   cf version
   ```

   For more information about upgrading the cf CLI, see Installing the cf CLI.

2. To push your app with multiple buildpacks, specify each buildpack with a `-b` flag by running:

   ```
   cf push YOUR-APP -b BUILDPACK-NAME-1 -b BUILDPACK-NAME-2 ... -b BUILDPACK-NAME-
   3
   ```

   Where:

   - `YOUR-APP` is the name of your app.
   - `BUILDPACK-NAME-1`, `BUILDPACK-NAME-2`, and `BUILDPACK-NAME-3` are the names of the buildpacks you want to push with your app.

   The last buildpack you specify is the **final buildpack**, which modifies the launch environment and sets the start command.

   To see a list of available buildpacks, run:

   ```
   cf buildpacks
   ```

For more information on multi buildpack order, see How buildpacks work.

For more information about using the cf CLI, see Using the cf CLI Cloud Foundry command line interface.

Using a proxy server with Cloud Foundry buildpacks

You can use a proxy server with buildpacks for your application.

Using a proxy server

You can assign proxy servers to environment variables. Proxy servers can be used to monitor your application’s traffic or to fetch your application’s dependencies.

A buildpack uses a proxy server if that buildpack contacts the internet during staging.

The binary buildpack does not use a proxy server because it does not access the internet during staging.

Setting environment variables

If you are using a Java buildpack, the `http_proxy` and `https_proxy` environment variables are not supported at runtime. The Java buildpack does not add the functionality to make proxies work at runtime.

To set your environment variables:

1. Add the following to the `env` block of your application manifest YAML file:

   ```
   ---
   env:
     http_proxy: http://YOUR-HTTP-PROXY:PORT
     https_proxy: https://YOUR-HTTPS-PROXY:PORT
   ```

   Where:

   - `YOUR-HTTP-PROXY` is the address of your proxy server for HTTP requests.
   - `YOUR-HTTPS-PROXY` is the address of your proxy server for HTTPS requests.
   - `PORT` is the port number you are using for your proxy server.

2. Set the environment variables with the Cloud Foundry Command Line Interface (cf CLI) `cf set-env` command:

   ```
   cf set-env YOUR-APP http_proxy "http://YOUR-HTTP-PROXY:PORT"
   ```

   ```
   cf set-env YOUR-APP https_proxy "https://YOUR-HTTPS-PROXY:PORT"
   ```

   Where:

   - `YOUR-APP` is the name of your application.
   - `YOUR-HTTP-PROXY` is the address of your proxy server for HTTP requests.
   - `YOUR-HTTPS-PROXY` is the address of your proxy server for HTTPS requests.
   - `PORT` is the port number you are using for your proxy server.

Un-setting environment variables

Removing an environment variable from the application manifest YAML file is not sufficient to unset the environment variable.

You must also unset the environment variables with the Cloud Foundry Command Line Interface (cf CLI) `cf unset-env` command:

~~~
```console
cf unset-env YOUR-APP ENV_VAR_NAME
```
~~~

For example:

~~~
```console
cf unset-env YOUR-APP https_proxy
```

Where:
<ul>
  <li><code>YOUR-APP</code> is the name of your application.</li>
~~~

```
  <li><code>ENV_VAR_NAME</code> is the name of the environment variable.</li>
</ul>
```

Supported binary dependencies in Cloud Foundry buildpacks

Each buildpack supports only the stable patches for each dependency listed in the buildpack's `manifest.yml`file and also in the GitHub releases page. For example, see the php-buildpack releases page.

If you try to use an unsupported binary, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:
...
 !
 !     exit
 !
Staging failed: Buildpack compilation step failed
```

Configuring the production server for Cloud Foundry apps

You can configure a production server for your apps in Cloud Foundry.

When you deploy an app, Cloud Foundry determines the command that is used to start the app through the following process:

1. If you use the command `cf push -c COMMAND`, then Cloud Foundry uses `COMMAND` to start the app.
2. If you create a file called a Procfile, Cloud Foundry uses Procfile to configure the command that launches the app. See About Procfiles.
3. If you do not use `cf push -c COMMAND` and do not create a Procfile, then Cloud Foundry does one of the following, depending on the buildpack:
    - Uses a default start command.
    - Fails to start the app and shows a warning that the app is missing a Procfile.

Procfiles

Use a Procfile to specify a start command for buildpacks when a default start command is not provided. Some buildpacks use Python to work on a variety of frameworks and do not attempt to provide a default start command.

Another reason to use a Procfile is to configure a production server for web apps.

When you use a Procfile, you declare required runtime processes, and called process types, for your web app. Process managers in a server use the process type to run and manage the workload.

In a Procfile, you declare one process type per line and use the following syntax:

```
PROCESS-TYPE: COMMAND
```

Where:

- `PROCESS-TYPE` is `web`. A `web` process handles HTTP traffic.
- `COMMAND` is the command line to launch the process.

For example, a Procfile with the following content starts the launch script created by the build process for a Java app:

```
web: build/install/MY-PROJECT-NAME/bin/MY-PROJECT-NAME
```

Procfile support is integrated into the buildpack lifecycle. However, due to differing behavior of buildpacks, it might not be suitable with all buildpacks.

Procfiles can be used with the following buildpacks:

- Binary buildpack
- .NET Core buildpack
- Go buildpack
- NGINX buildpack
- Node.js buildpack
- Python buildpack
- R buildpack
- Ruby buildpack

Specifying a web server

Follow these steps to specify a web server using a Procfile. For more information about configuring a web server for Rails apps, see Configuring a Ruby web server.

1. Create a blank file with a command line for a `web` process type.
2. Save it as a file named `Procfile` with no extension in the root directory of your app.
3. Push your app.

Configuring a Ruby web server

Cloud Foundry uses the default standard Ruby web server library WEBrick for Ruby and Ruby on Rails apps. However, Cloud Foundry can support a more robust production web server. For example, Phusion Passenger, Puma, Thin, or Unicorn.

To instruct Cloud Foundry to use a web server other than WEBrick, use the following steps:

1. Add the gem for the web server to your Gemfile.
2. In the `config` directory of your app, create a new configuration file or modify an existing file. See your web server documentation for how to configure this file.

   The following example uses the Puma web server:

```
<pre class="terminal">
    &#35; config/puma.rb
    threads 8,32
    workers 3

    on_worker_boot do
    # things workers do
    end
</pre>
```

3. In the root directory of your app, create a Procfile and add a command line for a `web` process type that points to your web server. For information about configuring the specific command for a process type, see your web server documentation.

   The following example shows a command that starts a Puma web server and specifies the app runtime environment, TCP port, paths to the server state information, and configuration files:

   ```
   web: bundle exec puma -e $RAILS_ENV -p 1234 -S ~/puma -C config/puma.rb
   ```

Sidecar buildpacks

Learn how buildpacks deploy sidecar processes alongside your app.

Sidecar processes are additional dependent processes that are run in the same container as the main app process. For more information, see How to Push an App to Cloud Foundry with Sidecars.

Specifying sidecar processes

A buildpack can specify sidecar processes with a `launch.yml` file at the root of the `deps` directory during the `supply` phase. The `deps` directory is a combination of the `DEPS_DIR` and `DEP_IDX` values. For more information about how these two values are passed to the buildpack interface, see How buildpacks work.

The following defines the relevant fields of the `launch.yml` file in more detail:

- `type`: A key that is mapped to a individual command.

  It is possible for multiple buildpacks to write their own `launch.yml` files. If two of these files have the same `type` keys, the `launch.yml` file written last overwrites the previous commands of the same `type`.

- `command`: The command to be run in the container. Container health is dependent on this command. The container fails when the command is no longer running.
- `limits`: Resource limits that are placed on the sidecar process.
- `memory`: The memory limit is in MB. If you use the sidecar buildpack with a Java app, you must configure this field to allocate memory to the sidecar. If you do not configure the field, the Java buildpack allocates all of the available memory to the app.
- `platforms`: A key specifying the data that must be read by a platform. For example, Cloud Foundry only reads data under the `cloudfoundry` key.

- `sidecar_for`: A list of `types` that the `command` uses for a sidecar process.

  Each `type` requires a health check. The `command` cannot run until each health check is passed. The more `types` that are listed, the more health checks are required. If any health check fails, the `command` does not run.

Example `launch.yml` file:

```
---
processes:
- type: "PROCESS-NAME"
  command: "COMMAND"
  limits:
    memory: 10
  platforms:
    cloudfoundry:
      sidecar_for: [ "TYPE-1", "TYPE-2", "TYPE-3"]
```

Where:

- `PROCESS-NAME` is the command that is mapped to the `type` field.
- `COMMAND` is the command that is mapped to the `command` field. For example, `./binary` or `java -jar java-file.jar`.
- `TYPE-1`, `TYPE-2` and `TYPE-3` are the `types` that the `command` uses for the sidecar process.

Buildpack example

The following buildpack example functions as a supply buildpack, or the non final buildpack in a multi buildpack build. The only purpose for this function is to write the `launch.yml` file to the buildpack's `deps` directory.

For the full sidecar buildpack, see example-sidecar-buildpack repository on GitHub.

The `example-sidecar-buildpack` has the following directory structure:

```
bin/
bin/supply
manifest.yml
VERSION
```

The `supply` script writes a `launch.yml` file to a specific location, `bin/supply`, as in the following example:

```
#!/bin/bash

set -euo pipefail

BUILD_DIR=$1
CACHE_DIR=$2
DEPS_DIR=$3
DEPS_IDX=$4

LAUNCH_CONTENTS='---
processes:
- type: "sidecar_process"
```

```
  command: "while true; do echo hello from a sidecar process; sleep 10; done"
  platforms:
    cloudfoundry:
      sidecar_for: [ "web"]
'

echo "-----> Running sidecar supply"

export BUILDPACK_DIR=`dirname $(readlink -f ${BASH_SOURCE%/*})`

pushd "$BUILDPACK_DIR"
  echo "$LAUNCH_CONTENTS" > "$DEPS_DIR"/"$DEPS_IDX"/launch.yml
popd
```

Customizing and Developing Buildpacks

In this section:

- Customizing and Developing Buildpacks
- Creating Custom Buildpacks
- Packaging Dependencies for Offline Buildpacks
- Merging from Upstream Buildpacks
- Upgrading Dependency Versions
- Releasing a New Buildpack Version
- Updating Buildpack-Related Gems

Customizing and developing buildpacks in Cloud Foundry

Buildpacks enable you to package frameworks and runtime support for your application. Cloud Foundry provides system buildpacks and an interface for customizing existing buildpacks and developing new buildpacks.

Customizing and creating buildpacks

If your application uses a language or framework that the Cloud Foundry system buildpacks do not support, do one of the following:

- Use a Cloud Foundry Community Buildpack.
- Use a Heroku Third-Party Buildpack.
- Customize an existing buildpack or create your own custom buildpack.

A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream.

For information about customizing an existing buildpack or creating your own, see the following:

```
+ <a href="./custom.html" class="subnav">Creating Custom Buildpacks</a>
+ <a href="./depend-pkg-offline.html" class="subnav">Packaging Dependencies for Offlin
```

```
e Buildpacks</a>
```

Maintaining buildpacks

After you have modified an existing buildpack or created your own, it is necessary to maintain it.

See the following topics to maintain your own buildpacks:

- Merging with upstream buildpacks
- Upgrading dependency versions

To configure a production server for your web app, see Configuring a production server.

Using CI for buildpacks

For information about updating and releasing a new version of a Cloud Foundry buildpack through the Cloud Foundry Buildpacks Team Concourse pipeline, see Using CI for buildpacks. You can use this as a model when working with Concourse to build and release new versions of your own buildpacks.

Creating custom buildpacks for Cloud Foundry

You can create custom buildpacks for Cloud Foundry.

For more information about how buildpacks work, see How Buildpacks Work.

Packaging custom buildpacks

Cloud Foundry buildpacks can work with limited or no internet connectivity. The Buildpack Packager gives the same flexibility to custom buildpacks, enabling them to work in partially or completely disconnected environments. For more information, see the Buildpack Packager repository on GitHub.

Using the Buildpack Packager

To use the Buildpack Packager:

1. Download the Buildpack Manager from the Buildpack Packager repository on GitHub.
2. Create a `manifest.yml` file in your buildpack.
3. Run the packager in cached mode:

   ```
   buildpack-packager build -cached -any-stack
   ```

The packager adds everything in your buildpack directory into a ZIP file, and excludes anything marked for exclusion in your manifest.

In cached mode, the packager downloads, and adds dependencies as described in the manifest.

For more information, see Buildpack Packager repository on GitHub.

Using and sharing the packaged buildpack

After you have packaged the buildpack using Buildpack Packager, you can use the ZIP file locally, or share it with others by uploading it to any network location that is accessible to the CLI. You can can then specify the buildpack with the `-b` option when you push apps. For more information, see Deploying Apps with a Custom Buildpack.

Offline buildpack packages might contain proprietary dependencies that require distribution licensing or export control measures. For more information about offline buildpacks, see About Offline Buildpacks section of the *Packaging dependencies for offline Buildpacks* topic.

Specifying a default version

As of Buildpack Packager v2.3.0, you can specify the default version for a dependency by adding a `default_versions` object to the `manifest.yml` file. The `default_versions` object has two properties, `name` and `version`.

For example:

```
default_versions:
- name: go
  version: 1.6.3
- name: other-dependency
  version: 1.1.1
```

To specify a default version:

1. Add the `default_version` object to your manifest, following the guidance in Rules for Specifying a Default Version. For a complete example, see manifest.yml in the Cloud Foundry Go(Lang) Buildpack repository in GitHub.
2. Run the `default_version_for` script from the compile-extensions repository, passing the path of your `manifest.yml` and the dependency name as arguments. Run:

   ```
   ./compile-extensions/bin/default_version_for manifest.yml DEPENDENCY-NAME
   ```

   Where `DEPENDENCY-NAME` is the `name` property from the `default_versions` object in your `manifest.yml` file.

For more information, see Buildpack Packager v2.3.0 in the Buildpack Packager repository on GitHub.

Rules for specifying a default version

The Buildpack Packager script validates this object according to the following rules:

- You can create at most one entry under `default_versions` for a single dependency. The following example causes Buildpack Packager to fail with an error because the manifest file specifies two default versions for the same `go` dependency.

  ```
  default_versions:
  - name: go
    version: 1.6.3
  - name: go
    version: 1.7.5
  ```

- If you specify a `default_version` for a dependency, you must also list that dependency and version under the `dependencies` section of the manifest. The following example causes Buildpack Packager to fail with an error because the manifest specifies `version: 1.9.2` for the `go` dependency, but lists `version: 1.7.5` under `dependencies`.

```
default_versions:
- name: go
  version: 1.9.2

dependencies:
- name: go
  version: 1.7.5
  uri: https://storage.googleapis.com/golang/go1.7.5.linux-amd64.tar.gz
  md5: c8cb76e2308c792e2705c2eb1b55de95
  cf_stacks:
  - cflinuxfs3
```

Important

To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

Core buildpack communication contract

Learn about the communication contract followed by the Cloud Foundry core buildpacks. This contract enables buildpacks to interact with one another, so that you can use multiple buildpacks with your apps.

You must ensure your custom buildpacks follow the contract.

This section uses the following placeholders:

- `IDX` is the zero-padded index matching the position of the buildpack in the priority list.
- `MD5` is the MD5 checksum of the buildpack's URL.

For all buildpacks that supply dependencies through `/bin/supply`:

- The buildpack must create `/tmp/deps/IDX/config.yml` to provide a name to subsequent buildpacks. This file might also contain miscellaneous configuration for subsequent buildpacks.
- The `config.yml` file must be formatted as:

  ```
  name: BUILDPACK
  config: YAML-OBJECT
  ```

  Where:

  - `BUILDPACK` is the name of the buildpack providing dependencies.
  - `YAML-OBJECT` is the YAML object that contains buildpack-specific configuration.

- The following directories can be created inside of `/tmp/deps/IDX/` to provide dependencies to subsequent buildpacks:
    - `/bin`: Contains binaries intended for `$PATH` during staging and launch.
    - `/lib`: Contains libraries intended for `$LD_LIBRARY_PATH` during staging and launch.
    - `/include`: Contains header files intended for compilation during staging.
    - `/pkgconfig`: Contains `pkgconfig` files intended for compilation during staging.
    - `/env`: Contains environment variables intended for staging, loaded as `FILENAME=FILECONTENTS`.
    - `/profile.d`: Contains scripts intended for `/app/.profile.d`, sourced before launch.
- The buildpack might make use of previous non-final buildpacks by scanning `/tmp/deps/` for index-named directories containing `config.yml`.

For the last buildpack:

- To make use of dependencies provided by the previously applied buildpacks, the last buildpack must scan `/tmp/deps/` for index-named directories containing `config.yml.`
- To make use of dependencies provided by previous buildpacks, the last buildpack:
    - Can use `/bin` during staging, or make it available in `$PATH` during launch.
    - Can use `/lib` during staging, or make it available in `$LD_LIBRARY_PATH` during launch.
    - Can use `/include`, `/pkgconfig`, or `/env` during staging.
    - Can copy files from `/profile.d` to `/tmp/app/.profile.d` during staging.
    - Can use the supplied config object in `config.yml` during the staging process.

Deploying apps with a custom buildpack

Once a custom buildpack has been created and pushed to a public Git repository, the Git URL can be passed through the cf CLI when pushing an app.

For example, you can use a buildpack that has been pushed to GitHub by running:

```
cf push YOUR-APP -b git://github.com/REPOSITORY/BUILDPACK.git
```

Where:

- `YOUR-APP` is the name of your app.
- `REPOSITORY` is the name of your public Git repository.
- `BUILDPACK` is the name of your custom buildpack.

Alternatively, you can use a private Git repository, with HTTPS, and username and password authentication:

```
cf push YOUR-APP -b https://USERNAME:PASSWORD@github.com/REPOSITORY/BUILDPACK.git
```

Where:

- `YOUR-APP` is the name of your app.

- `USERNAME` is your Git username.
- `PASSWORD` is the name of your Git password.
- `REPOSITORY` is the name of your public Git repository.
- `BUILDPACK` is the name of your custom buildpack.

By default, Cloud Foundry uses the default branch of the buildpack's Git repository. You can specify a different branch using the Git URL:

```
cf push YOUR-APP -b https://github.com/REPOSITORY/BUILDPACK.git#BRANCH
```

Where:

- `YOUR-APP` is the name of your app.
- `REPOSITORY` is the name of your public Git repository.
- `BUILDPACK` is the name of your custom buildpack.
- `BRANCH` is the branch you want to use.

Additionally, you can use tags in a Git repository:

```
cf push YOUR-APP -b https://github.com/REPOSITORY/BUILDPACK#TAG
```

Where:

- `YOUR-APP` is the name of your app.
- `REPOSITORY` is the name of your public Git repository.
- `BUILDPACK` is the name of your custom buildpack.
- `TAG` is the Git repository tag you want to use.

The app is then deployed to Cloud Foundry, and the buildpack is cloned from the repository and applied to the app.

If a buildpack is specified using `cf push -b`, the `detect` step is skipped. As a result, no buildpack `detect` scripts are run.

Disable Custom Buildpacks

Operators can choose to disable custom buildpacks. For more information, see the Disabling Custom Buildpacks section of the *Managing Custom Buildpacks* topic.

A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. To merge upstream patches to your custom buildpack, see Syncing a fork in the GitHub documentation.

Packaging dependencies for offline buildpacks

Learn about the dependency storage options that are available to you when creating offline buildpacks.

About offline buildpacks

Online, or uncached, buildpacks require an internet connection to download dependencies. For example, language interpreters, and compilers. Alternatively, you can create offline, or cached, buildpacks that are packaged with their dependencies. These offline buildpacks do not connect to the Internet when they are used to deploy Cloud Foundry apps.

Offline buildpacks might contain proprietary dependencies that require distribution licensing or export control measures.

You can find instructions for building offline packages in the `README.md` file for each buildpack repository. For example, see the Java buildpack.

Packaging dependencies in the buildpack

A simple way to package dependencies for a custom buildpack is to keep the dependencies in your buildpack source. However, this is not recommended. Keeping the dependencies in your source consumes unnecessary space.

To avoid keeping the dependencies in source control, load the dependencies into your buildpack, and provide a script for the operator to create a zipfile of the buildpack.

For example, you might complete the following process:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Creates a zipfile using your script
$ ./SomeScriptName
 ----> downloading-dependencies.... done
 ----> creating zipfile: ZippedBuildPackName.zip

$ # Adds the buildpack zipfile to the Cloud Foundry instance
$ cf create-buildpack SomeBuildPackName ZippedBuildPackName.zip 1
```

Pros

- Least complicated process for operators.
- Least complicated maintenance process for buildpack developers.

Cons

- Cloud Foundry admin buildpack uploads are limited to 1 GB, so the dependencies might not fit.
- Security and functional patches to dependencies require updating the buildpack.

Packaging selected dependencies in the buildpack

This is a variant of the package dependencies in the buildpack method. In this variation, the administrator edits a configuration file. For example, the `dependencies.yml` file to include a limited subset of the buildpack dependencies, packages, and uploads to the buildpack.

This approach is not recommended. See the Cons section for more information.

The administrator completes the following steps:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Selects dependencies
$ vi dependencies.yml # Or copy in a preferred config

$ # Builds a package using your script
$ ./package
 ----> downloading-dependencies.... done
 ----> creating zipfile: cobol_buildpack.zip

$ # Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
```

Pros

- It's possible to avoid the Cloud Foundry admin buildpack upload size limit in the following ways:
    - The administrator chooses a limited subset of dependencies.
    - The administrator maintains different packages for different dependency sets.

Cons

- More complex for buildpack maintainers.
- Security updates to dependencies require updating the buildpack.
- Proliferation of buildpacks require maintenance:
    - For each configuration, there is an update required for each security patch.
    - Culling orphan configurations can be difficult or impossible.
    - Administrators need to track configurations and merge them with updates to the buildpack.
    - Might result in a different configuration for each app.

Relying on a local mirror

In this method, the administrator provides a compatible file store of dependencies. When running the buildpack, the administrator specifies the location of the file store.

The administrator completes the following process:

```
$ # Clones your buildpack
$ git clone http://YOUR-GITHUB-REPOSITORY.example.com/repository
$ cd SomeBuildPackName

$ # Builds a package using your script
$ ./package https:///dependency/repository
 ----> creating zipfile: cobol_buildpack.zip

$ # Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
 ----> deploying app
 ----> downloading dependencies:
    https://OUR-INTERNAL-SITE.example.com/dependency/repository/dep1.tgz....
done
    https://OUR-INTERNAL-SITE.example.com/dependency/repository/dep2.tgz....
WARNING: dependency not found!
```

Pros

- Avoids the Cloud Foundry admin buildpack upload size limit.
- Leaves the administrator completely in control of providing dependencies.
- Security and functional patches for dependencies can be maintained separately on the mirror given the following conditions:
    - The buildpack is designed to use newer semantically versioned dependencies.
    - Buildpack behavior does not change with the newer functional changes.

Cons

- The administrator needs to set up and maintain a mirror.
- The additional config option presents a maintenance burden.

Merging with upstream buildpacks

Learn how to maintain your forked buildpack by merging it with the upstream Cloud Foundry buildpack. This process keeps your fork updated with changes from the original buildpack, providing patches, updates, and new features.

The following procedure assumes that you are maintaining a custom buildpack that was forked from a Cloud Foundry system buildpack. However, you can use the same procedure to update a buildpack forked from any upstream buildpack.

To sync your forked buildpack with an upstream Cloud Foundry buildpack:

1. Go to your forked repository on GitHub and click **Compare**. The **Comparing changes** page shows the unmerged commits between your forked buildpack and the upstream buildpack. Inspect unmerged commits and confirm that you want to merge them all.
2. Go to the forked repository and set the upstream remote as the Cloud Foundry buildpack repository.

   ```
   $ cd ~/workspace/ruby-buildpack
   $ git remote add upstream git@github.com:cloudfoundry/ruby-buildpack.git
   ```

3. Pull down the remote upstream changes.

   ```
   $ git fetch upstream
   ```

4. Merge the upstream changes into the intended branch. You might need to resolve merge conflicts. This example shows the merging of the `main` branch of the upstream buildpack into the `main` branch of the forked buildpack.

   ```
   $ git checkout main
   $ git merge upstream/main
   ```

   > **Important**
   >
   > When you merge upstream buildpacks, do not use `git rebase`. This approach is not sustainable because you confront the same merge conflicts repeatedly.

5. Run the buildpack test suite to ensure that the upstream changes do not break anything.

   ```
   $ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
   ```

6. Push the updated branch.

   ```
   $ git push
   ```

Your forked buildpack is now synced with the upstream Cloud Foundry buildpack.

For more information about syncing forks, see Syncing a fork.

Upgrading dependency versions for Cloud Foundry

This topic tells you how to upgrade a dependency version in a custom buildpack. These procedures enable Cloud Foundry (CF) operators to maintain custom buildpacks that contain dependencies outside of the dependencies in the CF system buildpacks.

Cloud Foundry buildpacks team process

The procedures in this topic refer to the tools used by the CF buildpacks team, but they do not require the following specific tools. You can use any continuous integration (CI) system and workflow management tool to update dependencies in custom buildpacks.

The CF buildpacks team uses the following tools to update dependencies:

- Concourse deployment of the buildpacks-ci pipelines.
- public-buildpacks-ci-robots GitHub repository.
- Pivotal Tracker for workflow management.

When the New Releases job in the notifications pipeline detects a new version of a tracked dependency in a buildpack, it creates a Tracker story about building and including the new version of the dependency in the buildpack manifests. It also posts a message as the `dependency-notifier` to the #buildpacks channel in the Cloud Foundry Slack channel.

Building binaries

For all dependencies, you must build the binary from source or acquire the binary as a tarball from a trusted source. For most dependencies, the CF buildpacks team builds the binaries from source.

The following steps assume you are using a Concourse deployment of the `buildpacks-ci` pipelines and Pivotal Tracker.

To build the binary for a dependency:

1. Go to the `public-buildpacks-ci-robots` directory and verify no uncommitted changes exist.

   ```
   $ cd ~/workspace/public-buildpacks-ci-robots
   $ git status
   ```

2. Run the `git pull` command in the directory to ensure it contains the most recent version of the contents.

   ```
   $ git pull -r
   ```

3. Go to the `binary-builds` directory.

   ```
   $ cd binary-builds
   ```

4. Locate the YAML file for the buildpack that you want to build a binary. The directory contains YAML files for all the packages and dependencies tracked by the CF buildpacks team. Each YAML file correlates to the build queue for one dependency or package, and uses the naming format `DEPENDENCY-NAME.yml`. For example, the YAML file tracking the build queue for Ruby is named `ruby-builds.yml` and contains the following contents:

   ```
   ---
   ruby: []
   ```

5. Different buildpacks use different signatures for verification. Determine which signature your buildpack requires by consulting the list in the buildpacks section of this topic and follow the instructions to locate the SHA256, MD5, or GPG signature for the binary:

6. For the SHA256 of a file, run `shasum -a 256 FILE-NAME`.
7. For the MD5 of a file, run `md5 FILE-NAME`.
8. For the GPG signature (for Nginx), see the Nginx Downloads page.
9. Add the version and verification for the new binary to the YAML file as attributes of an element under the dependency name. For example, to build the Ruby 2.3.0 binary verified with SHA256, add the following to the YAML file:

   ```
   ---
   ruby:
   - version: 2.3.0
     sha256: ba5ba60e5f1aa21b4ef8e9bf35b9ddb57286cb546aac4b5a28c71f459467e507
   ```

   Important

   Do not preface the version number with the name of the binary or language. For example, specify `2.3.0` for `version` instead of `ruby-2.3.0`.

   You can enqueue builds for multiple versions at the same time. For example, to build both the Ruby 2.3.0 binary and the Ruby 2.3.1 binary, add the following to the YAML file:

   ```
   ---
   ruby:
   - version: 2.3.0
     sha256: ba5ba60e5f1aa21b4ef8e9bf35b9ddb57286cb546aac4b5a28c71f459467e507
   - version: 2.3.1
     sha256: b87c738cb2032bf4920fef8e3864dc5cf8eae9d89d8d523ce0236945c5797dcd
   ```

10. Use the `git add` command to stage your changes:

    ```
    $ git add .
    ```

11. Use the `git commit -m "YOUR-COMMIT-MESSAGE [#STORY-NUMBER]"` command to commit your changes using the Tracker story number. Replace `YOUR-COMMIT-MESSAGE` with your commit message, and `STORY-NUMBER` with the number of your Tracker story.

    ```
    $ git commit -m "make that change [#1234567890]"
    ```

12. Run `git push` to push your changes to the remote origin.

    ```
    $ git push
    ```

13. Pushing your changes initiates the binary building process, which you can monitor at the binary-builder pipeline of your own `buildpacks-ci` Concourse deployment. When the build completes, it adds a link to the Concourse build run to the Tracker story specified in the commit message for the new release.

    Binary builds are run by the Cloud Foundry Binary Builder and the dependency-builds pipeline.

Updating buildpack manifests

After you build the binary for a dependency that you want to access and download from a URL, follow the instructions to add the dependency version to the buildpack manifest

The following steps assume you are using a Concourse deployment of the `buildpacks-ci` pipelines and Pivotal Tracker.

To add the dependency version to the buildpack manifest, use these steps:

1. Go to the directory of the buildpack for which you want to update dependencies and run `git checkout develop` to check out the `develop` branch.

   ```
   $ cd ~/workspace/ruby-buildpack
   $ git checkout develop
   ```

2. Edit the `manifest.yml` file for the buildpack to add or remove dependencies.

   ```
   dependencies:
     - name: ruby
       version: 2.3.0
       md5: 535342030a11abeb11497824bf642bf2
       uri: https://pivotal-buildpacks.s3.amazonaws.com/concourse-binaries/ruby/ruby-2.3.0-linux-x64.tgz
       cf_stacks:
         - cflinuxfs4
   ```

   - Follow the current structure of the manifest. For example, if the manifest includes the two most recent patch versions for each minor version of the language, you can also include the two most recent patch versions for each minor version of the language, such as both `ruby-2.1.9` and `ruby-2.1.8`.

   - Copy the `uri` and the `md5` from the `build-BINARY-NAME` job that ran in the Concourse dependency-builds pipeline and add them to the manifest.

   In the PHP buildpack, you might see a `modules` line for each PHP dependency in the manifest. Do not include this `modules` line in your new PHP dependency entry. The `modules` line is added to the manifest by the `ensure-manifest-has-modules` Concourse job in the `php-buildpack` when you commit and push your changes. You can see this in the output logs of the `build-out` task.

3. Replace any other mentions of the old version number in the buildpack repository with the new version number. The CF buildpack team uses Ag for text searching.

   ```
   $ ag OLD-VERSION
   ```

4. Run the following command to package and upload the buildpack, set up the org and space for tests in the specified CF deployment, and run the CF buildpack tests:

   ```
   $ BUNDLE_GEMFILE=cf.Gemfile buildpack-build
   ```

   If the command fails, you might need to fix or change the tests, fixtures, or other parts of the buildpack.

5. Once the test suite completely passes, use git commands to stage, commit, and push your changes:

   ```
   $ git add .
   $ git commit -m "YOUR-MESSAGE[#TRACKER-STORY-ID]"
   $ git push
   ```

6. Monitor the `LANGUAGE-buildpack` pipeline in Concourse. Once the test suite builds, the `specs-lts-develop` job and `specs-edge-develop` job, pass for the buildpack, you can deliver the Tracker story for the new Dependency release. Copy and paste links for the successful test suite builds into the Tracker story.

Buildpacks

The following list contains information about the buildpacks maintained by the CF buildpacks team:

Go:

- **Built from**: A tarred binary, `GO-VERSION.linux-amd64.tar.gz`, provided by Google on the Go Downloads page.
- **Verified with**: The MD5 of the tarred binary.
- **Example**: Using the Google Tarred Binary for Go 1.6.2.

Godep:

- **Built from**: A source code `.tar.gz` file from the Godep GitHub releases page.
- **Verified with**: The SHA256 of the source **Example**: Automated enqueuing of binary build for Godep 72.

  The buildpacks-ci dependency-builds pipeline automates the process of detecting, uploading, and updating Godep in the manifest.

Node.js buildpack

Node:

- **Verified with**: The SHA256 of the `node-vVERSION.tar.gz` file listed on `https://nodejs.org/dist/vVERSION/SHASUMS256.txt` For example, for Node version 4.4.6, the CF buildpacks team verifies with the SHA256 for `node-v4.4.6.tar.gz` on its SHASUMS256 page.
- **Example**: Enqueuing binary builds for Node 4.4.5 and 6.2.0.

Python buildpack

Python:

- **Verified with**: The MD5 of the `Gzipped source tarball`, listed on `https://www.python.org/downloads/release/python-VERSION/`, where `VERSION` has no periods. For example, for Python version `2.7.12`, use the MD5 for the `Gzipped source tarball` on its downloads page.
- **Example**: Enqueuing binary build for Python 2.7.12.

Java buildpack

OpenJDK:

- **Built from**: The tarred OpenJDK files managed by the CF Java Buildpack team.
- **Verified with**: The MD5 of the tarred OpenJDK files.

Ruby buildpack

JRuby:

- **Verified with**: The MD5 of the Source `.tar.gz` file from the JRuby Downloads page.
- **Example**: Enqueuing binary build for JRuby 9.1.2.0.

Ruby:

- **Verified with**: The SHA256 of the source from the Ruby Downloads page.
- **Example**: Enqueuing binary builds for Ruby 2.2.5 and 2.3.1.

Bundler:

- **Verified with**: The SHA256 of the `.gem` file from Rubygems.
- **Example**: Enqueuing binary build for Bundler 1.12.5.

PHP buildpack

PHP:

- **Verified with**: The SHA256 of the `.tar.gz` file from the PHP Downloads page.
- To enqueue builds for PHP, you need to edit a file in the `public-buildpacks-ci-robots` repository. For PHP5 versions, the CF buildpacks team enqueues builds in the `binary-builds/php-builds.yml` file. For PHP7 versions, the CF buildpacks team enqueues builds in the `binary-builds/php7-builds.yml` file.
- **Example**: Enqueuing binary builds for PHP 7.2.5 and 7.0.30.

Nginx:

- **Verified with**: The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` is different. This information is located on the Nginx Downloads page.
- **Example**: Enqueuing binary build for Nginx 1.11.0.

HTTPD:

- **Verified with**: The MD5 of the `.tar.bz2` file from the HTTPD Downloads page.
- **Example**: Enqueuing binary build for HTTPD 2.4.20.

Composer:

- **Verified with**: The SHA256 of the `composer.phar` file from the Composer Downloads page.
- For Composer, there is no build process as the `composer.phar` file is the binary. In the manual process, connect to the appropriate S3 bucket using the correct AWS credentials.

Create a new directory with the name of the composer version, for example `1.0.2`, and put the appropriate `composer.phar` file into that directory.

For Composer `v1.0.2`, connect and create the `php/binaries/trusty/composer/1.0.2` directory. Then place the `composer.phar` file into that directory so the binary is available at `php/binaries/trusty/composer/1.0.2/composer.phar`.

The buildpacks-ci dependency-builds pipeline automates the process of detecting, uploading, and updating Composer in the manifest.

- **Example**: Automated enqueuing of binary build for Composer 1.1.2.

Staticfile buildpack

Nginx:

- **Verified with**: The `gpg-rsa-key-id` and `gpg-signature` of the version. The `gpg-rsa-key-id` is the same for each version/build, but the `gpg-signature` is different. This information is located on the Nginx Downloads page.
- **Example**: Enqueuing binary build for Nginx 1.11.0.

Binary buildpack

The Binary buildpack has no dependencies.

Releasing a new Cloud Foundry buildpack version

You can update and release a new version of a Cloud Foundry (CF) buildpack through the CF Buildpacks Team Concourse pipeline. Concourse is a continuous integration (CI) tool for software development teams. This process is used by the CF Buildpacks Team and other CF buildpack development teams. You can use this process as a model for using Concourse to build and release new versions of your own buildpacks.

The Concourse pipelines for Cloud Foundry buildpacks are located in the buildpacks-ci GitHub repository.

Releasing a new buildpack version

To release a new buildpack version, do the following:

1. Download the `buildpacks-ci` repository:

   ```
   $ git clone https://github.com/cloudfoundry/buildpacks-ci.git
   ```

2. From the buildpack directory, check out the `develop` branch of the buildpack:

   ```
   $ cd /system/path/to/buildpack
   $ git checkout develop
   ```

3. Make sure you have the most current version of the repository:

   ```
   $ git pull -r
   ```

4. Run `bump` to update the version in the buildpack repository:

   ```
   $ /system/path/to/buildpacks-ci/scripts/bump
   ```

5. Modify the `CHANGELOG` file manually to condense recent commits into relevant changes. For more information, see Modify Changelogs.

6. Add and commit your changes:

   ```
   $ git add VERSION CHANGELOG
   $ git commit -m "Bump version to $(cat VERSION) [{insert story #}]"
   ```

7. Push your changes to the `develop` branch:

   ```
   $ git push origin develop
   ```

Concourse buildpack workflow

If `buildpacks-ci` is not deployed to Concourse, manually add a Git tag to the buildpack, and mark the tag as a release on GitHub.

If `buildpacks-ci` is deployed to Concourse, the buildpack update passes through the following life cycle:

1. Concourse starts the `buildpack-to-master` job in the pipeline for the updated buildpack. This job merges into the master or main branch of the buildpack.

2. The `detect-new-buildpack-and-upload-artifacts` job starts in the pipeline for the updated buildpack. This job creates a cached and uncached buildpack, and uploads them to an AWS S3 bucket.

3. The `specs-lts-master` and `specs-edge-master` jobs start and run the buildpack test suite, and the buildpack-specific tests of the Buildpack Runtime Acceptance Tests (BRATS).

4. If you use Pivotal Tracker, paste the links for the `specs-edge-master` and `specs-lts-master` builds in the related buildpack release story, and deliver the story.

5. Your project manager can manually start the `buildpack-to-github` job on Concourse as part of the acceptance process. This releases the buildpack to GitHub.

6. After the buildpack has been released to GitHub, the `cf-release` pipeline is started using the manual initiation of the `recreate-bosh-lite` job in that pipeline. If the new buildpack has been released to GitHub, the CF that is deployed for testing in the `cf-release` pipeline is tested against that new buildpack.

7. After the `cats` job has successfully completed, your project manager can include the new buildpacks in the `cf-release` repository and create the new buildpack BOSH release by manually starting the `ship-it` job.

If errors occur during this workflow, you might need to remove unwanted tags. For more information, see Handle Unwanted Tags.

Modifying changelogs

The Ruby Buildpack changelog shows an example layout, and content of a changelog. In general, changelogs follow these conventions:

- Reference public tracker stories whenever possible.
- Exclude unnecessary files.
- Combine and condense commit statements into individual stories containing valuable changes.

Handling unwanted tags

If you encounter problems with the commit that contains the new version, change the target of the release tag by performing the following:

1. Verify that the repository is in a valid state and is building successfully.
2. Remove the tag from your local repository and from GitHub.
3. Start a build. The pipeline build script tags the build again if it is successful.

Updating buildpack related gems in Cloud Foundry

Learn how to update your buildpack-packager and machete CF buildpack test framework, which are used for CF system buildpack development.

The `buildpack-packager` packages buildpacks and `machete` provides an integration test framework.

The CF Buildpacks team uses the gems-and-extensions pipeline to:

- Run integration tests for `buildpack-packager` and `machete`.
- Update the gems in the buildpacks managed by the team.

Running the update process

The following steps assume you are using a Concourse deployment of the `buildpacks-ci` pipelines.

At the end of the process, there is a new GitHub release. Updates are then applied to the buildpacks.

To update the version of either gem in a buildpack:

1. Verify that the test job `<gemname>-specs` for the gem was updated successfully and ran on the commit you plan to update.
2. Start the `<gemname>-tag` job to update (“bump”) the version of the gem.

   The `<gemname>-release` job starts and creates a new GitHub release of the gem.
3. Each of the buildpack pipelines, for example, the [go-buildpack pipeline, (https://buildpacks.ci.cf-app.com/teams/main/pipelines/go-buildpack)) has a job that watches for new releases of the gem. When a new release is detected, the buildpack’s `cf.Gemfile` is updated to that release version.
4. The commit made to the buildpack’s `cf.Gemfile` starts the full integration test suite for that buildpack.

The final step starts all buildpack test suites simultaneously, and causes contention for available shared BOSH lite test environments.

Deploying Apps

In this section:

Pushing Apps to TAS for VMs

In this section:

- Deploying a Large App
- Starting, Restarting, and Restaging Apps
- Pushing an App with Multiple Processes
- Running cf push Sub-Step Commands
- Rolling App Deployments
- Pushing Apps with Sidecar Processes
- Using Blue-Green Deployment to Reduce Downtime and Risk
- Troubleshooting App Deployment and Health

Pushing your app with Cloud Foundry CLI (cf push)

The cf CLI command `cf push` pushes apps to TAS for VMs. There are two main ways to run the `cf push` command:

- Run `cf push APP-NAME` to push an app the easiest way, using default settings.
- Run the `cf push` command with flags and helper files to customize:
  - How the pushed app runs, including its route (URL), instance count, and memory limits.
  - How the push process works: whether it's configured with a manifest, runs a startup script, or limits files uploaded to the Cloud Controller.

The Push with Defaults and Push with Custom Settings sections below detail both of these options.

For an explanation of what TAS for VMs does when you run `cf push`, see How Apps are Staged.

For information about the lifecycle of an app, see App Container Lifecycle

Prerequisites

Before you push your app to TAS for VMs, make sure that:

- Your app is *cloud-ready*. TAS for VMs behaviors related to file storage, HTTP sessions, and port usage may require modifications to your app. For help preparing your app to be pushed to TAS for VMs, see:
  - Considerations for Designing and Running an App in the Cloud
  - Any Buildpacks guides specific to your app language or framework, such as Getting Started Deploying Ruby on Rails Apps
- Your TAS for VMs installation supports the type of app you are going to push, or you have the URL of an externally-available buildpack that can stage the app.
- All required app resources are uploaded. For example, you may need to include a database driver.
- You have your target and credentials:
  - The API endpoint for your TAS for VMs installation. Also known as the target URL, this is the URL of the Cloud Controller in your TAS for VMs instance.

    - Your username and password for your TAS for VMs installation.

- You are logged into your app’s target org and space.
    1. Decide the org and space where you want to push your app. You may have access to one or more org and space.
    2. Log into this target org and space with `cf login`.
- Your app can access every service that it uses, because an instance of the service runs in, or is shared with, the app’s space.
    - See Sharing Service Instances for how to share service instances across spaces.
    - Typical services that cloud apps use include databases, message queues, and key-value stores.

Push with defaults

To push an app with default settings, do the following:

1. Choose a name for the app.
    - The app name must consist of alphanumeric characters.
    - The app name must be unique to your TAS for VMs installation.
        - To use an app name that is not unique, customize the app’s route as described in Customize the Route below.
        - Apps running at their default routes require unique names because default routes are based on app names, and all routes must be globally unique.
2. Run the following command:

    ```
    cf push APP-NAME
    ```

    Where `APP-NAME` is the name of the app.

Default route

An app’s route is the URL that it runs at. TAS for VMs assembles the route for a pushed app from a hostname and a domain.

By default, TAS for VMs sets the hostname and domain as follows:

- **Hostname**: The name of the app, as specified in the `cf push` command.
    - If the app name contains underscores, TAS for VMs converts them to hyphens when creating the app’s route.
- **Domain**: The default apps domain for the TAS for VMs installation.

For example, an app named `my-app-1234` running on TAS for VMs with an apps domain `apps.example.com` would, by default, run at the URL `https://my-app-1234.apps.example.com`.

For more information about routes and domains, see Routes and Domains.

Example session

The following session illustrates how TAS for VMs assigns default values to app when given a `cf push` command.

```
        $ cf push my-app-1234
        Creating app my-app-1234 in org example-org / space development as a.
user@shared-domain.example.com...
        OK

        Creating route my-app-1234.shared-domain.example.com...
        OK

        Binding my-app-1234.shared-domain.example.com to my-app-1234...
        OK

        Uploading my-app-1234...
        Uploading app: 560.1K, 9 files
        OK

        Starting app my-app-1234 in org example-org / space development as a.
user@shared-domain.example.com...
        -----> Downloaded app package (552K)
        OK
        -----> Using Ruby version: ruby-1.9.3
        -----> Installing dependencies using Bundler version 1.3.2
               Running: bundle install --without development:test --path
                 vendor/bundle --binstubs vendor/bundle/bin --deployment
               Installing rack (1.5.1)
               Installing rack-protection (1.3.2)
               Installing tilt (1.3.3)
               Installing sinatra (1.3.4)
               Using bundler (1.3.2)
               Updating files in vendor/cache
               Your bundle is complete! It was installed into ./vendor/bundle
               Cleaning up the bundler cache.
        -----> Uploading droplet (23M)

        1 of 1 instances running

        App started

        Showing health and status for app my-app-1234 in org example-org / sp
ace development as a.user@shared-domain.example.com...
        OK

        requested state: started
        instances: 1/1
        usage: 1G x 1 instances
        urls: my-app-1234.shared-domain.example.com

             state     since                    cpu    memory        disk
        #0   running   2014-01-24 05:07:18 PM   0.0%   18.5M of 1G   52.5M of
```

```
1G
```

Push with custom settings

Pushing an app with with custom settings typically proceeds as follows:

1. (Optional) Customize Basic App Settings
2. (Optional) Customize the Route
3. (Optional) Limit the Upload Files
4. (Optional) Configure App Initialization
5. Custom Push the App
6. (Optional) Configure App Services

The sections below detail these steps.

(Optional) Customize basic app settings

Basic settings to customize when pushing an app include:

- **Name**: You can use any series of alphanumeric characters as the name of your app.
- **Instances**: Generally speaking, the more app instances you run, the less downtime your app will experience. If your app is still in development, running a single instance can simplify troubleshooting. For any production app, VMware recommends a minimum of two instances.
- **Memory Limit**: The maximum amount of memory that each instance of your app can consume. If an instance exceeds this limit, TAS for VMs restarts the instance. If an instance exceeds its memory limit repeatedly in a short period of time, TAS for VMs delays restarting the instance.
- **Start Command**: This is the command that TAS for VMs uses to start each instance of your app. This start command differs by app framework.

(Optional) customize the route

To customize an app's route, do the following:

1. **(Optional) Customize the Hostname**
    - Pass the custom hostname into `cf push` with the `-n` flag.
2. **(Optional) Customize the Domain**
    - Pass the custom domain into `cf push` with the `-d` flag. The custom domain must be registered, and mapped to the org that contains the app's target space
3. **Ensure Route Uniqueness**
    - The app's route must be globally unique, whether you customize its host or domain, or let it use the Default Route described above.

  - To help ensure route uniqueness, pass the `--random-route` option into `cf push`. `--random-route` creates a route that includes the app name and random words

(Optional) Limit the upload files

By default, TAS for VMs uploads all app files except version control files and folders with names such as `.svn`, `.git`, and `_darcs`.

VMware recommends that you explicitly exclude extraneous files residing within your app directory, particularly if your app is large. For example, if you build your app locally and push it as a binary, you can save resources by not uploading any of the app's source code.

To exclude files from upload:

1. Create a `.cfignore` file that lists the files to exclude.
2. Save the `.cfignore` file to the directory where you run the `cf push` command.

For more information, see the Ignore Unnecessary Files When Pushing section of the Considerations for Designing and Running an App in the Cloud topic.

(Optional) Configure app initialization

You can configure `cf push` to run custom initialization tasks for an app.

These tasks run after TAS for VMs loads the app droplet but before it launches the app itself to let the initialization script access the app language runtime environment. For example, your script can map values from `$VCAP_SERVICES` into other environment variables or a config file that the app uses.

Note: See below for important information about configuring app initialization when you use certain buildpacks:

- **Java:** Initialization scripts for the Java buildpack require additional configuration. For more information, see How to Modify the Application Container Environment prior to Application Execution in the Knowledge Base.
- **PHP:** TAS for VMs does not support initialization scripts for the PHP buildpack versions prior to v4.3.18. If you use one of these buildpack versions, your app hosts the `.profile` script's contents. This means that any app staged using the affected buildpack versions can leak credentials placed in the `.profile` script.

To run initialization tasks:

1. Create a `.profile` script that contains the initialization tasks.
2. Save the `.profile` script to the directory where you run the `cf push` command.

As an example, the following `.profile` file uses `bash` to set a value for the environment variable `LANG`. Setting this value at the operating system level lets the app determine which language to use for error messages and instructions, collating sequences, and date formats:

```
# Set the default LANG for your apps
export LANG=en_US.UTF-8
```

Your app root directory may also include a `.profile.d` directory that contains bash scripts that perform initialization tasks for the buildpack. Developers should not edit these scripts unless they are using a custom buildpack.

Initialization tasks as described here are also called “pre-runtime hooks” and “`.profile` tasks”.

Custom push the app

To specify custom options when pushing an app with `cf push`, you can:

- Include them in the `cf push APP-NAME` command itself.
- Include them in a manifest file.
    - The manifest file must be named `manifest.yml` and reside in the directory where you run `cf push`.
    - The manifest can include the app name, which lets you run `cf push` with no arguments.
    - The manifest can also include a Services block that lists service instances for the app to bind to automatically.
- Both of the above, command-line options and a manifest.

See Deploying with App Manifests to learn how app settings change from push to push, and how command-line options, manifests, and commands like `cf scale` interact.

See the Cloud Foundry CLI Reference Guide for a full list of `cf push` options.

(Optional) Configure app services

If a newly-pushed app has the same name and route as an older app version, the new app retains the service bindings and service configuration of the previously-pushed version.

For apps that are not already set up for the services that they use, you need to:

1. Bind the services to the app. For more information about services, see the Services Overview topic.
2. (Optional) Configure the app with the service URL and credentials, if needed. For more information, see Configuring Service Connections.

App updates and downtime

When you push an app that is already running, TAS for VMs stops all existing instances of that app. Users who try to access the app get a “404 Not Found” message while `cf push` runs.

With some app updates, old and new versions of your code should never run at the same time. A worst-case example is if your app update migrates a database schema, causing old app instances to fail and lose user data. To prevent this, you need to stop all running instances of your app before you push the new version.

When old and new versions of your app can run simultaneously, you can avoid app downtime by using the blue-green method to swap routes between app versions running in parallel.

Troubleshoot app push problems

If your app does not start on TAS for VMs, first ensure that the app can run locally.

For how to troubleshoot your app in the cloud using the cf CLI, see Troubleshoot App Deployment and Health.

Deploying with app manifests

App manifests provide consistency and reproducibility, and can help you automate deploying apps. This topic provides basic procedures and guidance for deploying apps with a manifest file to VMware Tanzu Application Service for VMs (TAS for VMs). Both manifests and command line options allow you to override the default attribute values of `cf push`. These attributes include the number of app instances, disk space limit, memory limit, and log rate limit.

`cf push` follows rules of precedence when setting attribute values:

- Manifests override most recent values, including defaults and values set by commands such as `cf scale`.
- Command line options override manifests.

For a full list of attributes you can specify in an app manifest, see App manifest attribute reference.

Deploy an app with a manifest

To deploy an app with a manifest:

1. Create a `manifest.yml` file in the root directory of your app.

   > **Note:** By default, the `cf push` command uses the `manifest.yml` file in the app directory. To specify a different location for the manifest, pass its local path to the `-f` flag when you run `cf push`.

2. Add the following content to the file:

   ```
   ---
   applications:
   - name: YOUR-APP
   ```

   Where `YOUR-APP` is the name of your app.

3. Run:

   ```
   cf push
   ```

   If you specify any values with command-line flags, they override the values specified in the manifest. For more information, see Deploy multiple apps with one manifest.

For more information about manifest format and attributes, see App manifest attribute reference.

Deploy multiple apps with one manifest

This section describes how to deploy multiple apps with a minimal manifest. For more information about manifest format and attributes, see App Manifest Attribute Reference.

General rules

Follow these general rules when deploying multiple apps with one manifest:

- Use a `no-route` line in the description of any app that provides background services to another app.
- You cannot use any command line options with `cf push` except for `-f` and `--no-start`.
    - If your manifest is not named `manifest.yml` or not in the current working directory, use the `-f` command line option.
- To push a single app rather than all of the apps described in the manifest, provide the desired app name by running `cf push YOUR-APP`.

Procedure

To deploy multiple apps with a manifest:

> **Note:** Each app must be in a subdirectory under the same parent directory.

1. Create a `manifest.yml` file in the directory that contains the apps.
2. Add each app and its directory to the file. VMware Tanzu Application Service for VMs pushes the apps in the order specified in the manifest.

   > **Note:** If you push multiple apps using a manifest and one fails to deploy, TAS for VMs does not attempt to push apps specified after the app that failed.

   ```
   ---
   applications:
   - name: APP-ONE
     path: ./APP-ONE-DIRECTORY
   - name: APP-TWO
     path: ./APP-TWO-DIRECTORY
   ```

   Where:

    - `APP-ONE` is the name of the first app you want TAS for VMs to push.
    - `APP-ONE-DIRECTORY` is the directory containing the first app.
    - `APP-TWO` is the name of the second app you want TAS for VMs to push.
    - `APP-TWO-DIRECTORY` is the directory containing the second app.
3. From the directory that contains the apps and the manifest, run:

   ```
   cf push
   ```

App Manifest attribute reference

App properties and behavior can be managed using cf CLI commands or the app manifest (a YAML properties file). This topic describes manifest formatting and provides a full list of attributes available for app manifests. You can use it alongside Deploying with App Manifests, which provides basic procedures and guidance for deploying apps with manifests.

For more information about V3 manifest properties, see the Cloud Foundry API (CAPI) V3 documentation.

Manifest format

Manifests are written in YAML. The following manifest illustrates some YAML conventions:

- The manifest begins with three dashes.
- The `version` property specifies a manifest schema version. This property is optional. For more information, see Add Schema Version to a Manifest below.
- The `applications` block begins with a heading followed by a colon.
- The app `name` is preceded by a single dash and one space.
- Subsequent lines in the block are indented two spaces to align with `name`.

```
---
version: 1
applications:
- name: my-app
  memory: 512M
  instances: 2
```

Note: If your app name begins with the dash character (`-`), you cannot interact with the app using the cf CLI. This is because the cf CLI interprets the dash as a flag.

Add schema version to a manifest

Note: This attribute is available with CAPI V3 only. To push a manifest that uses this attribute, do either of the following:

- Use cf CLI v7 or v8. See Upgrading to cf CLI v7 or Upgrading to cf CLI v8.
- Run `cf v3-push APP-NAME`. See v3-push - Cloud Foundry CLI.

You can specify the schema version in the `versions` property of the manifest. This property is optional.

The only supported version is `1`. If not specified, the `versions` property defaults to `1`.

Add variables to a manifest

You can use variables to create app manifests with values shared across all applicable environments in combination with references to environment-specific differences defined in separate files.

To add variables to an app manifest, do the following:

1. Create a file called `vars.yml`.
2. Add attributes to your `vars.yml` file. See the following example:

   ```
   instances: 2
   memory: 1G
   ```

3. Add the variables to your app manifest file using the following format: `((VARIABLE-NAME))`. See the following example:

   ```
   ---
   applications:
   - name: test-app
     instances: ((instances))
     memory: ((memory))
     buildpacks:
     - go_buildpack
     env:
       GOPACKAGENAME: go_calls_ruby
     command: go_calls_ruby
   ```

   Note: You can also use variables for partial values. For example, you can specify `host` in your variables file and `- route: ((host)).env.com` in your manifest file.

4. Run `cf push`:

   ```
   cf push --vars-file /PATH/vars.yml
   ```

   Where `PATH` is the path to the file you created.

Minimize duplication with YAML anchors

Note: Top-level attributes have been deprecated in favor of YAML anchors. For more information, see Deprecated App Manifest Features.

In manifests where multiple apps share settings or services, you may see duplicated content. While the manifests still work, duplication increases the risk of typographical errors, which cause deployments to fail.

You can declare shared configuration using a YAML anchor, which the manifest refers to in app declarations by using an alias.

```
---
defaults: &defaults
  buildpacks:
    - staticfile_buildpack
```

```
  memory: 1G


applications:
- name: bigapp
  <<: *defaults
- name: smallapp
  <<: *defaults
  memory: 256M
```

This manifest pushes two apps, smallapp and bigapp, with the staticfile buildpack but with 256M memory for smallapp and 1G for bigapp.

App attributes

Application Attributes

This section explains how to describe optional app attributes in manifests. Each of these attributes can also be specified by a command line option. Command line options override the manifest.

> **Note:** In cf CLI v6, the route component attributes `domain`, `domains`, `host`, `hosts`, and `no-hostname` have been deprecated in favor of the `routes` attribute. In cf CLI v7, these attributes are removed. For more information, see Deprecated App Manifest Features.

buildpacks

You can refer to a buildpack by name in a manifest or a command line option. The `cf buildpacks` command lists the buildpacks that you can use.

See below for information on referencing buildpacks in a manifest. The command line option that overrides this attribute is `-b`.

- **Custom buildpacks**: If your app requires a custom buildpack, you can use the `buildpacks` attribute to specify it in a number of ways:
  - By name: `MY-BUILDPACK`.
  - By GitHub URL: `https://github.com/cloudfoundry/java-buildpack.git`.
  - By GitHub URL with a branch or tag: `https://github.com/cloudfoundry/java-buildpack.git#v3.3.0` for the `v3.3.0` tag.

    ```
    ---
    ...
    buildpacks:
      - buildpack_URL
    ```

- **Multiple buildpacks**: If you are using multiple buildpacks, you can provide an additional `-b` flag or add an additional value to your manifest:

  ```
  ---
    ...
  ```

```
  buildpacks:
    - buildpack_URL
    - buildpack_URL
```

Note: This feature does not work with Deprecated App Manifest Features.

Note: You must specify multiple buildpacks in the correct order: the buildpack will use the app start command given by the final buildpack. See the multi-buildpack repository for more information.

Also see Pushing an App with Multiple Buildpacks for more information.

command

Some languages and frameworks require that you provide a custom command to start an app. Refer to the buildpack documentation to determine if you need to provide a custom start command.

You can provide the custom start command in your app manifest or on the command line. See Starting, Restarting, and Restaging Apps for more information about how Cloud Foundry determines its default start command.

To specify the custom start command in your app manifest, add it in the `command: START-COMMAND` format as the following example shows:

```
---
  ...
  command: bundle exec rake VERBOSE=true
```

The start command you specify becomes the default for your app. To return to using the original default start command set by your buildpack, you must explicitly set the attribute to `null` as follows:

```
---
  ...
  command: null
```

On the command line, use the `-c` option to specify the custom start command as the following example shows:

```
$ cf push my-app -c "bundle exec rake VERBOSE=true"
```

Note: The `-c` option with a value of 'null' forces `cf push` to use the buildpack start command. See Forcing cf push to use the Buildpack Start Command for more information.

If you override the start command for a Buildpack app, Linux uses `bash -c COMMAND` to invoke your app. If you override the start command for a Docker app, Linux uses `sh -c COMMAND` to invoke your app. Because of this, if you override a start command, you should prefix `exec` to the final command in your custom composite start command.

An app needs to catch termination signals and clean itself up appropriately. Because of the way that shells manage process trees, the use of custom composite shell commands, particularly those that create child processes using `&`, `&&`, `||`, etc., can prevent your app from receiving signals that are sent to the top level bash process.

To resolve this issue, you can use `exec` to replace the bash process with your own process. For example:

- `bin/rake cf:on_first_instance db:migrate && bin/rails server -p $PORT -e $RAILS_ENV` The process tree is bash -> ruby, so on graceful shutdown only the bash process receives the TERM signal, not the ruby process.
- `bin/rake cf:on_first_instance db:migrate && exec bin/rails server -p $PORT -e $RAILS_ENV` Because of the `exec` prefix included on the final command, the ruby process invoked by rails takes over the bash process managing the execution of the composite command. The process tree is only ruby, so the ruby web server receives the TERM signal and can shutdown gracefully for 10 seconds.

In more complex situations, like making a custom buildpack, you may want to use bash `trap`, `wait`, and backgrounded processes to manage your process tree and shut down apps gracefully. In most situations, however, a well-placed `exec` should be sufficient.

disk_quota

Use the `disk_quota` attribute to allocate the disk space for your app instance. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case.

```
---
  ...
  disk_quota: 1024M
```

The command line option that overrides this attribute is `-k`.

docker

If your app is contained in a Docker image, then you may use the `docker` attribute to specify it and an optional Docker username.

This attribute is a combination of `push` options that include `--docker-image` and `--docker-username`.

```
---
  ...
  docker:
    image: docker-image-repository/docker-image-name
    username: docker-user-name
```

The command line option `--docker-image` or `-o` overrides `docker.image`. The command line option `--docker-username` overrides `docker.username`.

The manifest attribute `docker.username` is optional. If it is used, then the password must be provided in the environment variable `CF_DOCKER_PASSWORD`. Additionally, if a Docker username is specified, then a Docker image must also be specified.

Note: Using the docker attribute in conjunction with the buildpacks or path attributes will result in an error.

health-check-http-endpoint

Use the health-check-http-endpoint attribute to customize the endpoint for the http health check type. If you do not provide a health-check-http-endpoint attribute, it uses endpoint /.

```
---
  ...
  health-check-type: http
  health-check-http-endpoint: /health
```

health-check-invocation-timeout

Use the health-check-invocation-timeout attribute to set the timeout in seconds for individual health check requests for HTTP and port health checks.

```
---
  ...
  health-check-invocation-timeout: 30
```

To override this attribute, run:

```
cf set-health-check APP-NAME http --invocation-timeout 10
```

Where APP-NAME is the name of your app.

Within the manifest, the health check timeout is controlled by the timeout attribute.

health-check-type

Use the health-check-type attribute to set the health_check_type flag to either port, process or http. If you do not provide a health-check-type attribute, it defaults to port.

```
---
  ...
  health-check-type: port
```

The command line option that overrides this attribute is -u.

In cf CLI v6, the value of none is deprecated in favor of process. In cf CLI v7, none is removed.

instances

Use the instances attribute to set the number of app instances.

```
---
  ...
  instances: 2
```

The default number of instances is 1.

To ensure that platform maintenance does not interrupt your app, run at least two instances.

memory

Use the `memory` attribute to specify the memory limit for all instances of an app. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case. For example:

```
---
  ...
  memory: 1024M
```

The default memory limit is 1G. You might want to specify a smaller limit to conserve quota space if you know that your app instances do not require 1G of memory.

The command line option that overrides this attribute is `-m`.

metadata

Use the `metadata` information to tag your apps with additional information. You can specify two types of metadata: `labels` and `annotations`. For more information, see Types of Metadata in *Using Metadata*.

See the following example:

```
metadata:
  annotations:
    contact: "bob@example.com jane@example.com"
  labels:
    sensitive: true
```

For more information about metadata, see Using Metadata.

no-route

Note: If you use the `no-route` flag attribute in the manifest or the flag option, it will override all route-related attributes.

By default, `cf push` assigns a route to every app. But, some apps process data while running in the background and should not be assigned routes.

You can use the `no-route` attribute with a value of `true` to prevent a route from being created for your app.

```
---
  ...
  no-route: true
```

The command line option that overrides this attribute is `--no-route`.

In the Diego architecture, `no-route` skips creating and binding a route for the app, but does not specify which type of health check to perform. If your app does not listen on a port because it is a worker or a scheduler app, then it does not satisfy the port-based health check, and TAS for VMs marks it as crashed. To prevent this, disable the port-based health check by running:

```
cf set-health-check APP-NAME process
```

Where `APP-NAME` is the name of your app.

To remove a route from an existing app, perform the following steps:

1. Remove the route using the `cf unmap-route` command.
2. Push the app again with the `no-route: true` attribute in the manifest or the `--no-route` command line option.

For more information, see Deploy Multiple Apps with One Manifest.

path

You can use the `path` attribute to tell Cloud Foundry the directory location where it can find your app.

The directory specified as the `path`, either as an attribute or as a parameter on the command line, becomes the location where the buildpack `Detect` script executes.

The command line option that overrides this attribute is `-p`.

```
---
  ...
  path: /path/to/app/bits
```

For more information, see How cf push Finds the App.

processes

> **Note**: This attribute is available with CAPI V3 only. To push a manifest that uses this attribute, do either of the following:
>
> - Use cf CLI v7 or v8. See Upgrading to cf CLI v7 or Upgrading to cf CLI v8.
> - Run `cf v3-push APP-NAME`. See v3-push - Cloud Foundry CLI.

Use the `processes` attribute to push apps that run multiple processes, such as a web app that has a UI process and a worker process. See the following example:

```
processes:
- type: web
  command: start-web.sh
  disk_quota: 512M
  health-check-http-endpoint: /healthcheck
  health-check-type: http
  health-check-invocation-timeout: 10
  instances: 3
  memory: 500M
  timeout: 10
- type: worker
  command: start-worker.sh
  disk_quota: 1G
  health-check-type: process
  instances: 2
  memory: 256M
  timeout: 15
```

For detailed information about the process-level configuration, see The app manifest specification in the CAPI documentation.

For more information about pushing an app with multiple processes, see Pushing an App with Multiple Processes.

random-route

If you push your app without specifying any route-related CLI options or app manifest flags, the cf CLI attempts to generate a route based on the app name, which can cause collisions.

You can use the `random-route` attribute to generate a unique route and avoid name collisions. When you use `random-route`, the cf CLI generates an HTTP route with a random host (if `host` is not set) or a TCP route with an unused port number.

See the following example use cases:

- You deploy the same app to multiple spaces for testing purposes. In this situation, you can use `random-route` to randomize routes declared with the route attribute in the app manifest.
- You use an app manifest for a classroom training exercise in which multiple users deploy the same app to the same space.

The command line option that overrides this attribute is `--random-route`.

```
---
  ...
  random-route: true
```

routes

Use the `routes` attribute to provide multiple HTTP and TCP routes. Each route for this app is created if it does not already exist.

This attribute is a combination of `push` options that include `--hostname`, `-d`, and `--route-path` that were used in cf CLI v6. These flags are not supported in cf CLI v7. You must use the `routes` property in the manifest from cf CLI v7 on.

Optionally, specify the `protocol` attribute to configure which network protocol the route uses for app ingress traffic. The available protocols are `http2`, `http1`, and `tcp`.

Note: The `protocol` route attribute is only available for TAS for VMs deployments with HTTP/2 routing enabled. For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.

Optionally, specify the `protocol` attribute to configure which network protocol the route uses for app ingress traffic. The available protocols are `http2`, `http1`, and `tcp`.

Note: The protocol route attribute is only available for TAS for VMs deployments with HTTP/2 routing enabled. For information about configuring support for HTTP/2 in TAS for VMs, see Configuring HTTP/2 Support.

```
---
  ...
  routes:
  - route: example.com
    protocol: http2
  - route: www.example.com/foo
  - route: tcp-example.com:1234
```

Manifest attributes

If you use the routes attribute in conjunction with the host, hosts, domain, domains, or no-hostname attributes, an error results.

Push flag options

This attribute has unique interactions with different command-line options. The table shows the cf push flags supported in cf CLI v6. In cf CLI v7, these cf push flags are no longer supported:

- --no-route
- -d1
- --hostname
- --route-path

Flag supported in cf CLI v6	Result
--no-route	All declared routes are ignored.
-d	Overrides the DOMAIN in all declared HTTP and TCP routes.
--hostname, -n	Sets or overrides the HOSTNAME in all HTTP routes. Does not impact TCP routes.
--route-path	Sets or overrides the PATH in all HTTP routes. Does not impact TCP routes.
--random-route	Sets or overrides the HOSTNAME in all HTTP routes. Sets or overrides the PORT in all TCP routes. The PORT and HOSTNAME are randomly generated.

sidecars

Note: This attribute is available with CAPI V3 only. To push a manifest that uses this attribute, do either of the following:

- Use cf CLI v7 or v8. See Upgrading to cf CLI v7 or Upgrading to cf CLI v8.
- Run cf v3-push APP-NAME. See v3-push - Cloud Foundry CLI.

Use the `sidecars` attribute to specify additional processes to run in the same container as your app. Each sidecar must have values for `name`, `process_types`, and `command`, whereas `memory` is optional. See the following example:

```
sidecars:
- name: authenticator
  process_types: [ 'web', 'worker' ]
  command: bundle exec run-authenticator
  memory: 800M
- name: upcaser
  process_types: [ 'worker' ]
  command: ./tr-server
  memory: 900M
```

For more information about sidecars, see Pushing Apps with Sidecar Processes.

stack

Use the `stack` attribute to specify which stack to deploy your app to.

To see a list of available stacks, run `cf stacks` from the cf CLI.

```
---
  ...
  stack: cflinuxfs3
```

The command line option that overrides this attribute is `-s`.

timeout

The `timeout` attribute defines the number of seconds that Cloud Foundry allocates for starting your app. It is related to the `health-check-type` attribute.

For example:

```
---
  ...
  timeout: 80
```

You can increase the timeout length for very large apps that require more time to start. The `timeout` attribute defaults to 60, but you can set it to any value up to the Cloud Controller's `cc.maximum_health_check_timeout` property.

`cc.maximum_health_check_timeout` has a maximum value of `600` seconds.

The command line option that overrides the timeout attribute is `-t`.

> **Important:** If you configure `timeout` with a value greater than `cc.maximum_health_check_timeout`, the Cloud Controller reports a validation error with the maximum limit.

Environment variables

The `env` block consists of a heading, then one or more environment variable/value pairs.

For example:

```
---
  ...
  env:
    RAILS_ENV: production
    RACK_ENV: production
```

`cf push` deploys the app to a container on the server. The variables belong to the container environment.

Note: You must name variables with alphanumeric characters and underscores. Non-conforming variable names may cause unpredictable behavior.

Note: Do not use user-provided environment variables for security sensitive information such as credentials as they might unintentionally show up in cf cli output and Cloud Controller logs. Use user-provided service instances instead. The system-provided environment variable VCAP_SERVICES is properly redacted for user roles such as Space Supporter and in Cloud Controller log files.

- To view all variables, run:

  ```
  cf env APP-NAME
  ```

  Where `APP-NAME` is the name of your app.

- To set an individual variable, run:

  ```
  cf set-env APP-NAME VARIABLE-NAME VARIABLE-VALUE
  ```

  Where:

  - `APP-NAME` is the name of your app.
  - `VARIABLE-NAME` is the environment variable you want to set.
  - `VARIABLE-VALUE` is the value of the environment value.

- To unset an individual variable, run:

  ```
  cf unset-env APP-NAME VARIABLE-NAME VARIABLE-VALUE
  ```

  Where:

  - `APP-NAME` is the name of your app.
  - `VARIABLE-NAME` is the environment variable you want to set.
  - `VARIABLE-VALUE` is the value of the environment value.

Environment variables interact with manifests in the following ways:

- When you deploy an app for the first time, Cloud Foundry reads the variables described in the environment block of the manifest and adds them to the environment of the container where the app is staged, and the environment of the container where the app is deployed.
- When you stop and then restart an app, its environment variables persist.

Services

Apps can bind to services such as databases, messaging, and key-value stores.

Apps are deployed into App Spaces. An app can only bind to services instances that exist in the target App Space before the app is deployed.

The `services` block consists of a heading, then one or more service instance names.

Whoever creates the service chooses the service instance names. These names can convey logical information, as in `backend_queue`, describe the nature of the service, as in `mysql_5.x`, or do neither, as in the example below.

```
---
  ...
  services:
   - instance_ABC
   - instance_XYZ
```

Binding to a service instance is a special case of setting an environment variable, namely `VCAP_SERVICES`. See the Bind a Service section of the *Delivering Service Credentials to an App* topic.

Deprecated app manifest features

These app manifest features have been deprecated in favor of other options, as described below.

> ⚠ **Caution:** Running `cf push app -f manifest.yml` fails if your manifest uses any of these deprecated features along with the feature that replaces it.

Top-level attributes

Previously, you could declare top-level attributes, which are also known as global attributes. For example, you can move an attribute above the `applications` block, where it need appear only once.

The following example illustrates how this was used to manage duplicated settings.

```
---
# all apps use these settings and services
domain: shared-domain.example.com
memory: 1G
instances: 1
services:
- clockwork-mysql
applications:
- name: springtock
```

```
  host: tock09876
  path: ./spring-music/build/libs/spring-music.war
- name: springtick
  host: tick09875
  path: ./spring-music/build/libs/spring-music.war
```

Now, you can use YAML aliases instead.

The following example illustrates how to declare shared configuration using a YAML anchor, which the manifest refers to in app declarations by using an alias.

```
---
defaults: &defaults
  buildpacks:
    - staticfile_buildpack
  memory: 1G


applications:
- name: bigapp
  <<: *defaults
- name: smallapp
  <<: *defaults
  memory: 256M
```

When pushing the app, make explicit the attributes in each app's declaration. To do this, assign the anchors and include the app-level attributes with YAML aliases in each app declaration.

Attribute routes Replaces domain, domains, host, hosts, and no-hostname

Note: These properties are removed in cf CLI v7.

Previously, you could specify routes by listing them all at once using the `routes` attribute, or by using their hosts and domains as shown below.

```
---
applications:
- name: webapp
  host: www
  domains:
  - example.com
  - example.io
```

The following route component attributes have been deprecated:

- `domain`
- `domains`
- `host`
- `hosts`
- `no-hostname`

Now you can only specify routes by using the `routes` attribute:

```
---
applications:
- name: webapp
  routes:
  - route: www.example.com/foo
  - route: tcp.example.com:1234
```

This app manifest feature has been deprecated, and a replacement option is under consideration.

Inheritance

This feature has been deprecated, and has been replaced by Variable Substitution.

With inheritance, child manifests inherited configurations from a parent manifest, and the child manifests could use inherited configurations as provided, extend them, or override them. This feature has been deprecated, and has been replaced by Variable Substitution.

Buildpack Field in Manifest Is Deprecated

The singular `buildpack` field in manifests is deprecated. It has been replaced by `buildpacks`, which is now an array which takes as a value multiple buildpacks.

Deploying your app with docker

You can use the Cloud Foundry Command Line Interface (cf CLI) to push an app with a new or updated Docker image. VMware Tanzu Application Service for VMs (TAS for VMs) then uses the Docker image to create containers for the app.

For an explanation of how Docker works in TAS for VMs, see Using Docker in TAS for VMs.

Requirements

To push apps with Docker, you need:

- A TAS for VMs deployment that has Docker support enabled. To enable Docker support, see the Enable Docker section of the *Using Docker in TAS for VMs* topic.
- A Docker image that meets the following requirements:
    - The Docker image must contain an `/etc/passwd` file with an entry for the `root` user. In addition, the home directory and the shell for that `root` user must be present in the image file system.
    - The total size of the Docker image file system layers must not exceed the disk quota for the app. The maximum disk allocation for apps is set by the Cloud Controller. The default maximum disk quota is 2048 MB per app.

    > **Note:** If the total size of the Docker image file system layers exceeds the disk quota, the app instances do not start.

- The location of the Docker image on Docker Hub or another Docker registry.

- A registry that supports the Docker Registry HTTP API V2 and presents a valid certificate for HTTPS traffic. For more information, see the Docker Registry HTTP API V2 spec in the Docker documentation.

Requirement for cf ssh Support

If you want to log in to your app container using the `cf ssh` command, you must make a shell such as `sh` or `bash` available in the container.

The SSH server in the container looks for the following executables in absolute locations or the `PATH` environment variable:

- `/bin/bash`
- `/usr/local/bin/bash`
- `/bin/sh`
- `bash`
- `sh`

Benefits of Specifying Tags

If you want your app container to be consistent after platform updates and code changes, specify a tag when you push your Docker image. Otherwise, the platform applies the `latest` tag without respecting changes to `PORT` or `ENTRYPOINT`.

If you push your Docker image without specifying a tag, you must run `cf restage` for the changes to take effect.

Port Configuration

By default, apps listen for connections on the port specified in the `PORT` environment variable for the app. TAS for VMs allocates this value dynamically.

When configuring a Docker image for TAS for VMs, you can control the exposed port and the corresponding value of `PORT` by specifying the `EXPOSE` directive in the image Dockerfile. If you specify the `EXPOSE` directive, then the corresponding app pushed to TAS for VMs listens on that exposed port. For example, if you set `EXPOSE` to `7070`, then the app listens for connections on port 7070.

If you do not specify a port in the `EXPOSE` directive, then the app listens on the value of the `PORT` environment variable as determined by TAS for VMs.

If you set the `PORT` environment variable via an `ENV` directive in a Dockerfile, TAS for VMs overrides the value with the system-determined value.

TAS for VMs supports only one exposed port on the image.

For more information about the `PORT` environment variable, see the PORT section of the *TAS for VMs Environment Variables* topic. For more information about the `EXPOSE` directive, see the EXPOSE section of the *Dockerfile reference* topic in the Docker documentation.

Start Command

By default, Docker uses the start command specified in the Docker image. You can override the start command either by using a command-line parameter or by specifying it in a manifest file.

For more information about command-line parameters for `docker start`, see docker start in the Docker Documentation.

Push a Docker Image from a Registry

TAS for VMs supports pushing apps from container registries such as Docker Hub, Google Container Registry (GCR), and Amazon Elastic Container Registry (ECR).

How you run `cf push` with apps stored in container registries depends on which registry you use and how it authenticates requests for the container image.

The sections below explain how to push apps under different container registry scenarios.

Docker Hub

To deploy a Docker image from a Docker Hub repository, run:

```
cf push APP-NAME --docker-image REPO/IMAGE:TAG
```

Where:

- `APP-NAME` is the name to give the pushed app on TAS for VMs.
- `REPO` is the name of the repository where the image is stored.
- `IMAGE` is the name of the app image on Docker Hub.
- (Optional, but recommended) `TAG` is the tag or version for the image.

For example, the following command pushes the `your-image` image from Docker Hub to a TAS for VMs app:

```
cf push your-app --docker-image cloudfoundry/your-image
```

Private Container Registry without Authentication

As an alternative to Docker Hub, you can use any Docker image registry that presents a valid certificate for HTTPS traffic, such as a company-internal Docker registry.

To push an app as a Docker image using a specified Docker registry, run:

```
cf push APP-NAME --docker-image YOUR-PRIVATE-REGISTRY.DOMAIN:PORT/REPO/IMAGE:TAG
```

Where:

- `APP-NAME` is the name to give the pushed app on TAS for VMs.
- `YOUR-PRIVATE-REGISTRY.DOMAIN` is the path to the Docker registry.
- `PORT` is the port where the registry serves traffic.
- `REPO` is the name of the repository where the image is stored.
- `IMAGE` is the name of the app image being pushed.

- (Optional, but recommended) `TAG` is the tag or version for the image.

For example, the following command pushes the `v2` version of the `your-image` image from the `your-repo` repository of the `internal-registry.example.com` registry on port `5000`:

```
cf push your-app --docker-image internal-registry.example.com:5000/your-repo/your-imag
e:v2
```

Private Container Registry with Basic Authentication

Many Docker registries control access to Docker images by authenticating with a username and password.

To push an app as a Docker image from a registry that uses basic username and password authentication, run:

```
CF_DOCKER_PASSWORD=YOUR-PASSWORD cf push APP-NAME --docker-image REPO/IMAGE:TAG --dock
er-username USER
```

Where:

- `YOUR-PASSWORD` is the password to use for authentication with the Docker registry.
    - Setting `CF_DOCKER_PASSWORD` prepended to the `cf push --docker-image` makes the value temporary, which is more secure than setting the environment variable indefinitely with `export`.
- `APP-NAME` is the name to give the pushed app on TAS for VMs.
- `REPO` is the repository where the image is stored.
    - For Docker Hub, this is just the repository name.
    - For a private registry, this includes the registry address and port, as described in Push a Docker Image from a Private Registry, in the format `YOUR-PRIVATE-REGISTRY.DOMAIN:PORT/REPO`.
- `IMAGE` is the name of the app image being pushed.
- (Optional, but recommended)`TAG` is the tag or version for the image.
- `USER` is the username to use for authentication with the registry.

> **Note:** If container registry credentials change, you must push the app with the new credentials. Apps require access to the container registry when starting. If you do not push the app with the new credentials, TAS for VMs fails to start the app. When you rotate container credentials, VMware recommends using a set of two credentials, where the `old` credentials can be deactivated after all apps have been pushed with the `new` credentials.

Amazon Elastic Container Registry (ECR)

TAS for VMs supports pushing apps from images hosted on Amazon Web Services ECR, which authenticates with temporary password tokens.

To push an app as a Docker image from ECR, run:

```
CF_DOCKER_PASSWORD=AWS-SECRET-ACCESS-KEY cf push APP-NAME --docker-image REPO/IMAGE:TAG --docker-username AWS-ACCESS-KEY-ID
```

Where:

- `AWS-SECRET-ACCESS-KEY` is the AWS Secret Access Key for the IAM user accessing the ECR registry.
    - Setting `CF_DOCKER_PASSWORD` prepended to the `cf push --docker-image` makes the value temporary, which is more secure than setting the environment variable indefinitely with `export`.
- `APP-NAME` is the name to give the pushed app on TAS for VMs.
- `REPO` is the ECR repository containing the image being pushed.
- `IMAGE` is the name of the app image being pushed.
- (Optional, but recommended) `TAG` is the tag or version for the image.
- `AWS-ACCESS-KEY-ID` is the AWS Access Key ID for the IAM user accessing the ECR registry.

Running `cf push` with an ECR registry triggers TAS for VMs to:

1. Use the AWS Secret Access Key and Access Key ID to retrieve the temporary ECR username and password.
2. Use the temporary tokens to retrieve the image.

Google Container Registry (GCR)

TAS for VMs supports pushing apps from images hosted on Google Container Registry (GCR) service. This feature requires that you use JSON key-based authentication. For more information about JSON key authentication, see the Google Cloud documentation.

Step 1: Authenticate with GCR

To authenticate with GCR, you must create a JSON key file and associate it with your project.

To create a JSON key file and associate it with your project:

1. Create a GCP service account. To create a GCP service account, see the Google Cloud documentation. Run:

    ```
    gcloud iam service-accounts create YOUR-ACCOUNT --display-name "YOUR-DISPLAY-NAME"
    ```

    Where:

    - `YOUR-ACCOUNT` is the name of your service account.
    - `YOUR-DISPLAY-NAME` is the display name of your service account.

2. Set your project ID by running:

    ```
    gcloud config set project YOUR-PROJECT-ID
    ```

   Where `YOUR-PROJECT-ID` is your project ID.

3. Create a JSON key file and associate it with the service account by running:

   ```
   gcloud iam service-accounts keys create key.json --iam-account=YOUR-ACCOUNT@YOUR-PROJECT-ID.iam.gserviceaccount.com
   ```

   Where:

   - `YOUR-ACCOUNT` is the name of your service account.
   - `YOUR-PROJECT-ID` is your project ID.

4. Add the IAM policy binding for your project and service account by running:

   ```
   gcloud projects add-iam-policy-binding YOUR-PROJECT --member serviceAccount:YOUR-ACCOUNT@YOUR-PROJECT-ID.iam.gserviceaccount.com --role roles/storage.objectViewer
   ```

   Where:

   - `YOUR-PROJECT` is the name of your project.
   - `YOUR-ACCOUNT` is the name of your service account.
   - `YOUR-PROJECT-ID` is your project ID.

Step 2: Deploy the GCP image

To deploy your GCR image using the cf CLI, run:

```
CF_DOCKER_PASSWORD="$(cat key.json)" cf push APP-NAME --docker-image docker://YOUR-REGISTRY-URL/YOUR-PROJECT/YOUR-IMAGE-NAME --docker-username _json_key`
```

Where:

- `APP-NAME` is the name of the app being pushed.
- `YOUR-REGISTRY-URL` is the URL of your registry.
- `YOUR-PROJECT` is the name of your project.
- `YOUR-IMAGE-NAME` is the name of your image.

Note: The `key.json` file must point to the file you created in the previous step.

Note: For information about specifying `YOUR-REGISTRY-URL`, see Pushing and Pulling Images in the Google Cloud documentation.

Docker volume support

You can use volume services with Docker apps. For more information about enabling volume support, see Using an External File System (Volume Services).

Deploying large apps

When you deploy apps larger than 750 MB to VMware Tanzu Application Service for VMs (TAS for VMs), you must observe additional constraints and recommended settings.

Deployment considerations and limitations

The deployment process involves uploading, staging, and starting the app. For more information about the default time limits for uploading, staging, and starting an app, see the Deployment section of the *App Container Lifecycle* topic.

To deploy large apps to TAS for VMs, ensure that:

- The total size of the files to upload for your app does not exceed the maximum app file size that an admin sets in the **App Developer Controls** pane of the TAS for VMs tile.
- Your network connection speed is sufficient to upload your app within the 15-minute limit. The minimum recommended speed is 874 KB per second.

  > **Note:** TAS for VMs provides an authorization token that is valid for a minimum of 20 minutes.

- You allocate enough memory for all instances of your app. Use either the `-m` flag with `cf push` or set an app memory value in your `manifest.yml` file.
- You allocate enough disk space for all instances of your app. Use either the `-k` flag with `cf push` or set a disk space allocation value in your `manifest.yml` file.
- If you use an app manifest file, `manifest.yml`, be sure to specify adequate values for your app for attributes such as app memory, app start timeout, and disk space allocation. For more information about using manifests, see Deploying with App Manifests.
- The total size of environment variables for your application does not exceed 130 KB. This includes TAS for VMs system environment variables like `VCAP_SERVICES` and `VCAP_APPLICATION`.
- You push only the files that are necessary for your app. To meet this requirement, push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files. For more information about specifying excluded files, see the Ignore Unnecessary Files When Pushing section in the *Considerations for Designing and Running an App in the Cloud* topic.
- You configure Cloud Foundry Command Line Interface (cf CLI) staging, startup, and timeout settings to override settings in the manifest, as necessary.
  - `CF_STAGING_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to stage after Cloud Foundry successfully uploads and packages the app. Value set in minutes.
  - `CF_STARTUP_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to start. Value set in minutes.

   - `cf push -t TIMEOUT`: Controls the maximum time that Cloud Foundry allows to elapse between starting an app and the first healthy response from the app. When you use this flag, the cf CLI ignores any app start timeout value set in the manifest. Value set in seconds. In cf CLI v7, you can also use the long form: `cf push --app-start-timeout TIMEOUT`

For more information about using the cf CLI to deploy apps, see the Push section of the *Getting Started with the cf CLI* topic.

> **Note:** Changing the timeout setting for the cf CLI does not change the timeout limit for Cloud Foundry server-side jobs such as staging or starting apps. You must change server-side timeouts in the manifest. Because of the differences between the Cloud Foundry and cf CLI timeout values, your app might successfully start even though the cf CLI reports `App failed`. Run `cf apps APP_NAME` to review the status of your app.

Default settings and limitations summary table

This table provides summary information of constraints and default settings to consider when you deploy a large app to TAS for VMs.

Setting	Note
App Package Size	Maximum: Set in the **App Developer Controls** pane of the TAS for VMs tile
Authorization Token Grace Period	Default: 20 minutes, minimum
`CF_STAGING_TIMEOUT`	cf CLI environment variable Default: 15 minutes
`CF_STARTUP_TIMEOUT`	cf CLI environment variable Default: 5 minutes
`cf push -t TIMEOUT`	App start timeout maximum Default: 60 seconds
Disk Space Allocation	Default: 1024 MB
Internet Connection Speed	Recommended Minimum: 874 KB per second

Starting, restarting, and restaging your apps

You can start, restart, and restage apps in Cloud Foundry, using the cf CLI or the app manifest (attributes in a YAML file).

Start your app

To start your app, run the following command from your app root directory:

```
$ cf push YOUR-APP
```

For more information about pushing apps, see the Pushing an app topic.

Cloud Foundry determines the start command for your app from one of the three following sources:

- The `-c` command-line option in the Cloud Foundry Command Line Interface (cf CLI). See the following example:

  ```
  $ cf push YOUR-APP -c "node YOUR-APP.js"
  ```

- The `command` attribute in the app manifest. For example:

  ```
  command: node YOUR-APP.js
  ```

- The buildpack, which provides a start command appropriate for a particular type of app.

The source that Cloud Foundry uses depends on factors explained below.

How Cloud Foundry determines its default start command

The first time you deploy an app, `cf push` uses the buildpack start command by default. After that, `cf push` defaults to whatever start command was used for the previous push.

To override these defaults, provide the `-c` option, or the command attribute in the manifest. When you provide start commands both at the command line and in the manifest, `cf push` ignores the command in the manifest.

Forcing Cloud Foundry to use the buildpack start command

To force Cloud Foundry to use the buildpack start command, specify a start command of `null`.

You can specify a null start command in one of two ways.

- Using the `-c` command-line option in the cf CLI:

  ```
  $ cf push YOUR-APP -c "null"
  ```

- Using the `command` attribute in the app manifest:

  ```
  command: null
  ```

This can be helpful after you have deployed while providing a start command at the command line or the manifest. At this point, a command that you provided, rather than the buildpack start command, has become the default start command. In this situation, if you decide to deploy using the buildpack start command, the `null` command makes that easy.

Start commands when migrating a database

Start commands are used in special ways when you migrate a database as part of an app deployment. For more information, see Services Overview.

Restart your app

To restart your app, run the following command:

```
$ cf restart YOUR-APP
```

Restarting your app stops your app and restarts it with the already compiled droplet. A droplet is a tarball that includes:

- stack
- buildpack
- app source code

The Diego cell unpacks, compiles, and runs a droplet on a container.

Restart your app to refresh the app's environment after actions such as binding a new service to the app or setting an environment variable that only the app consumes. However, if your environment variable is consumed by the buildpack in addition to the app, then you must restage the app for the change to take effect.

Restage your app

To restage your app, run the following command:

```
$ cf restage YOUR-APP
```

Restaging your app stops your app and restages it, by compiling a new droplet and starting it.

Restage your app if you have changed the environment in a way that affects your staging process, such as setting an environment variable that the buildpack consumes. The staging process has access to environment variables, so the environment can affect the contents of the droplet. You should also restage your app whenever you edit any configuration settings, such as when you rename it, add metadata, or configure health checks. The new settings often do not take effect until you restage the app.

Restaging your app compiles a new droplet from your app without updating your app source. If you must update your app source, re-push your app by following the steps in the section above.

Pushing an app with multiple processes

The Cloud Foundry API (CAPI) V3 supports using a single command to push apps that run multiple processes, such as a web app that has a UI process and a worker process. You can push an app with multiple processes using either a manifest or a Procfile.

For more information about processes, see Processes in the CAPI V3 documentation.

Push an app with multiple processes using a manifest

To push an app with multiple processes using a manifest:

1. Create a file that defines a manifest. Include each process with its start command. See the following example, which defines the app `example-app` with two processes:

   ```
   version: 1
   - name: example-app
   ```

```
  processes:
  - type: web
    command: bundle exec rackup config.ru -p $PORT
    instances: 3
  - type: worker
    command: bundle exec rake worker:start
    health-check-type: process
    instances: 2
```

2. Push the app using your manifest:

   ```
   cf push -f example-manifest.yml
   ```

For more information about defining processes with manifests, see processes in *App Manifest Attribute Reference*.

Push an app with multiple processes using a Procfile

To push an app with multiple processes using a Procfile:

> **Note:** Procfile support varies depending on the buildpack you use. For example, the Staticfile buildpack requires a Staticfile instead of a Procfile. For more information, see Staticfile Buildpack.

1. Create a file named `Procfile` in the root of your app directory.

   > **Note:** For more information about Procfiles, see Procfiles in the CAPI V3 documentation.

2. Add each process and its start command to the Procfile. See the following example:

   ```
   web: bundle exec rackup config.ru -p $PORT
   worker: bundle exec rake worker:start
   ```

3. Push the app:

   ```
   cf push myapp
   ```

By default, the web process has a route and one instance. Other processes have zero instances by default.

Scale a process

To scale an app process:

- **cf CLI v7:**
- Run:

  ```
  cf scale APP-NAME --process PROCESS-NAME -i INSTANCE-COUNT
  ```

Where

- `APP-NAME` is the name of your app.
- `PROCESS-NAME` is the name of the process you want to scale.
- `INSTANCE-COUNT` is the number of instances to which you want to scale the process.

View processes

To view the processes running as part of an app, run `cf APP-NAME`, where `APP-NAME` is the name of your app. See the following example in which the app has a `web` and a `worker` process.

```
$ cf app myapp
 Showing health and status for app myapp in org test / space test as admin...

name: myapp
 requested state: started
 processes: web:1/1, worker:2/2
 memory usage: 256M x 1, 256M x 2
 routes: myapp.cloudfoundry.example.com
 stack: cflinuxfs3
 buildpacks: ruby

web:1/1
 state since cpu memory disk
 #0 running 2017-09-25 15:43:26 PM 0.2% 18.9M of 256M 84.4M of 1G

worker:2/2
 state since cpu memory disk
 #0 running 2017-09-25 15:49:46 PM 0.1% 5M of 256M 73M of 1G
 #1 running 2017-09-25 15:49:46 PM 0.1% 5M of 256M 73M of 1G
```

Running cf push sub-step commands

The Cloud Foundry CLI (cf CLI) includes commands that provide detailed control over app pushes. When you use these commands, you can choose to perform only some steps of the `cf push` process or perform specific actions between the steps that are normally run as part of running `cf push`.

Here are some example use cases for the sub-step commands:

- Updating a third party system before staging an app
- Retrying failed stagings without incurring downtime
- Calling external services to report audit data during push
- Scanning a droplet before deploy
- Integrating with a change request system

To support these custom push workflows, Cloud Foundry divides apps into smaller building blocks. The following table describes these building blocks as *resources* and lists the command associated with each one.

For information on using these commands, see Example Workflows below.

> **Note:** The cf CLI v6 commands described in this topic are experimental and unsupported. Consider upgrading to cf CLI v7 to use supported versions of these commands. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.

Resource	Description	Command
App	The top-level resource that represents an app and its configuration. For more information, see Apps in the CAPI documentation.	• **cf CLI v6:** `cf v3-create-app` • **cf CLI v7:** `cf create-app`
Package	The source code that makes up an app. For more information, see Packages in the CAPI documentation.	• **cf CLI v6:** `cf v3-create-package` • **cf CLI v7:** `cf create-package`
Build	The staging process. Creating a build combines a Package with a Buildpack and builds it into an executable resource. For more information, see Builds in the CAPI documentation.	• **cf CLI v6:** `cf v3-stage` • **cf CLI v7:** `cf stage-package`
Droplet	An executable resource that results from a Build. For more information, see Droplet in the CAPI documentation.	• **cf CLI v6:** `cf v3-set-droplet` • **cf CLI v7:** `cf set-droplet`
Manifest	A file used when pushing your app to apply bulk configuration to an app and its underlying processes.. For more information, see Space Manifest in the CAPI documentation.	• **cf CLI v6:** `cf v3-apply-manifest` • **cf CLI v7:** `cf create-app-manifest`, `cf apply-manifest`

Example Workflows

The following sections describe example workflows for working with the `cf push` sub-step commands.

Push an App Using Sub-Step Commands

This example workflow describes how to push an app using sub-step commands instead of `cf push`.

1. Create your app with the cf CLI:
    - If you are using cf CLI v7, run:

```
    cf create-app APP-NAME
    ```

    Where `APP-NAME` is the name you give your app.

  - If you are using cf CLI v6, run:

    ```
    cf v3-create-app APP-NAME
    ```

    Where `APP-NAME` is the name you give your app.

2. From your app directory, create a package for your app.
    - If you are using cf CLI v7, run:

      ```
      cf create-package APP-NAME
      ```

      Where `APP-NAME` is the name of your app.

    - If you are using cf CLI v6, run:

      ```
      cf v3-create-package APP-NAME
      ```

      Where `APP-NAME` is the name of your app.

3. Locate and copy the `package guid` from the output of the previous step. See the following example output:

    ```
    Uploading and creating bits package for app APP-NAME in org test / space test as admin...
     package guid: 0dfca85a-8ed4-4f00-90d0-3ab08852dba8
     OK
    ```

4. Stage the package you created:
    - If you are using cf CLI v7, run:

      ```
      cf stage-package APP-NAME --package-guid PACKAGE-GUID
      ```

      Where:

        - `APP-NAME` is the name of your app.
        - `PACKAGE-GUID` is the package GUID you recorded in the previous step.

    - If you are using cf CLI v6, run:

      ```
      cf v3-stage APP-NAME --package-guid PACKAGE-GUID
      ```

      Where:

        - `APP-NAME` is the name of your app.
        - `PACKAGE-GUID` is the package GUID you recorded in the previous step.

5. Locate and copy the `droplet guid` from the output of the previous step. See the following example output:

```
Staging package for APP-NAME in org test / space test as admin...
...
Package staged
 droplet guid: f60d3464-415a-4202-9d40-26a70373a487
 state: staged
 created: Mon 25 Sep 16:37:45 PDT 2018
```

6. Assign the droplet to your app:

   - If you are using cf CLI v7, run:

     ```
     cf set-droplet APP-NAME -d DROPLET-GUID
     ```

     Where:

     - `APP-NAME` is the name of your app.
     - `DROPLET-GUID` is the droplet GUID you recorded in the previous step.

   - If you are using cf CLI v6, run:

     ```
     cf v3-set-droplet APP-NAME -d DROPLET-GUID
     ```

     Where:

     - `APP-NAME` is the name of your app.
     - `DROPLET-GUID` is the droplet GUID you recorded in the previous step.

7. Start your app:

   - If you are using cf CLI v7, run:

     ```
     cf start APP-NAME
     ```

     Where `APP-NAME` is the name of your app.

   - If you are using cf CLI v6, run:

     ```
     cf v3-start APP-NAME
     ```

     Where `APP-NAME` is the name of your app.

Roll Back to a Previous Droplet

This example workflow describes how to roll back to a previous droplet used by your app. You may want to use this, for example, if you update your app and it has a bug that causes it to crash.

1. List the droplets for your app:

   - If you are using cf CLI v7, run:

     ```
     cf droplets APP-NAME
     ```

     Where `APP-NAME` is the name of your app.

   - If you are using cf CLI v6, run:

```
    cf v3-droplets APP-NAME
    ```

    Where `APP-NAME` is the name of your app.

2. From the output, locate and copy the second-to-last GUID. In the following example, this is `66524145-5502-40e6-b782-47fe68e13c49`.

    ```
    Listing droplets of app APP-NAME in org test / space test as admin...

    guid state created
            66524145-5502-40e6-b782-47fe68e13c49 staged Mon 25 Sep 16:37:34
    PDT 2018
            0677ad93-9f77-4aaa-9a6b-44da022dcd58 staged Mon 25 Sep 16:44:55
    PDT 2018
    ```

3. Stop your app:

    - If you are using cf CLI v7, run:

        ```
        cf stop APP-NAME
        ```

        Where `APP-NAME` is the name of your app.

    - If you are using cf CLI v6, run:

        ```
        cf v3-stop APP-NAME
        ```

        Where `APP-NAME` is the name of your app.

4. Set the app to use the previous droplet:

    - If you are using cf CLI v7, run:

        ```
        cf set-droplet APP-NAME -d PREVIOUS-DROPLET-GUID
        ```

        Where:

        - `APP-NAME` is the name of your app.
        - `PREVIOUS-DROPLET-GUID` is the droplet GUID you recorded in a previous step.

    - If you are using cf CLI v6, run:

        ```
        cf v3-set-droplet APP-NAME -d PREVIOUS-DROPLET-GUID
        ```

        Where:

        - `APP-NAME` is the name of your app.
        - `PREVIOUS-DROPLET-GUID` is the droplet GUID you recorded in a previous step.

5. Start your app:

    - If you are using cf CLI v7, run:

        ```
        cf start APP-NAME
        ```

Where `APP-NAME` is the name of your app.

  - If you are using cf CLI v6, run:

    ```
    cf v3-start APP-NAME
    ```

    Where `APP-NAME` is the name of your app.

Configuring rolling app deployments

You can use Cloud Foundry Command Line Interface (cf CLI) commands or the Cloud Foundry API (CAPI) to push your apps to VMware Tanzu Application Service for VMs (TAS for VMs) using a rolling deployment.

For information about the traditional method for addressing app downtime while pushing app updates, see Using blue-green deployment to reduce downtime and risk.

For more information about CAPI, see the Cloud Foundry API (CAPI) documentation.

Prerequisites

The procedures in this topic require one of the following:

- **cf CLI v7:** Install cf CLI v7.
- **cf CLI v6:** If you use cf CLI v6:
  - You must install cf CLI v6.40 or later.
  - The rolling deployment feature must be enabled for your deployment. Ensure that the **Disable rolling app deployments** check box is not enabled in the **Advanced Features** pane of the VMware Tanzu Application Service for VMs (TAS for VMs) tile.
- **CAPI V3:** If you use CAPI V3, you must install the cf CLI.

Commands

This section describes the commands for working with rolling app deployments.

Deploy an app

To deploy an app without incurring downtime:

> ⚠ **Caution:** Ensure that you understand the limitations of this feature before running the command. For more information, see Limitations below.

- **For cf CLI v7, run:**

  ```
  cf push APP-NAME --strategy rolling
  ```

  Where `APP-NAME` is the name that you want to give your app.

> **Note:** cf CLI v7 exits when one instance of each process is healthy. It also includes a `--no-wait` flag on push for users who do not want to wait for the operation to complete. `cf push` used with the `--no-wait` flag exits as soon as one instance is healthy.

If the deployment hangs indefinitely, cancel the process and run it again with the step above. To cancel, see Cancel a Deployment below.

- **For cf CLI v6, run:**

  ```
  cf v3-zdt-push APP-NAME
  ```

  Where `APP-NAME` is the name that you want to give your app.

  > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.

- **For CAPI V3:**

  1. Log in to the cf CLI.

     ```
     cf login
     ```

  2. Create an empty app by running the following `curl` command with `POST /v3/apps`. Record the app GUID from the output.

     ```
     cf curl /v3/apps \
       -X POST \
       -H "Content-type: application/json" \
       -d '{
         "name": "APP-NAME",
         "relationships": {
           "space": {
             "data": {
               "guid": "SPACE-GUID"
             }
           }
         }
       }'
     ```

     Where:

     - `APP-NAME` is the name that you want to give your app.
     - `SPACE-GUID` is the space identifier that you want to associate with your app.

  3. Create a package with the following `curl` command with `POST /v3/packages`. Record the package GUID from the output.

     ```
     cf curl /v3/packages \
       -X POST \
       -H "Content-type: application/json" \
     ```

```
  -d '{
    "type": "bits",
    "relationships": {
      "app": {
        "data": {
          "guid": "APP-GUID"
        }
      }
    }
  }'
```

Where `APP-GUID` is the app GUID that you recorded in the previous step. This app GUID is a unique identifier for your app.

4. Upload the package bits by running the following `curl` command with `POST /v3/packages/PACKAGE-GUID/upload`.

   ```
   cf curl /v3/packages/PACKAGE-GUID/upload \
   -X POST \
   -F bits=@"PACKAGED-APP" \
   ```

   Where:

   - `PACKAGE-GUID` is the package GUID that you recorded in the previous step.
   - `PACKAGED-APP` is your app packaged in a file such as `.zip`.

5. Create the build by running the following `curl` command with `POST /v3/builds`. Record the droplet GUID from the output.

   ```
   cf curl /v3/builds \
     -X POST \
     -H "Content-type: application/json" \
     -d '{
       "package": {
          "guid": PACKAGE-GUID"
       }
     }'
   ```

   Where `PACKAGE-GUID` is the package GUID that you recorded in a previous step.

6. Deploy your app by running the following `curl` command with `POST /v3/deployments`. To verify the status of the deployment or take action on the deployment, record the deployment GUID from the output.

   ```
   cf curl /v3/deployments \
   -X POST \
   -H "Content-type: application/json" \
   -d '{
     "droplet": {
       "guid": "DROPLET-GUID"
     },
     "strategy": "rolling",
     "relationships": {
       "app": {
         "data": {
           "guid": "APP-GUID"
         }
   ```

```
    }
  }
}'
```

Where DROPLET-GUID and APP-GUID are the GUIDs that you recorded in previous steps.

For more information about this command, see How It Works below.

Cancel a deployment

To stop the deployment of an app that you pushed:

- **For cf CLI v7, run:**

  ```
  cf cancel-deployment APP-NAME
  ```

  Where APP-NAME is the name of the app.

- **For cf CLI v6, run:**

  ```
  cf v3-cancel-zdt-push APP-NAME
  ```

  Where APP-NAME is the name of the app.

  > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.

- **For CAPI V3, run:**

  ```
  cf curl /v3/deployments/DEPLOYMENT-GUID/actions/cancel" -X POST
  ```

  Where DEPLOYMENT-GUID is the GUID of the deployment that you recorded after following the CAPI procedure in Deploy an App above.

This reverts the app to its state from before the deployment started by doing the following:

- Scaling up the original web process
- Removing any deployment artifacts
- Resetting the `current_droplet` on the app

> **Note:** The cancel command is designed to revert the app to its original state as quickly as possible and does not guarantee zero downtime.

Restart an app

To restart your app without downtime, run the appropriate command below. Restart an app to apply configuration updates that require a restart, such as environment variables or service bindings.

- **For cf CLI v7, run:**

```
cf restart APP-NAME --strategy rolling
```

Where `APP-NAME` is the name of the app.

- **For cf CLI v6, run:**

```
cf v3-zdt-restart APP-NAME
```

Where `APP-NAME` is the name of the app.

> **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.

- **For CAPI V3, run:**

```
cf curl /v3/deployments \
-X POST \
-H "Content-type: application/json" \
-d '{
  "droplet": {
    "guid": "DROPLET-GUID"
  },
  "strategy": "rolling",
  "relationships": {
    "app": {
      "data": {
        "guid": "APP-GUID"
      }
    }
  }
}'
```

Where `DROPLET-GUID` and `APP-NAME` are the GUIDs that you recorded in previous steps.

How it works

This section describes the rolling deployments and their limitations.

Rolling deployment process

This section describes the process of pushing an app with a rolling deployment strategy.

1. The `cf push APP-NAME --strategy rolling` command does the following:
    1. Stages the updated app package.
    2. Creates a droplet with the updated app package.
    3. Creates a deployment with the new droplet and any new configuration.
        - This starts a new process with one instance that shares the route with the old process.

    - At this point, if you run `cf app` on your app, you see multiple `web` processes. For more information about the deployment object, see the Deployments section of the CAPI V3 documentation.

2. After the command creates the deployment, the `cc_deployment_updater` BOSH job runs in the background, updating deployments as follows:
    1. Adds another instance of the new web process and removes an instance from the old web process.

        > **Note:** This only happens if all instances of the new web process are currently running.

    2. Repeats the above step until the new web process reaches the desired number of instances.
    3. Removes the old web process. The new web process now fully replaces the old web process.
    4. Restarts all non-web processes of the app.
    5. Sets the deployment to `DEPLOYED`.

Limitations

The following table describes the limitations of when using rolling deployments.

Limitation	Description
App Manifests	The `cf v3-zdt-push` command does not support providing an app manifest with the `-f` flag. If you have a `manifest.yml` file in your app directory, it is ignored. **Note:** This limitation only applies to cf CLI v6.
SSH to app instances	Pushing updates to your app with a `cf v3-zdt-push` command causes the new web process and app GUID to mismatch. `cf ssh` does not handle this scenario. You must use the `cf v3-ssh` command instead. **Note:** This limitation only applies to cf CLI v6.
Multiple app versions	During a deployment, Cloud Foundry serves both the old and new version of your app at the same route. This could lead to user issues if you push backwards-incompatible API changes.
Database migrations	Deployments do not handle database migrations. Migrating an app database when the existing app is not compatible with the migration can result in downtime.
Non-web processes	Rolling deployments only run web processes through the rolling update sequence described above. The commands restart worker and other non-web processes in bulk after updating all web processes. The CAPI V3 API introduces the concept of processes as runnable units of an app. Each app has a web process by default. You can specify additional processes with a Procfile, and in some cases buildpacks create additional processes. For more information about processes, see Processes in the CAPI V3 documentation.

Quotas	Pushing updates to your app using a rolling deployment strategy creates an extra instance of your app. If you lack sufficient quota, the deployment fails. Admins might need to increase quotas to accommodate rolling deployments.
Simultaneous apps when interrupting a push	If you push app before your previous push command for the same app has completed, your first push gets interrupted. Until the last deployment completes, there might be many versions of the app running at the same time. Eventually, the app runs the code from your most recent push.

View the status of rolling deployments

You can use CAPI to view the status of rolling deployments.

To view the status of a rolling deployment:

1. Log in to the cf CLI:

   ```
   cf login
   ```

2. Find the GUID of your app by running:

   ```
   cf app APP-NAME --guid
   ```

   Where `APP-NAME` is the name of the app.

3. Find the deployment for that app by running:

   ```
   cf curl GET /v3/deployments?app_guids=APP-GUID&status_values=ACTIVE
   ```

   Where `APP-GUID` is the GUID of the app. Deployments are listed in chronological order, with the latest deployment displayed as the last in a list.

4. Run:

   ```
   cf curl GET /v3/deployments/DEPLOYMENT-GUID
   ```

   Where `DEPLOYMENT-GUID` is the GUID of the rolling deployment.

`cf curl GET /v3/deployments/DEPLOYMENT-GUID` returns the following properties about the status of rolling deployments:

- `status.value`: Indicates if the deployment is `ACTIVE` or `FINALIZED`.
- `status.reason`: Provides detail about the deployment status.
- `status.details`: Provides the timestamp for the most recent successful health check. The value of the `status.details` property can be `nil` if there is no successful health check for the deployment. For example, there might be no successful health check if the deployment was cancelled.

The following table describes the possible values for the `status.value` and `status.reason` properties:

`status.value`	`status.reason`	Description
`ACTIVE`	`DEPLOYING`	The deployment is deploying.

`ACTIVE`	`CANCELLING`	The deployment is cancelling.
`FINALIZED`	`DEPLOYED`	The deployment was deployed.
`FINALIZED`	`CANCELLED`	The deployment was cancelled.
`FINALIZED`	`SUPERSEDED`	The deployment was stopped and did not finish deploying because there was another deployment created for the app.
`FINALIZED`	`DEGENERATE`	The deployment was created incorrectly by the system.

Pushing apps with sidecar processes

You can run additional processes in the same container as your app. These additional processes are called sidecar processes, or sidecars. An example of a sidecar is an Application Performance Monitoring (APM) tool.

About sidecars

When you provide a sidecar for your app, VMware Tanzu Application Service for VMs (TAS for VMs) packages the required code and configuration needed to run the sidecar and app in the same droplet. It deploys this droplet in a single container on Diego. Both processes within the container undergo health checks independently.

You can push sidecar processes with your app by using one of two methods:

- Using an app manifest. For instructions, see Push an App with a Sidecar Using an App Manifest below.
- With a custom buildpack. For instructions, see Sidecar Buildpacks.

For additional information about sidecars, see Sidecars in the Cloud Foundry API (CAPI) documentation.

For sample apps that use sidecars, see the capi-sidecar-samples repository on GitHub. These sample apps use an app manifest.

Use Cases

You can use sidecars for processes that depend on each other or must run in the same container.

For example, you can use sidecars for processes that must:

- Communicate over a Unix socket or through localhost
- Share the same filesystem
- Be scaled and placed together
- Have fast interprocess communication

Limitations

Sidecars have these limitations:

- The start and stop order of app processes and their sidecars is undefined.

- App processes and sidecars are codependent. If either crashes or exits, the other does also.
- Sidecars are currently not independently scalable. Sidecars share resources with the main app process and other sidecars within the container.
- Sidecars only support PID-based health checks. HTTP health checks for sidecars are not currently supported.

Requirements for Java Apps

These sections describe several requirements that are specific to pushing sidecars with Java apps.

Reserving Memory

You must allocate memory to the sidecar. If you do not, the Java buildpack allocates all of the available memory to the app. As a result, the sidecar does not have enough memory and the app fails to start.

To allocate memory to the sidecar, use the `memory` property in the app manifest. For example:

```
sidecars:
- name: SIDECAR-NAME
        process_types: [ 'PROCESS-TYPES' ]
        command: START-COMMAND
        memory: 256MB
```

Where:

- `SIDECAR-NAME` is a name you give your sidecar.
- `PROCESS-TYPES` is a list of app processes for the sidecar to attach to, such as `web` or `worker`. You can attach multiple sidecars to each process type your app uses.
- `START-COMMAND` is the command used to start the sidecar. For example, `./binary` or `java -jar java-file.jar`.

You must also allocate memory to sidecars that you push with a custom buildpack. For more information, see Sidecar Buildpacks.

Packaging Binaries

If your sidecar is a binary file rather than a set of buildable source files, then you must package the binary file with your Java app.

In some cases, the Java buildpack requires you to push a `.jar` file. If this is the case with your app, you must include the sidecar binary in the `.jar` file.

To package the sidecar binary with the `.jar` file, run:

```
zip JAR -u SIDECAR-BINARY
```

Where:

- `JAR` is your `.jar` file.
- `SIDECAR-BINARY` is your sidecar binary.

For more information about packaging assets with your Java app, see Tips for Java Developers.

Push an App with a Sidecar Using an App Manifest

These sections explain how to push an app with a sidecar using an app manifest. For an example that you can try yourself, see Sidecar Tutorial below.

> **Note:** When pushing a Java app, ensure that you follow the requirements listed in Requirements for Java Apps above.

Prerequisites

Before you can push an app with a sidecar with an app manifest, you must have:

- An app that is currently running or ready to be pushed.
- A file that TAS for VMs can execute inside the app container as a sidecar process. For example, an executable binary, a Java .jar file, or Ruby scripts.

Procedure

To push an app with a sidecar:

1. Create an app or use an existing app. To create an app:
   - If you are using cf CLI v7, run:

     ```
     cf create-app APP-NAME
     ```

     Where `APP-NAME` is the name you give your app.
   - If you are using cf CLI v6, run:

     ```
     cf v3-create-app APP-NAME
     ```

     Where `APP-NAME` is the name you give your app.

     > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.
2. Create a manifest file in the root directory of your app, such as `manifest.yml`. Otherwise, use an existing manifest file for your app. For more information, see Deploying with App Manifests.
3. Add the values below to your app manifest file under the `applications` key:

   ```
   sidecars:
     - name: SIDECAR-NAME
       process_types: [ 'PROCESS-TYPES' ]
       command: START-COMMAND
   ```

Where:

- `SIDECAR-NAME` is a name you give your sidecar.
- `PROCESS-TYPES` is a list of app processes for the sidecar to attach to, such as `web` or `worker`. You can attach multiple sidecars to each process type your app uses.
- `START-COMMAND` is the command used to start the sidecar. For example, `./binary` or `java -jar java-file.jar`.

This example manifest file includes multiple sidecars:

```
---
applications:
- name: my-app
  sidecars:
     - name: authenticator
       process_types: [ 'web', 'worker' ]
       command: bundle exec run-authenticator
     - name: performance monitor
       process_types: [ 'web' ]
       command: bundle exec run-performance-monitor
```

4. To apply the manifest file to your app:

   - If you are using cf CLI v7, run:

     ```
     cf apply-manifest -f PATH-TO-MANIFEST
     ```

     Where `PATH-TO-MANIFEST` is the path to your manifest file. If no flags are passed, the command defaults to using the manifest located in your `pwd`.

   - If you are using cf CLI v6, run:

     ```
     cf v3-apply-manifest -f PATH-TO-MANIFEST
     ```

     Where `PATH-TO-MANIFEST` is the path to your manifest file.

     > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7*.

5. To push your app:

   - If you are using cf CLI v7, run:

     ```
     cf push APP-NAME
     ```

     Where `APP-NAME` is the name of your app.

   - If you are using cf CLI v6, run:

     ```
     cf v3-push APP-NAME
     ```

Where APP-NAME is the name of your app.

> **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7.*

Sidecar Tutorial

You can explore sidecars using the app in the capi-sidecar-samples repository on GitHub. The sections below describe the app, how to build and push the app, and some ways to observe the app and its processes after pushing.

> **Note:** This tutorial assumes that you are pushing the Ruby sample app. You can also follow this tutorial for a Java app using the sidecar-dependent-java-app and push_java_app_with_binary_sidecar.sh in the samples repository. When pushing a Java app, ensure that you follow the requirements listed in Requirements for Java Apps above.

About the Sample App

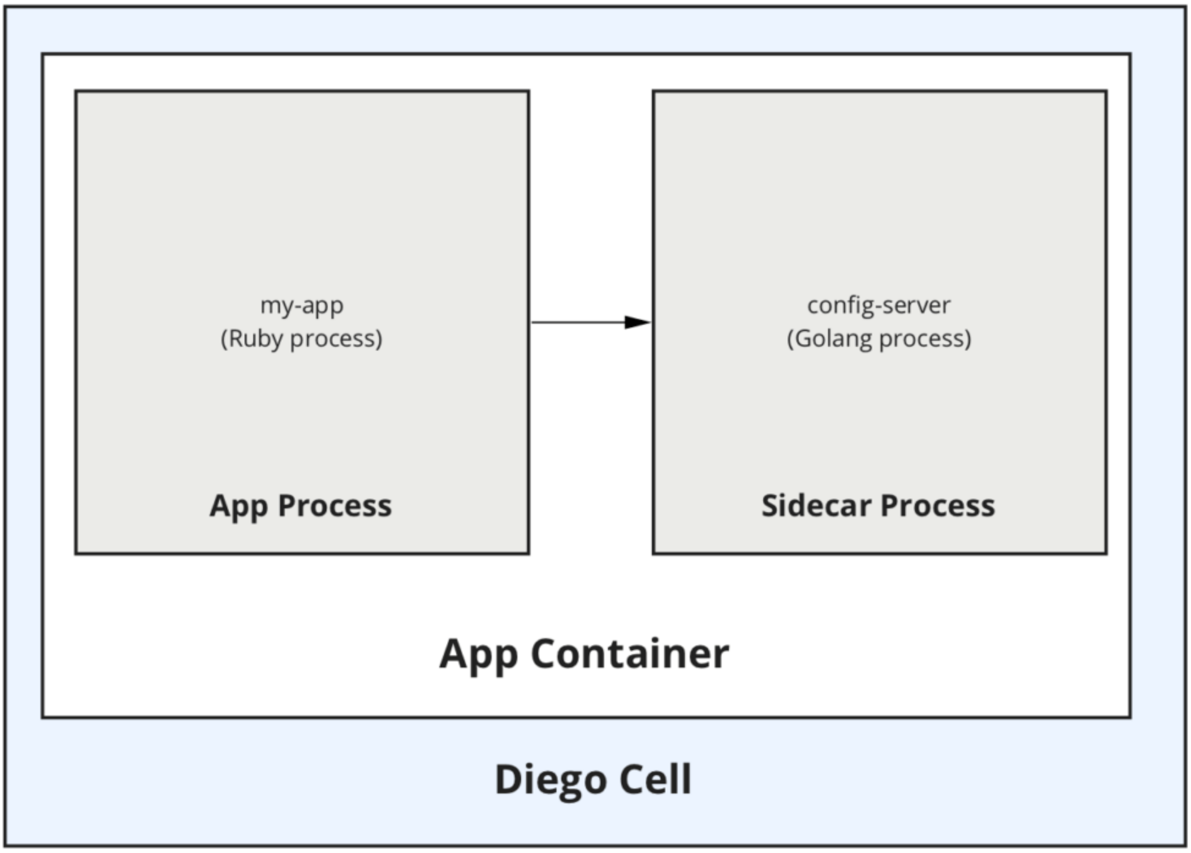
The capi-sidecar-samples repository contains:

- **A simple Ruby app**: This app is named sidecar-dependent-app. It includes a /config endpoint that calls to the sidecar and prints the response, as shown in this code snippet:

```
get '/config' do
puts "Sending a request to the config-server sidecar at localhost:#{ENV['CONFIG_SERVER_PORT']}/config/"
response = Typhoeus.get("localhost:#{ENV['CONFIG_SERVER_PORT']}/config/")
puts "Received #{response.body} from the config-server sidecar"
response.body
end
```

- **A Golang sidecar**: The config-server-sidecar produces a config-server binary. It provides apps with their required configuration over its /config endpoint. It also accepts connections only over localhost on the CONFIG_SERVER_PORT port. This means the sidecar must be co-located in the same container as the app, so that it shares the same network namespace as the main app.

The diagram below illustrates the app architecture:



Push the App and Sidecar

To push the app and sidecar:

1. In a terminal window, clone the Git repository to your workspace by running:

   ```
   git clone https://github.com/cloudfoundry-samples/capi-sidecar-samples.git
   ```

2. Navigate to the `config-server-sidecar` directory.
3. Build the binary for the sidecar by running:

   ```
   GOOS=linux GOARCH=amd64 go build -o config-server .
   ```

   Note: If you do not have Go installed, download the `config-server_linux_x86-64` binary from Releases in the `capi-sidecar-samples` repository in GitHub.

4. To create the app:
   - If you are using cf CLI v7, run:

     ```
     cf create-app sidecar-dependent-app
     ```

   - If you are using cf CLI v6, run:

```
cf v3-create-app sidecar-dependent-app
```

> **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7.*

5. Navigate to the `sidecar-dependent-app` directory.
6. Open and review the `manifest.yml` file. Under `sidecars`, the sidecar is specified with a name, process type, and start command. Under `env`, there is an environment variable that defines the port on which the app and sidecar communicate.
7. To apply the manifest to the app:
    - If you are using cf CLI v7, run:

      ```
      cf apply-manifest
      ```

    - If you are using cf CLI v6, run:

      ```
      cf v3-apply-manifest
      ```

      > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7.*

8. To push the app:
    - If you are using cf CLI v7, run:

      ```
      cf push sidecar-dependent-app
      ```

    - If you are using cf CLI v6, run:

      ```
      cf v3-push sidecar-dependent-app
      ```

      > **Note:** This command is experimental and unsupported. Consider upgrading to cf CLI v7 to use a supported version of this command. To upgrade to cf CLI v7, see Install cf CLI v7 in *Upgrading to cf CLI v7.*

After you push the app, you can further explore it in View the Processes Running in the Container and View the Web URL and App Logs below.

View the Processes Running in the Container

To view the app and sidecar process running in the container:

1. SSH into the app container by running:

   ```
   cf ssh sidecar-dependent-app
   ```

2. To see both the `rackup` process for the main app and `config-server` process for the sidecar, run:

   ```
   ps aux
   ```

   The output you see should resemble the output below:

   ```
   vcap@f00949bd-6601-4731-6f7e-e859:~$ ps aux
   USER         PID %CPU %MEM    VSZ      RSS   TTY   STAT   START TIME  C
   OMMAND
   root           1 0.0  0.0     1120     0     ?     S      22:17 0:00  /
   tmp/garden-init
   vcap           7 0.0  0.0     106716   4508  ?     S      22:17 0:00
   ./config-server
   vcap          13 0.0  0.1     519688   35412 ?     S      22:17 0:00  /
   home/vcap/deps/0/vendor_bundle/ruby/2.4.0/bin/rackup config.ru -p 8080
   vcap          24 0.0  0.0     116344   10792 ?     S      22:17 0:00  /
   tmp/lifecycle/diego-sshd --allowedKeyExchanges= --address=0.0.0.0:2222
   --allowUnauthenticatedClients=false --inhe
   root          82 0.0  0.0     108012   4548  ?     S      22:17 0:00  /
   etc/cf-assets/healthcheck/healthcheck -port=8080 -timeout=1000ms -liven
   ess-interval=30s
   vcap         215 0.3  0.0     70376    3756  pts/0 S      23:12 0:00  /
   bin/bash
   vcap         227 0.0  0.0     86268    3116  pts/0 R      23:12 0:00  p
   s aux
   ```

3. To see that the sidecar is listening on the port specified by `CONFIG_SERVER_PORT` and that the main `ruby` process is connected to it, run:

   ```
   lsof -i | grep $CONFIG_SERVER_PORT
   ```

   The output you see should resemble the output below:

   ```
   vcap@f00949bd-6601-4731-6f7e-e859:~$ lsof -i | grep $CONFIG_SERVER_PORT
   config-se   7 vcap 3u  IPv4 17265901     0t0 TCP *:8082 (LISTEN)
   config-se   7 vcap 5u  IPv4 17265992     0t0 TCP localhost:8082->localh
   ost:42266 (ESTABLISHED)
   ruby       13 vcap 11u  IPv4 17274965    0t0 TCP localhost:42266->local
   host:8082 (ESTABLISHED)
   ```

View the Web URL and App Logs

To view the Web URL and logs for the app:

1. In a browser, navigate to the `config` endpoint of the `sidecar-dependent-app`. For example: `https://sidecar-dependent-app.example.com/config`.

2. See that the browser displays `Scope` and `Password` information. This is the configuration that the app fetches from the `config-server` sidecar.

3. In a terminal window, begin streaming logs for the app by running:

   ```
   cf logs sidecar-dependent-app
   ```

4. In your browser, refresh the `/config` endpoint page and observe that the log stream in your terminal displays logs for both the sidecar and the main app process.

5. In a separate terminal window from your log stream, SSH into the app container by running:

   ```
   cf ssh sidecar-dependent-app
   ```

6. Terminate the sidecar process by running:

   ```
   kill -9 $(pgrep config-server)
   ```

7. View the output in the terminal window where you are streaming the app logs. The app logs indicate that the sidecar process crashed and that Diego restarted the app container. For example:

   ```
   2019-04-17T16:48:55.41-0700 [API/0] OUT App instance exited with guid
   21df1eb8-f25d-43b2-990b-c1a417310553 payload:
   {"instance"=>"a8db0eed-7371-4805-5ad3-4596", "index"=>0,
   "cell_id"=>"86808ce7-afc2-47da-9e79-522a62a48cff", "reason"=>"CRASHED",
   "exit_description"=>"APP/PROC/WEB/SIDECAR/CONFIG-SERVER: Exited with st
   atus 137",
   "crash_count"=>1, "crash_timestamp"=>1555544935367052708,
   "version"=>"50892dcb-274d-4cf6-b944-3eda1e000283"}
   ```

Using blue-green deployment to reduce downtime

Blue-green deployment is a technique that helps you reduce app downtime and risk by running two identical production environments: Blue and Green.

At any time, only one of the environments is live, with the live environment serving all production traffic.

For the example discussed here, Blue is live and Green is idle. As you prepare a new version of your software, deployment and the final stage of testing takes place in the environment that is *not* live: in this example, Green. After you have deployed and fully tested the software in Green, you switch the router so that all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

This technique can eliminate downtime due to app deployment. In addition, blue-green deployment reduces risk: if something unexpected happens with your new version on Green, you can immediately roll back to the lastS version by switching back to Blue.S

Note: If your app uses a relational database, blue-green deployment can lead to discrepancies between your Green and Blue databases during an update. To maximize data integrity, configure a single database for backward and forward compatibility.

Note: You can adjust the route mapping pattern to display a static maintenance page during a maintenance window for time-consuming tasks, such as migrating a database. In this scenario, the router switches all incoming requests from Blue to Maintenance to Green.

Blue-Green Deployment with Cloud Foundry Example

For this example, we'll start with a simple app: "demo-time." This app is a web page that displays the words "Blue time" and the date/time on the server.

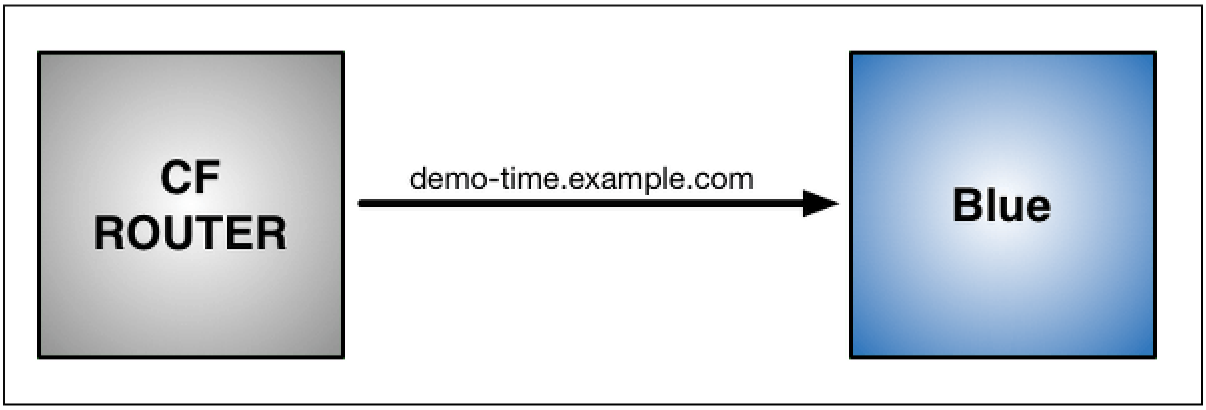
Step 1: Push an App

Use the Cloud Foundry Command Line Interface (cf CLI) to push the app. Name the app "Blue" with the subdomain "demo-time."

```
$ cf push Blue -n demo-time
```

As shown in the graphic below:

- Blue is now running on Cloud Foundry.
- The CF Router sends all traffic for `demo-time.example.com` traffic to Blue.



Step 2: Update App and Push

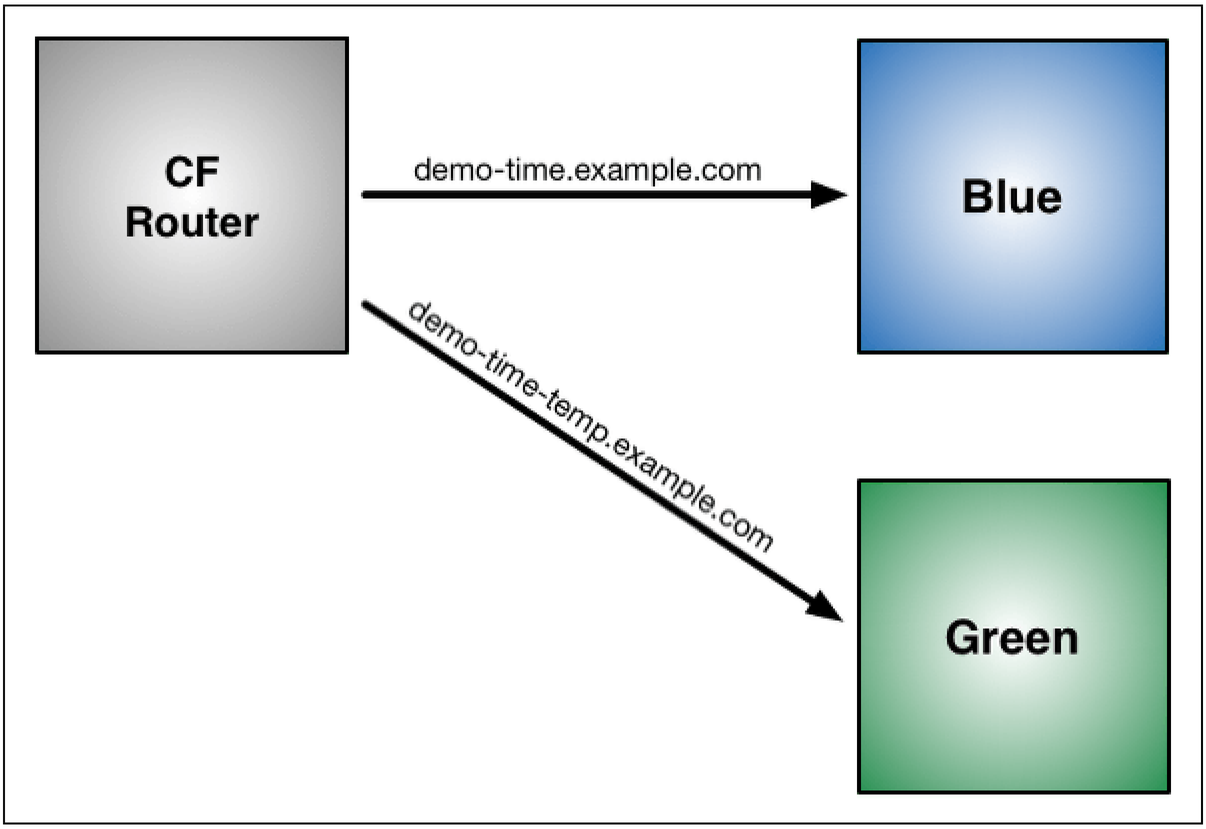
Now make a change to the app. 1. First, replace the word "Blue" on the web page with "Green," then rebuild the source file for the app. 1. Run `cf push` again, but use the name "Green" for the app and provide a different subdomain to create a temporary route:

```
$ cf push Green -n demo-time-temp
```

After this push:

- Two instances of our app are now running on Cloud Foundry: the original Blue and the updated Green.

- The CF Router continues sending all traffic for `demo-time.example.com` to Blue. The router now also sends any traffic for `demo-time-temp.example.com` to Green.



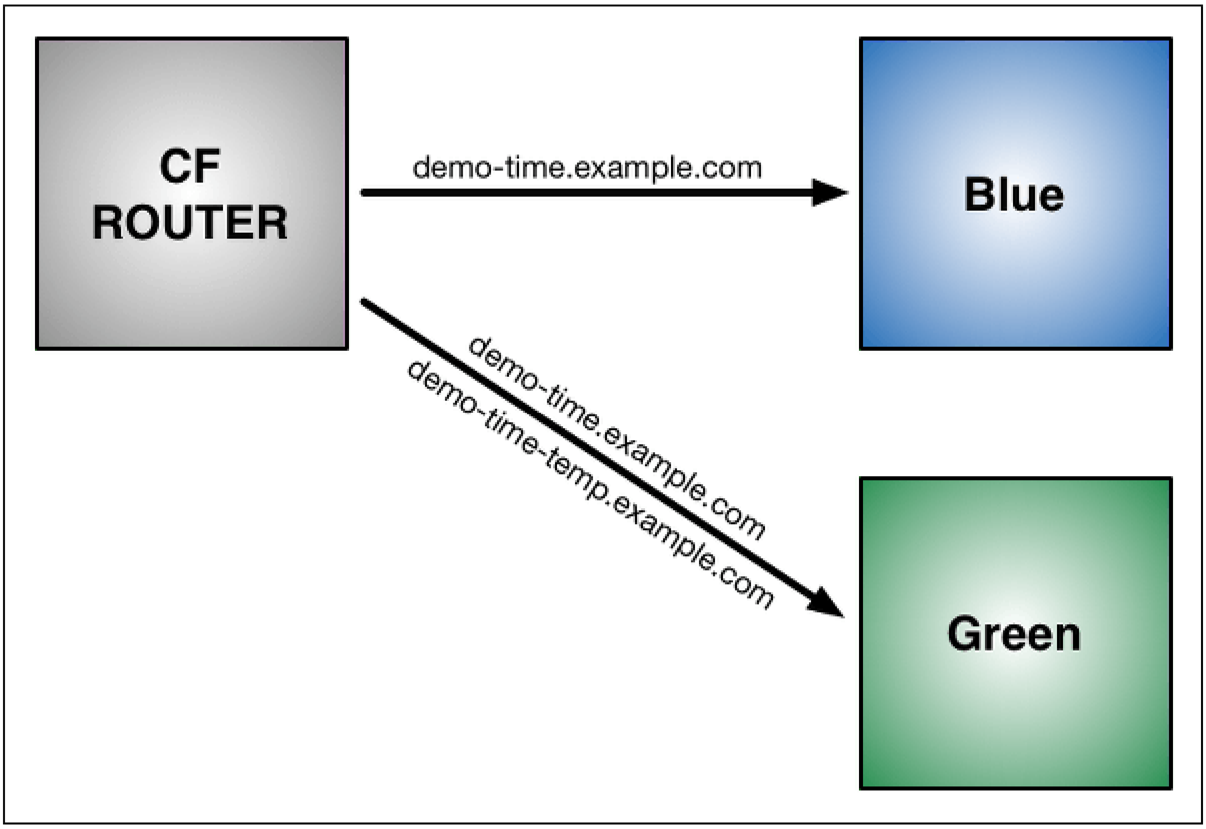
Step 3: Map Original Route to Green

Now that both apps are up and running, switch the router so all incoming requests go to the Green app *and* the Blue app. Do this by mapping the original URL route (`demo-time.example.com`) to the Green app using the cf map-route command.

```
$ cf map-route Green example.com -n demo-time
Binding demo-time.example.com to Green... OK
```

After the `cf map-route` command :

- The CF Router continues sending traffic for `demo-time-temp.example.com` to Green.
- Within a few seconds, the CF Router begins load balancing traffic for `demo-time.example.com` between Blue and Green.



Step 4: Unmap Route to Blue

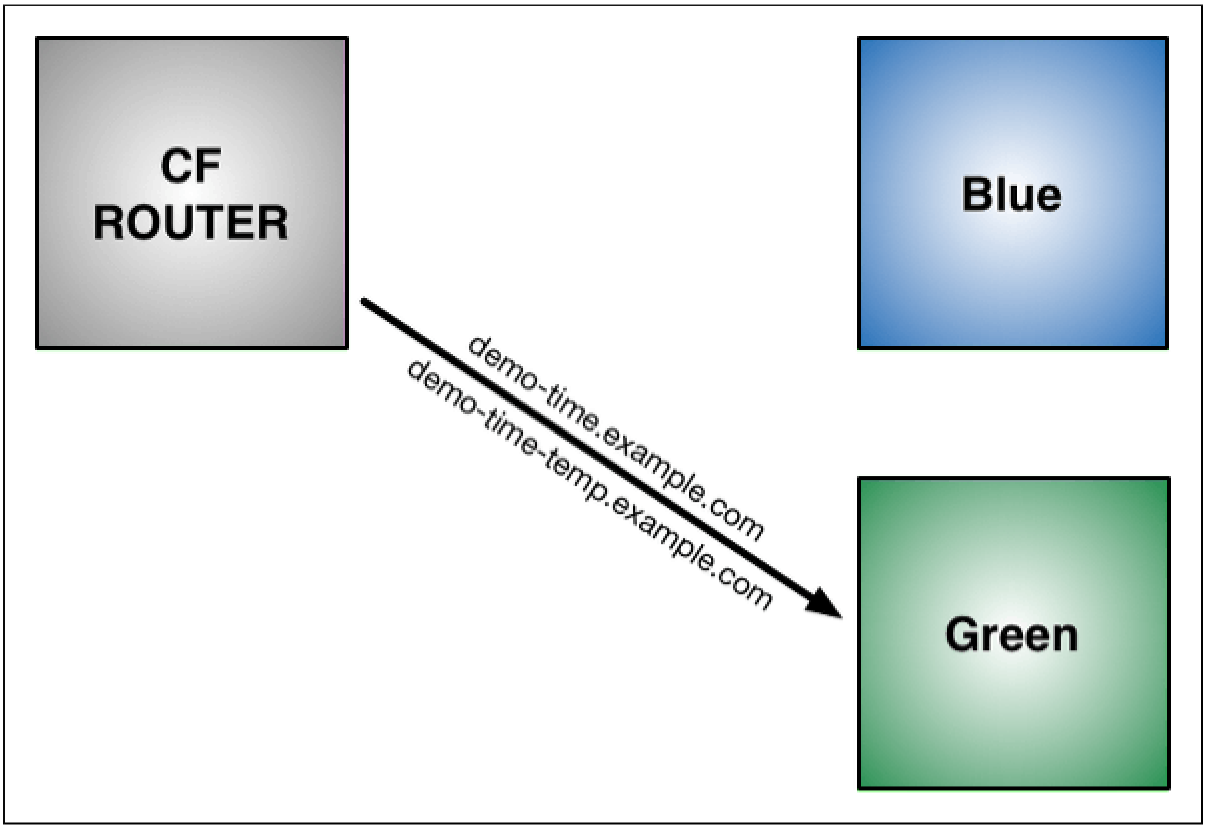
After you verify Green is running as expected, stop routing requests to Blue using the cf unmap-route command:

```
$ cf unmap-route Blue example.com -n demo-time
Unbinding demo-time.example.com from blue... OK
```

After the `cf unmap-route` command:

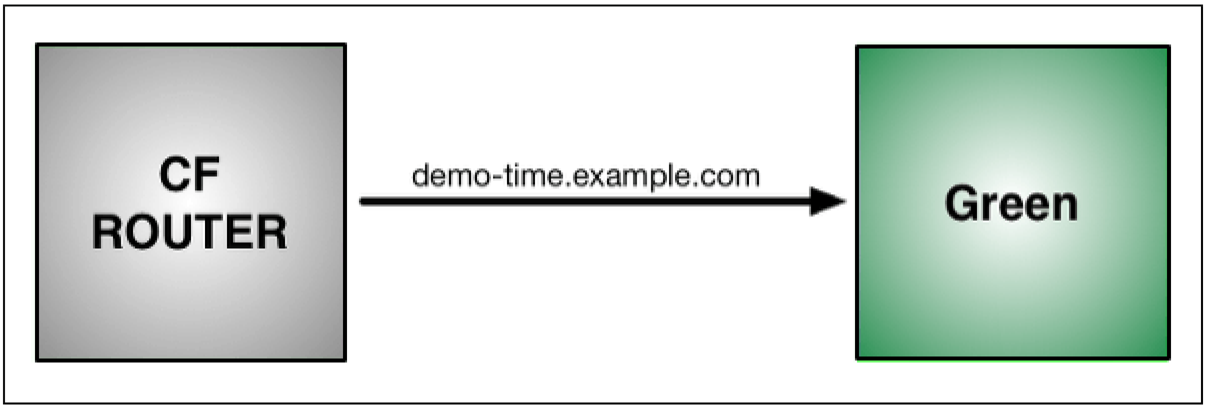
- The CF Router stops sending traffic to Blue.

Now all traffic for `demo-time.example.com` is sent to Green.



Step 5: Remove temporary route to Green

You can now use `cf unmap-route` to remove the route `demo-time-temp.example.com` from Green. The route can be deleted using `cf delete-route` or it can be reserved for later use. You can also decommission Blue, or keep it in case you need to roll back your changes.



Implementation

Cloud Foundry community members have written a plugin to automate the blue-green deployment process:

- BlueGreenDeploy: cf-blue-green-deploy is a plugin, written in Go, for the Cloud Foundry Command Line Interface (cf CLI) that automates a few steps involved in zero-downtime deploys.

Troubleshooting app deployment and health

In this topic, you will learn how to diagnose and resolve common issues when you deploy and run apps on VMware Tanzu Application Service for VMs (TAS for VMs).

Common issues

The following sections describe common issues you might encounter when attempting to deploy and run your app, and suggest possible resolutions.

cf push times out

If your deployment times out during the upload or staging phase, you may receive one of the following error messages:

- `504 Gateway Timeout`
- `Error uploading application`
- `Timed out waiting for async job JOB-NAME to finish`

If this happens:

- **Check your network speed.** Depending on the size of your app, your `cf push` could be timing out because the upload is taking too long. VMware recommends an Internet connection speed of at least 768 KB/s (6 Mb/s) for uploads.
- **Make sure you are pushing only needed files.** By default, `cf push` will push all the contents of the current working directory. Make sure you are pushing only the directory for your app. If your app is too large, or if it has many small files, Ops Manager may time out during the upload. To reduce the number of files you are pushing, ensure that you push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files.
- **Set the CF_STAGING_TIMEOUT and CF_STARTUP_TIMEOUT environment variables.** By default your app has 15 minutes to stage and 5 minutes to start. You can increase these times by setting `CF_STAGING_TIMEOUT` and `CF_STARTUP_TIMEOUT`. Type `cf push -h` at the command line for more information.
- **If your app contains a large number of files, try pushing the app repeatedly.** Each push uploads a few more files. Eventually, all files have uploaded and the push succeeds. This is less likely to work if your app has many small files.

App too large

If your app is too large, you may receive one of the following error messages on `cf push`:

- `413 Request Entity Too Large`
- `You have exceeded your organization's memory limit`

If this happens:

- **Make sure your org has enough memory for all instances of your app.** You will not be able to use more memory than is allocated for your organization. To view the memory quota for your org, use `cf org ORG_NAME`.

Your total memory usage is the sum of the memory used by all apps in all spaces within the org. Each app's memory usage is the memory allocated to it multiplied by the number of instances.

- **Make sure your app is less than 1 GB.** By default, Ops Manager deploys all the contents of the current working directory. To reduce the number of files you are pushing, ensure that you push only the directory for your app, and remove unneeded files or use the `.cfignore` file to specify excluded files. The following limits apply:
  - The app files to push cannot exceed 1 GB.
  - The droplet that results from compiling those files cannot exceed 1.5 GB. Droplets are typically a third larger than the pushed files.
  - The combined size of the app files, compiled droplet, and buildpack cache cannot total more than 4 GB of space during staging.

Unable to detect a supported app type

If Ops Manager cannot identify an appropriate buildpack for your app, you see an error message that states `Unable to detect a supported app type`.

You can view what buildpacks are available with the `cf buildpacks` command.

If you see a buildpack that you believe should support your app, see the buildpack documentation for details about how that buildpack detects apps it supports.

If you do not see a buildpack for your app, you may still be able to push your app with a custom buildpack using `cf push -b` with a path to your buildpack.

App deploy fails

Even when the deploy fails, the app might exist on TAS for VMs. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or About Apps Manager, or you might have to delete the app and redeploy.

If you push multiple apps using a manifest and one fails to deploy, Ops Manager does not attempt to push apps specified after the app that failed.

Common reasons deploying an app fails include:

- The app does not successfully bind to a service, such as a PostgreSQL or MongoDB. The service may not exist, may be down, or may reject the binding.
- The app does not have a unique URL. See App Requires Unique URL.

App requires unique URL

TAS for VMs requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and TAS for VMs cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

App fails to start

After `cf push` stages the app and uploads the droplet, the app may fail to start, commonly with a pattern of starting and crashing similar to the following example:

```
-----> Uploading droplet (23M)
...
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 down
...
0 of 1 instances running, 1 failing
FAILED
Start unsuccessful
```

If this happens:

- **Find the reason app is failing and modify your code.** Run `cf events APP-NAME` and `cf logs APP-NAME --recent` and look for messages similar to this:

  ```
  2018-07-20T15:53:26.00-0700   app.crash      app-name   index: 2, reason: CRASHED, cell_id: d874ad05-d0ca-4c63-9f5a-0b1ddd90dd5d, instance: f406c53e-b1ca-4a0f-6140-dddd, exit_description: APP/PROC/WEB: Exited with status 1
  ```

  These messages may identify a memory or port issue. If they do, take that as a starting point when you re-examine and fix your app code.

- **Make sure your app code uses the `PORT` environment variable.** Your app may be failing because it is listening on the wrong port. Instead of hard coding the port on which your app listens, use the `PORT` environment variable.

For example, this Ruby snippet assigns the port value to the `listen_here` variable:

~~~
```
listen_here = ENV['PORT']
```
For more examples specific to your app framework, see the appropriate [buildpack documentation](../../buildpacks/index.html) for your app's language.
~~~

- **Make sure your app adheres to the principles of the Twelve-Factor App and Prepare to Deploy an App.** These texts explain how to prevent situations where your app builds locally but fails to build in the cloud.

- **Verify the timeout configuration of your app.** App health checks use a timeout setting when an app starts up for the first time. See Using App Health Checks. If an app fails to start up due to health check timeout, you might see messages in the logs similar to the following:

  ```
  2017-01-30T14:07:20.39-0800 [CELL/0]     OUT Creating container
  ```

2017-01-30T14:07:20.65-0800 [CELL/0] OUT Successfully created container 2017-01-30T14:07:22.30-0800 [CELL/0] OUT Starting health monitoring of container 2017-01-30T14:08:23.52-0800 [CELL/0] ERR Timed out after 1m0s: health check never passed. 2017-01-

30T14:08:23.57-0800 [CELL/0] OUT Destroying container 2017-01-30T14:08:23.59-0800 [API/0] OUT Process has crashed with type: “web” 2017-01-30T14:08:23.59-0800 [CELL/0] OUT Creating container 2017-01-30T14:08:23.60-0800 [API/0] OUT App instance exited with guid 91086440-bac0-44f0-808f-a034a1ec2ed0 payload: {“instance”=>"", “index”=>0, “reason”=>“CRASHED”, “exit_description”=>“2 error(s) occurred:\n\n* 1 error(s) occurred:\n\n* Exited with status 6\n* 2 error(s) occurred:\n\n* cancelled\n* cancelled”, “crash_count”=>1, “crash_timestamp”=>1485814103565763172, “version”=>“3e6e4232-7e19-4168-9583-1176833d2c71”} 2017-01-30T14:08:23.83-0800 [CELL/0] OUT Successfully destroyed container 2017-01-30T14:08:23.84-0800 [CELL/0] OUT Successfully created container 2017-01-30T14:08:25.41-0800 [CELL/0] OUT Starting health monitoring of container

App consumes too much memory, then crashes

An app that `cf push` has uploaded and started can crash later if it uses too much memory.

Make sure your app is not consuming more memory than it should. When you ran `cf push` and `cf scale`, that configured a limit on the amount of memory your app should use. Check your app’s actual memory usage. If it exceeds the limit, modify the app to use less memory.

Routing conflict

TAS for VMs allows multiple apps, or versions of the same app, to be mapped to the same route. This feature enables Blue-Green deployment. For more information see Using Blue-Green Deployment to Reduce Downtime and Risk.

Routing multiple apps to the same route may cause undesirable behavior in some situations by routing incoming requests randomly to one of the apps on the shared route.

If you suspect a routing conflict, run `cf routes` to check the routes in your installation.

If two apps share a route outside of a Blue-Green deploy strategy, choose one app to re-assign to a different route and follow the procedure below:

1. Remove the existing route from that app by running:

   ```
   cf unmap-route APP-NAME OLD-ROUTE
   ```

2. Map the app to a new, unique route by running:

   ```
   cf map-route APP-NAME NEW-ROUTE
   ```

Gather diagnostic information

Use the techniques in this section to gather diagnostic information and troubleshoot app deployment issues.

Examine environment variables

`cf push` deploys your app to a container on the server. The environment variables in the container govern your app.

You can set environment variables in a manifest created before you deploy. For more information, see Deploying with App Manifests.

You can also set an environment variable with a `cf set-env` command followed by a `cf push` command. You must run `cf push` for the variable to take effect in the container environment.

Use the `cf env` command to view the environment variables that you have set using the `cf set-env` command and the variables in the container environment:

```
$ cf env my-app
 Getting env variables for app my-app in org My-Org / space development as admin...
 OK

 System-Provided:
 {
   "VCAP_SERVICES": {
     "p-mysql-n/a": [
       {
         "credentials": {
           "uri":"postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lraa
         },
         "label": "p-mysql-n/a",
         "name": "p-mysql",
         "syslog_drain_url": "",
         "tags": ["postgres","postgresql","relational"]
       }
     ]
   }
 }

 User-Provided:
 my-env-var: 100
 my-drain: http://drain.example.com
```

View logs

To view app logs streamed in real-time, use the `cf logs APP-NAME` command.

To aggregate your app logs to view log history, bind your app to a syslog drain service. For more information, see Streaming app Logs to Log Management Services.

> **Note:** The Diego architecture does not support the `cf files` command, and in cf CLI v7, the `cd files` command is deprecated for all uses.

Trace Cloud Controller REST API calls

If a command fails or produces unexpected results, re-run it with HTTP tracing enabled to view requests and responses between the Cloud Foundry Command Line Interface (cf CLI) and the Cloud Controller REST API.

For example:

- Re-run `cf push` with `-v`:

  ```
  cf push APP-NAME -v
  ```

- Re-run `cf push` while appending API request diagnostics to a log file:

  ```
  CF_TRACE=PATH-TO-TRACE.LOG cf push APP-NAME
  ```

These examples enable HTTP tracing for a single command only. To enable it for an entire shell session, set the variable first:

```
export CF_TRACE=true
```

```
export CF_TRACE=PATH-TO-TRACE.LOG
```

> **Note:** `CF_TRACE` is a local environment variable that modifies the behavior of the cf CLI. Do not confuse `CF_TRACE` with the variables in the container environment where your apps run.

Analyze Zipkin trace IDs

When the Zipkin feature is enabled in Ops Manager, the Gorouter adds or forwards Zipkin trace IDs and span IDs to HTTP headers. For more information about what the Gorouter provides in the HTTP header, see the HTTP Headers section of the *HTTP Routing* topic.

After adding Zipkin HTTP headers to app logs, developers can use `cf logs myapp` to correlate the trace and span IDs logged by the Gorouter with the trace IDs logged by their app. To correlate trace IDs for a request through multiple apps, each app must forward appropriate values for the headers with requests to other apps.

Use troubleshooting commands

You can investigate app deployment and health using the cf CLI.

Some cf CLI commands may return connection credentials. Remove credentials and other sensitive information from command output before you post the output a public forum.

- `cf apps`: Returns a list of the apps deployed to the current space with deployment options, including the name, current state, and routes for each app.
- `cf app APP-NAME`: Returns the health and status of each instance of a specific app in the current space, including current state and how long it has been running.

  > **Note:** CPU values returned by `cf app` show the total usage of each app instance on all CPU cores on a host VM, where each core contributes 100%. For example, the CPU of a single-threaded app instance on a Diego cell with one core cannot exceed 100%, and four instances sharing the cell cannot exceed an average CPU of 25%. A multi-threaded app instance running alone on a cell with eight cores can draw up to 800% CPU.

- `cf env APP-NAME`: Returns environment variables set using `cf set-env` and variables existing in the container environment.
- `cf events APP-NAME`: Returns information about app crashes, including error codes. Shows that an app instance exited. For more detail, look in the app logs. For a list of Ops Manager errors, see https://github.com/cloudfoundry/errors.
- `cf logs APP-NAME --recent`: Dumps recent logs. For more information, see Viewing Logs in the Command Line Interface.
- `cf logs APP-NAME`: Returns a real-time stream of the app STDOUT and STDERR. Use **Ctrl-C** (^C) to exit the real-time stream.

Commands available in cf CLI v6, but deprecated in cf CLI v7:

- `cf files APP-NAME`: Lists the files in an app directory. Given a path to a file, outputs the contents of that file. Given a path to a subdirectory, lists the files within. Use this to explore individual logs.

> **Note:** Your app should direct its logs to STDOUT and STDERR. The `cf logs` command also returns messages from any log4j facility that you configure to send logs to STDOUT.

Access apps with SSH

If you need to troubleshoot an instance of an app, you can gain SSH access to the app with the SSH proxy and daemon. For more information, see App SSH Overview.

Send requests to app instance

To obtain debug data without SSH, you can make HTTP requests to a specific instance of an app by using the `X-CF-APP-INSTANCE` HTTP header. For more information, see App Instance Routing section of the *HTTP Routing*.

Pushing Apps to TAS for VMs [Windows]

In this section:

- Deploying .NET Apps
- Using SMB Volumes in .NET Apps
- Tips for .NET Framework Developers

Deploying .NET Apps

This topic lists resources for .NET app developers.

For general information about app deployment, see Pushing an App.

.NET Framework on Windows

HWC Buildpack describes how to push .NET Framework apps to VMware Tanzu Application Service for VMs [Windows] using the HWC buildpack. It also describes what features of IIS (Internet Information Services) and HWC (Hostable Web Core) are supported, and how to configure your .NET Framework apps to run on TAS for VMs [Windows].

Console Applications in the *.NET Cookbook* provides information about pushing compiled .NET Framework Console apps to TAS for VMs [Windows] using the binary buildpack.

.NET Core on Windows

Binary Buildpack describes how to push compiled .NET Core apps to Windows Diego Cells on TAS for VMs [Windows] using the binary buildpack.

.NET Core on Linux

.NET Core Buildpack describes how to push .NET Core apps to Linux Diego Cells on TAS for VMs using the .NET Core buildpack. For supported ASP.NET Core versions, see the .NET Core buildpack release notes in the .NET Core buildpack repository on GitHub.

Other Resources

Tips for .NET Framework Developers describes common errors and challenges faced when pushing .NET Framework apps to TAS for VMs [Windows].

The .Net Cookbook provides code samples, guidelines, tips, and other recipes for making changes to both new and existing .NET Framework and .NET Core apps to run successfully on TAS for VMs [Windows].

Using SMB Volumes in .NET Apps

This topic describes how to use Server Message Block (SMB) Volumes in a .NET app. In this example, you create an app that reads and writes to a `note.txt` file within an SMB Volume share.

Prerequisites

To use SMB Volumes in your .NET App, you must have:

- Visual Studio
- An IaaS account
- VMware Tanzu Application Service for VMs [Windows]
- An accessible SMB Volume in an IaaS with a UNC path
- An existing .NET Framework app
- An SMB username
- An SMB password

Using SMB Volumes in .NET Apps with Steeltoe

Steeltoe is a set of libraries that help your team write cloud native apps. Steeltoe contains functionality for mounting SMB Volumes. Using Steeltoe requires modifying your app code. For more information, see the Steeltoe documentation.

If you are using Steeltoe, add the following environment variables to your Config Server Provider:

- `SMB_PATH`
- `SMB_USERNAME`
- `SMB_PASSWORD`

Where:

- `SMB_PATH` is the UNC path to your SMB.
- `SMB_USERNAME` is your IaaS account username.
- `SMB_PASSWORD` is your IaaS account password.

For more information about adding environment variables to your Config Server Provider, see Config Server Provider in the Steeltoe documentation.

After adding your environment variables to your Config Server Provider, proceed to SMB Mounting below.

Using SMB Volumes in .NET Apps with Your Batch Profile

If you do not want to modify your app code to include Steeltoe, you can use mount SMB Volumes without Steeltoe.

To mount SMB Volumes:

1. Retrieve the UNC of your existing SMB share. The UNC is your machine's fully-qualified domain name (FQDN).
2. Create a `.profile.bat` file in your .NET app's root directory.
3. Use Apps Manager or the Cloud Foundry Command Line Interface (cf CLI) to set the following environment variables:
    - `SMB_PATH`
    - `SMB_USERNAME`
    - `SMB_PASSWORD`
4. Add the following command to your `.profile.bat` file:

    ```
    net use z: %SMB_PATH% %SMB_PASSWORD% /USER:%SMB_USERNAME%
    ```

5. Proceed to SMB Mounting below.

SMB Mounting

To configure SMB mounting:

1. Using the cf CLI or Apps Manager, add `SMB_PATH`, `SMB_USERNAME`, and `SMB_PASSWORD` as environment variables to your local computer or the computer you are using to deploy your

app.

> **Note:** If you are using SMB Volumes with your batch profile and have already added these environment variables, ignore this step.

> **Note:** If Visual Studio does not detect these new environment variables, restart Visual Studio.

2. In Visual Studio, create a new file in your solution named `SMBConfiguration.cs`. This file is the single representation of your SMB Volume configuration and reads the connection data from the environment variables you established previously. Include the following in your `SMBConfiguration.cs` file:

```
// SMBConfiguration.cs
using System;

namespace NetFrameworkApp.Controllers
{
    public class SMBConfiguration
    {
        public String GetSharePath()
        {
            return Environment.GetEnvironmentVariable("SMB_PATH");
        }

        public String GetUserName()
        {
            return Environment.GetEnvironmentVariable("SMB_USERNAME");
        }

        public String GetPassword()
        {
            return Environment.GetEnvironmentVariable("SMB_PASSWORD");
        }
    }
}
```

3. In Visual studio, create a new MVC Controller named `NoteController`. This file creates a controller endpoint that reads the example note file. For more information about creating a controller, see Add a controller to an ASP.NET Core MVC app in the Microsoft Documentation.

4. Use your package manager to add `Steeltoe.Common` and `Steeltoe.Common.Net` to your app. If you are not using Steeltoe, ignore this step.

5. Edit `NoteController.cs` to read from a file named `note.txt`. This `note.txt` does not exist yet but will be created by the `FileMode.OpenOrCreate` method. See the example code snippet below, which reads the contents of the note file and stashes the `note.txt` content in the Viewbag. If you are not using Steeltoe, ignore the reference to `Steeltoe.Common.Net` in the following code snippet.

```
// NoteController.cs
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Steeltoe.Common.Net;

namespace NetFrameworkApp.Controllers
{
    public class NoteController : Controller
    {
        SMBConfiguration configuration = new SMBConfiguration();
        public ActionResult Index()
        {

            var credential = new NetworkCredential(configuration.GetUserName(),
configuration.GetPassword());

            using (var share = new WindowsNetworkFileShare(configuration.GetSha
rePath(), credential))
            using (var inputStream = new FileStream(Path.Combine(configuration.
GetSharePath(), "note.txt"), FileMode.OpenOrCreate))
            using (var streamReader = new StreamReader(inputStream))
            {
            // Never display raw user input as HTML. Do not do this in producti
on code.
                ViewBag.Note = streamReader.ReadToEnd();
            }


            return View();
        }
    }
}
```

6. In Visual Studio, create a subdirectory in **Views** named `Note`.
7. In Visual Studio, create a new View named `Index`. For more information about Views, see Views in ASP.NET Core MVC and Add a view to an ASP.NET Core MVC app in the Microsoft documentation.
8. In Visual Studio, create an `Index.cshtml` file that contains the following:

```
// Index.cshtml

@ViewBag.Note
```

If you were to run the app now, you would see an empty page with no errors.

9. Modify the `Index.cshtml` file to contain a form. This form posts to a yet-to-be-created update endpoint and also displays our note inside a text area.

```
<big>
<xmp>
// Index.cshtml
...
<form action="/note/update" method="post">
```

```
    <textarea name="note">@ViewBag.Note</textarea>
    <div>
        <button type="submit">Update</button>
    </div>
</form>
</xmp>
</big>
```

10. Modify the `NoteController.cs` to have an update endpoint. This endpoint saves the data posted to the endpoint back into the `note.txt`. See the example code snippet in the following step.

11. Modify the `NoteController.cs` to include the ControllerBase class `RedirectToAction` method, which redirects to the index page so the user can see what was just saved. For more information about the ControllerBase class, see ControllerBase.RedirectToAction Method in the Microsoft documentation. See the example code snippet below:

```
// NoteController.cs
namespace NetFrameworkApp.Controllers
{
    public class NoteController : Controller
    {
      ...

        [HttpPost]
        public ActionResult Update(String note)
        {
            var credential = new NetworkCredential(configuration.GetUserName(),
configuration.GetPassword());

            using (var share = new WindowsNetworkFileShare(configuration.GetSha
rePath(), credential))
            using (var outputStream = new FileStream(Path.Combine(configuratio
n.GetSharePath(), "note.txt"), FileMode.Create))
            using (var streamWriter = new StreamWriter(outputStream))
            {
                streamWriter.Write(note);
            }

            return RedirectToAction("Index");
        }
    }
}
```

Known Issues

Inability to SSH Into App Instance

Even though you have successfully created the SMB mapping, you cannot `cf ssh` into that app instance. Trying to access the created mapping through SSH results in an unspecified path error.

Tips for .NET Framework Developers

This topic describes how to push .NET Framework apps to VMware Tanzu Application Service for VMs [Windows] Diego Cells.

For information about how to develop microservices for .NET Framework and .NET Core using Steeltoe, see the Steeltoe documentation.

Prerequisites

The TAS for VMs [Windows] tile must be installed and configured. To install TAS for VMs [Windows], see Installing and Configuring TAS for VMs [Windows].

Operators must also install the Cloud Foundry Command Line Interface (cf CLI) to run the commands on this topic. For information about installing the cf CLI, see Installing the cf CLI.

Overview

After operators install and configure the TAS for VMs [Windows] tile, developers can push .NET Framework apps to the Windows Diego Cell. Developers can push both OWIN and non-OWIN apps, and can push apps that are served by Hostable Web Core or self-hosted.

If you have upgraded to TAS for VMs [Windows] and have apps that you want to migrate, see Upgrading Windows Diego Cells.

Develop .NET Framework Apps

.NET on Ops Manager Cookbook

For useful tips and recipes for migrating and developing .NET Framework apps to run on TAS for VMs [Windows], see the .NET Cookbook.

Push a .NET Framework App

By default, Ops Manager serves .NET Framework apps with Hostable Web Core (HWC). HWC is a lighter version of the Internet Information Services (IIS) server that contains the core IIS functionality.

To push a .NET Framework app to a Windows Diego Cell:

1. Target the Cloud Controller of your Ops Manager deployment by running:

   ```
   cf api api.APP-DOMAIN
   ```

   Where `APP-DOMAIN` is your app's public domain name. For example, `example.com`.

2. Run one of the following commands to deploy your .NET app:

   - To deploy an app with `.bat` or `.exe` files, run:

     ```
     cf push -s windows -b binary_buildpack
     ```

   - To deploy a .NET Framework app, run:

```
cf push APP-NAME -s windows -b hwc_buildpack
```

Where APP-NAME is the name of your app.

> **Note:** The -s windows option instructs Ops Manager to run the app in the Windows Diego Cell.

> **Note:** If you are not pushing your app from its directory, use the -p option and specify the path to the directory that contains the app.

3. Wait for your app to stage and start. If you see an error message, see Troubleshoot App Errors below.

Context Path Routing Support for ASP.NET Apps

Context path routing enables multiple apps to share the same route hostname, such as app1.example.com/app2. ASP.NET developers can host apps under a route path. Within Windows Diego Cells, you can have multiple routes to an app, but those routes cannot have different context paths.

Making an app accessible under another app's URL requires a pair of commands. To define a context path route, such as app1.example.com/app2:

1. Push the top-level app by running:

   ```
   cf push TOP-LEVEL-APP-NAME
   ```

   Where TOP-LEVEL-APP-NAME is the name of your top-level app.

2. Push the lower-level app by running:

   ```
   cf push LOWER-LEVEL-APP-NAME -d APP-DOMAIN --hostname TOP-LEVEL-APP-NAME --rout
   e-path LOWER-LEVEL-APP-NAME
   ```

   Where:

   - TOP-LEVEL-APP-NAME is the name of your top-level app.
   - LOWER-LEVEL-APP-NAME is the name of your lower-level app.
   - APP-DOMAIN is your app's public domain name. For example, example.com.

   > **Note:** The -d parameter is only needed when pushing an app to a non-default domain.

Enable Graceful Shutdown for a .NET App

Developers can configure .NET apps to shut down gracefully after running cf stop. When you run cf stop, the .NET app receives a CTRL_SHUTDOWN_EVENT and is allowed ten seconds to shut down. To enable graceful shutdown, you must include a control handler in the app.

For more information, see Graceful Shutdown in the .NET Cookbook.

Push a Self-Hosted App

Developers can choose to push a self-hosted app instead of using Hostable Web Core. Self-hosted apps combine server code with the app code.

To push a self-hosted app:

1. Target the Cloud Controller of your Ops Manager deployment by running:

   ```
   cf api api.APP-DOMAIN
   ```

   Where `APP-DOMAIN` is your app's public domain name. For example, `example.com`.

2. Push your .NET app from the app root by running:

   ```
   cf push APP-NAME -s windows -b binary_buildpack -c PATH-TO-BINARY
   ```

   Where:

   - `APP-NAME` is the name of your app.
   - `PATH-TO-BINARY` is the path to your executable.

3. Wait for your app to stage and start. If you see an error message, see Troubleshoot App Errors below.

Push a SOAP Service

Developers can push Simple Object Access Protocol (SOAP) web services to their Ops Manager deployment by following the procedures in the sections below.

Step 1: Deploy Your Web Service

To deploy a SOAP web service:

1. Develop the service as an ASMX web service in Microsoft Visual Studio.
2. Publish the service to your local file system.
3. Open a command line to the directory containing the published web service.
4. Push your service by running:

   ```
   cf push SOAP-SERVICE-NAME -s windows -b hwc_buildpack -p WEB-SERVICE-DIRECTORY
   -u none
   ```

   Where:

   - `SOAP-SERVICE-NAME` is the name of your service.
   - `WEB-SERVICE-DIRECTORY` is the path to the directory containing the published web service.

   For example:

   ```
   $ cf push webservice -s windows -b hwc_buildpack -u none
   ```

```
requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: webservice.example.com
```

> **Note:** The push command must include the `-s` flag to specify the stack, which instructs Ops Manager to run the app in the Windows Diego Cell.

> **Note:** The `-p` and `-u` parameters are optional parameters. The `-p` parameter is needed only when pushing your service from a directory that does not contain the published web service. The `-u` parameter is needed only when disabling the health check when you do not have a route serving `/`.

5. Confirm your service is running by finding your service's URL in the push command's output and browsing to it. In the example above, the URL for the service would be `http://webservice.example.com`.

Step 2: Modify the WSDL File

Your SOAP web service is now deployed on Ops Manager, but the service's WSDL file contains incorrect port information. The WSDL file must be modified to enable an app to consume your web service. Either you or the service developer can perform the needed modification.

See the following portion of an example WSDL file:

```
- <wsdl:service name="WebService1">
  - <wsdl:port name="WebService1Soap" binding="tns:WebService1Soap">
      <soap:address location="http://webservice.example.com:62492/WebService1.asmx"/>
    </wsdl:port>
  - <wsdl:port name="WebService1Soap12" binding="tns:WebService1Soap12">
      <soap12:address location="http://webservice.example.com:62492/WebService1.asmx"/
>
    </wsdl:port>
- </wsdl:service>
```

The WSDL file provides the port number for the SOAP web service as `62492`. This is the port that the web service listens on in the Garden container, but external apps cannot access the service on this port. Instead, external apps must use port 80, and the Gorouter routes requests to the web service in the container. For more information about the Garden container, see Container Mechanics in *Container Security*. For more information about the Gorouter, see TAS for VMs Routing Architecture.

The URL of the web service in the WSDL file must be modified to remove `62492`. With no port number, the URL defaults to port 80. In the example above, the modified URL would be `http://webservice.example.com/WebService1.asmx`.

SOAP web service developers can resolve this problem in one of two ways:

- Modify the WSDL file by following the instructions in Modify a Web Service's WSDL Using a SoapExtensionReflector from the Microsoft Developers Network.

- Instruct the developers of external apps that consume the web service to follow the procedure in Consume the SOAP Web Service.

Consume the SOAP Web Service

Developers of external apps that consume the SOAP web service can use a modified version of the WSDL file.

To use a modified version of the WSDL file:

1. In a browser, navigate to the WSDL file of the web service, using the following URL:

   ```
   https://SOAP-SERVICE-NAME.APP-DOMAIN/ASMX-FILE?wsdl
   ```

   Where:

   - `SOAP-SERVICE-NAME` is the name of your service.
   - `APP-DOMAIN` is your site's public domain name.
   - `ASMX-FILE` is the filename of your asmx file.

   For example:

   ```
   https://webservice.example.com/WebService1.asmx?wsdl
   ```

2. Download the WSDL file to your local machine.
3. Edit the WSDL file to eliminate the container port, as described in Modify the WSDL File.
4. In Microsoft Visual Studio, right-click on your app in the **Solution Explorer** and select **Add** > **Service Reference**.
5. Under **Address**, enter the local path to the modified WSDL file. For example:

   ```
   C:\Users\example\wsdl.xml
   ```

6. Click **OK**. Microsoft Visual Studio generates a client from the WSDL file that you can use in your codebase.

Context Path Routing Support for SOAP Web Services

Developers can push SOAP web services to their Ops Manager deployment with context path routing. For more information, see Context Path Routing Support for ASP.NET Apps.

Troubleshoot App Errors

If a .NET app fails to start, see the following list of errors and their possible solutions:

- `NoCompatibleCell`: Your Ops Manager deployment cannot connect to your Windows Diego Cell. For more information about troubleshooting your Windows Diego Cell configuration, see Troubleshooting Windows Diego Cells.
- `Start unsuccessful`: Your app may may be misconfigured or lacks the required DLL files and dependencies.

  - Ensure that your app directory contains either a valid `.exe` binary or a valid `Web.config` file.
  - Ensure that you are pushing from a directory containing your app dependencies. If it does not, specify the app dependency directory with the `-p` flag.

Getting Started Deploying Java Apps

In this section:

- Getting Started Deploying Grails Apps
- Getting Started Deploying Ratpack Apps
- Getting Started Deploying Spring Apps

Deploying your Grails apps to Cloud Foundry

This information walks you through deploying a Grails app to Cloud Foundry. If you experience a problem following the steps, refer to the Known Issues or Troubleshooting Application Deployment and Health topics for more information.

If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_grails.git` to clone the `pong_matcher_grails` app from GitHub, and follow the instructions in the Sample app step sections.

Ensure that your Grails app runs locally before continuing with this procedure.

Deploy a Grails application

This section describes how to deploy a Grails application to Cloud Foundry.

Prerequisites

- A Grails app that runs locally on your workstation
- Intermediate to advanced Grails knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

You can develop Grails applications in Groovy, Java 7 or 8, or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7 as described in *Step 8: Configure the Deployment Manifest* of this topic.

Step 1: Declare app dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle, Grails, and Maven, and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide

Build Tool	Build Script	Documentation
Grails	`BuildConfig.groovy`	Grails: Configuration - Reference Documentation
Maven	`pom.xml`	Apache Maven Project Documentation

> **Note**
>
> You can skip this step. The `pong_matcher_grails/app/grails-app/conf/BuildConfig.groovy` file contains the dependencies for the `pong_matcher_grails` sample app, as shown in the following example.

```
dependencies {
        // specify dependencies here under either 'build', 'compile', 'runtime', 'tes
t' or 'provided' scopes e.g.
        // runtime 'mysql:mysql-connector-java:5.1.29'
        // runtime 'org.postgresql:postgresql:9.3-1101-jdbc41'
        test "org.grails:grails-datastore-test-support:1.0-grails-2.4"
        runtime 'mysql:mysql-connector-java:5.1.33'
    }
```

Step 2: Allocate sufficient memory

Run the Cloud Foundry Command Line Interface (cf CLI) `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

> **Note**
>
> You can skip this step. In the `manifest.yml` of the `pong_matcher_grails` sample app, the `memory` sub-block of the `applications` block allocates 1 GB to the app.

Step 3: Provide a JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

> **Note**
>
> You can skip this step. The `pong_matcher_grails` sample app declares a MySQL JDBC driver in the `pong_matcher_grails/app/grails-app/conf/DataSource.groovy`

file because the app uses ClearDB, which is a database-as-service for MySQL-powered apps. The example below shows this declaration.

```
dataSource {
    pooled = true
    jmxExport = true
    driverClassName = "com.mysql.jdbc.Driver"
    dialect = org.hibernate.dialect.MySQL5InnoDBDialect
    uri = new URI(System.env.DATABASE_URL ?: "mysql://foo:bar@localhost")
    username = uri.userInfo ? uri.userInfo.split(":")[0] : ""
    password = uri.userInfo ? uri.userInfo.split(":")[1] : ""
    url = "jdbc:mysql://" + uri.host + uri.path

    properties {
        dbProperties {
            autoReconnect = true
        }
    }
}
```

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

Note

ou can skip this step. The `pong_matcher_grails` app does not require a Procfile.

Step 5: Create and bind a service instance for a Grails application

This section describes using the cf CLI to configure a ClearDB managed service instance for an app. You can use either the CLI or Apps Manager to perform this task. For more information about using Apps Manager, see About Apps Manager.

Cloud Foundry supports two types of service instances:

- Managed services integrate with Cloud Foundry through service brokers that offer services and plans and manage the service calls between Cloud Foundry and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Create a service instance

1. View managed and user-provided services and plans available to you by running:

   ```
   cf marketplace
   ```

The example shows two of the available managed database-as-a-service providers and their offered plans: `cleardb` database-as-a-service for MySQL-powered apps and `postgresql-10-odb` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development
as clouduser@example.com...
OK

service             plans                                     descripti
on

cleardb             spark, boost, amp                         Highly av
ailable MySQL for your apps
postgresql-10-odb   standalone, standalone-replica, general   PostgreSQ
L as a Service
```

2. Create a service instance for your app by running:

```
cf create-service SERVICE PLAN SERVICE-INSTANCE
```

Where:

- `SERVICE` and `PLAN` are chosen from the output of the previous step.
- `SERVICE-INSTANCE` is a unique name you provide for the service instance.

Note

Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `spark` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as c
louduser@example.com....
OK
```

Bind a service instance

When you bind an app to a service instance, Cloud Foundry writes information about the service instance to the `VCAP_SERVICES` app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider’s documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE-INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using Apps Manager. For more information about using Apps Manager, see Adding and Binding services using Apps Manager.

Note

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
...
  services:
    - mysql
```

Step 6: Configure the deployment manifest file

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with application manifests topic for more information.

Note

You can skip this step. The `manifest.yml` file for the `pong_matcher_grails` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and target the API endpoint

Enter your log in credentials, and select a space and org.

```
cf login -a API-ENDPOINT
```

Where `API-ENDPOINT` is the URL of the Cloud Controller in your TAS for VMs instance.

Note

You must do this step to run the sample app.

Step 8: Deploying the application

You must use the cf CLI to deploy apps.

From the root directory of your application, run the following command to deploy your application:

```
cf push APP-NAME -p PATH-TO-FILE.war
```

You must deploy the `.war` artifact for a Grails app, and you must include the path to the `.war` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Grails section in the Tips for Java Developers topic.

The `cf push` command creates a URL route to your application in the form `HOST.DOMAIN`, where `HOST` is your `APP-NAME` and `DOMAIN` is specified by your administrator. Your `DOMAIN` is`shared-domain.example.com`.

For example, `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that Cloud Foundry hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Note

1. Change to the `app`directory, and run `./grailsw war` to build the app.
2. Run `cf push pong_matcher_grails -n HOST-NAME` to push the app.

```
<br />
Example: <code>cf push pong_matcher_grails -n my-grails-app</code>
<br /><br />
<b>This example works for cf CLI v6. The <code>-n</code> flag is not supported for cf
CLI v7/v8. Hostname must be set using the <code>routes</code> property in the manifes
t.</b>
```

You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The following example shows the terminal output of deploying the `pong_matcher_grails` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and follows the instructions in the manifest to start two instances of the app, allocating 1 GB of memory between the instances. After the instances start, the output displays their health and status. This example works for cf CLI v6. The `-n` flag is not supported for cf CLI v7/v8. Hostname must be set using the `routes` property in the manifest.

```
$ cf push pong_matcher_grails -n my-grails-app
Using manifest file /Users/example/workspace/pong_matcher_grails/app/manifes
```

```
t.yml

Creating app pong_matcher_grails in org Cloud-Apps / space development as clo
uduser@example.com...
OK

Creating route my-grails-app.cfapps.io...
OK

Binding my-grails-app.cfapps.io to pong_matcher_grails...
OK

Uploading pong_matcher_grails...
Uploading app files from: /Users/example/workspace/pong_matcher_grails/app/ta
rget/pong_matcher_grails-0.1.war
Uploading 4.8M, 704 files
OK
Binding service mysql to app pong_matcher_grails in org Cloud-Apps / space de
velopment as clouduser@example.com...
OK

Starting app pong_matcher_grails in org Cloud-Apps / space development as clo
uduser@example.com...
OK
-----> Downloaded app package (38M)
-----> Java Buildpack Version: v2.5 | https://github.com/cloudfoundry/java-bu
ildpack.git#840500e
-----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.i
o/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz (1.5s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://dow
nload.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_RELEASE.
jar (0.0s)
       Modifying /WEB-INF/web.xml for Auto Reconfiguration
-----> Downloading Tomcat Instance 8.0.14 from https://download.run.pivotal.i
o/tomcat/tomcat-8.0.14.tar.gz (0.4s)
       Expanding Tomcat to .java-buildpack/tomcat (0.1s)
-----> Downloading Tomcat Lifecycle Support 2.4.0_RELEASE from https://downlo
ad.run.pivotal.io/tomcat-lifecycle-support/tomcat-lifecycle-support-2.4.0_REL
EASE.jar (0.0s)
-----> Downloading Tomcat Logging Support 2.4.0_RELEASE from https://downloa
d.run.pivotal.io/tomcat-logging-support/tomcat-logging-support-2.4.0_RELEASE.
jar (0.0s)
-----> Downloading Tomcat Access Logging Support 2.4.0_RELEASE from https://d
ownload.run.pivotal.io/tomcat-access-logging-support/tomcat-access-logging-su
pport-2.4.0_RELEASE.jar (0.0s)

-----> Uploading droplet (83M)

0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
```

```
0 of 2 instances running, 2 starting
2 of 2 instances running

App started

Showing health and status for app pong_matcher_grails in org Cloud-Apps / spa
ce development as clouduser@example.com...
OK

requested state: started
instances: 2/2
usage: 1G x 2 instances
urls: my-grails-app.cfapps.io

     state     since                    cpu    memory         disk
#0   running   2014-11-10 05:07:33 PM   0.0%   686.4M of 1G   153.6M of 1G
#1   running   2014-11-10 05:07:36 PM   0.0%   677.2M of 1G   153.6M of 1G
```

Step 9: Testing your deployed app

Use the cf CLI or About Apps Manager to review information and administer your app and your Cloud Foundry account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. For more information about using Apps Manager, see Using Apps Manager.

Sample app step
To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
`curl -v -X DELETE $HOST/all`
You should get a response of `200`.

3. To request a match as "andrew", run:
`curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'`
You should again get a response of `200`.

4. To request a match as a different player, run:
`curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'`

5. To check the status of the first match request, run:
`curl -v -X GET $HOST/match_requests/firstrequest`
The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:
```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' {
"match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova" }'
```
You receive a `201 Created` response.

Manage your application with the cf CLI

Run the `cf help` command to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

You cannot perform certain tasks in the CLI or About Apps Manager because these are commands that only an administrator can run. If you are not an administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with Apps Manager, depending on your user role, see Getting Started with Apps Manager.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App deploy fails

Even when the deploy fails, the app might exist on Cloud Foundry. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or About Apps Manager, or you might have to delete the app and redeploy it.

App requires a unique URL

Cloud Foundry requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Cloud Foundry cannot successfully deploy the app. You can fix this issue by running `cf push` with the `--random-route` flag to create a unique URL. Using `--random-route` to create a URL that includes the app name and random words might create a long URL, depending on the number of words that the app name includes.

Deploying Ratpack apps to Cloud Foundry

This information walks you through deploying a Ratpack app to Cloud Foundry. If you experience a problem following the steps, check the Known Issues topic or refer to the Troubleshooting Application Deployment and Health topic.

Sample app step
If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_groovy.git` to clone the `pong_matcher_groovy` app from GitHub, and follow the instructions in the Sample app step sections.

Ensure that your Ratpack app runs locally before continuing with this procedure.

Deploy a Ratpack application

This section describes how to deploy a Ratpack application to Cloud Foundry.

Prerequisites

- A Ratpack app that runs locally on your workstation
- Intermediate to advanced Ratpack knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

You can develop Ratpack applications in Java 7 or 8 or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7. Refer to Step 8: Configure the Deployment Manifest.

Step 1: Declare app dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide
Maven	`pom.xml`	Apache Maven Project Documentation

You can skip this step. The `build.gradle` file contains the dependencies for the `pong_matcher_groovy` sample app, as the example below shows.

```
dependencies {
  // SpringLoaded enables runtime hot reloading.
  // It is not part of the app runtime and is not shipped in the distribution.
  springloaded "org.springframework:springloaded:1.2.0.RELEASE"

  // Default SLF4J binding.  Note, this is a blocking implementation.
  // See here for a non blocking appender http://logging.apache.org/log4j/2.x/manual/a
sync.html
  runtime 'org.slf4j:slf4j-simple:1.7.7'

  compile group: 'redis.clients', name: 'jedis', version: '2.5.2', transitive: true

  testCompile "org.spockframework:spock-core:0.7-groovy-2.0"
}
```

Step 2: Allocate sufficient memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

You can skip this step. In the `manifest.yml` of the `pong_matcher_groovy` sample app, the `memory` sub-block of the `applications` block allocates 512 MB to the app.

Step 3: Provide a JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

You can skip this step. The `pong_matcher_groovy` sample app does not require a JDBC driver.

Step 4: (Optional) Configure a Procfile

Use a Procfile to declare required runtime processes for your web app and to specify your web server. For more information, see the Configuring a Production Server topic.

Sample app step
You can skip this step. The `pong_matcher_groovy` app does not require a Procfile.

Step 5: Create and bind a service instance for a Ratpack application

Learn how to use the CLI to configure a Redis managed service instance for an app. You can use either the CLI or Apps Manager to perform this task. For more information about using Apps Manager, see About Apps Manager.

Cloud Foundry supports the following types of service instances:

- Managed services integrate with Cloud Foundry through service brokers that offer services and plans and manage the service calls between Cloud Foundry and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the Services Overview topic.

Creating a service instance

1. View managed and user-provided services and plans available to you by running:

   ```
   cf marketplace
   ```

   The example shows two of the available managed database-as-a-service providers and their offered plans: `postgresql-10-odb` PostgreSQL as a Service and `rediscloud` Enterprise-Class Redis for Developers.

   ```
   $ cf marketplace
   Getting services from marketplace in org Cloud-Apps / space development
   as clouduser@example.com...
   OK
   ```

```
service            plans                                       descripti
on

postgresql-10-odb  standalone, standalone-replica, general    PostgreSQ
L as a Service

rediscloud         30mb, 100mb, 1gb, 10gb, 50gb               Enterpris
e-Class Redis for Developers
```

2. Create a service instance for your app by running:

   ```
   cf create-service SERVICE PLAN SERVICE-INSTANCE
   ```

   Where:

   - `SERVICE` and `PLAN` are chosen from the output of the previous step.
   - `SERVICE-INSTANCE` is a unique name you provide for the service instance.

Run `cf create-service rediscloud 30mb baby-redis`. This creates a service instance named `baby-redis` that uses the `rediscloud` service and the `30mb` plan, as the following example shows.

```
$ cf create-service rediscloud 30mb baby-redis
Creating service baby-redis in org Cloud-Apps / space development as clouduse
r@example.com....
OK
```

Bind a service instance

When you bind an app to a service instance, Cloud Foundry writes information about the service instance to the `VCAP_SERVICES` app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command

```
cf bind-service APPLICATION SERVICE_INSTANCE
```

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using Apps Manager. For more information about using Apps Manager, see Adding and Binding services using Apps Manager.

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
...
  services:
    - baby-redis
```

Step 6: Configure the deployment manifest

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to Deploying with application manifests for more information.

You can skip this step. The `manifest.yml` file for the `pong_matcher_groovy` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and target the API endpoint

Enter your log in credentials, and select a space and org.

```
cf login -a API-ENDPOINT
```

Where `API-ENDPOINT` is the URL of the Cloud Controller in your TAS for VMs instance.

Sample app step
You must perform this step to run the sample app.

Step 8: Deploy the app

You must use the cf CLI to deploy apps.

From the root directory of your app, run the following command to deploy your application:

```
cf push APP-NAME -p PATH-TO-FILE.distZip
```

You must deploy the `.distZip` artifact for a Ratpack app, and you must include the path to the `.distZip` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the Tips for Java Developers topic.

The `cf push` command creates a URL route to your application in the form `HOST.DOMAIN`, where `HOST` is your `APP-NAME` and `DOMAIN` is specified by your administrator. Your `DOMAIN` is`shared-domain.example.com`.

For example, `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that Cloud Foundry hosts or the push fails. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample app step 1. Change to the `app` directory, and run `./gradlew distZip` to build the app. 1. Run `cf push pong_matcher_groovy -n HOST-NAME` to push the app. Example: `cf push pong_matcher_groovy -n groovy-ratpack-app`

This example is valid for cf CLI v6. In cf CLI v7 and onwards, the `-n` flag is not supported. The host name must be specified in the manifest, using the `routes` attribute.

You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The following example shows the terminal output of deploying the `pong_matcher_groovy` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `baby-redis` service and follows the instructions in the manifest to start one instance of the app with 512 MB. After the app starts, the output displays the health and status of the app. Note, this example is valid for cf CLI v6. In cf CLI v7 and onwards, the `-n` flag is not supported. The host name must be specified in the manifest, using the `routes` attribute.

```
$ cf push pong_matcher_groovy -n groovy-ratpack-app
Using manifest file /Users/example/workspace/pong_matcher_groovy/app/manifes
t.yml

Creating app pong_matcher_groovy in org Cloud-Apps / space development as clo
uduser@example.com...
OK

Creating route groovy-ratpack-app.cfapps.io...
OK

Binding groovy-ratpack-app.cfapps.io to pong_matcher_groovy...
OK

Uploading pong_matcher_groovy...
Uploading app files from: /Users/example/workspace/pong_matcher_groovy/app/bu
ild/distributions/app.zip
Uploading 138.2K, 18 files
OK
Binding service baby-redis to app pong_matcher_groovy in org Cloud-Apps / spa
ce development as clouduser@example.com...
OK

Starting app pong_matcher_groovy in org Cloud-Apps / space development as clo
uduser@example.com...
OK
-----> Downloaded app package (12M)
Cloning into '/tmp/buildpacks/java-buildpack'...
-----> Java Buildpack Version: 9e096be | https://github.com/cloudfoundry/java
```

```
-buildpack#9e096be
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.3s)
-----> Uploading droplet (49M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_groovy in org Cloud-Apps / spa
ce development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: groovy-ratpack-app.cfapps.io

     state     since                    cpu    memory           disk
#0   running   2014-10-28 04:48:58 PM   0.0%   193.5M of 512M   111.7M of 1G
```

Step 9: Test your deployed app

Use the cf CLI or About Apps Manager to review information and administer your app and your account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the Manage Your Application with the cf CLI section for more information. For more information about using Apps Manager, see Using Apps Manager.

Sample app step
To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:
`curl -v -X DELETE $HOST/all`
You should get a response of `200`.

3. To request a match as "andrew", run:
`curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'`
You should again get a response of `200`.

4. To request a match as a different player, run:
`curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'`

5. To check the status of the first match request, run:

```
curl -v -X GET $HOST/match_requests/firstrequest
```

The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' {
"match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova" }'
```

You should receive a `201 Created` response.

Manage your application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

You cannot perform certain tasks in the CLI or About Apps Manager because these are commands that only an administrator can run. If you are not an administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with Apps Manager, depending on your user role, see Getting Started with Apps Manager.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App deploy fails

Even when the deploy fails, the app might exist on Cloud Foundry. Run `cf apps` to review the apps in the targeted org and space. You might be able to correct the issue using the CLI or About Apps Manager, or you might have to delete the app and redeploy.

App requires a unique URL

Cloud Foundry requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Cloud Foundry cannot successfully deploy the app. You can fix this issue by running `cf push` with the `--random-route` flag to create a unique URL. Using `--random-route` to create a URL that includes the app name and random words might create a long URL, depending on the number of words that the app name includes.

Deploying Spring apps to Cloud Foundry

This guide walks you through deploying a Spring app to Cloud Foundry. You can choose whether to push a sample app, your own app, or both.

If you experience a problem following the steps, see the Known Issues topic, or refer to the Troubleshooting Application Deployment and Health topic.

If you want to go through this tutorial using the sample app, run `git clone https://github.com/cloudfoundry-samples/pong_matcher_spring` to clone the `pong_matcher_spring` app from GitHub, and follow the instructions in the Sample app step sections.

Ensure that your Spring app runs locally before continuing with this procedure.

Deploy a Spring application

This section describes how to deploy your Spring application to Cloud Foundry.

Prerequisites

- A Spring app that runs locally on your workstation
- Intermediate to advanced Spring knowledge
- The Cloud Foundry Command Line Interface (cf CLI)
- JDK 1.6, 1.7, or 1.8 for Java 6, 7, or 8 configured on your workstation

The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to an earlier version. For more information, refer to Creating custom buildpacks

Step 1: Declare app dependencies

Make sure to declare all the dependency tasks for your app in the build script of your chosen build tool.

The Spring Getting Started Guides demonstrate features and functionality you can add to your app, such as consuming RESTful services or integrating data. These guides contain Gradle and Maven build script examples with dependencies. You can copy the code for the dependencies into your build script.

The following table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	`build.gradle`	Gradle User Guide
Maven	`pom.xml`	Apache Maven Project Documentation

Sample app step

You can skip this step. The `pom.xml` file contains the dependencies for the `pong_matcher_spring` sample app, as the example below shows.

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.jayway.jsonpath</groupId>
        <artifactId>json-path</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Make sure you are not building fully executable jars because application push might fail.

Step 2: Allocate sufficient memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

You can skip this step. The Cloud Foundry Java buildpack uses settings declared in the sample app to allocate 1 GB of memory to the app.

Step 3: Provide a JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample app step
You can skip this step. In the `pong_matcher_spring` sample app, the `src/main/resources/application.yml` file declares the JDBC driver, and the `pom.xml` file includes the JDBC driver as a dependency.

Step 4: Configure service connections for a Spring app

Cloud Foundry provides extensive support for creating and binding a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. For more information about creating and binding a service connection for your app, refer to Configuring Service Connections.

Sample app step: Create a Service Instance

Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `spark` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as a.user@exampl
e.com....
OK
```

Sample app step: Bind a Service Instance

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
...
  services:
    - mysql
```

Step 5: Configure the deployment manifest

You can specify deployment options in a manifest file `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the Deploying with Application Manifests topic for more information.

Sample app step

You can skip this step. The `manifest.yml` file for the `pong_matcher_spring` sample app does not require any additional configuration to deploy the app.

Step 6: Log in and target the API endpoint

Enter your log in credentials, and select a space and org.

```
cf login -a API-ENDPOINT
```

Where `API-ENDPOINT` is the URL of the Cloud Controller in your TAS for VMs instance.

You must do this step to run the sample app.

Step 7: Deploying your app

You must use the cf CLI to deploy apps.

From the root directory of your app, run the following command to deploy your application.

```
cf push APP-NAME -p PATH-TO-FILE.jar
```

Most Spring apps include an artifact, such as a `.jar`, `.war`, or `.zip` file. You must include the path to this file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. The example shows how to specify a path to the `.jar` file for a Spring app. Refer to the Tips for Java Developers topic for CLI examples for specific build tools, frameworks, and languages that create an app with an artifact.

The `cf push` command creates a URL route to your application in the form `HOST.DOMAIN`, where `HOST` is your `APP-NAME` and `DOMAIN` is specified by your administrator. Your `DOMAIN` is `shared-domain.example.com`.

For example, `cf push my-app` creates the URL `my-app.shared-domain.example.com`.

The URL for your app must be unique from other apps that Cloud Foundry hosts or the push fails. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, go to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample app step 1. Run `brew install maven`. 1. Change to the `app` directory, and run `mvn package` to build the app. 1. Run `cf push pong_matcher_spring -n HOSTNAME` to push the app. Example: `cf push pong_matcher_spring -n my-spring-app`.

This example works for cf CLI v6. The `-n` flag is not supported for cf CLI v7/v8. Hostname must be set using the `routes` property in the manifest.

You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The following example shows the terminal output of deploying the `pong_matcher_spring` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and starts one instance of the app with 1 GB of memory. After the app starts, the output displays the health and status of the app. This example works for cf CLI v6. The `-n` flag is not supported for cf CLI v7/v8. Hostname must be set using the `routes` property in the manifest.

```
$ cf push pong_matcher_spring -n spring1119
Using manifest file /Users/example/workspace/pong_matcher_spring/manifest.yml

Creating app pong_matcher_spring in org Cloud-Apps / space development as a.u
ser@example.com...
OK

Creating route spring1119.cfapps.io...
OK

Binding spring1119.cfapps.io to pong_matcher_spring...
OK
```

```
Uploading pong_matcher_spring...
Uploading app files from: /Users/example/workspace/pong_matcher_spring/targe
t/pong-matcher-spring-1.0.0.BUILD-SNAPSHOT.jar
Uploading 797.5K, 116 files
OK
Binding service mysql to app pong_matcher_spring in org Cloud-Apps / space de
velopment as a.user@example.com...
OK

Starting app pong_matcher_spring in org Cloud-Apps / space development as a.u
ser@example.com...
OK
-----> Downloaded app package (25M)
-----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.i
o/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz (1.2s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://dow
nload.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_RELEASE.
jar (0.1s)

-----> Uploading droplet (63M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_spring in org Cloud-Apps / spa
ce development as a.user@example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: spring1119.cfapps.io

     state     since                    cpu    memory         disk
#0   running   2014-11-19 12:29:27 PM   0.0%   553.6M of 1G   127.4M of 1G
```

Step 8: Testing your deployed app

Use the cf CLI or About Apps Manager to review information and administer your app and your Cloud Foundry account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See Manage your application with the cf CLI for more information. For more information about using Apps Manager, see Using Apps Manager.

Sample app step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

1. To clear the database from any previous tests, run:
```
curl -v -X DELETE $HOST/all
```
You should get a response of 200.

1. To request a match as "andrew", run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d
'{"player": "andrew"}'
```
You should again get a response of `200`.

1. To request a match as a different player, run:
```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d
'{"player": "navratilova"}'
```

1. To check the status of the first match request, run:
```
curl -v -X GET $HOST/match_requests/firstrequest
```
The last line of the output shows the `match_id`.

1. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:
```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' {
"match_id":"MATCH_ID", "winner":"andrew", "loser":"navratilova" }'
```
You should receive a `201 Created` response.

Manage your app with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the Cloud Foundry Command Line Interface (cf CLI) topics, especially the Getting Started with cf CLI topic.

You cannot perform certain tasks in the CLI or About Apps Manager because these are commands that only an administrator can run. If you are not an administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with Apps Manager, depending on your user role, see Getting Started with Apps Manager.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the Troubleshooting Application Deployment and Health topic to learn more about troubleshooting.

App deploy fails

Even when the deploy fails, the app might exist on Cloud Foundry. Run the `cf apps` command to review the apps in the targeted org and space. You might be able to correct the issue using the

CLI or About Apps Manager, or you might have to delete the app and redeploy it.

App requires a Content-Type

If you specify a `Content-Encoding` header of `gzip` but do not specify a `Content-Type` within your application, Cloud Foundry might send a `Content-Type` of `application/x-gzip` to the browser. This scenario might cause the deploy to fail if it conflicts with the actual encoded content of your app. To avoid this issue, be sure to explicitly set `Content-Type` within your app.

App requires a unique URL

Cloud Foundry requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Cloud Foundry cannot successfully deploy the app. You can fix this issue by running `cf push` with the `--random-route` flag to create a unique URL. Using `--random-route` to create a URL that includes the app name and random words might create a long URL, depending on the number of words that the app name includes.

Managing Apps

In this section:

Using Apps Manager

In this section:

About Apps Manager

The web-based Apps Manager application helps you manage users, organizations, spaces, and applications.

Apps Manager provides a visual interface for performing the following subset of functions available through the Cloud Foundry Command Line Interface (cf CLI):

- **Orgs**: You can create and manage orgs.
- **Spaces**: You can create, manage, and delete spaces.
- **Apps**: You can scale apps, bind apps to services, manage environment variables and routes, view logs and usage information, start and stop apps, and delete apps.
- **Services**: You can bind services to apps, unbind services from apps, choose and edit service plans, and rename and delete service instances.
- **Users**: You can invite new users, manage user roles, and delete users.

To access Apps Manager as the admin user, see Logging in to Apps Manager.

Getting Started with Apps Manager

This topic describes the functions and scope of Apps Manager, a web-based tool for managing VMware Tanzu Application Service for VMs (TAS for VMs) organizations, spaces, apps, services, and users.

Browser Support

Apps Manager is compatible with the latest major versions of the following browsers:

- Apple Safari
- Google Chrome
- Microsoft Edge
- Microsoft Internet Explorer
- Mozilla Firefox

VMware recommends using Chrome, Firefox, Edge, or Safari for the best Apps Manager experience.

About Permissions

Your ability to perform actions in Apps Manager depends on your user role and the feature flags that the admin sets. For more information about feature flags, see Using Feature Flags.

The following table shows the relationship between specific org and space management actions and the non-admin user roles who can perform them. A non-admin user must be a member of the org and space to perform these actions.

Admin users can perform all of these actions using either the cf CLI or by logging into Apps Manager as an Org Manager, using the UAA Admin credentials.

Space Managers assign and remove users from spaces by setting and unsetting their roles within the space.

Action	CLI command	Org Manager	Space Manager	Org Auditor, Space Developer, or Space Auditor
Create an org	`create-org`	†	†	†
Delete an org	`delete-org`	No	No	No
Rename an org	`rename-org`	Yes	No	No
View org members	`org-users`	Yes	Yes	Yes
Assign user a role in org	`set-org-role`	‡	‡	No
Remove org role from user	`unset-org-role`	‡	‡	No
View space members	`space-users`	Yes	Yes	Yes
Assign user a role in space	`set-space-role`	‡	‡	No

Remove space role from user	`unset-space-role`	‡	‡	No

[†]Defaults to no. Yes if feature flag `user_org_creation` is set to `true`.

[‡]Defaults to no. Yes if feature flags `set_roles_by_username` and `unset_roles_by_username` are set to `true`.

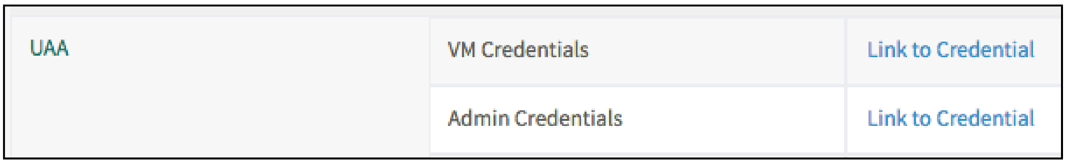
Logging In to Apps Manager

This topic describes how to log in to Apps Manager.

Log In as Admin User

To log in to Apps Manager as the Admin user:

1. If you do not know the system domain for the deployment, select the **Domains** pane in the VMware Tanzu Application Service for VMs (TAS for VMs) tile and see the value configured in the **System domain** field.
2. Open a browser and go to `apps.SYSTEM-DOMAIN`, where `SYSTEM-DOMAIN` is the system domain you retrieved in the previous step. For example, if the system domain is `system.example.com`, then point your browser to `apps.system.example.com`.
3. Log in using UAA credentials for the Admin user. To obtain these credentials:
    1. In the TAS for VMs tile, select the **Credentials** tab.
    2. Under **UAA**, see the **Admin Credentials**.



UAA	VM Credentials	Link to Credential
	Admin Credentials	Link to Credential

Managing Orgs and Spaces Using Apps Manager

This topic explains how to view and manage orgs and spaces in Apps Manager.

Note: To manage a space, you must have Space Manager permissions in that space.

Overview

You can use Apps Manager to manage orgs and spaces. This includes creating and deleting orgs, domains, and spaces, managing member permissions, adding metadata, and more.

For information about managing apps and service instances, see Managing Apps and Service Instances Using Apps Manager.

You can manage orgs and spaces across multiple foundations. For more information, see [Configuring Multi-Foundation Support in Apps Manager](../operating/configure-multi-foundation.html).

Prerequisite

To do the following, you must first log in to Apps Manager with an account that has adequate permissions.

For more information, see About Permissions in *Getting Started with Apps Manager*.

Manage an Org

You can manage org resources, including the spaces, domains, and members associated with the selected space, in Apps Manager. You can also manage org settings.

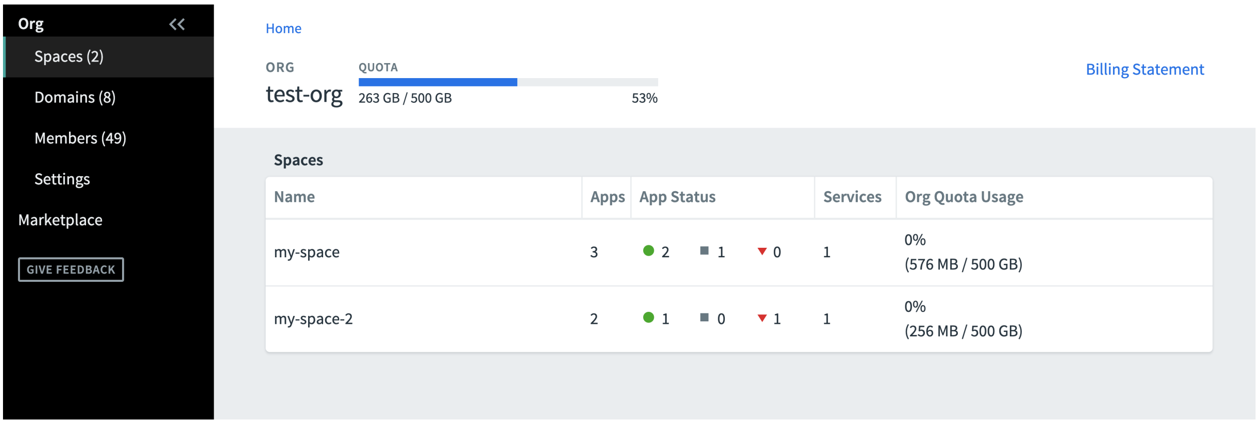
You can manage an org by navigating to the org in Apps Manager.

To navigate to an org, do one of the following:

- On the Apps Manager home page, under **Orgs**, click the name of the org that you want to manage.
- Search for the org by entering its name in the search bar.

By default, the landing page for an org in Apps Manager is the org **Spaces** page. You can use the panel on the left side of the screen to navigate to other pages for resources in the org, such as domains and members.

The following image shows an example of the landing page for an org in Apps Manager.



ORG
test-org

QUOTA
263 GB / 500 GB 53%

Spaces

Name	Apps	App Status	Services	Org Quota Usage
my-space	3	● 2 ■ 1 ▼ 0	1	0% (576 MB / 500 GB)
my-space-2	2	● 1 ■ 0 ▼ 1	1	0% (256 MB / 500 GB)

You can manage the following resources in an org using Apps Manager:

- Spaces. See Manage Spaces.
- Domains. See Manage Domains.
- Members. See Manage Members.
- Org settings. See Manage Settings.

Manage Spaces

In the org **Spaces** tab, you can do the following:

- See the following information for each space:
  - Name
  - Number of apps

  - Number of apps in running, stopped, and crashed states
  - Number of services
  - Memory quota usage
- Create a new space.

Manage Domains

In the org **Domains** page, you can do the following:

- See each **Domain** in your org.
- See the **Type** associated with each domain. A type is either **Globally Shared** or **Privately Shared**. For more information about globally and locally shared domains, see Shared Domains and Private Domains in *Configuring Routes and Domains*.
- Delete a domain by clicking the vertical three-dot icon and selecting **Delete**. You must have permission to manage the domain to delete it.
- Create a new private domain by clicking **Add a Domain**. You must have Org Manager permissions to create a private domain.

> **Note:** To create globally shared domains, you must have Admin permissions and use the Cloud Foundry Command Line Interface (cf CLI). For more information, see [Create a Shared Domain](https://docs.pivotal.io/application-service/2-12//devguide/deploy-apps/routes-domains.html#create-a-shared-domain).

Manage Members

In the org **Members** page, you can do the following:

- See member email addresses
- Toggle member permissions
- Remove member from org
- Invite a new member

Manage Settings

In the org **Settings** page, you can do the following:

- Change the org name
- See spaces assigned to isolation segments. For more information, see Isolation Segments in *TAS for VMs Security*.
- Delete the org. You must have admin permissions to delete an org.

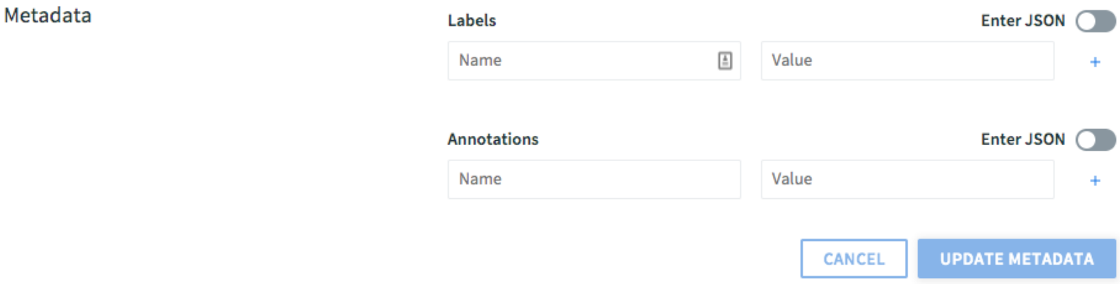
Add Metadata

You can add metadata, including labels and annotations, to orgs using Apps Manager.

For more information about adding metadata to objects in VMware Tanzu Application Service for VMs (TAS for VMs), see Using Metadata.

To add labels or annotations to an org using Apps Manager:

1. Click the **Settings** tab of the desired org and navigate to the **Metadata** section.



2. For **Labels**, complete the **Name** and **Value** fields. Or, select **Enter JSON** to enter the label in JSON.
3. For **Annotations**, complete the **Name** and **Value** fields. Or, select **Enter JSON** to enter the annotation in JSON.
4. To add more labels or annotations, click **+** next to the **Labels** or **Annotations** field.
5. Click **Update Metadata**.

Manage a Space

You can manage space resources, including the apps, service instances, routes, and members associated with the selected space, in Apps Manager. You can also manage space settings.

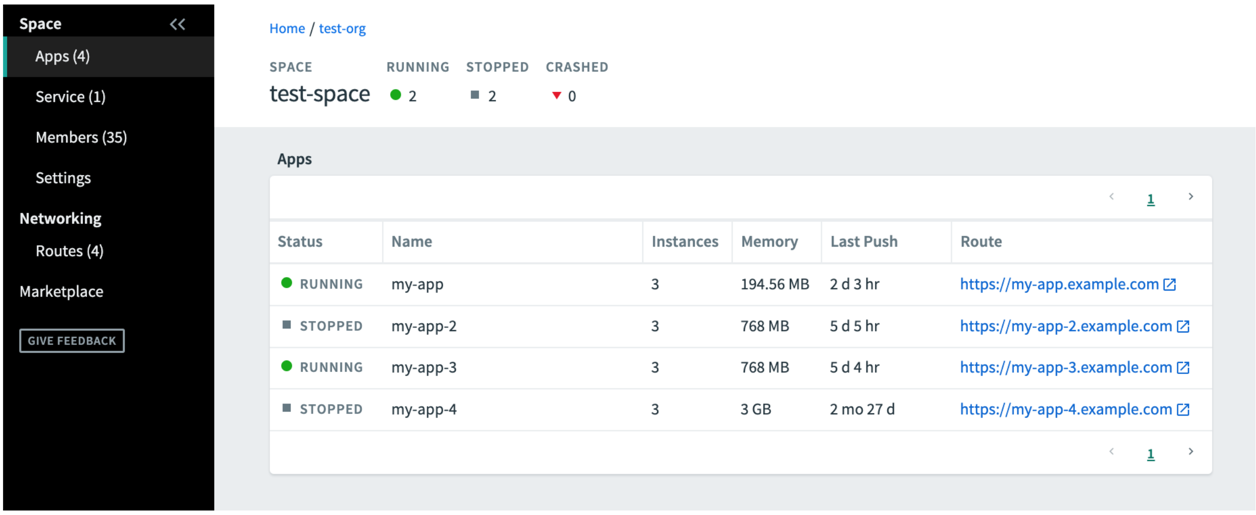
You can manage a space by navigating to the space in Apps Manager.

To navigate to the space, do one of the following:

- On the org page, click the name of the space you want to manage.
- Search for a space by entering its name in the search bar.

By default, the landing page for a space in Apps Manager is the space **Apps** page. You can use the panel on the left side of the screen to navigate to other pages for resources in the space, such as services and members.

The following image shows an example of the landing page for a space in Apps Manager.



SPACE	RUNNING	STOPPED	CRASHED
test-space	● 2	■ 2	▼ 0

Apps

Status	Name	Instances	Memory	Last Push	Route
● RUNNING	my-app	3	194.56 MB	2 d 3 hr	https://my-app.example.com
■ STOPPED	my-app-2	3	768 MB	5 d 5 hr	https://my-app-2.example.com
● RUNNING	my-app-3	3	768 MB	5 d 4 hr	https://my-app-3.example.com
■ STOPPED	my-app-4	3	3 GB	2 mo 27 d	https://my-app-4.example.com

You can manage the following resources in a space using Apps Manager:

- Apps. See Manage Apps.
- Services. See Manage Services.
- Members. See Manage Members.
- Routes. See Manage Routes.
- Space settings. See Manage Settings.

Manage Apps

The **Apps** page lists information about all apps in the space. For each app, it shows the **Status**, **Name**, number of **Instances**, amount of **Memory** available, time since the **Last Push**, and associated **Route**.

Manage Services

The **Services** page shows the **Service**, **Name**, number of **Bound Apps**, and **Plan** for each service instance in the space.

For more information about configuring services, see Services.

Manage Members

The **Members** page lists all members of the space. For each member, the page also lists the associated user roles.

The following user roles can be assigned to space members on the **Members** page:

- **Space Manager**: Space Managers can invite and manage users in the space.
- **Space Developer**: Space Developers can create, manage, and delete apps and services in a space.
- **Space Auditor**: Space Auditors have read-only access to space details, app logs, and reports.

Manage Routes

The **Routes** page lists the routes associated with the space.

For each **Route**, the page shows the **Route Service** and **Bound Apps**. For more information about route services, see Route Services.

Manage Settings

From the space **Settings** page, you can do the following:

- Modify the space name.
- View the Application Security Groups (ASGs) associated with the space.
- View the space's isolation segment assignment. For more information, see Isolation Segments in *TAS for VMs Security*.
- Add metadata to the space. For more information, see Add Metadata.
- Delete the space.

Add Metadata

You can add metadata, including labels and annotations spaces using Apps Manager.

For more information about adding metadata to objects in TAS for VMs, see Using Metadata.

To add labels or annotations to a space using Apps Manager:

1. Click the **Settings** tab of the desired space and navigate to the **Metadata** section.
2. For **Labels**, complete the **Name** and **Value** fields. Or, select **Enter JSON** to enter the label in JSON.
3. For **Annotations**, complete the **Name** and **Value** fields. Or, select **Enter JSON** to enter the annotation in JSON.
4. To add more labels or annotations, click **+** next to the **Labels** or **Annotations** field.
5. Click **Update Metadata**.

Managing User Roles with Apps Manager

This topic explains how to manage user roles with Apps Manager.

> **Note:** The procedures described here are not compatible with using SAML or LDAP for user identity management. To create and manage user accounts in a SAML or LDAP-enabled Ops Manager deployment, see Adding Existing SAML or LDAP Users to an Ops Manager Deployment.

Overview

Ops Manager uses role-based access control, with each role granting the permissions in either an org or an app space.

A user account can be assigned one or more roles. The combination of these roles defines the actions a user can perform in an org and within specific app spaces in that org. For information about the actions that each role allows, see Orgs, Spaces, Roles, and Permissions. For example, to assign roles to user accounts in a space, you must have Space Manager role assigned to the user in that space.

You can also modify permissions for existing users by adding or removing the roles associated with the user account. User roles are assigned on a per-space basis, so you must modify the user account for each space that you want to change.

Admins, Org Managers, and Space Managers can assign user roles with Apps Manager or with the Cloud Foundry Command Line Interface (cf CLI). For more information, see Users and Roles in *Getting Started with the cf CLI*.

You can manage user roles across multiple foundations. For more information, see Configuring Multi-Foundation Support in Apps Manager.

Manage Org Roles

Valid org roles are Organization Manager and Organization Auditor.

To grant or revoke org roles:

1. Go to the **Home** page
2. Select an org.
3. In the panel on the left side of the screen, click **Members**. Edit the roles assigned to each user by selecting or clearing the check boxes under each user role. Apps Manager saves your changes automatically.
4. The **Members** panel displays all members of the org. Select a check box to grant an org role to a user, or clear a check box to revoke a role from a user.

Manage App Space Roles

Valid app space roles are Space Manager, Space Developer, and Space Auditor.

To grant or revoke app space roles:

1. Go to the page for a space.
2. In the panel on the left side of the screen, click **Members**. The **Members** panel displays all members of the space.
3. Select a check box to grant an app space role to a user, or clear a check box to revoke a role from a user.
    - **Space Managers** can invite and manage users and enable features for a given space. Assign this role to managers or other users who need to administer the account.
    - **Space Developers** can create, delete, and manage apps and services, and have full access to all usage reports and logs. Space Developers can also edit apps, including the number of instances and memory footprint. Assign this role to app developers or other users who need to interact with apps and services.

- **Space Auditors** have view-only access to all space information, settings, reports, and logs. Assign this role to users who need to view but not edit the app space.

Invite New Users

> **Note**: The **Enable invitations** check box in the **Apps Manager** pane of the TAS for VMs tile must be enabled to invite new users.

To invite new users to an org:

1. Go to the org page.
2. In the panel on the left side of the screen, click **Members**.
3. Click **Invite New Members**. The **Invite New Team Member(s)** form appears.
4. In the **Add Email Addresses** text field, enter the email addresses of the users that you want to invite. Enter multiple email addresses as a comma-delimited list.
5. The **Assign Org Roles** and **Assign Space Roles** tables list the current org and available spaces with check boxes corresponding to each possible user role. Select the check boxes that correspond to the permissions that you want to grant to the invited users.
6. Click **Send Invite**. The Apps Manager sends an email containing an invitation link to each email address that you specified.

Remove a User From an Org

Removing a user from org also removes them from all spaces in the org.

To remove a user from the org:

1. Go to the org page.
2. In the panel on the left side of the screen, click **Members**.
3. Locate the user account that you want to remove.
4. Under the user’s email address, click on the **Remove User** link. A warning dialog appears.
5. Click **Remove** to confirm user account deletion from the org.

Remove a User From a Space

To remove a user from a space:

1. Go to the page for a space.
2. In the panel on the left side of the screen, click **Members**. The **Members** panel displays all members of the space.
3. Locate the user account that you want to remove.
4. Under the user’s email address, click on the **Remove User** link. A warning dialog appears.
5. Click **Remove** to confirm user account deletion from the space.

Use Apps Manager to add and bind services in TAS for VMs

You can use Apps Manager to add and bind service instances through the Marketplace for your TAS for VMs deployment. The Marketplace offers self-service, on-demand provisioning of add-on services.

For more information about how to add Managed Services to your Cloud Foundry deployment, refer to the Services topics.

To use a service with your application, you must access the Services Marketplace, create and configure an instance of the service, then bind the service instance to your application.

Step 1: Access the Marketplace

Follow the steps below to access the Marketplace.

1. Log in to Apps Manager for your Cloud Foundry deployment.
2. In the left navigation panel, click **Marketplace**.

> **Note**: You can also access the Marketplace from a Space page or from an App Dashboard.

Step 2: Create and configure a service instance

Follow the steps below to create and configure an instance of a service.

1. In the Marketplaces, select a service.
2. Select a plan from the left column, and click **Select this plan**.
3. Complete the **Configure Instance** form with the following information:
    - **Instance Name**: Create a name for this instance of the service. Service instance names must be unique within a space.
    - **Add to Space**: From the drop-down list, select the space where you want to add the service instance.
    - **Bind to App**: From the drop-down list, select an app. Select **[do not bind]** if you do not want to bind the service instance to any app at this time.

    Click **Add** to add the service instance. PWS creates the service instance in the selected space. If you specify an app, PWS binds the service instance to the app.

Step 3: Bind a service instance to an application

Follow the steps below to bind an existing service instance to an application.

1. In the **Spaces** section of the left navigation panel, select a space.
2. In the **Apps** table of the Space page, click the row listing the application you want to bind. This opens the App Dashboard.
3. On the App Dashboard, select the **Services** tab.

4. In the Services panel, click **Bind a Service**.
5. Select a service from the **Select a service** drop-down.
6. Click **Bind**.

Managing Apps and Service Instances using Apps Manager

This topic tells you how to view and manage apps and service instances with Apps Manager.

Overview

You can use Apps Manager to manage apps, service instances, service keys, and route services. This includes tasks such as scaling apps, binding apps to services, generating services keys, and more.

For information about managing orgs and spaces, see Managing Orgs and Spaces Using Apps Manager.

You can manage apps and service instances across multiple foundations.

Prerequisite

To view and manage apps and service instances with Apps Manager, you must log in to Apps Manager with an account that has adequate permissions.

For more information, see About Permissions in *Getting Started with Apps Manager*.

Manage an App

This section describes how to manage an app in Apps Manager.

You can do the following tasks to manage apps in Apps Manager:

- View app summary information
- Start and stop apps
- Scale apps
- View sidecar processes
- View key app metrics
- Bind apps to services
- Manage environment variables and routes
- View logs and usage information
- View app revisions
- Re-deploy app revisions
- Delete apps
- Terminate specific instances of apps

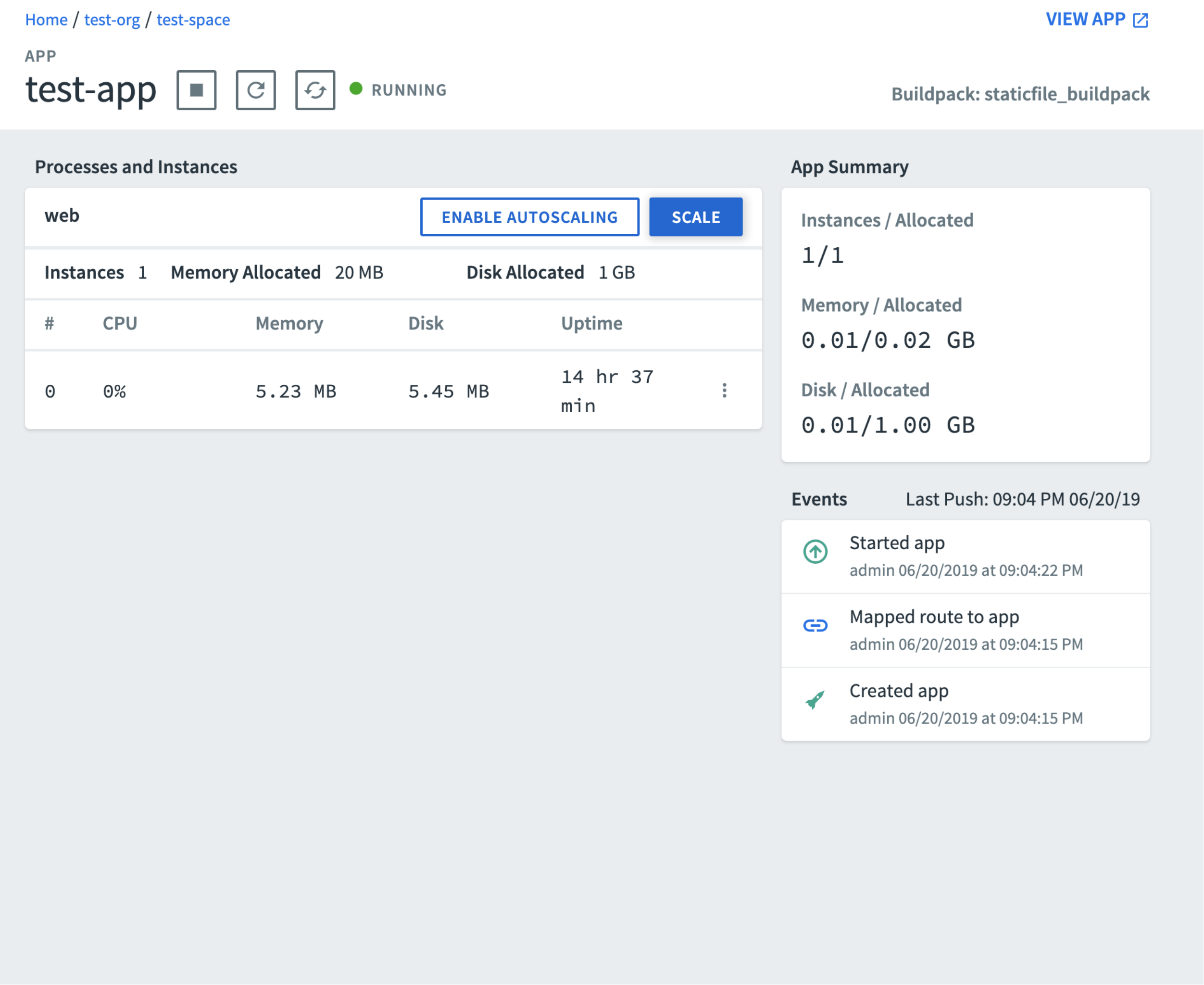
View App Overview

The **Overview** page for an app includes app summary information, such as app processes and instances, app memory and disk space, and app events.

These are the ways to go to the app **Overview** page:

- On the space page, click the app you want to manage.
- Search for an app by entering its name in the search bar.

The following image shows an example of an app **Overview** page.



test-app ● RUNNING

Buildpack: staticfile_buildpack

Processes and Instances

web

Instances 1 Memory Allocated 20 MB Disk Allocated 1 GB

#	CPU	Memory	Disk	Uptime
0	0%	5.23 MB	5.45 MB	14 hr 37 min

App Summary

Instances / Allocated
1/1

Memory / Allocated
0.01/0.02 GB

Disk / Allocated
0.01/1.00 GB

Events Last Push: 09:04 PM 06/20/19

Started app
admin 06/20/2019 at 09:04:22 PM

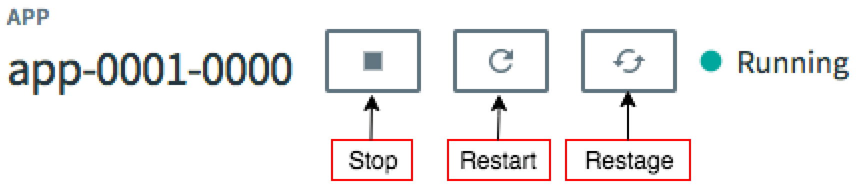
Mapped route to app
admin 06/20/2019 at 09:04:15 PM

Created app
admin 06/20/2019 at 09:04:15 PM

Start, Stop, or Restage an App

You can start, stop, and restage from the **Overview** page of the app. The start, stop, and restage buttons are located on the app **Overview** page next to the name of the app.

The following image shows the location of the start, stop, and restage buttons.



For more information about starting, stopping, and restaging apps with the Cloud Foundry Command Line Interface (cf CLI), see Starting, Restarting, and Restaging Apps.

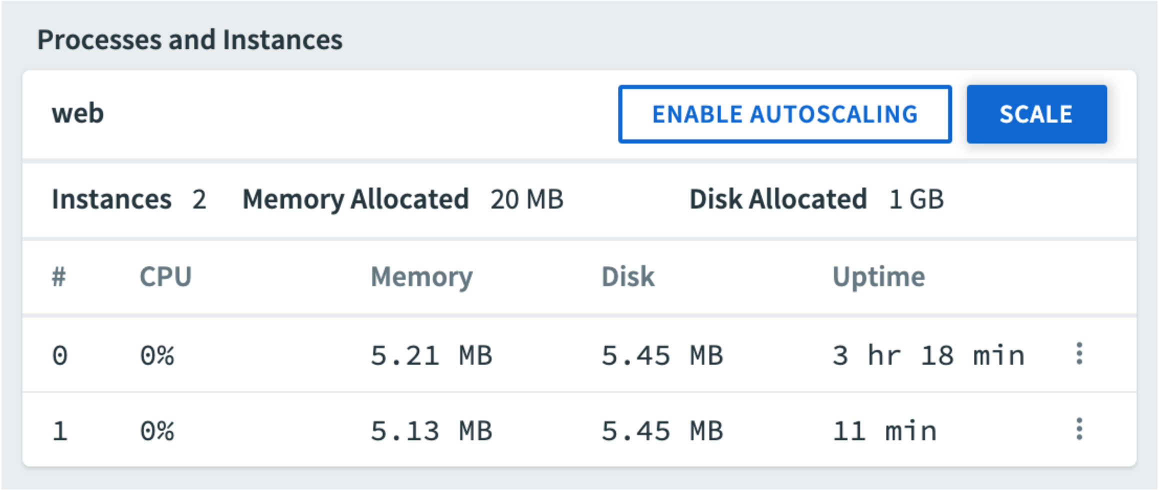
Scale an App

From the app **Overview** page, you can scale an app manually or configure App Autoscaler to scale it automatically.

Scale an App Manually

To manually scale an app:

1. Go to the app **Overview** page.
2. Under **Processes and Instances**, click **Scale** to open the **Scale app** dialog.



Processes and Instances

web

Instances 2 Memory Allocated 20 MB Disk Allocated 1 GB

#	CPU	Memory	Disk	Uptime
0	0%	5.21 MB	5.45 MB	3 hr 18 min
1	0%	5.13 MB	5.45 MB	11 min

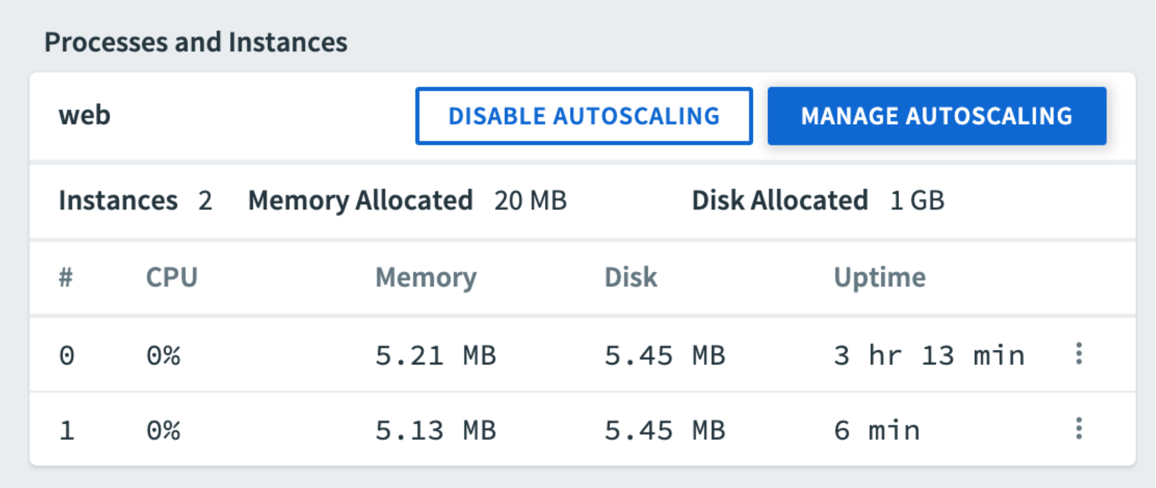
3. Edit the number of **Instances**, the **Memory Limit**, and the **Disk Limit** as desired.
4. Click **Apply Changes**.

Enable App Autoscaler

You can enable App Autoscaler to automatically scale your apps. For information about how to configure App Autoscaler to scale an app automatically, see Configure Autoscaling for an App in *Scaling an App Using App Autoscaler*.

To enable App Autoscaler for an app:

1. Go to the app **Overview** page.
2. Under **Processes and Instances**, click **Enable Autoscaling** to enable App Autoscaler.



Instances 2 Memory Allocated 20 MB Disk Allocated 1 GB

#	CPU	Memory	Disk	Uptime
0	0%	5.21 MB	5.45 MB	3 hr 13 min
1	0%	5.13 MB	5.45 MB	6 min

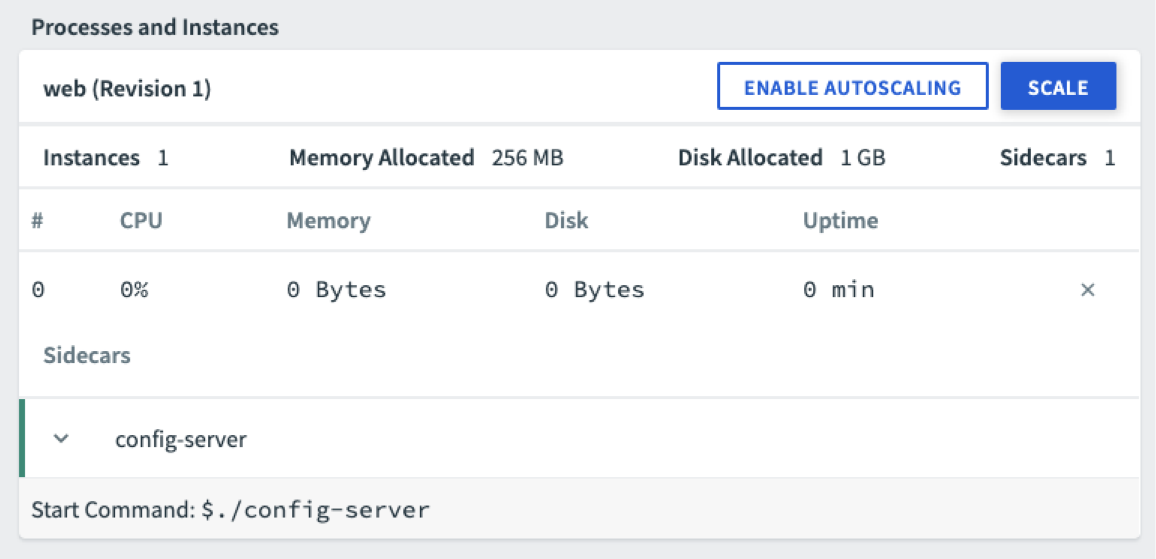
3. Click **Manage Autoscaling** to open App Autoscaler.

View Sidecar Processes

You can view the sidecar processes associated with your app from the app **Overview** page.

To view sidecar processes associated with your apps:

1. Go to the app **Overview** page.
2. Under **Processes and Instances**, see **Sidecars**. This section only appears if your app has any sidecar processes.



Instances 1 Memory Allocated 256 MB Disk Allocated 1 GB Sidecars 1

#	CPU	Memory	Disk	Uptime
0	0%	0 Bytes	0 Bytes	0 min

Sidecars

config-server

Start Command: `$./config-server`

For more information about sidecar processes, see Pushing Apps with Sidecar Processes.

View Key Metrics (Beta)

If Metric Store is installed, you can view the following key metrics for an app in the **Key Metrics** section of the app **Overview** page:

- CPU
- Memory
- Disk

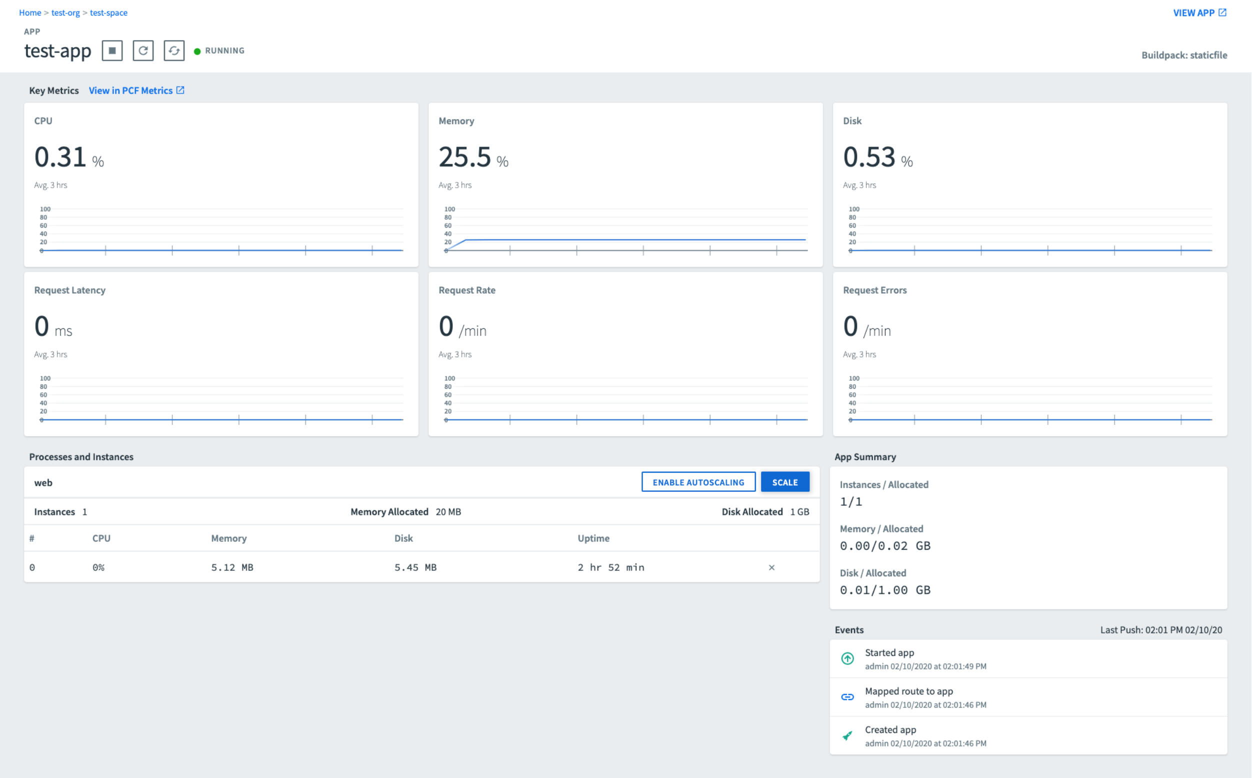
- Request Latency
- Request Rate
- Request Errors

For more information about each metric, see Interpret Metrics in *Monitoring and Troubleshooting Apps with Metrics*.

For each metric, the app **Overview** page includes a graph that shows metric behavior over the past three hours. You can hover over the data points on the graph to view the value of the metric. The page also includes the average value for the metric over the past three hours.

Metric Store is required for Apps Manager to display the **Key Metrics** section. For information about installing and configuring Metric Store, see Metric Store.

The following image is an example of an app **Overview** page, showing key metrics for the app.



test-app

Buildpack: staticfile

Key Metrics

CPU 0.31 % Avg. 3 hrs

Memory 25.5 % Avg. 3 hrs

Disk 0.53 % Avg. 3 hrs

Request Latency 0 ms Avg. 3 hrs

Request Rate 0 /min Avg. 3 hrs

Request Errors 0 /min Avg. 3 hrs

Processes and Instances

web

Instances 1 | Memory Allocated 20 MB | Disk Allocated 1 GB

#	CPU	Memory	Disk	Uptime	
0	0%	5.12 MB	5.45 MB	2 hr 52 min	×

App Summary

Instances / Allocated
1/1

Memory / Allocated
0.00/0.02 GB

Disk / Allocated
0.01/1.00 GB

Events — Last Push: 02:01 PM 02/10/20

Started app
admin 02/10/2020 at 02:01:49 PM

Mapped route to app
admin 02/10/2020 at 02:01:46 PM

Created app
admin 02/10/2020 at 02:01:46 PM

Bind or Unbind Services

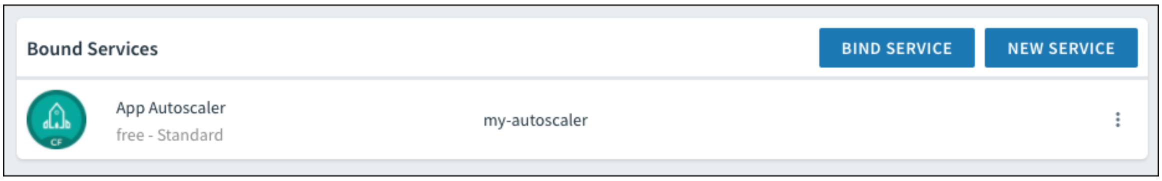
You can bind your app to new or existing service instances. You can also unbind your app from service instances.

Bind an App to an Existing Service

> **Note:** For services that use asynchronous bindings, Apps Manager displays the status of the service while the bind is still pending. Asynchronous bindings provide more flexibility for services that require additional time before returning a successful bind response.

To bind your app to an existing service:

1. Go to the app **Overview** page.
2. In the panel on the left side of the screen, under **Application**, click **Services**.



Bound Services

App Autoscaler

free - Standard

my-autoscaler

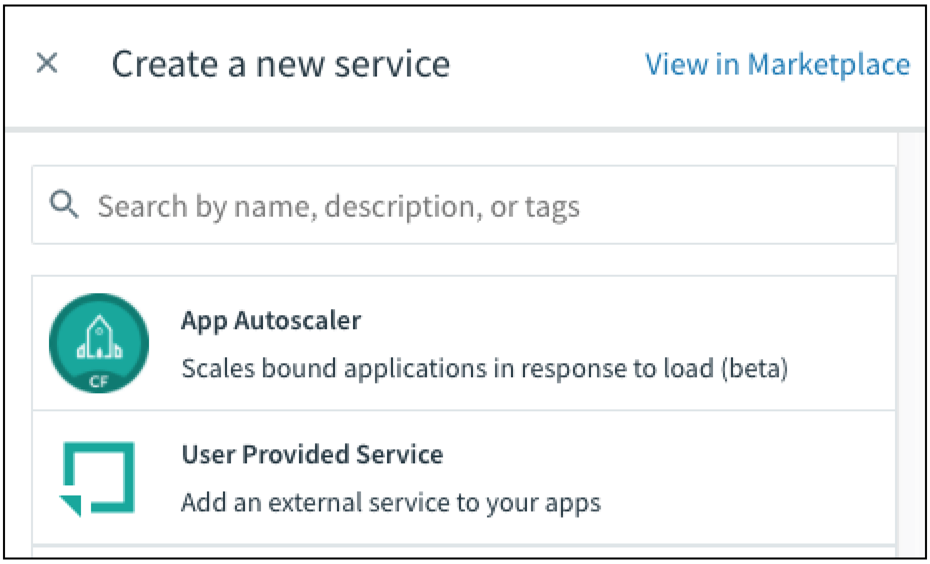
3. Click **Bind Service**.
4. For **Service to Bind**, select the service instance from the drop-down menu.
5. (Optional) For **Binding Name**, enter a binding name in all lowercase letters. For example, `bind-usr-serv`.
6. (Optional) For **Add Parameters**, specify additional parameters.
7. Click **Bind**.

Bind an App to a New Service

To bind your app to a new service instance:

1 Go to the app **Overview** page.

1. In the panel on the left side of the screen, under **Application**, click **Services**.
2. Click **New Service**.



Create a new service

App Autoscaler

Scales bound applications in response to load (beta)

User Provided Service

Add an external service to your apps

Note: If you prefer to create the new service instance in the Marketplace, you can click **View in Marketplace** at any time.

3. Click the service.
4. Select a plan and click **Select Plan**.
5. Under **Instance Name**, enter a name for the instance.
6. (Optional) For **Binding Name**, enter a binding name in all lowercase letters. For example, `bind-usr-serv`.

7. (Optional) For **Add Parameters**, specify additional parameters. For a list of supported configuration parameters, consult the documentation for the service.
8. Click **Create**.

Unbind a Service

To unbind your app from a service instance:

1. Go to the app **Overview** page.
2. Go to **Application** and click **Services**.
3. Locate the service instance in the **Bound Services** list.
4. Click the three-dot icon.
5. Select **Unbind** from the drop-down menu.

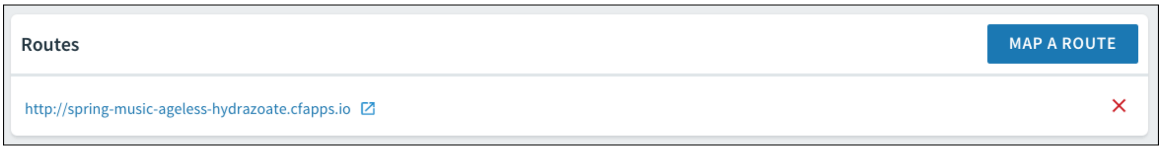
Map or Unmap Routes

The **Routes** page shows the routes associated with your app. You can use this page to map and unmap routes for your app.

Map Routes

To map routes to your app:

1. Go to the app **Overview** page.
2. Go to **Networking**, and click **Routes**.
3. To add a new route, click **Map a Route**.



4. Enter the route and click **Map**.

Unmap Routes

To unmap a route from your app:

1. Go to the app **Overview** page.
2. Go to **Networking**, and click **Routes**.
3. Locate the route from the list and click the **x**.
4. Click **Unmap** in the pop-up window to confirm.

Create Container-to-Container Networking Policies

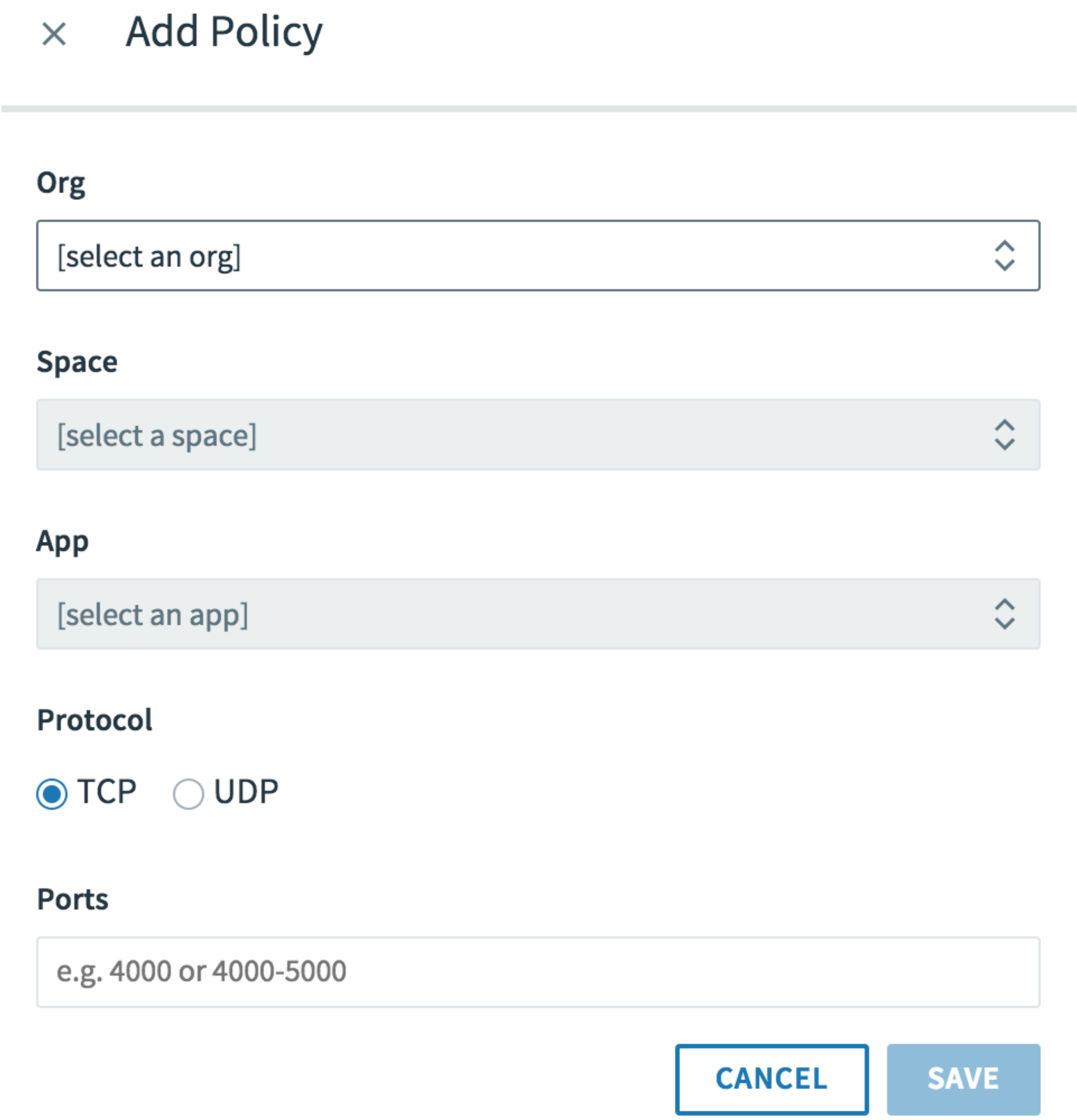
Container networking policies enable app instances to communicate with each other directly. You can create container networking policies in the **Networking** tab.

For more information about container-to-container networking in Ops Manager, see Container-to-Container Networking.

Note: To view and use the **Networking** tab, you must have either the `network.write` or `network.admin` UAA scope. If you do not see the **Networking** tab, request one of the previous scopes from your Ops Manager administrator.

To create container-to-container networking policies:

1. Go to the app **Overview** page.
2. Go to **Networking**, and click **Container Networking**. The page displays any container networking policies associated with the app.
3. To add a new network policy, click **Create Policy** or **Add a network policy**.



Protocol

● TCP ○ UDP

4. In the **Add Policy** window, configure the following:

- For **Org**, select the org of the destination app.
- For **Space**, select the space of the destination app.
- For **App**, select the destination app.
- For **Protocol**, select **TCP** or **UDP**.
- For **Ports**, enter the ports at which to connect to the destination app. The allowed range is from `1` to `65535`. You can specify a single port, such as `8080`, or a range of ports, such as `8080-8090`.

5. Click **Save**.

Manage App Revisions

A revision represents code and configuration used by an app at a specific time. It is a Cloud Foundry API (CAPI) object that can contain references to a droplet, custom start command, and environment variables. The most recent revision for a running app represents code and configuration currently running in Ops Manager.

You can view revisions, list environment variables associated with revisions, and re-deploy previous revisions in the **Revisions** tab.

For more information about app revisions in Ops Manager, see App Revisions.

View App Revisions

To list revisions for an app:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Revisions**.
3. View app revisions in the **Revisions** table.

List Environment Variables for an App Revision

To list environment variables associated with an app revision:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Revisions**.
3. In the **Revisions** table, select the row that contains the revision. The row expands to reveal metadata and environment variables associated with the revision.

Roll Back to a Previous App Revision

To roll back an app to a previous revision:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Revisions**.
3. In the **Revisions** table, select the row that contains the revision you want to roll back to. The row expands to reveal metadata.
4. Click **Redeploy**.

View Logs

You can view logs for an app in Apps Manager. You can also use Apps Manager to view a live stream of app logs.

To view logs for an app:

1. Go to the app **Overview** page.
2. Go to **Observability**, and click **Logs**.
3. Click the Play button to view a live version of the logs.

Manage Tasks

The **Tasks** page includes jobs and tasks associated with an app. It displays a table containing **Task ID**, **State**, **Start Time**, **Task Name**, and **Command**.

From the **Tasks** page, you can view tasks, run tasks, and enable task scheduling.

View Tasks

You can view tasks for an app on the Tasks page in Apps Manager.

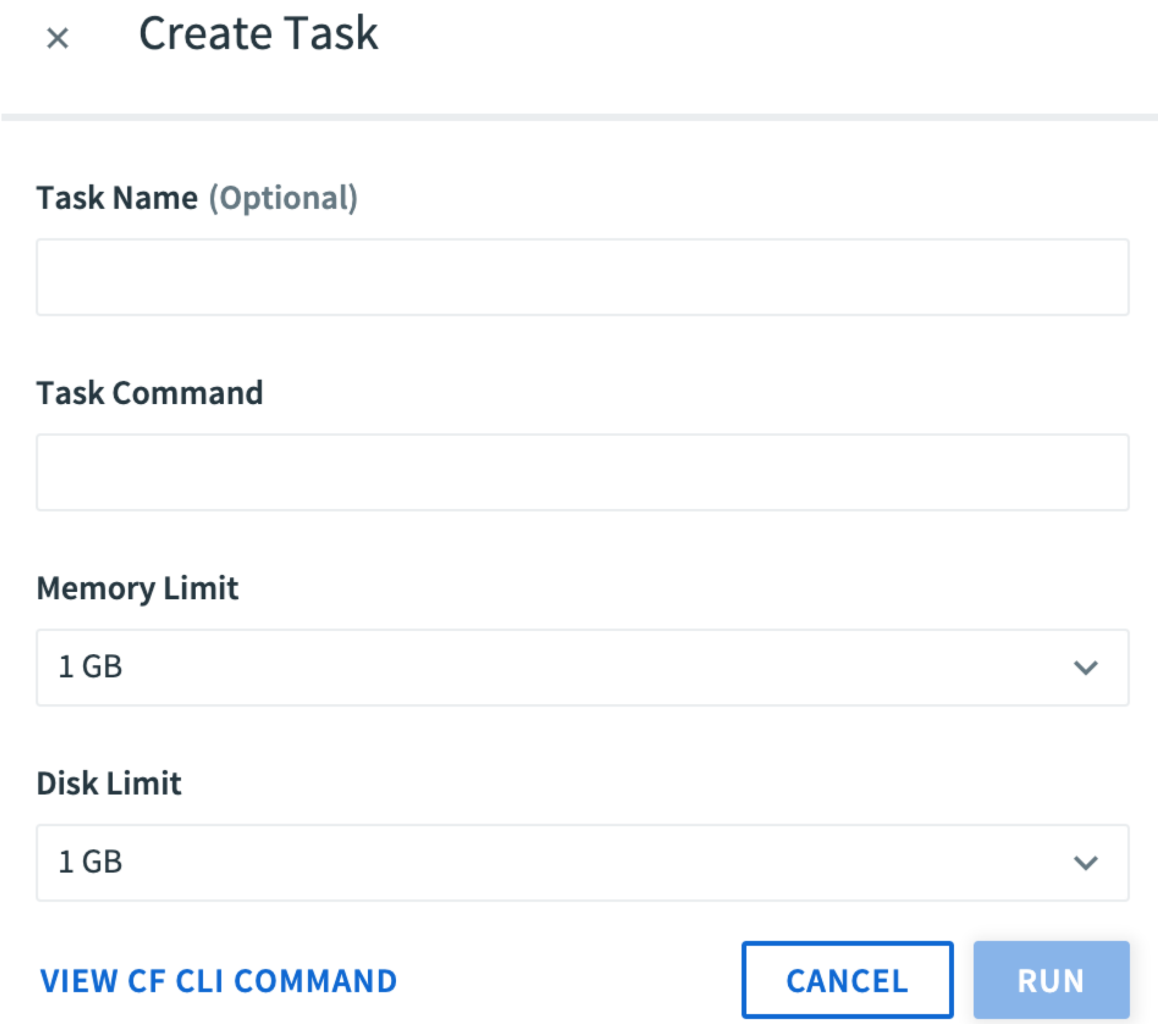
To access the **Tasks page**:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Tasks**.

Run a Task

To run a task for an app:

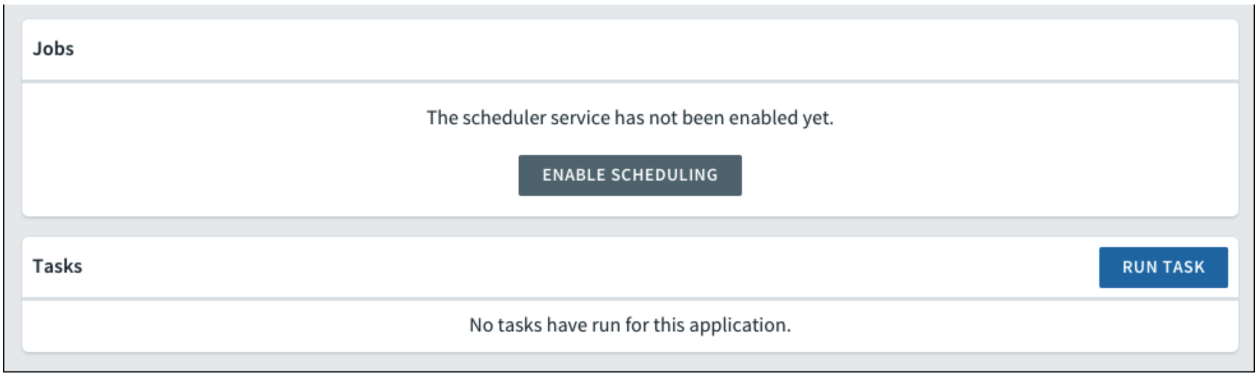
1. Go to the app **Overview** page.
2. Go to **Application**, and click **Tasks**.
3. Click **Run Task** to create a task.



4. (Optional) Enter a **Task Name**.
5. Enter the **Task Command**.
6. Select a **Memory Limit** and a **Disk Limit** for the task.
7. Click **Run**.

Enable Task Scheduling

In the **Tasks** tab, click **Enable Scheduling** to bind the Scheduler service to your app. For more about Scheduler, see Scheduling Jobs in the Scheduler documentation.



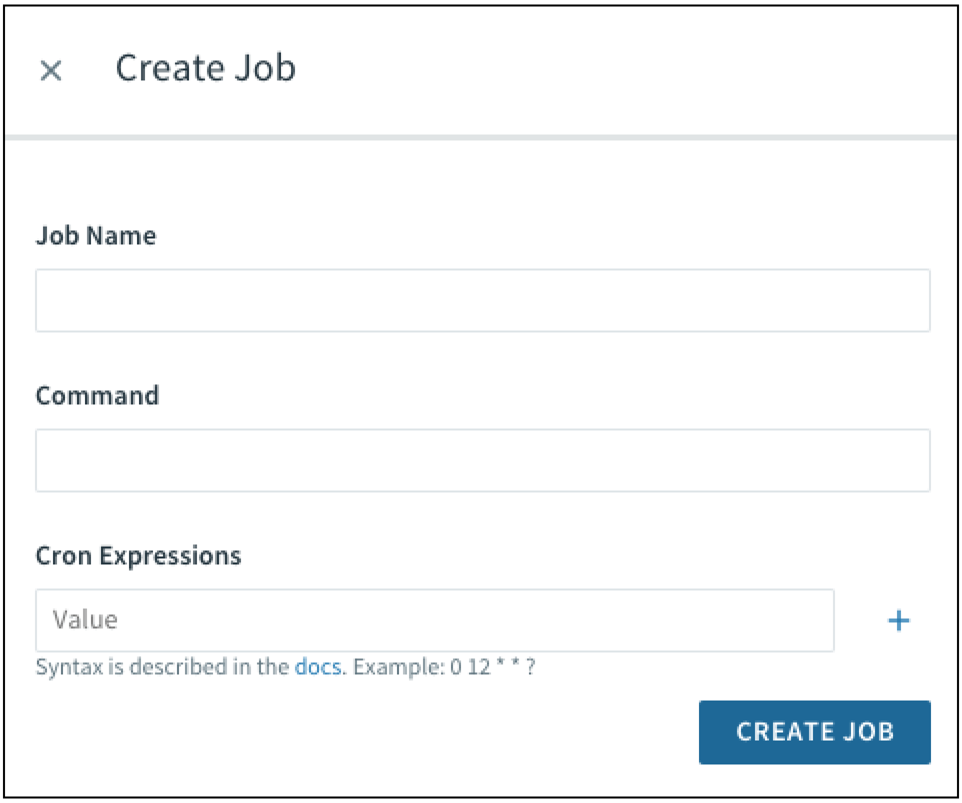
The scheduler service has not been enabled yet.

No tasks have run for this application.

Schedule a Task

To schedule a task:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Tasks**.
3. Click **Create Job** to schedule a task.



Syntax is described in the docs. Example: 0 12 * * ?

4. Enter a **Job Name**.
5. Enter a **Command**.
6. Enter one or more **Cron Expressions** for your desired task schedule or schedules. For more information on cron expression syntax, see Schedule a Job in *Scheduling Jobs* in the Scheduler documentation.
7. Click **Create Job**.

View and Manage Settings

To view settings for an app, in the panel on the left side of the screen, under **Application**, click **Settings**.

You can do the following from the **Settings** page:

- Rename the app.
- View information about the buildpacks, start command, and stack.
- Configure health checks.
- View or add Environment Variables associated with the app.
- Add metadata to the app.
- View the App Security Groups (ASGs) associated with the app.
- Delete the app. When you click **Delete App**, you also have the option to delete the app's routes.

Configure Health Checks

To configure health checks for your app:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Settings**.
3. In the **Health Check** section, click the **Type** drop down menu and select **port**, **http**, or **process**. Depending on which option you select, additional fields might appear.
4. For **Type**, select **port**, **http**, or **process**.
5. Complete the fields that appear based on your selection.
6. Click **Update**.

For more information, see Using App Health Checks.

View Environment Variables

To view all environment variables for an app:

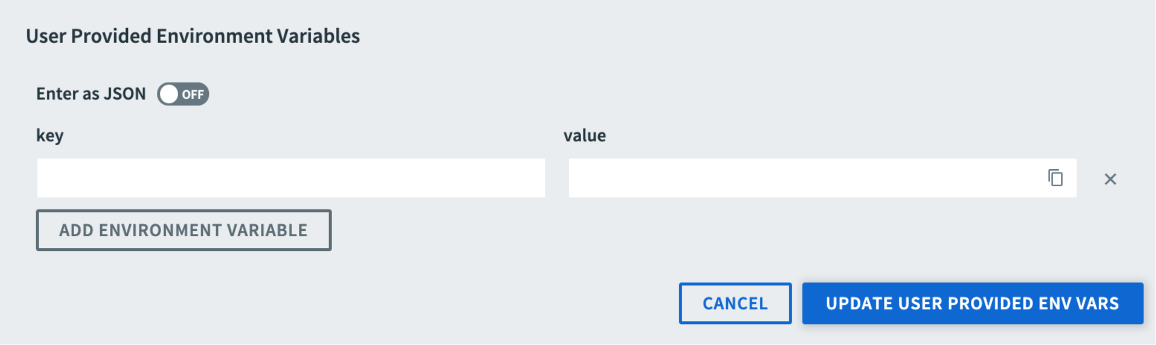
1. Go to the app **Overview** page.
2. Go to **Application**, and click **Settings**.
3. Under **User Provided Environment Variables**, click **Reveal Env Vars**.

Add Environment Variables

To add a user-provided environment variable:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Settings**.
3. Click **Reveal User Provided Env Vars**.
4. Click **Add Environment Variable**.

5. Enter the **key** and **value** of the variable. Alternatively, toggle **Enter as JSON** to enter the variable in JSON.


User Provided Environment Variables
Enter as JSON
OFF
key
value
ADD ENVIRONMENT VARIABLE
CANCEL
UPDATE USER PROVIDED ENV VARS

6. Click **Save**.

> **Note:** Changes to environment variables, service bindings, and service unbindings require restarting the app to take effect. You can restart the app from the Apps Manager or with the Cloud Foundry Command Line Interface `cf restage` command.

Add Metadata

You can add metadata, including labels and annotations, to apps and spaces using Apps Manager.

For more information about adding metadata to objects in VMware Tanzu Application Service for VMs (TAS for VMs), see Using Metadata.

To add labels or annotations to an app using Apps Manager:

1. Go to the app **Overview** page.
2. Go to **Application**, and click **Settings**.
3. For **Labels**, provide a name and value pair or select **Enter JSON** to enter the label in JSON.
4. For **Annotations**, provide a name and value pair or select **Enter JSON** to enter the annotation in JSON.
5. To add more labels or annotations, click **+** next to the **Labels** or **Annotations** field.
6. Click **Update Metadata**.

Manage a Service Instance

This section describes how to manage service instances in Apps Manager.

You can do the following tasks to manage service instances in Apps Manager:

- Bind or unbind apps.
- Bind or unbind routes.
- View or change your service plan.
- Manage service keys.
- Rename or delete your service instance.

Note: For services that use on-demand brokers, the service broker creates, updates, or deletes the service instance in the background and notifies you when it finishes.

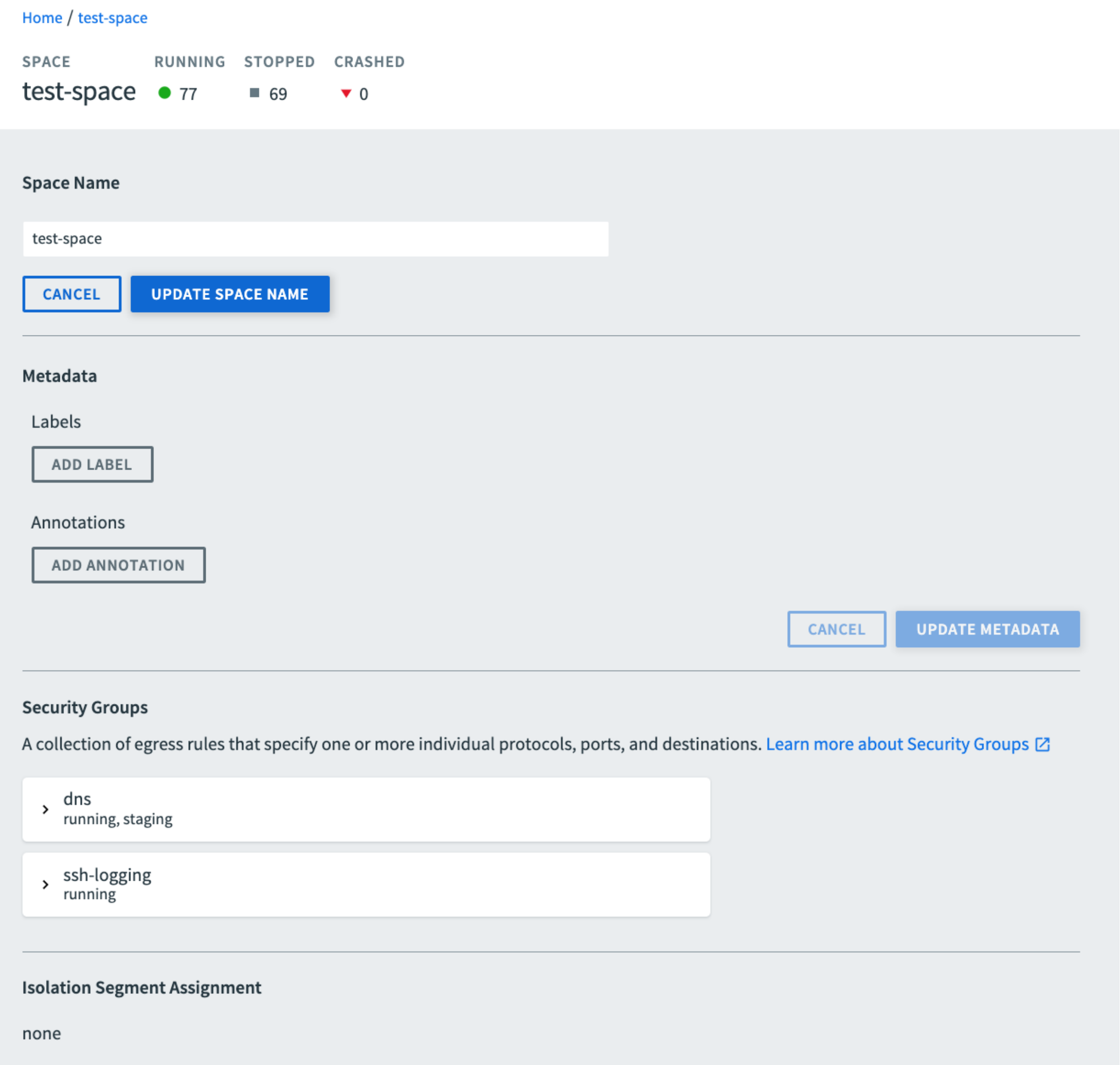
View Service Overview

The **Overview** page for a service instance includes information about bound apps, bound routes, and service key credentials for the service.

To locate the service instance **Overview** page:

1. Go to the landing page for the space of the service instance.
2. Go to **Space**, and click **Services**.
3. On the **Services** page, click the name of the service instance.

The following image shows an example of a service instance **Overview** page.



Home / test-space

SPACE	RUNNING	STOPPED	CRASHED
test-space	● 77	■ 69	▼ 0

Space Name

test-space

Metadata

Labels

Annotations

Security Groups

A collection of egress rules that specify one or more individual protocols, ports, and destinations. Learn more about Security Groups

dns
running, staging

ssh-logging
running

Isolation Segment Assignment

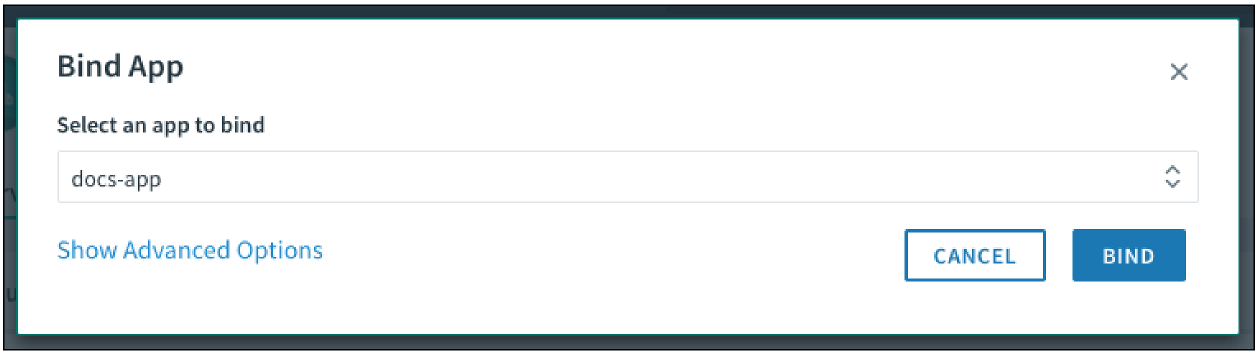
none

Bind an App

Note: For services that use asynchronous bindings, Apps Manager displays the status of the service while the bind is still pending. Asynchronous bindings provide more flexibility for services that require additional time before returning a successful bind response.

To bind and app to a service:

1. On the service **Overview** page, click **Bind App**.
2. In the **Bind App** pop-up window, select the app to bind to the service instance.



Bind App

Select an app to bind

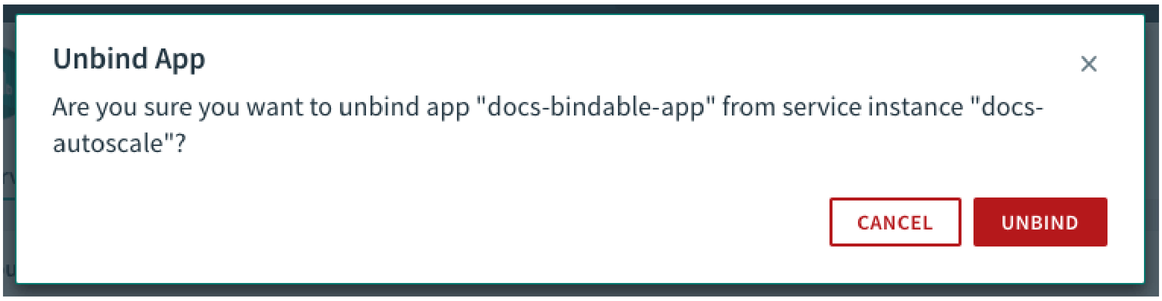
docs-app

1. (Optional) To attach parameters to the binding, click **Show Advanced Options**. Under **Arbitrary Parameters**, enter any additional service-specific configuration.
2. Click **Bind**.

Unbind an App

To unbind an app from a service instance:

1. On the service **Overview** page, go to **Bound Apps**, and click the **×** to the right of the app name. An **Unbind App** popup appears.



Unbind App

Are you sure you want to unbind app "docs-bindable-app" from service instance "docs-autoscale"?

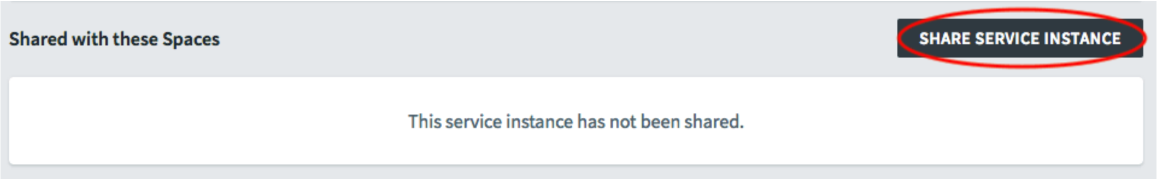
2. Click **Unbind** to confirm.

Share Service Instance

From the service **Overview** page, you can share your service instance across spaces.

To share a service instance:

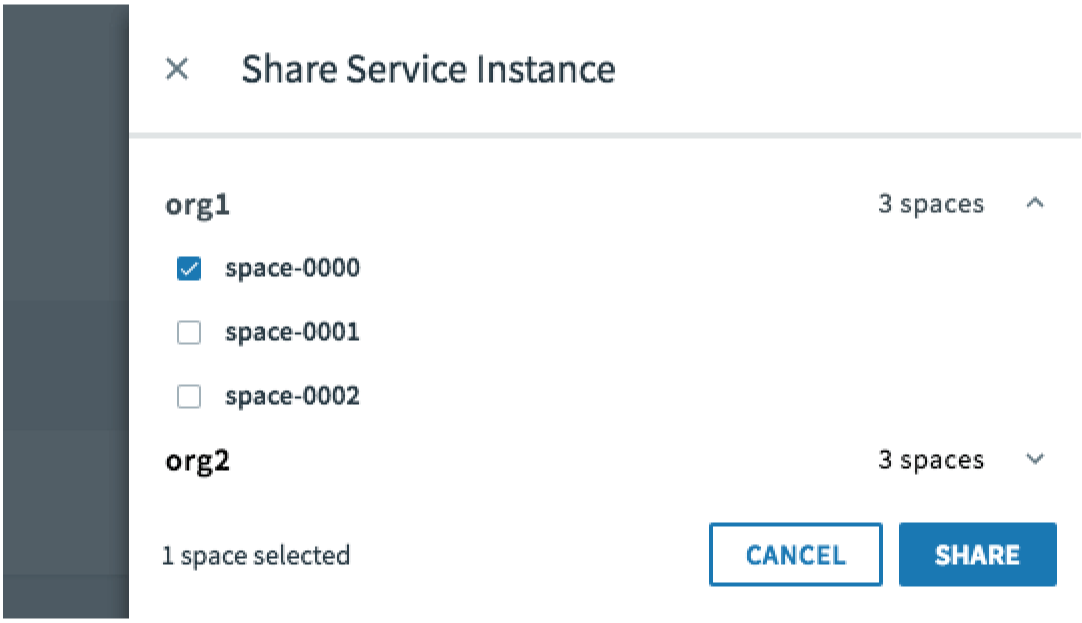
1. On the service **Overview** page, click **Share Service Instance**.



Shared with these Spaces

This service instance has not been shared.

2. Select the spaces with which you want to share your service instance.



Share Service Instance

org1 3 spaces

☑ space-0000

□ space-0001

□ space-0002

org2 3 spaces

1 space selected

3. Click **Share**.

View or Change Your Service Plan

To view or change your service plan:

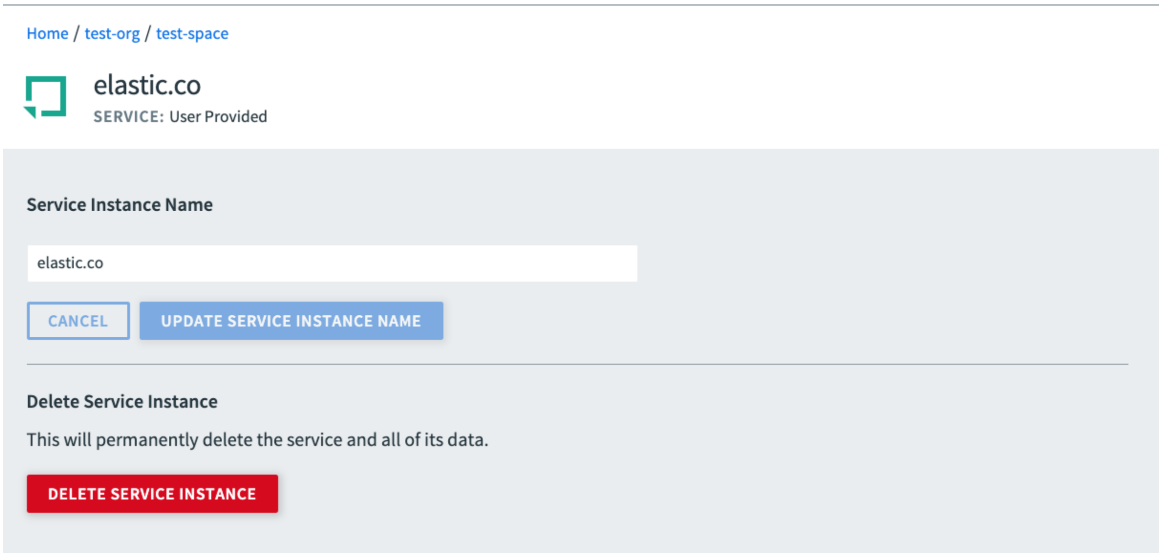
1. Go to the service **Overview** page.
2. Go to **Service**, and click **Plan**.
3. Review your current plan information.
4. To change your plan, select a new plan from the list and click **Select This Plan** or **Upgrade Your Account**.

> **Note:** Not all services support upgrading. If your service does not support upgrading, the service plan page only displays the selected plan.

Rename or Delete Your Service Instance

To rename or delete your service instance:

1. Go to the service **Overview** page.
2. Go to **Service**, and click **Settings**.



Home / test-org / test-space
elastic.co
SERVICE: User Provided
Service Instance Name
elastic.co
CANCEL
UPDATE SERVICE INSTANCE NAME
Delete Service Instance
This will permanently delete the service and all of its data.
DELETE SERVICE INSTANCE

3. Do one of these steps:
   - To change the service instance name, enter the new name and click **Update**.
   - To add configuration parameters to the service instance, enter the parameters in the **Name** and **Value** fields and then click **Update**. Alternatively, enter your configuration parameters using the **Enter JSON** toggle and then click **Update**.
   - To delete the service instance, click **Delete Service Instance**.

Note: The service broker supports creating, updating, and deleting service instances asynchronously. When the service broker completes one of these operations, a status banner appears in Apps Manager.

View and Update Spring Cloud Services Configurations

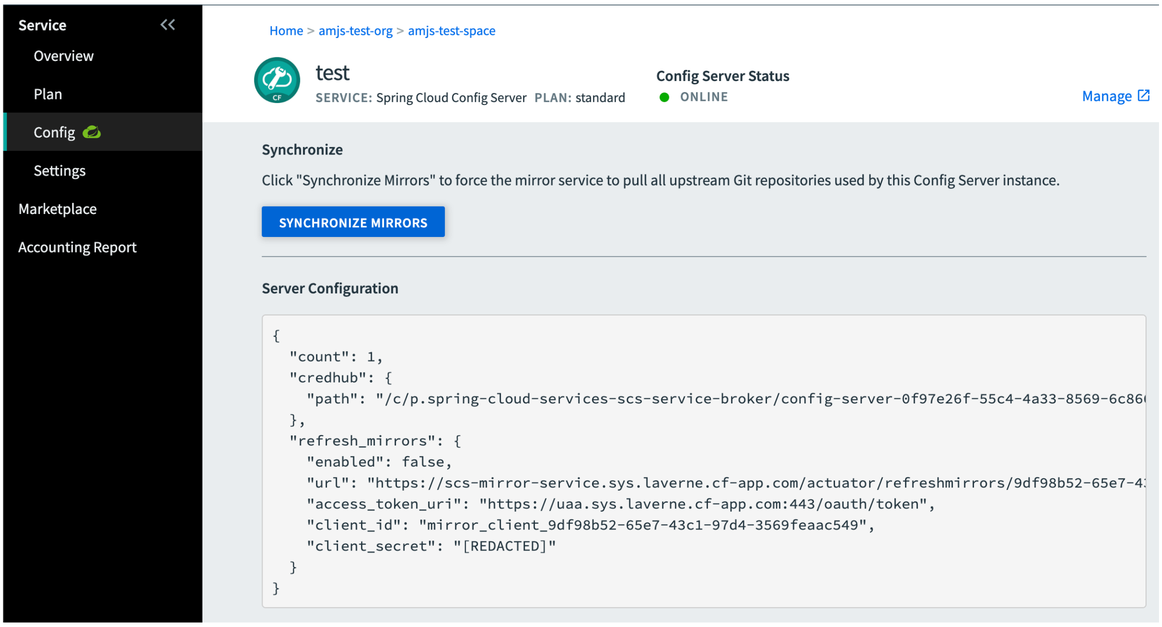
For Spring Cloud Services (SCS) Config Server instances, Apps Manager lets you:

- See the Git repos that the SCS Config Server uses to configure the client apps that it serves. For more information, see Configuring with Git in the SCS documentation.
- Update client app configurations, by updating the local mirrors of their configuration repos.
- See the current status and configuration of the SCS Config Server itself.

To see and update this configuration and status information for an SCS Config Server instance:

1. Go to the service **Overview** page.
2. Go to **Service**, and click **Config**.

   Note: The **Configuration** pane only appears for Spring Cloud Services instances.



Synchronize

Click "Synchronize Mirrors" to force the mirror service to pull all upstream Git repositories used by this Config Server instance.

Server Configuration

```
{
  "count": 1,
  "credhub": {
    "path": "/c/p.spring-cloud-services-scs-service-broker/config-server-0f97e26f-55c4-4a33-8569-6c86
  },
  "refresh_mirrors": {
    "enabled": false,
    "url": "https://scs-mirror-service.sys.laverne.cf-app.com/actuator/refreshmirrors/9df98b52-65e7-4
    "access_token_uri": "https://uaa.sys.laverne.cf-app.com:443/oauth/token",
    "client_id": "mirror_client_9df98b52-65e7-43c1-97d4-3569feaac549",
    "client_secret": "[REDACTED]"
  }
}
```

The **Config** pane includes:

3. The current status of the SCS Config Server.
4. The JSON object that configures settings for the SCS Config Server itself.
    - Passed in with `-c` flag to `cf create-service` and `cf update-service`.
5. A **Synchronize Mirrors** button for the app configurations that the SCS Config Server serves.
    - Click this button to refresh the local Git repository mirrors. Refreshing the mirrors updates client app configuration settings, as described in Mirror Service in the SCS documentation.
6. A **Manage** link to the SCS Config Server dashboard, at the `/dashboard` URL endpoint of the SCS service instance.
    - Click this link to see the Git repos that the SCS Config Server uses to configure its apps.
    - For more information about the SCS Config Server dashboard, see Dashboard Information in *Using the Dashboard* in the SCS documentation.

Configure User-Provided Service Instance

You can create a user-provided service instance from the Marketplace. For more information, see User-Provided Service Instances.

To configure settings for a user-provided service instance:

1. Go to the service **Overview** page.
2. IGo to **Service**, and click **Configuration**.

> **Note:** The **Configuration** pane only appears for user-provided service instances.

![alt-text="Configure settings for serevice instances."](./images/config-user-provided.png)

3. Enter the **Credential Parameters**, **Syslog Drain Url**, and **Route Service Url**.

   Breaking Change: In TAS for VMs v2.10, aggregate syslog drains contain only logs by default, and do not contain metrics. If you rely on metrics sent through aggregate syslog drains, you must add `?include-metrics-deprecated=true` to your aggregate drain URLs to continue to receive metrics in the drains.

   For more information, see Aggregate Syslog Drains Contain Logs Only in the *TAS for VMs v2.10 Release Notes*.

4. Click **Update Service**.

Manage Service Keys

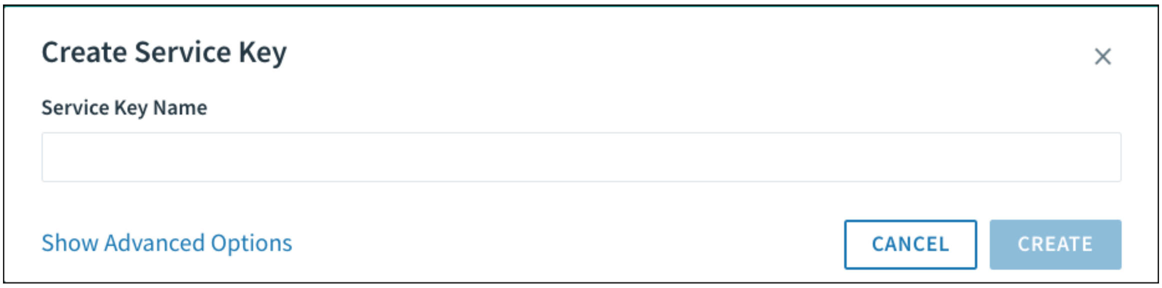
To manage service keys:

1. Go to the service **Overview** page.
2. Go to **Service Key Credentials**, generate a new service key, get the credentials for a service key, or delete a service key.

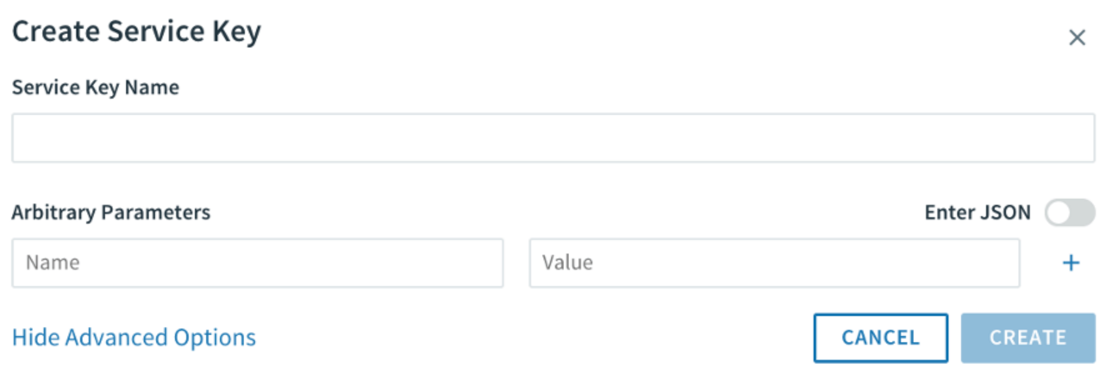
Generate a Service Key

To generate a new service key:

1. Go to the service **Overview** page.
2. Go to **Service Key Credentials**, and click **Create Service Key**.
3. Enter a **Service Key Name**.



4. (Optional) Click **Show Advanced Options**. Under **Arbitrary Parameters**, enter any additional service-specific configuration in the **Name** and **Value** fields.

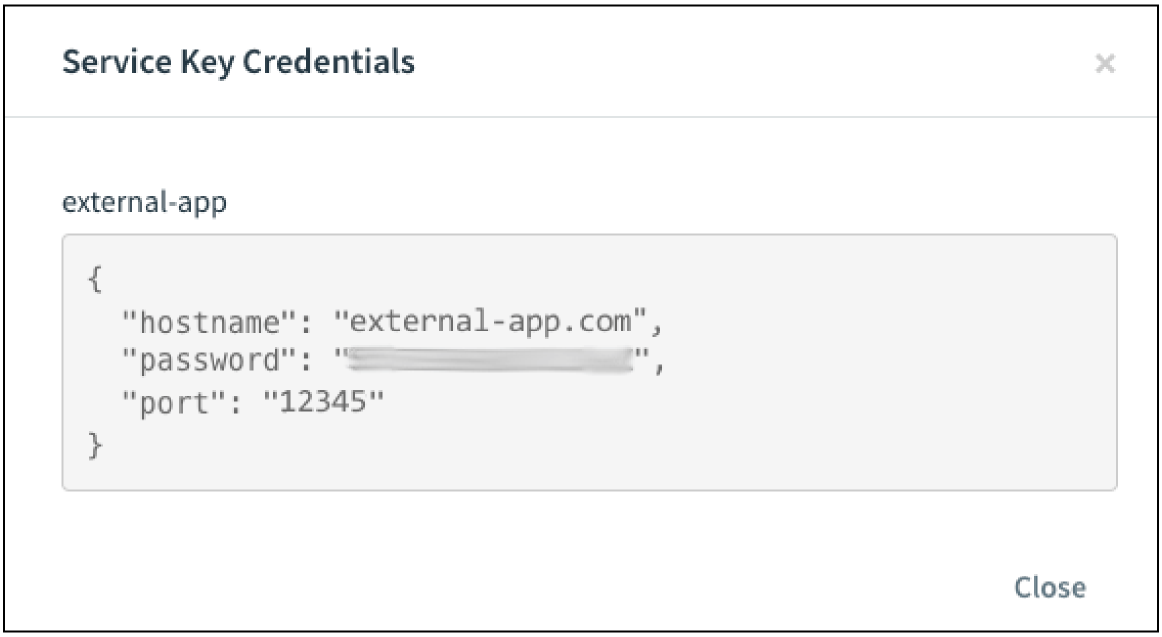


5. Click **Create** to generate the service key.

View Credentials for a Service Key

To view the credentials for a service key:

1. Go to the service **Overview** page.
2. Go to **Service Key Credentials**, and click the service instance name. The JSON object containing the credentials appears.



external-app

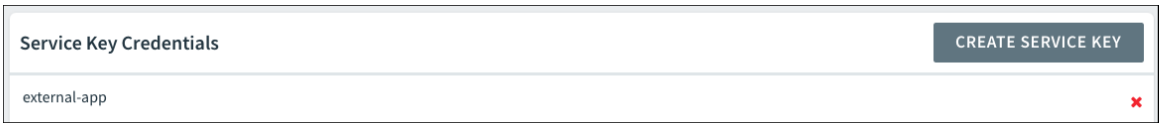
```
{
  "hostname": "external-app.com",
  "password": "[redacted]",
  "port": "12345"
}
```

3. Click **Close**.

Delete Service Key

To delete a service key:

1. Go to the service **Overview** page.
2. Go to **Service Key Credentials**, and click the **x** next to the service instance name.



Manage Route Services

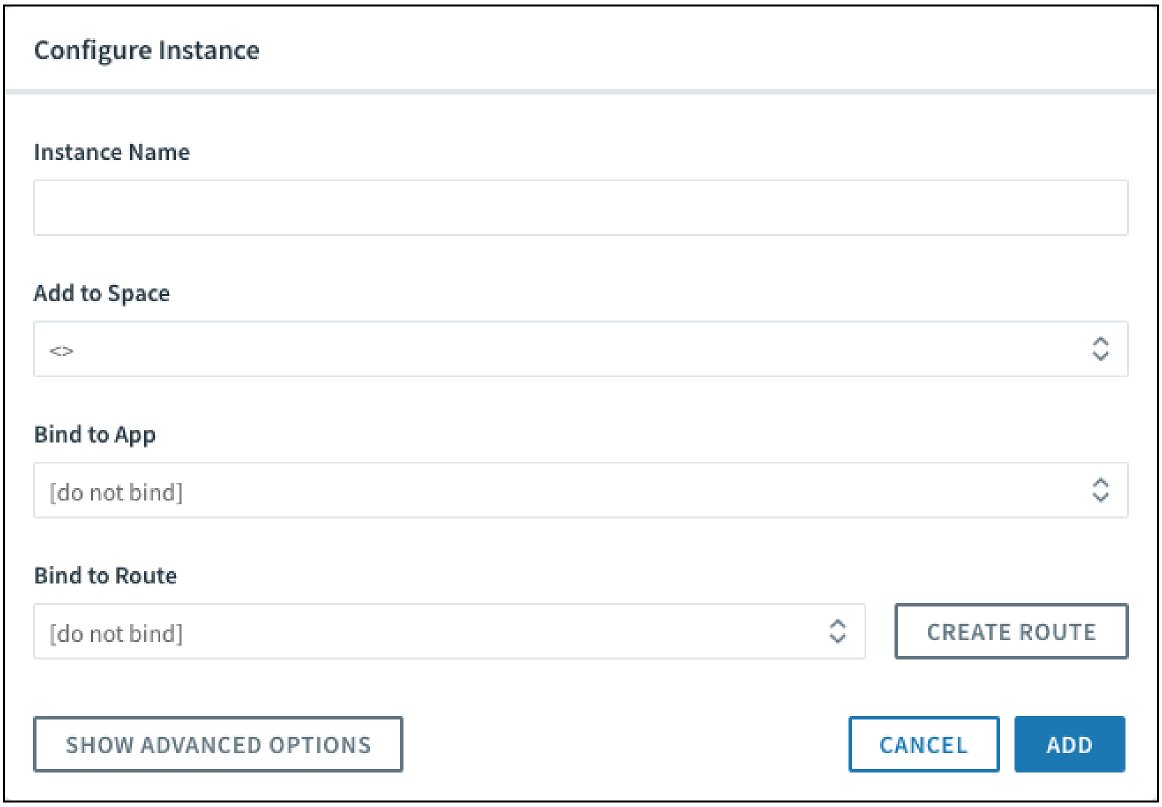
You can bind a new service instance to a route when you create the instance in the Marketplace, or you can manage route services for an existing service instance on the service instance page.

For more information about route services, see Route Services.

Bind a New Service Instance to a Route

To bind a new service instance to a route:

1. Select the service from the Marketplace.



Configure Instance

Instance Name

Add to Space: <>

Bind to App: [do not bind]

Bind to Route: [do not bind]

2. Under **Bind to Route**, either bind the service instance to an existing route or click **Create Route** to create a new custom route.

> **Note:** You must choose a Marketplace service compatible with route services for the **Bind to Route** field to appear.

3. Complete the remaining fields and click **Add** to create the service instance.

Bind an Existing Service Instance to a Route

To bind an existing service instance to a route:

1. Go to the service **Overview** page.
2. Go to **Bound Routes**, and click **Bind Route**.

> **Note:** If the service is not compatible with route services, the text "This service does not support route binding" appears under **Bound Routes**.

![alt-text=""](./images/bind-route-old.png)

3. Do one of the following steps:
   - For **Select a route to bind**, select an existing route.
   - For **Create Custom Route**, enter a new route.
4. Click **Bind**.

Unbind a Route from a Service Instance

To unbind a route from a service instance:

1. Go to the service **Overview** page.
2. Go to **Bound Routes**, and click the **x** next to the name of the route.

Viewing ASGs in Apps Manager

About ASGs

Application Security Groups (ASGs) are a collections of egress rules that specify the protocols, ports, and IP address ranges where app or task instances send traffic. The platform sets up rules to filter and log outbound network traffic from app and task instances. ASGs apply to both buildpack-based and Docker-based apps and tasks.

When apps or tasks begin staging, they need traffic rules permissive enough to allow them to pull resources from the network. After an app or task is running, the traffic rules can be more restrictive and secure. To distinguish between these two security requirements, administrators can define one ASG for app and task staging, and another for app and task runtime. For more information about staging and running apps, see Application Container Lifecycle.

To provide granular control when securing a deployment, an administrator can assign ASGs to apply to all app and task instances for the entire deployment, or assign ASGs to spaces to apply only to apps and tasks in a particular space.

Only admin users can create and modify ASGs. For information about creating and configuring ASGs, see App Security Groups.

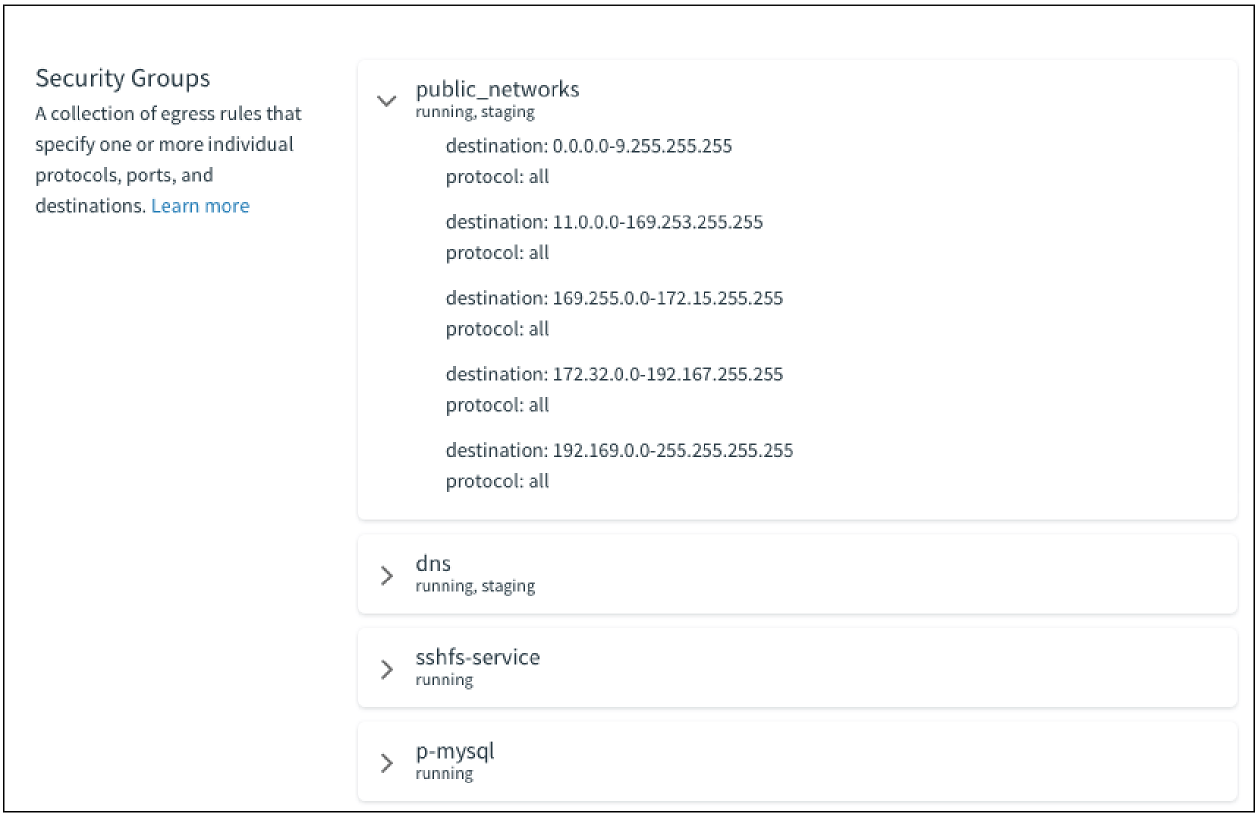
Displaying ASGs for a Space

To view the ASGs associated with a space, perform the following steps.

1. Log in to Apps Manager.
2. From the **Home** page, select the **Org** that contains the space you want to view.
3. Select the **Space** you want to view.
4. Click on the **Settings** tab.

5. In the **Security Groups** section, Apps Manager displays ASGs associated with the selected space.

6. Click on an ASG to expand its egress rules.



Security Groups

A collection of egress rules that specify one or more individual protocols, ports, and destinations. Learn more

public_networks
running, staging

destination: 0.0.0.0-9.255.255.255
protocol: all

destination: 11.0.0.0-169.253.255.255
protocol: all

destination: 169.255.0.0-172.15.255.255
protocol: all

destination: 172.32.0.0-192.167.255.255
protocol: all

destination: 192.169.0.0-255.255.255.255
protocol: all

dns
running, staging

sshfs-service
running

p-mysql
running

Configuring Spring Boot Actuator Endpoints for Apps Manager

The Apps Manager UI supports several production-ready endpoints from Spring Boot Actuator. This topic describes the Actuator endpoints and how you can configure your app to display data from the endpoints in Apps Manager.

For more information about Spring Boot Actuator, see the Spring Boot Actuator documentation.

> **Note:** This feature requires Spring Boot v1.5 or later.

Overview

The Apps Manager integration with Spring Boot does not use the standard Spring Boot Actuators. Instead, it uses a specific set of actuators that are secured using the Space Developer role for the space that the application runs in. Authentication and authorization are automatically delegated to the Cloud Controller and the User Account and Authentication server without any configuration from the user.

By default, actuators are secure and cannot be accessed without explicit configuration by the user, even if Spring Security is not included. This allows users to take advantage of the Spring Boot Apps Manager integration without accidentally exposing their actuators without security.

Actuator Endpoints

The table below describes the Spring Boot Actuator endpoints supported by Apps Manager. To integrate these endpoints with Apps Manager, you must first Activate Spring Boot Actuator for Your App.

Endpoint	About
`/info`	• **Description**: Exposes details about app environment, git, and build. To send build and Git information to this endpoint, see Configure the Info Actuator. • **How to use in Apps Manager**: See View Build and Git Information for Your App.
`/health`	• **Description**: Shows health status or detailed health information over a secure connection. Spring Boot Actuator includes the auto-configured health indicators specified in the Auto-configured HealthIndicators section of the Spring Boot documentation. If you want to write custom health indicators, see the Writing custom HealthIndicators section of the Spring Boot documentation. • **How to use in Apps Manager**: See View App Health.
`/loggers`	• **Description**: Lists and allows modification of the levels of the loggers in an app. • **How to use in Apps Manager**: See Manage Log Levels.
`/dump`	• **Description**: Generates a thread dump. • **How to use in Apps Manager**: See View Thread Dump.
`/trace`	• **Description**: Displays trace information from your app for each of the last 100 HTTP requests. For more information, see the Tracing section of the Spring Boot documentation. • **How to use in Apps Manager**: See View Request Traces.
`/heapdump`	• **Description**: Generates a heap dump and provides a compressed file containing the results. • **How to use in Apps Manager**: See Download Heap Dump.
`/mappings`	• **Description**: Displays the endpoints an app serves and other related details. • **How to use in Apps Manager**: See View Mappings.

Activate Spring Boot Actuator for Your App

You must add a `spring-boot-starter-actuator` dependency to your app project for the production-ready HTTP endpoints to return values. For more information, see the Enabling production-ready features section of the Spring Boot documentation.

1. Follow the instructions below that correspond to your project type.

    - **Maven**: If you use Maven, add the following to your project:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
</dependencies>
```

- **Gradle**: If you use Gradle, add the following to your project:

  ```
  dependencies {
      compile("org.springframework.boot:spring-boot-starter-actuator")
  }
  ```

2. If you use self-signed certificates in your Ops Manager deployment for UAA or the Cloud Controller, specify in your `application.properties` file to skip SSL validation:

   ```
   management.cloudfoundry.skip-ssl-validation=true
   ```

For more information, see Cloud Foundry support in the Spring Boot Actuator documentation.

Configure the Info Actuator

The `/info` endpoint provides information about the project build for your app, as well as its git details.

Add Build Information

To add build information to the `/info` endpoint, follow the instructions below that correspond to your project type.

Maven

Add the following to your app project:

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>1.4.2.RELEASE</version>
            <executions>
                <execution>
                    <goals>
                        <goal>build-info</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Gradle

Add the following to your app project:

```
springBoot  {
    buildInfo()
```

```
}
```

Add Git Information

To add git information to the `/info` endpoint, follow these instructions:

1. Add the following property to your `application.properties` file:

   ```
   management.info.git.mode=full
   ```

2. Follow the instructions below that correspond to your project type.

Maven

Add the following plugin to your project:

```
<build>
    <plugins>
        <plugin>
            <groupId>pl.project13.maven</groupId>
            <artifactId>git-commit-id-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Gradle

Add the following plugin to your project:

```
plugins {
    id "com.gorylenko.gradle-git-properties" version "1.4.17"
}
```

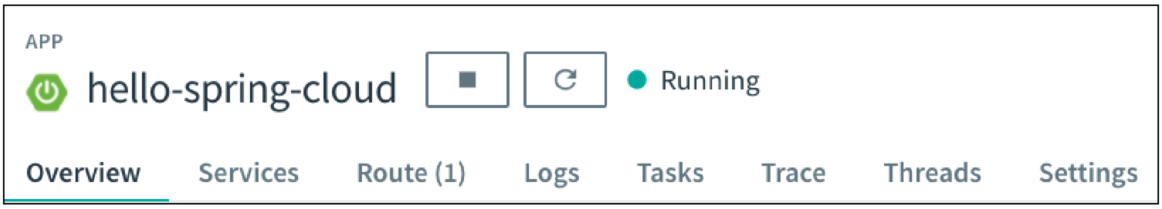
Using Spring Boot Actuators with Apps Manager

This document describes how to view and manage app information from Spring Boot Actuator in Apps Manager.

Prerequisites

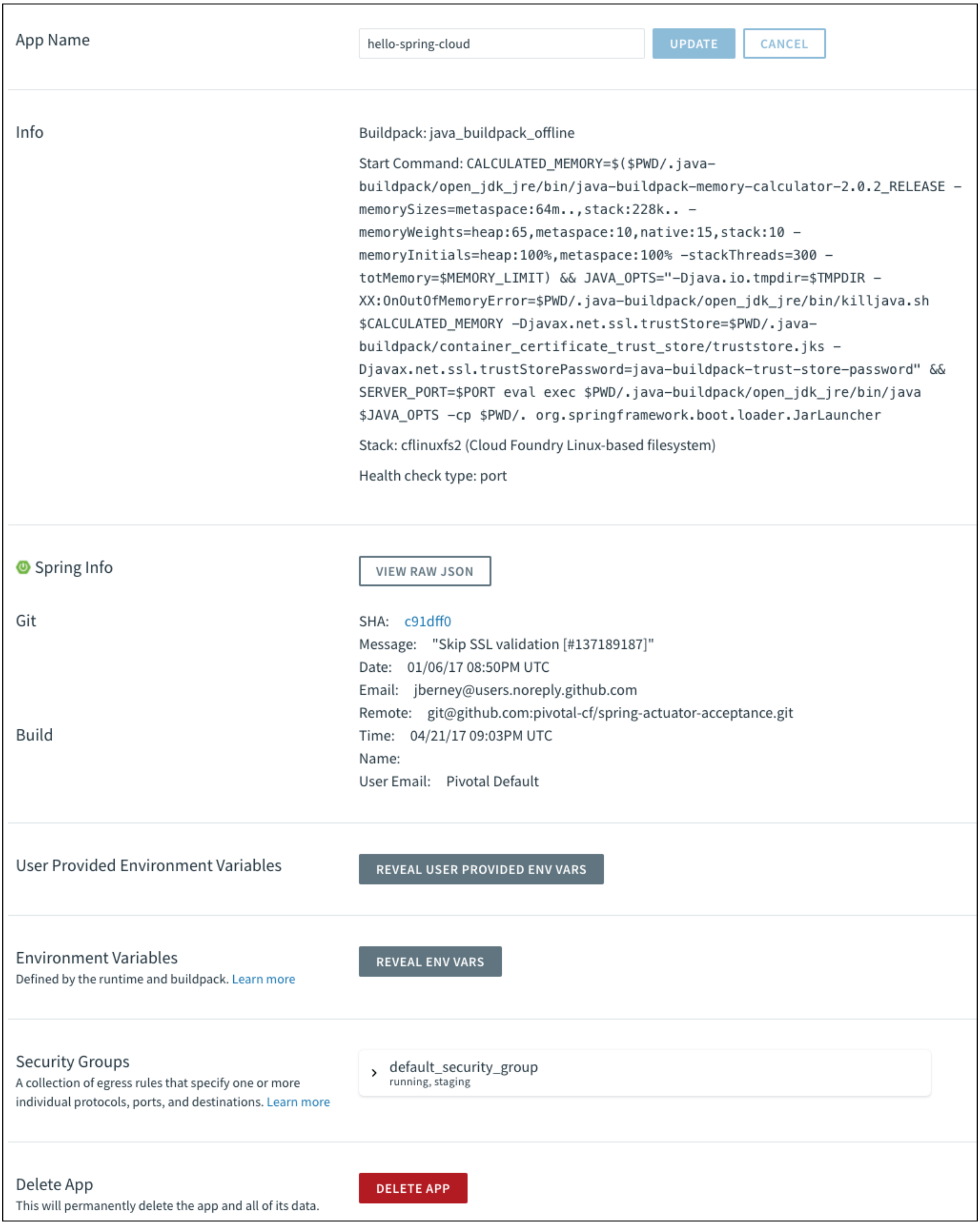
The Apps Manager integration with Spring Boot Actuator requires:

- A TAS for VMs user with the `SpaceDeveloper` role. For more information, see App Space Roles.
- Spring Boot v1.5 or later.
- Completing the procedures in Configure Spring Boot Actuator Endpoints for Apps Manager. After your configure your app, Apps Manager displays the Spring Boot logo next to the name of your app on the app page:



View Build and Git Information for Your App

To view the data that your app sends to its `/info` Actuator endpoint, select the **Settings** tab:



App Name: hello-spring-cloud

Info

Buildpack: java_buildpack_offline

Start Command: CALCULATED_MEMORY=$($PWD/.java-buildpack/open_jdk_jre/bin/java-buildpack-memory-calculator-2.0.2_RELEASE -memorySizes=metaspace:64m..,stack:228k.. -memoryWeights=heap:65,metaspace:10,native:15,stack:10 -memoryInitials=heap:100%,metaspace:100% -stackThreads=300 -totMemory=$MEMORY_LIMIT) && JAVA_OPTS="-Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh $CALCULATED_MEMORY -Djavax.net.ssl.trustStore=$PWD/.java-buildpack/container_certificate_trust_store/truststore.jks -Djavax.net.ssl.trustStorePassword=java-buildpack-trust-store-password" && SERVER_PORT=$PORT eval exec $PWD/.java-buildpack/open_jdk_jre/bin/java $JAVA_OPTS -cp $PWD/. org.springframework.boot.loader.JarLauncher

Stack: cflinuxfs2 (Cloud Foundry Linux-based filesystem)

Health check type: port

Spring Info

Git

SHA: c91dff0
Message: "Skip SSL validation [#137189187]"
Date: 01/06/17 08:50PM UTC
Email: jberney@users.noreply.github.com
Remote: git@github.com:pivotal-cf/spring-actuator-acceptance.git

Build

Time: 04/21/17 09:03PM UTC
Name:
User Email: Pivotal Default

User Provided Environment Variables

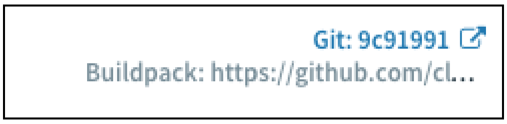
Environment Variables
Defined by the runtime and buildpack. Learn more

Security Groups
A collection of egress rules that specify one or more individual protocols, ports, and destinations. Learn more

default_security_group
running, staging

Delete App
This will permanently delete the app and all of its data.

In the upper right of the app page, Apps Manager also displays the SHA of your app code repository from the latest build:

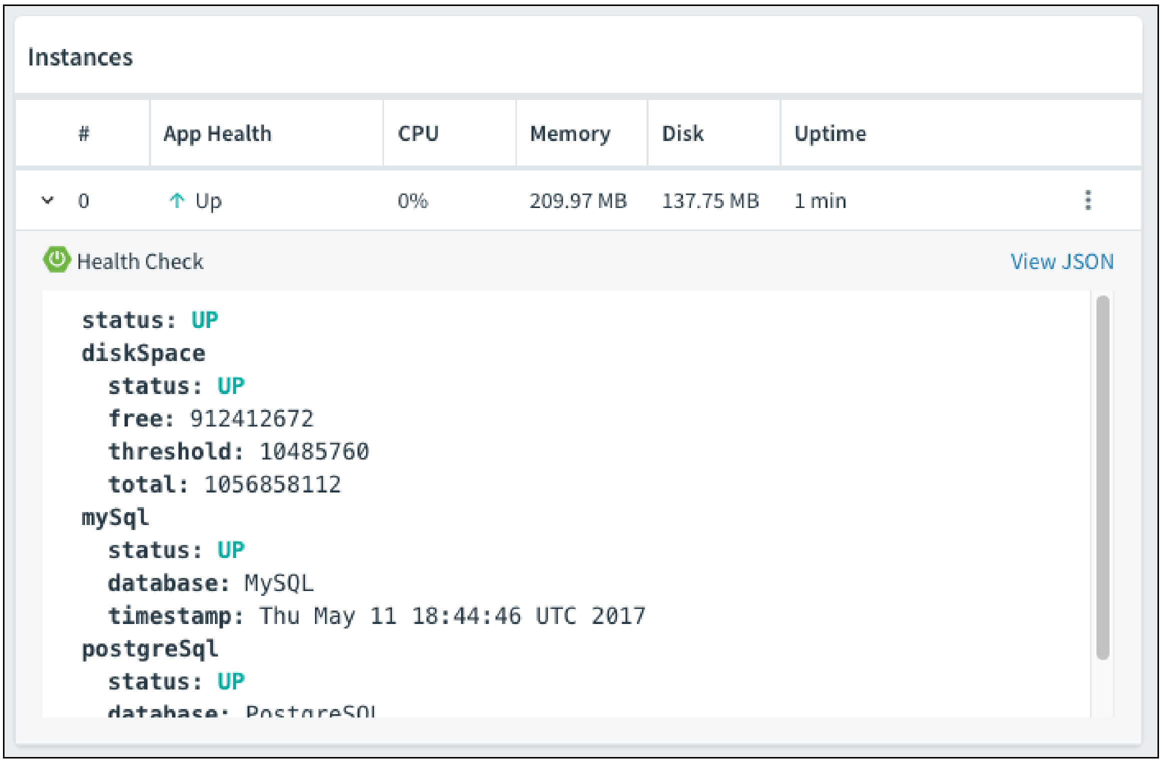


Git: 9c91991
Buildpack: https://github.com/cl...

View App Health

To view the health-check data that your app sends to its `/health` Actuator endpoints:

1. Select the **Overview** tab.
2. Click an instance under the **Instances** section:



Instances

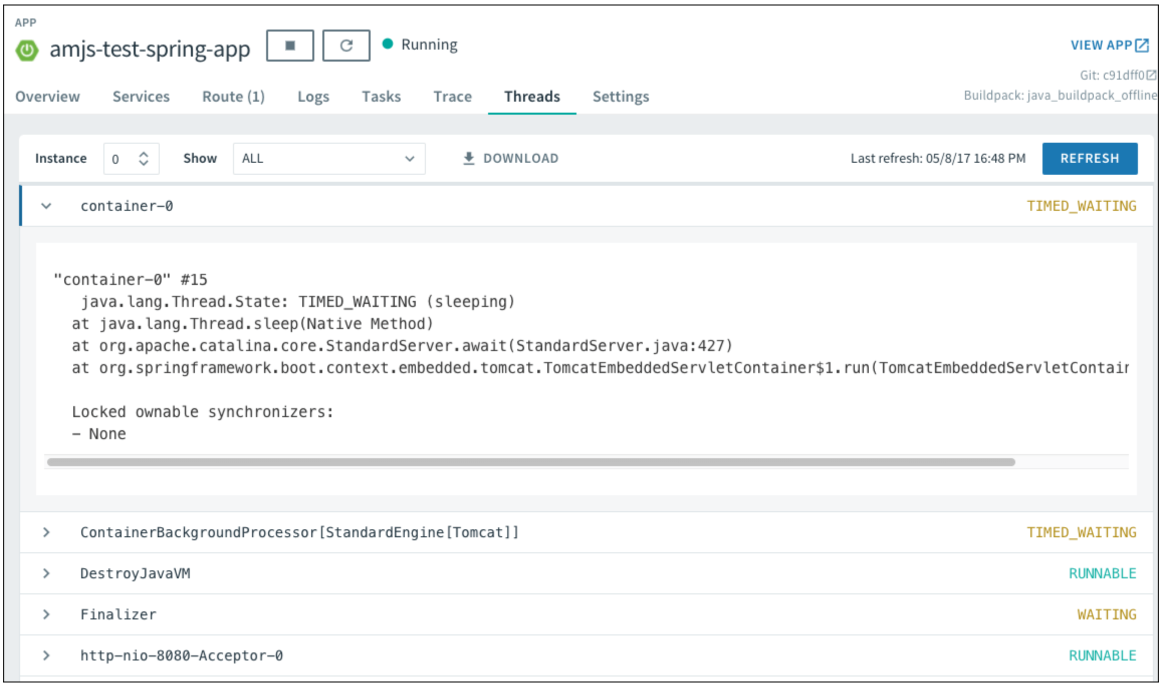
#	App Health	CPU	Memory	Disk	Uptime
0	↑ Up	0%	209.97 MB	137.75 MB	1 min

```
status: UP
diskSpace
  status: UP
  free: 912412672
  threshold: 10485760
  total: 1056858112
mySql
  status: UP
  database: MySQL
  timestamp: Thu May 11 18:44:46 UTC 2017
postgreSql
  status: UP
  database: PostgreSQL
```

View Thread Dump

To trigger and view a thread dump from your app to its `/dump` Actuator endpoint:

1. Select the **Threads** tab.
2. Click **Refresh**.



Instance 0 Show ALL Last refresh: 05/8/17 16:48 PM

container-0 TIMED_WAITING

```
"container-0" #15
   java.lang.Thread.State: TIMED_WAITING (sleeping)
  at java.lang.Thread.sleep(Native Method)
  at org.apache.catalina.core.StandardServer.await(StandardServer.java:427)
  at org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer$1.run(TomcatEmbeddedServletContair

  Locked ownable synchronizers:
  - None
```

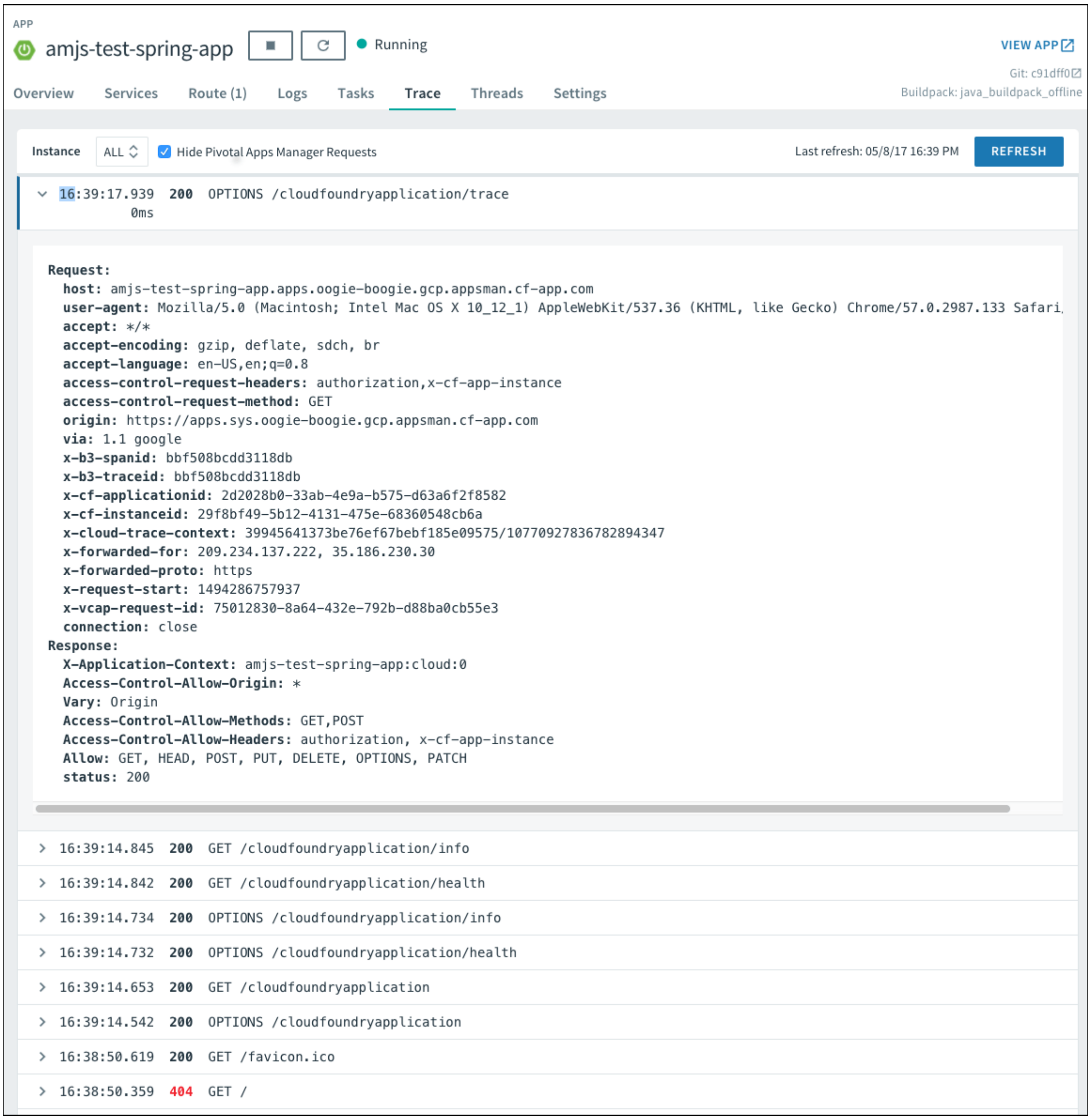
```
ContainerBackgroundProcessor[StandardEngine[Tomcat]]    TIMED_WAITING
DestroyJavaVM                                           RUNNABLE
Finalizer                                               WAITING
http-nio-8080-Acceptor-0                                RUNNABLE
```

You can click each thread to expand and view its details. You can also modify which threads appear on the page using the **Instance** and **Show** dropdowns.

View Request Traces

To retrieve and view tracing information from the `/trace` Actuator endpoint of your app:

1. Select the **Trace** tab.
2. Click **Refresh**.



```
⌄ 16:39:17.939  200  OPTIONS /cloudfoundryapplication/trace
           0ms

Request:
  host: amjs-test-spring-app.apps.oogie-boogie.gcp.appsman.cf-app.com
  user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari,
  accept: */*
  accept-encoding: gzip, deflate, sdch, br
  accept-language: en-US,en;q=0.8
  access-control-request-headers: authorization,x-cf-app-instance
  access-control-request-method: GET
  origin: https://apps.sys.oogie-boogie.gcp.appsman.cf-app.com
  via: 1.1 google
  x-b3-spanid: bbf508bcdd3118db
  x-b3-traceid: bbf508bcdd3118db
  x-cf-applicationid: 2d2028b0-33ab-4e9a-b575-d63a6f2f8582
  x-cf-instanceid: 29f8bf49-5b12-4131-475e-68360548cb6a
  x-cloud-trace-context: 39945641373be76ef67bebf185e09575/10770927836782894347
  x-forwarded-for: 209.234.137.222, 35.186.230.30
  x-forwarded-proto: https
  x-request-start: 1494286757937
  x-vcap-request-id: 75012830-8a64-432e-792b-d88ba0cb55e3
  connection: close
Response:
  X-Application-Context: amjs-test-spring-app:cloud:0
  Access-Control-Allow-Origin: *
  Vary: Origin
  Access-Control-Allow-Methods: GET,POST
  Access-Control-Allow-Headers: authorization, x-cf-app-instance
  Allow: GET, HEAD, POST, PUT, DELETE, OPTIONS, PATCH
  status: 200

› 16:39:14.845  200  GET /cloudfoundryapplication/info
› 16:39:14.842  200  GET /cloudfoundryapplication/health
› 16:39:14.734  200  OPTIONS /cloudfoundryapplication/info
› 16:39:14.732  200  OPTIONS /cloudfoundryapplication/health
› 16:39:14.653  200  GET /cloudfoundryapplication
› 16:39:14.542  200  OPTIONS /cloudfoundryapplication
› 16:38:50.619  200  GET /favicon.ico
› 16:38:50.359  404  GET /
```

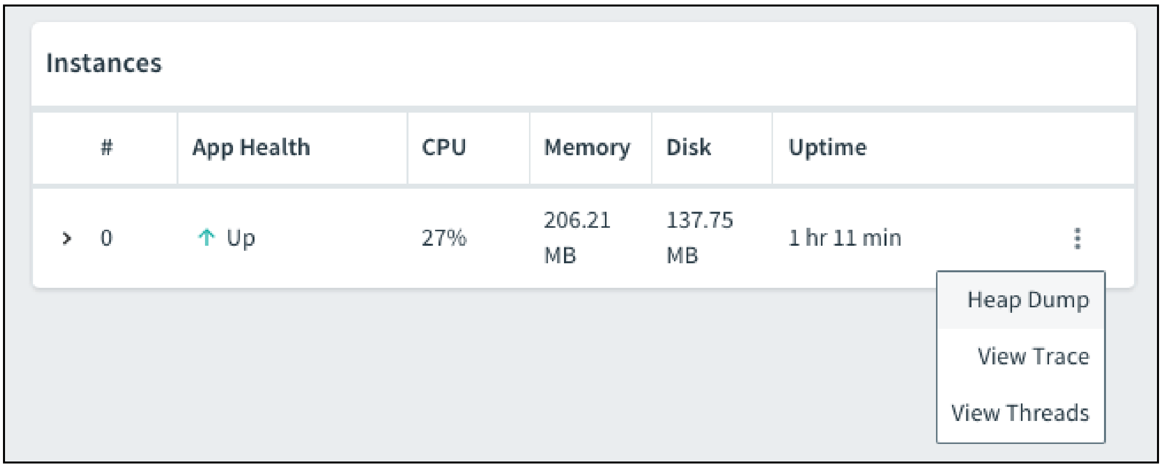
This page displays the last 100 requests from your app. You can click each individual request to expand and view its trace details. You can modify which requests appear on the page using the **Instance** dropdown.

By default, the **Trace** tab does not show requests and responses from Apps Manager polling app instances for data. To include these requests, clear the **Hide Apps Manager Requests** checkbox next to the **Instance** dropdown.

Download Heap Dump

To trigger and view a heap dump from your app to its `/heapdump` endpoint:

1. Select the settings dropdown for an instance of your app.
2. Click **Heap Dump**. This downloads a `.zip` file.



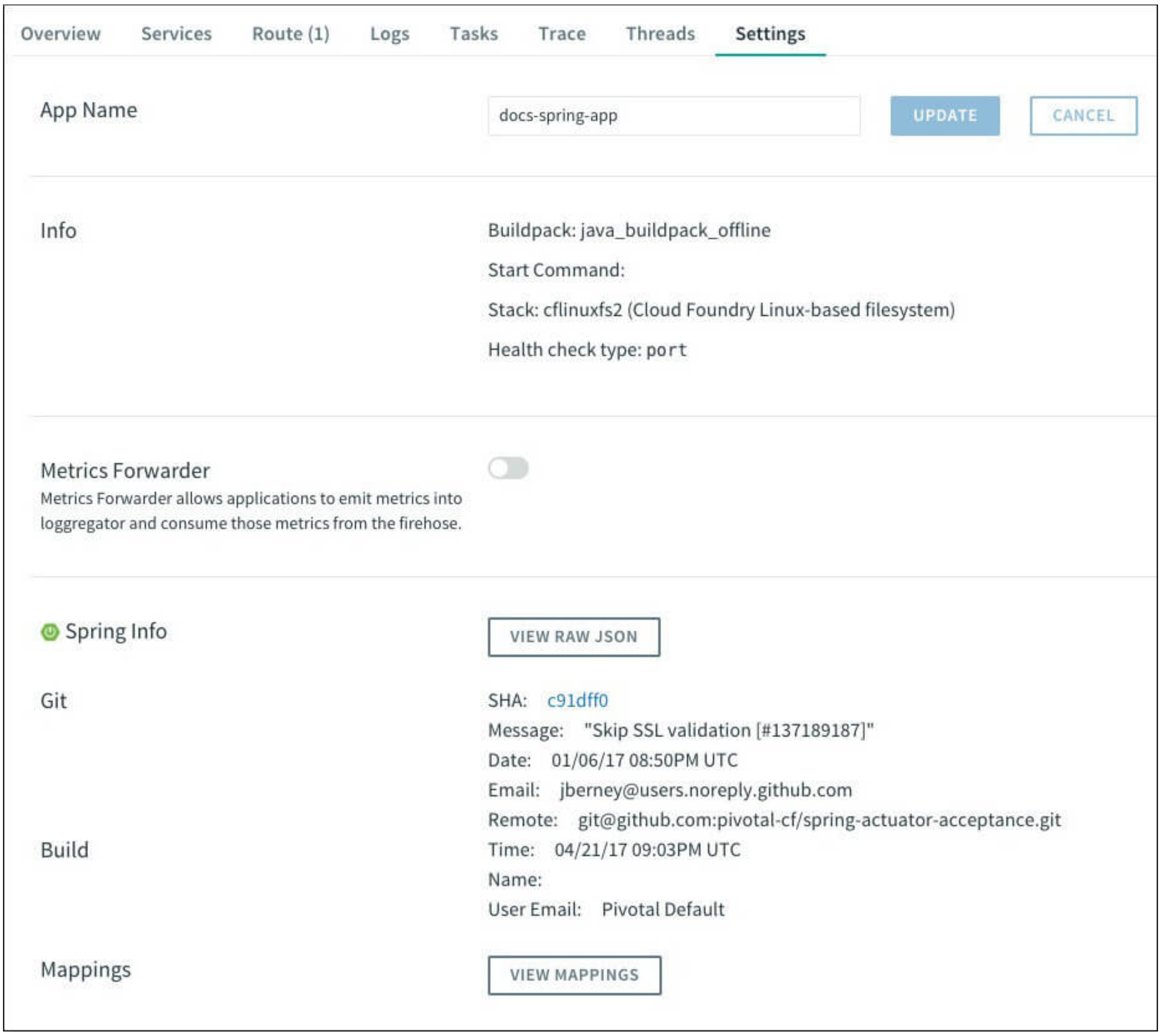
Instances

#	App Health	CPU	Memory	Disk	Uptime
0	↑ Up	27%	206.21 MB	137.75 MB	1 hr 11 min

View Mappings

To view a collated list of the endpoints an app serves:

1. Select the **Settings** tab.
2. Click **View Mappings**.



App Name: docs-spring-app

Info

Buildpack: java_buildpack_offline

Start Command:

Stack: cflinuxfs2 (Cloud Foundry Linux-based filesystem)

Health check type: `port`

Metrics Forwarder

Metrics Forwarder allows applications to emit metrics into loggregator and consume those metrics from the firehose.

Spring Info

Git

SHA: c91dff0

Message: "Skip SSL validation [#137189187]"

Date: 01/06/17 08:50PM UTC

Email: jberney@users.noreply.github.com

Remote: git@github.com:pivotal-cf/spring-actuator-acceptance.git

Build

Time: 04/21/17 09:03PM UTC

Name:

User Email: Pivotal Default

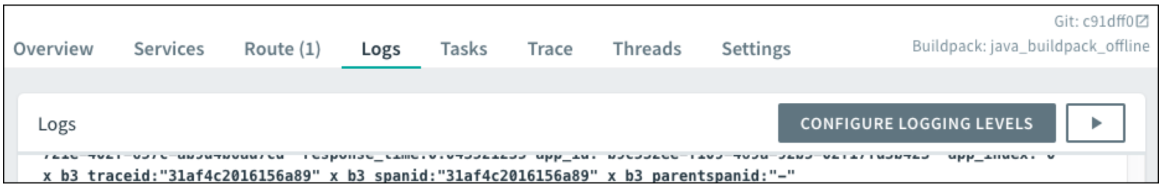
Mappings

Manage Log Levels for Apps Mapped to External Routes

Spring Boot apps include *loggers* for many provided and user components of the app. You can set the log level for each logger in Apps Manager.

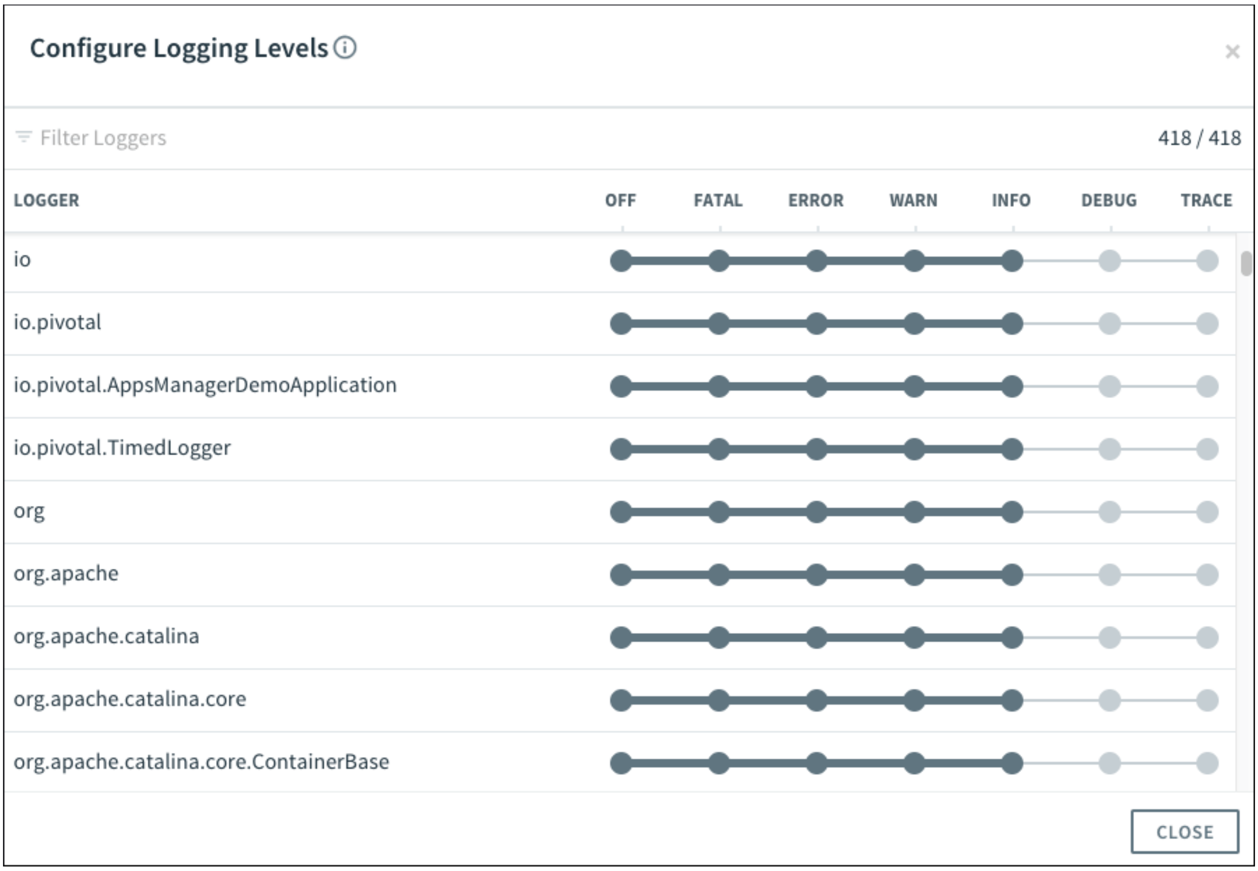
To view the **Configure Logging Levels** screen:

1. Select the **Logs** tab.
2. Click **Configure Logging Levels**.



```
x_b3_traceid:"31af4c2016156a89" x_b3_spanid:"31af4c2016156a89" x_b3_parentspanid:"-"
```

Apps Manager displays the default log level for each logger in gray.



Configure Logging Levels ⓘ

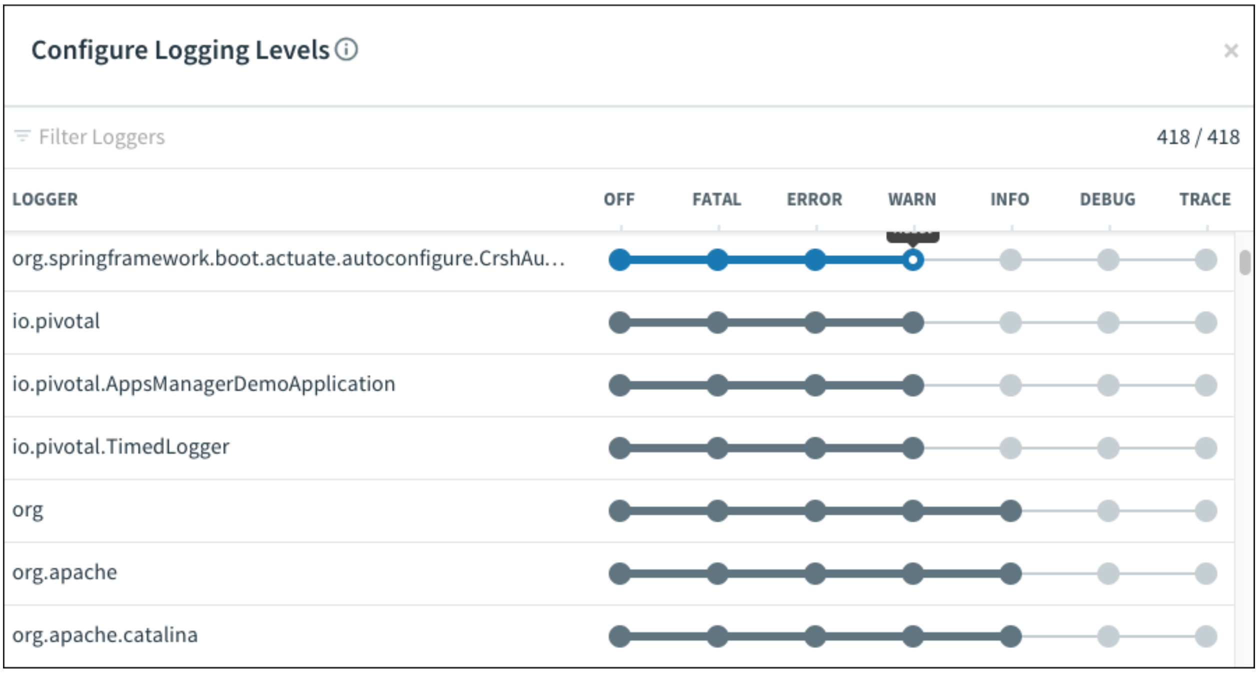
Filter Loggers 418 / 418

LOGGER	OFF	FATAL	ERROR	WARN	INFO	DEBUG	TRACE
io	●	●	●	●	●	○	○
io.pivotal	●	●	●	●	●	○	○
io.pivotal.AppsManagerDemoApplication	●	●	●	●	●	○	○
io.pivotal.TimedLogger	●	●	●	●	●	○	○
org	●	●	●	●	●	○	○
org.apache	●	●	●	●	●	○	○
org.apache.catalina	●	●	●	●	●	○	○
org.apache.catalina.core	●	●	●	●	●	○	○
org.apache.catalina.core.ContainerBase	●	●	●	●	●	○	○

You can modify the log level for a logger by clicking the desired level in the logger row, as shown in the following image. Whenever you set a log level, the following happens:

- The log level displays in blue to indicate that it is user-configured.
- Each child namespace of the logger inherits the log level.

> **Note:** You can manually set any of the child loggers to override this inheritance.

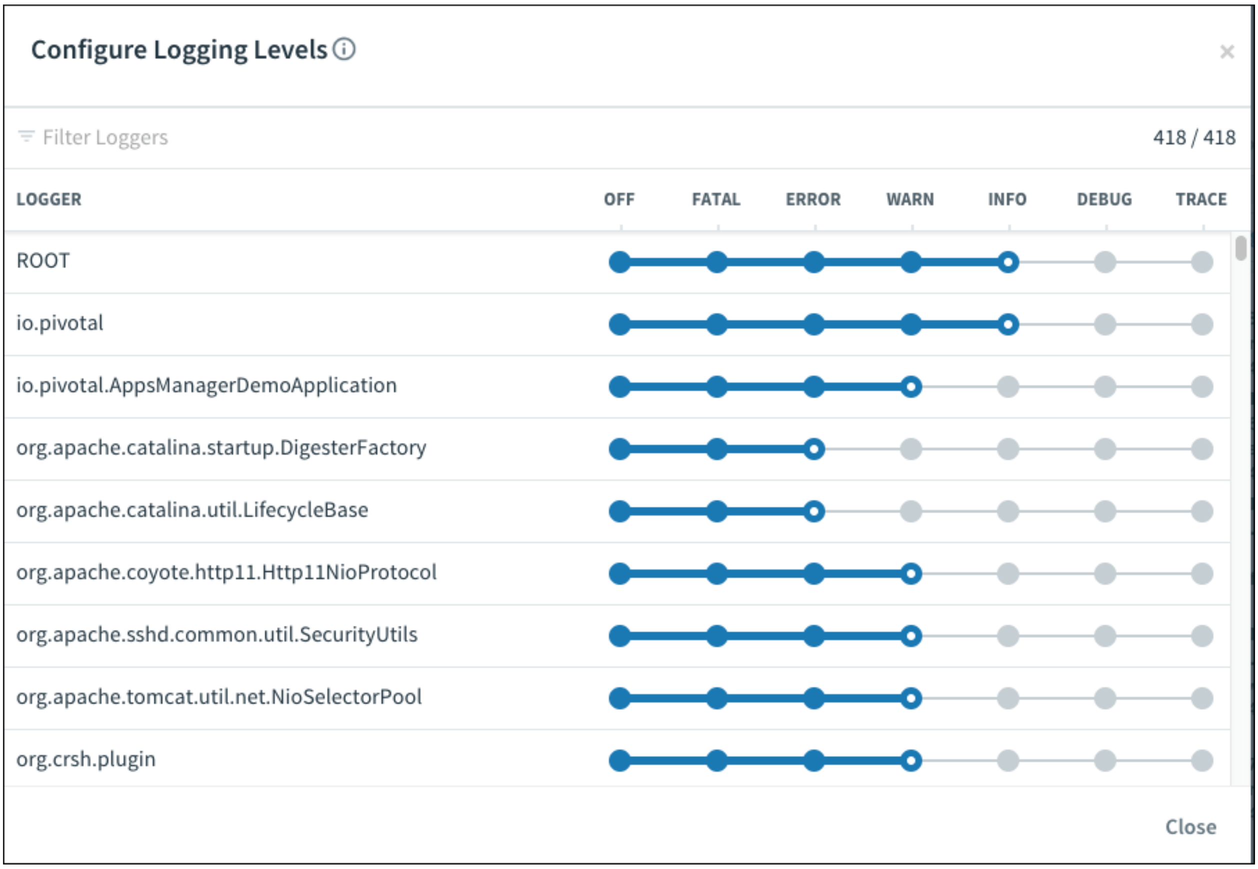


Configure Logging Levels ⓘ ×

Filter Loggers 418 / 418

LOGGER	OFF	FATAL	ERROR	WARN	INFO	DEBUG	TRACE
org.springframework.boot.actuate.autoconfigure.CrshAu...	●	●	●	◉	○	○	○
io.pivotal	●	●	●	●	○	○	○
io.pivotal.AppsManagerDemoApplication	●	●	●	●	○	○	○
io.pivotal.TimedLogger	●	●	●	●	○	○	○
org	●	●	●	●	●	○	○
org.apache	●	●	●	●	●	○	○
org.apache.catalina	●	●	●	●	●	○	○

All of the loggers with user-configured logging levels float to the top of the list.



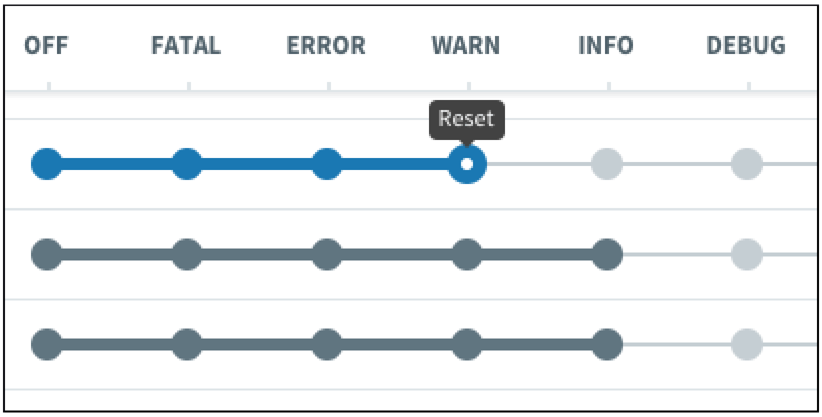
Configure Logging Levels ⓘ ×

Filter Loggers 418 / 418

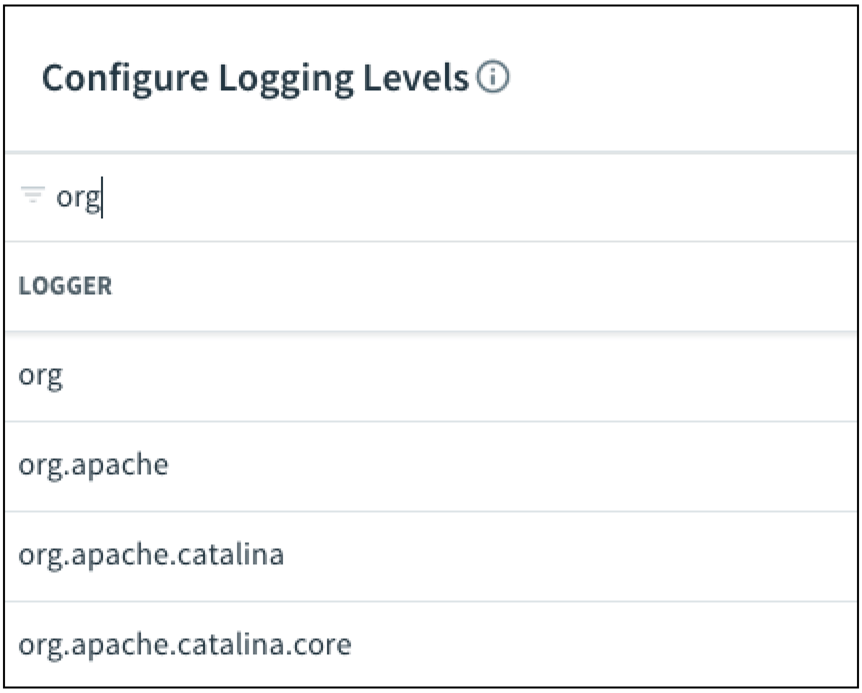
LOGGER	OFF	FATAL	ERROR	WARN	INFO	DEBUG	TRACE
ROOT	●	●	●	●	◉	○	○
io.pivotal	●	●	●	●	◉	○	○
io.pivotal.AppsManagerDemoApplication	●	●	●	◉	○	○	○
org.apache.catalina.startup.DigesterFactory	●	●	◉	○	○	○	○
org.apache.catalina.util.LifecycleBase	●	●	◉	○	○	○	○
org.apache.coyote.http11.Http11NioProtocol	●	●	●	◉	○	○	○
org.apache.sshd.common.util.SecurityUtils	●	●	●	◉	○	○	○
org.apache.tomcat.util.net.NioSelectorPool	●	●	●	◉	○	○	○
org.crsh.plugin	●	●	●	◉	○	○	○

Close

You can reset log levels by clicking the white dot displayed on the current log level.



You can also filter which loggers you see using the **Filter Loggers** textbox.


Configure Logging Levels

org

LOGGER

org

org.apache

org.apache.catalina

org.apache.catalina.core

Troubleshoot Spring Boot Actuator Integration

This section describes how to troubleshoot common issues with the integration of Apps Manager and Spring Boot Actuator.

/cloudfoundryapplication Failed Request

Symptom

You see the following failed request message in your app logs:

```
Could not find resource for relative : /cloudfoundryapplication of full path:
http://example.com/cloudfoundryapplication
```

Explanation

Apps Manager uses the `/cloudfoundryapplication` endpoint as the root for Spring Boot Actuator integrations. It calls this endpoint for an app when you view the app in the Apps Manager UI, regardless of whether you have configured Spring Boot Actuator endpoints for Apps Manager.

Solution

If you are not using the Spring Boot Actuator integrations for Apps Manager, you can ignore this failed request message.

Configuring Multi-Foundation Support in Apps Manager

This topic tells you how to configure multi-foundation support in Apps Manager.

> ⚠ **Caution**: Multi-foundation support in Apps Manager is in beta. You can use this feature to connect foundations that use the same major and minor version of VMware Tanzu Application Service for VMs (TAS for VMs). If you connect foundations that use different TAS for VMs versions, Apps Manager functionality might not work.

Overview

Configuring multi-foundation support in Apps Manager allows you to search, view, and manage orgs, spaces, apps, and service instances across multiple foundations from a single interface.

Configure Multi-Foundation Support

This section explains the procedures for configuring multi-foundation support in Apps Manager.

Configure TAS for VMs

This section tells you how to configure VMware Tanzu Application Service for VMs (TAS for VMs) on a foundation to enable multi-foundation support in Apps Manager.

To configure multi-foundation support in Apps Manager:

1. In the TAS for VMs tile of one of your foundations, select **Apps Manager**.
2. For **Multi-foundation configuration (beta)**, enter a JSON object for each additional foundation that you want to manage. Use the following format for the JSON object:

```
{
  "FOUNDATION-NAME": {
    "ccUrl": "https://api.FOUNDATION-SYSTEM-DOMAIN.com",
    "systemDomain": "FOUNDATION-SYSTEM-DOMAIN.com",
    "usageServiceUrl": "https://app-usage.FOUNDATION-SYSTEM-DOMAIN.com",
    "invitationsServiceUrl": "https://p-invitations.FOUNDATION-SYSTEM-DOMAIN.co
m",
    "logoutUrl": "https://login.FOUNDATION-SYSTEM-DOMAIN.com/logout.do",
    "metricsUrl": "https://metrics.FOUNDATION-SYSTEM-DOMAIN.com",
    "uaaUrl": "https://login.FOUNDATION-SYSTEM-DOMAIN.com"
  }
  }
```

Where:

  - `FOUNDATION-NAME` is a name for the foundation.
  - `FOUNDATION-SYSTEM-DOMAIN` is the system domain of the foundation. The system domain is listed in the **Domains** pane of the TAS for VMs tile.
  - `metricsUrl` is an optional field. If App Metrics is installed on the foundation, provide the App Metrics URL.

3. For **Redirect URIs**, enter a comma-separated list of the URI for each additional foundation on which you enabled multi-foundation support. Use the following format for each URI:

   ```
   https://apps.FOUNDATION-SYSTEM-DOMAIN.com/**
   ```

   Where `FOUNDATION-SYSTEM-DOMAIN` is the system domain of the foundation on which you enabled multi-foundation support.

Add Trusted Certificate Authorities

Apps Manager must be able to validate the certificate authorities (CAs) used by all foundations.

If any foundation uses a certificate for TLS termination that is not signed by a globally-trusted CA, add the CA that signed the TLS certificate as a trusted CA on the foundation on which you enabled multi-foundation support.

Configure the SAML Identity Provider (Optional)

VMware recommends that all foundations use the same external SAML identity provider. This allows Apps Manager to automatically authenticate with the identity provider.

To configure all foundations to use the same SAML identity provider:

1. In the TAS for VMs tile, select **Authentication and Enterprise SSO**.
2. Verify that the foundation uses the correct provider for **Provider name**.
3. Repeat the preceding steps for each foundation in your deployment.

For more information on UAA and identity providers, see Identity Providers in UAA.

For more information on how to set up identity providers in UAA, see Adding Existing SAML or LDAP Users to an Ops Manager Deployment.

Scaling an App Using App Autoscaler

In this section:

About App Autoscaler

You can use App Autoscaler with TAS for VMs to scale your apps and control costs of running them. This article gives a detailed overview of App Autoscaler.

App Autoscaler overview

App Autoscaler is a VMware Tanzu Application Service for VMs (TAS for VMs) Marketplace service that scales apps in your environment based on app performance metrics or a schedule. This controls the cost of running apps while maintaining app performance.

You can use App Autoscaler to do the following:

- Configure scaling rules that adjust app instance counts based on metrics thresholds
- Modify the maximum and minimum number of instances for an app, either manually or following a schedule
- Configure scaling factors so that the app scales more quickly. Use caution when setting the scaling factors by which to scale your applications up or down.

For example, you can configure App Autoscaler to scale down the number of instances for an app over the weekend. You can also configure App Autoscaler to scale up the number of instances for an app when the value of the CPU Usage metric increases above a custom threshold.

About App Autoscaler scaling rules

This section describes how App Autoscaler decides when to scale an app up or down.

It also provides information about the custom metrics, comparison metrics, and default metrics that you can use when you create scaling rules for an app in App Autoscaler.

How App Autoscaler decides when to scale

Every 35 seconds, App Autoscaler makes a decision about whether to scale up, scale down, or keep the same number of instances.

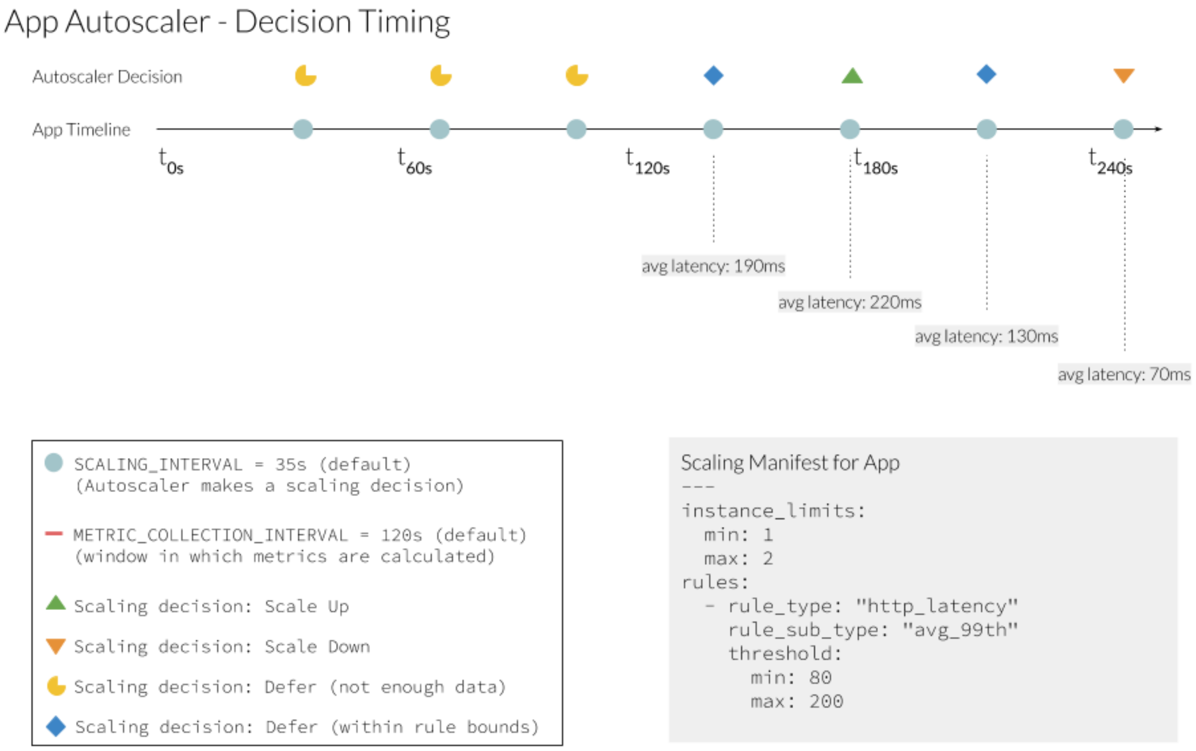
To make a scaling decision, App Autoscaler averages the values of a given metric for the most recent 120 seconds.

> **Note:** Operators can edit the 35 second scaling interval and the 120 second metric collection interval for all apps within the org. For more information, see (Optional) Configure App Autoscaler in *Configuring TAS for VMs*.

App Autoscaler scales apps as follows:

- Increment the instances by the app's scale up factor when any metric exceeds its maximum threshold.
- Decrement the instances by the app's scale down factor when all metrics fall below their minimum thresholds.
- Keep the same number of instances when app metrics do not exceed thresholds.

The following diagram provides an example of how App Autoscaler makes scaling decisions:


App Autoscaler - Decision Timing
Autoscaler Decision
App Timeline
t_{0s}
t_{60s}
t_{120s}
t_{180s}
t_{240s}
avg latency: 190ms
avg latency: 220ms
avg latency: 130ms
avg latency: 70ms
● SCALING_INTERVAL = 35s (default)
(Autoscaler makes a scaling decision)
— METRIC_COLLECTION_INTERVAL = 120s (default)
(window in which metrics are calculated)
▲ Scaling decision: Scale Up
▼ Scaling decision: Scale Down
◔ Scaling decision: Defer (not enough data)
◆ Scaling decision: Defer (within rule bounds)
Scaling Manifest for App

```
---
instance_limits:
  min: 1
  max: 2
rules:
  - rule_type: "http_latency"
    rule_sub_type: "avg_99th"
    threshold:
      min: 80
      max: 200
```

As shown in the diagram, an app has a maximum threshold of 200 milliseconds and a minimum threshold of 80 milliseconds for an HTTP latency metric. The scale up factor and scale down factor are not set in the scaling manifest, so the default value is one.

If HTTP latency averages 220 milliseconds for 120 seconds, App Autoscaler scales the app up one instance.

If HTTP latency then averages 70 milliseconds over the next 120 second window and the app's other scaling metrics also fall below their minimum thresholds, App Autoscaler scales the app down one instance.

If the average value for HTTP latency over a 120 second window is below the maximum threshold of 200 milliseconds and above the minimum threshold of 80 milliseconds, App Autoscaler maintains the same number of instances for the app.

You can also set a maximum and minimum number of instances. For example, if an app exceeds the maximum threshold of a given metric, but the number of instances is already at the maximum number of allowed instances, App Autoscaler does not scale up the app.

Default metrics for scaling rules

App Autoscaler includes several default metrics for which you can create scaling rules.

> **Note:** VMware recommends that you define custom metrics for scaling rules instead of using the default metrics. Custom metrics allow you to more accurately monitor the performance of your apps based on your environment.

The following table lists the default metrics for App Autoscaler:

Metric	Description	Notes
CPU Utilization	Average CPU percentage for all instances of the app.	App CPU utilization data can vary greatly based on the number of CPU cores on Diego Cells and app density. For more information, see App Autoscaler advisory for scaling Apps based on the CPU utilization in the Knowledge Base.
Container Memory Utilization	Average memory percentage for all instances of the app.	
HTTP Throughput	Total HTTP requests per second (divided by the total number of app instances).	
HTTP Latency	Average latency of apps response to HTTP requests. This does not include Gorouter processing time or other network latency. Average is calculated on the middle 99% or middle 95% of all HTTP requests.	
RabbitMQ Depth	The queue length of the specified queue.	

Custom metrics for scaling rules

VMware recommends that you define custom metrics for App Autoscaler scaling rules. Custom metrics allow you to define the metrics that are the best indicators of app performance for your environment.

You can configure apps to emit custom metrics out of the Loggregator Firehose using Metric Registrar. For steps on how to configure your apps to emit custom metrics with Metric Registrar, see Registering Custom App Metrics.

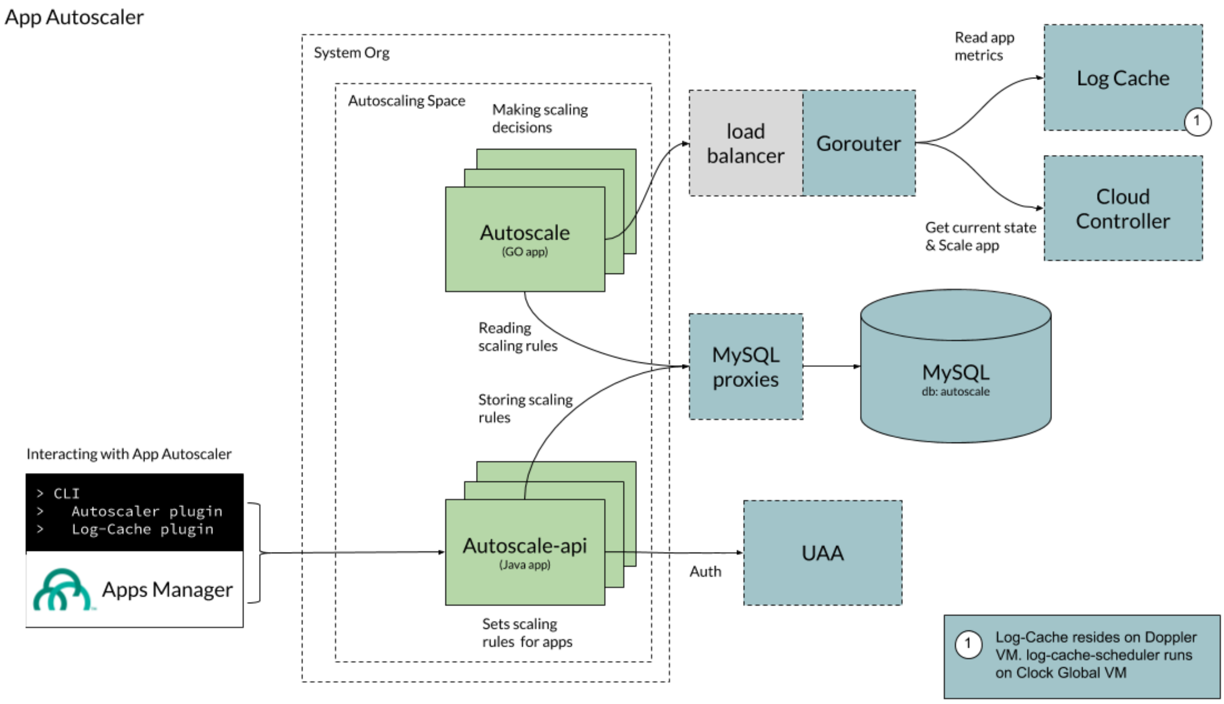
Comparison metrics for scaling rules

You can use the **Comparison Metric** field in App Autoscaler to define a scaling rule that divides one custom metric by another.

When you add a scaling rule, the **Metric** field is the dividend and the **Comparison Metric** field is the divisor.

App Autoscaler architecture

The following diagram shows the components and architecture of App Autoscaler. It also shows how App Autoscaler components interact with VMware Tanzu Application Service for VMs (TAS for VMs) components to make app scaling decisions.



View a larger version of this image.

As demonstrated in the architecture diagram, App Autoscaler makes scaling decisions based on autoscaling rules that users configure by using either the Cloud Foundry Command Line Interface (cf CLI) or Apps Manager. The Autoscale API stores these autoscaling rules in a MySQL database.

At a predefined interval, known as the scaling interval, the App Autoscaler app reads the scaling rules and retrieves app metric data from the Loggregator Log Cache. Then, App Autoscaler makes a scaling decision and communicates with the Cloud Controller to scale the app, if necessary.

For more information about Loggregator Log Cache, see Loggregator Architecture. For more information about the Cloud Controller, see Cloud Controller.

Using App Autoscaler to scale apps in TAS for VMs

With the Space Developer role, you can configure the App Autoscaler service in Apps Manager to automatically scale apps in your TAS for VMs deployment.

You can use the App Autoscaler command-line interface (CLI) plugin to configure App Autoscaler rules from the command line. For more information, see Using the App Autoscaler CLI.

> **Note:** Space Managers, Space Auditors, and all Org roles do not have permission to use App Autoscaler. For help managing user roles, see Managing User Accounts and Permissions Using Apps Manager.

Overview

To use App Autoscaler to automatically scale your apps in you VMware Tanzu Application Service for VMs (TAS for VMs) deployment, you must create and bind the App Autoscaler service, then configure scaling rules for App Autoscaler.

For more information about App Autoscaler, including information about how to create scaling rules, see About App Autoscaler.

Prerequisite

Before using the Autoscaler service in Apps Manager, you must configure the `search-server` app for any deployments that have restrictive networking policies around request proxying.

Autoscaler controls may not appear for apps in Apps Manager if your TAS for VMs deployment has restrictive networking policies around request proxying.

To resolve this issue and enable Autoscaler controls in Apps Manager:

1. Using the cf CLI, log in to the system org and system space.
2. Locate the `search-server` app.
3. Update the `no_proxy` environment variable for the `search-server` app to include your system domain.

   ```
   cf set-env search-server no_proxy '*.SYSTEM-DOMAIN'
   ```

   where `SYSTEM-DOMAIN` is the system domain configured for your TAS for VMs deployment. For example:

   ```
   cf set-env search-server no_proxy '*.example.com'
   ```

4. Restage the `search-server` app.

   ```
   cf restage search-server
   ```

Configure App Autoscaler

To configure App Autoscaler:

1. Create and bind the App Autoscaler service to your app. See Create and Bind the App Autoscaler Service.

2. Configure autoscaling instance limits for your app. See Configure Autoscaling for an App.

Create and bind the App Autoscaler service

To use App Autoscaler, you must create an instance of the App Autoscaler service and bind it to any app you want to autoscale. You can do this using either the Apps Manager or from the Cloud Foundry Command Line Interface (cf CLI):

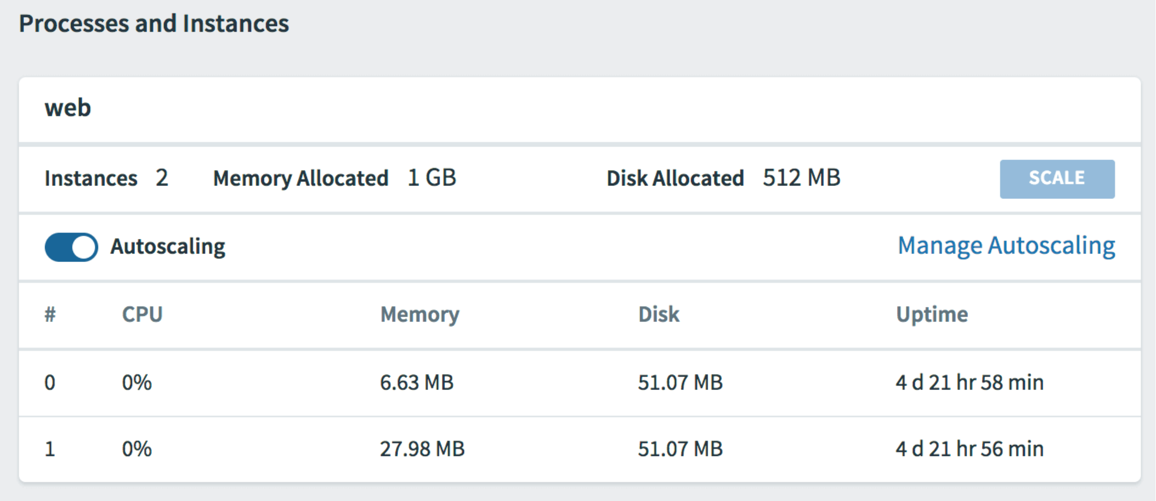
- **Apps Manager**:
  1. Create an instance of the service.
  2. Bind the service to an app.
- **cf CLI**:
  1. Create an instance of the service.
  2. Bind the service to an app.

> **Note:** Manual scaling overrides scaling rules that you configure with App Autoscaler. If you manually scale an app bound to an App Autoscaler service instance, the App Autoscaler instance automatically unbinds from that app, and the app scales to the manual setting. For more information, see Managing Apps and Service Instances Using Apps Manager.

To enable App Autoscaler for an app:

1. In Apps Manager, select an app from the space in which you created the App Autoscaler service.

2. Under **Processes and Instances**, enable **Autoscaling**.



Processes and Instances

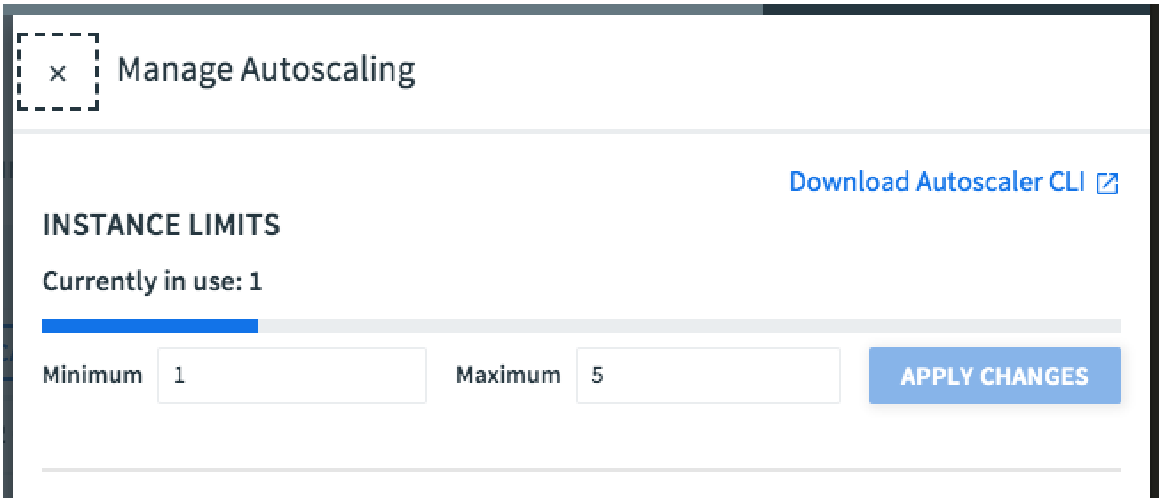
web

Instances 2 **Memory Allocated** 1 GB **Disk Allocated** 512 MB

● **Autoscaling**

#	CPU	Memory	Disk	Uptime
0	0%	6.63 MB	51.07 MB	4 d 21 hr 58 min
1	0%	27.98 MB	51.07 MB	4 d 21 hr 56 min

3. Click **Manage Autoscaling** to configure instance limits, scaling rules, and scheduled limit changes for the app. For more information, see Configure Autoscaling for an App.



Manage Autoscaling

INSTANCE LIMITS

Currently in use: 1

Minimum 1

Maximum 5

Configure autoscaling for an app

App Autoscaler keeps instance counts within an allowable range defined by minimum and maximum values. The minimum and maximum values are called instance limits.

For more information about how App Autoscaler makes scaling decisions for an app, see How App Autoscaler Determines When to Scale in *About App Autoscaler*.

Follow the procedures in the sections below:

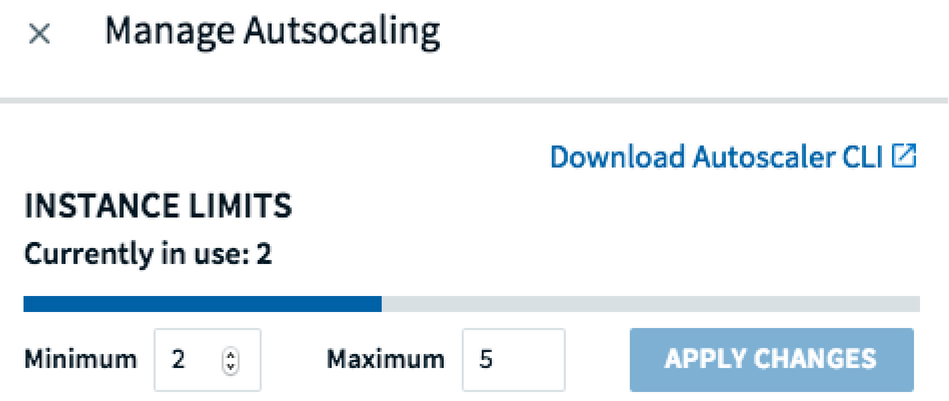
- Create or Modify Instance Limits
- Add or Delete Scaling Rules
- Create or Modify Scheduled Limit Changes

Create or modify instance limits

This section describes how to create and edit instance limits for app scaling rules. You can also schedule changes to your instance limits for a specific date and time. For more information, see Scheduled Limit Changes.

To manually modify instance limits:

1. In Apps Manager, navigate to the **Overview** page for your app. Under **Processes and Instances**, click **Manage Autoscaling**.
2. In the **Instance Limits** section, set values for **Minimum** and **Maximum**.



INSTANCE LIMITS

Currently in use: 2

Minimum 2

Maximum 5

3. Click **Apply Changes**.

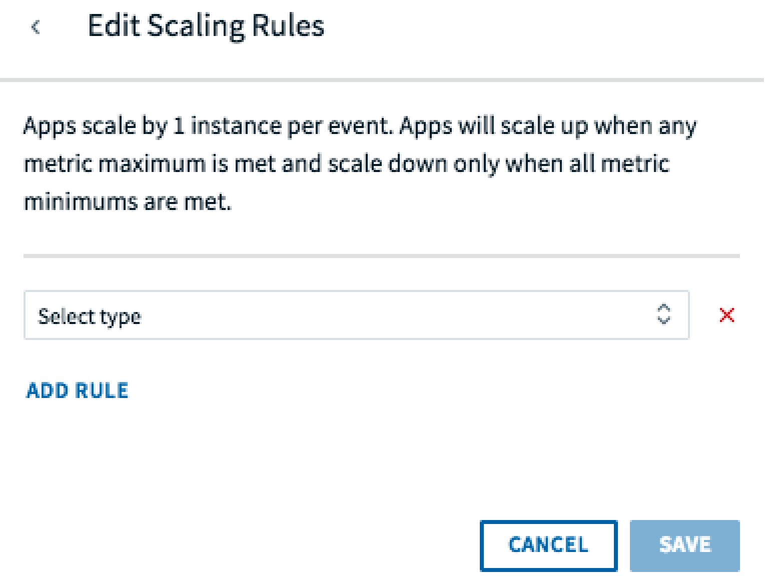
Add or Delete scaling rules

The following procedures describe how to add and delete scaling rules for your apps with App Autoscaler.

For more information about scaling rules and metrics in App Autoscaler, see About App Autoscaler Scaling Rules.

To add a scaling rule for an app:

1. In the **Manage Autoscaling** pane, click **Edit** next to **Scaling Rules**. The **Edit Scaling Rules** pane appears.
2. Click **Add Rule**.
3. In the **Select type** dropdown, select the metric for the new scaling rule.



Apps scale by 1 instance per event. Apps will scale up when any metric maximum is met and scale down only when all metric minimums are met.

4. Set the minimum and maximum thresholds for the metric. For information about setting instance limits, see About App Autoscaler Scaling Rules in *About App Autoscaler*.
5. Select or fill in any other fields that appear under the threshold fields:

   - If you are adding an **HTTP Latency** rule, configure **Percent of traffic to apply**.
   - If you are adding a **RabbitMQ** depth rule, provide the name of the queue to measure.
   - If you are adding a **Custom** rule, enter your custom **Metric**.
   - If you are adding a **Compare** rule, enter values in the **Metric** and **Comparison Metric** fields.
6. Click **Save**.

To delete a scaling rule for an app:

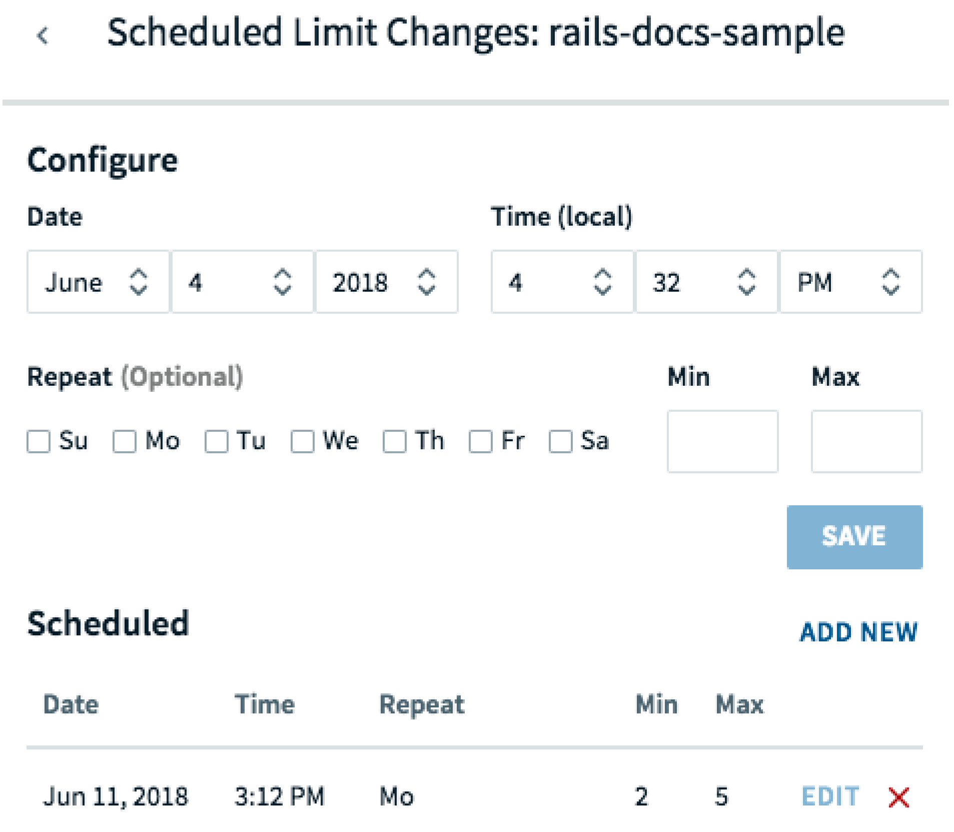
1. Click the **x** icon next to the rule you want to delete.
2. Click **Save**.

Create or modify scheduled limit changes

Because app demand often follows a weekly, daily, or hourly schedule, you can schedule App Autoscaler to change the allowable instance range to track expected surges or quiet periods.

To create or modify a scheduled limit change:

1. Click **Edit** next to **Scheduled Limits**.
2. Click **Add New** to add a new scheduled limit or select an existing entry to modify by clicking **EDIT** next to the entry.
3. Edit the following values:
   - **Date** and **Time (local)**: Set the date and time of the change.
   - **Repeat (Optional)**: Set the day of the week for which you want to repeat the change.
   - **Min** and **Max**: Set the allowable range within which App Autoscaler can change the instance count for an app.



Scheduled Limit Changes: rails-docs-sample

Configure

Date: June, 4, 2018

Time (local): 4, 32, PM

Repeat (Optional)

□ Su □ Mo □ Tu □ We □ Th □ Fr □ Sa

Min

Max

Scheduled

Date	Time	Repeat	Min	Max	
Jun 11, 2018	3:12 PM	Mo	2	5	EDIT ✕

4. Click **Save**.

To delete an existing entry, click the **x** icon next to the entry you want to delete.

For example, to schedule an app to scale down for a weekend, you can enter two rules as follows:

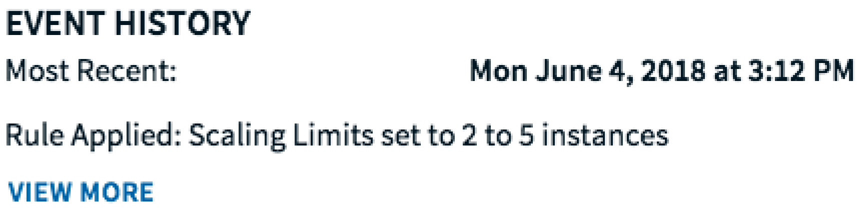
1. Scale down to a single instance on Friday evening:
   - **Date** and **Time (local)**: `Dec`, `2`, `2018` and `7:00 PM`
   - **Repeat (Optional)**: `Fr`
   - **Min** and **Max**: `1` and `1`
2. Increase instances to between 3 and 5 on Monday morning:
   - **Date** and **Time (local)**: `Dec`, `5`, `2018` and `7:00 AM`
   - **Repeat (Optional)**: `M`
   - **Min** and **Max**: `3` and `5`

View and manage App Autoscaler events and notifications

App Autoscaler logs all autoscaling events. You can view event history and manage notifications for App Autoscaler.

View event history

To view all autoscaling events in the past 24 hours, click **View More** in the **Event History** section of the **Manage Autoscaling** pane.



EVENT HISTORY

Most Recent: Mon June 4, 2018 at 3:12 PM

Rule Applied: Scaling Limits set to 2 to 5 instances

Manage App Autoscaler notifications

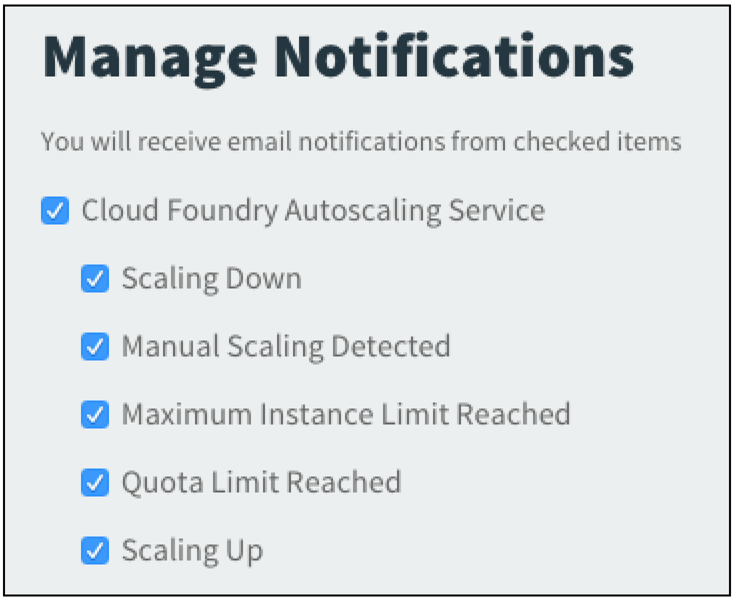
App Autoscaler emails or texts its event notifications to all users with the Space Developer role by default.

To subscribe or unsubscribe from autoscaling event notifications:

1. Navigate to the **Manage Notifications** page in Ops Manager.

   > **Note:** If installed, Notifications Management should be available at `https://notifications-ui.SYSTEM-DOMAIN/preferences`, where `SYSTEM-DOMAIN` is your system domain.

2. Choose which notifications you want to receive from App Autoscaler:



Manage Notifications

You will receive email notifications from checked items

☑ Cloud Foundry Autoscaling Service

- ☑ Scaling Down
- ☑ Manual Scaling Detected
- ☑ Maximum Instance Limit Reached
- ☑ Quota Limit Reached
- ☑ Scaling Up

Tutorial: Scaling a Spring App on a custom metric

With App Autoscaler, you can scale an app in your TAS for VMs deployment based on a custom metric. This tutorial walks you through the steps.

Overview

In a VMware Tanzu Application Service for VMs (TAS for VMs) deployment, Autoscaler can automatically scale apps based on custom metrics. The following table describes the main components involved in this workflow and how they correspond to steps in this tutorial.

Component	Description	Related Tutorial Steps
App	The app must emit custom metrics that are created with the open-source tool, Prometheus. This tutorial includes a sample Spring app that does that.	Review the Sample App and Push the Sample App
Metric Registrar	The Metric Registrar is a component of TAS for VMs that allows app developers to export custom app metrics to the logging system. It has its own CLI plugin.	Register a Custom Metrics Endpoint
App Autoscaler	App Autoscaler is a service integrated with Apps Manager that automatically scales apps in your environment based on app metrics or a schedule. See the Scaling an App Using App Autoscaler topic for more information.	Create and Autoscaling Rule and Trigger Scaling

Prerequisites

This tutorial requires the following:

- A TAS for VMs environment with the Metric Registrar enabled. You can confirm this with your platform operator.
- Access to Apps Manager in the TAS for VMs environment.
- The ability to push an app to the TAS for VMs environment. For example, you need space developer permissions in at least one space.
- The Cloud Foundry Command Line Interface (cf CLI). See Installing the cf CLI.
- Access to a command-line for running commands such as cf CLI and git commands.

Review the sample app

The sample app code is in the pivotal-cf/metric-registrar-examples GitHub repository. It is a Spring app with a simple UI that includes several buttons to call different endpoints. Some of these endpoints are instrumented to produce metrics.

You can see what the UI looks like in the Push the Sample App section, which includes a screenshot. The following sections provide some details about the code.

Dependencies

You can view the app dependencies in the build.gradle file:

```
dependencies {
    implementation('io.micrometer:micrometer-registry-prometheus')
    implementation('org.springframework.boot:spring-boot-starter-actuator')
    implementation('org.springframework.boot:spring-boot-starter-security')
    implementation('org.springframework.boot:spring-boot-starter-web')
    testImplementation('org.springframework.boot:spring-boot-starter-test')
    testImplementation('org.springframework.security:spring-security-test')
}
```

The dependencies include the Micrometer Prometheus library, which does the following:

- Creates a metrics endpoint at `/actuator/prometheus` in a format supported by the Metric Registrar.
- Allows you to instrument the app by creating new metrics. See Instrumentation below.

The Spring Security dependency exposes the endpoints so they can be reached by Metric Registrar.

Instrumentation

This section describes how the app is *instrumented*. Instrumentation refers to how metrics have been added for a particular function.

See the ExampleController.java file from the sample app code:

- The `ExampleController` class includes a `MeterRegistry` object, which is passed and set in the constructor.

  ```
  private MeterRegistry registry;
  private AtomicLong custom;

  public ExampleController(MeterRegistry registry) {
     this.registry = registry;
     this.custom = new AtomicLong(0L);
  }
  ```

- It also includes a `custom` variable that is initialized in the constructor. In the customMetric handler, this variable gets passed to the registry and incremented or decremented.

  ```
  public ResponseEntity<String> customMetric(@RequestParam(value="inc", defaultVa
  lue="") String increment) {
      AtomicLong customGauge = registry.gauge("custom", this.custom);
      if (!"".equals(increment)) {
          customGauge.incrementAndGet();
      } else {
          customGauge.decrementAndGet();
      }
  ```

  Note: The App Autoscaler only scales on a gauge, or metric, that can go up and down. The standard metrics of CPU, disk, HTTP throughput, and HTTP latency are all gauges.

Push the sample app

To push the sample app:

1. Open a terminal window and clone the git repository that contains the sample app:

   ```
   git clone git@github.com:pivotal-cf/metric-registrar-examples.git
   ```

2. Navigate to the app directory:

   ```
   cd metric-registrar-examples/java-spring-security
   ```

3. Build the app:

   ```
   ./gradlew build
   ```

4. Push the app with a random route:

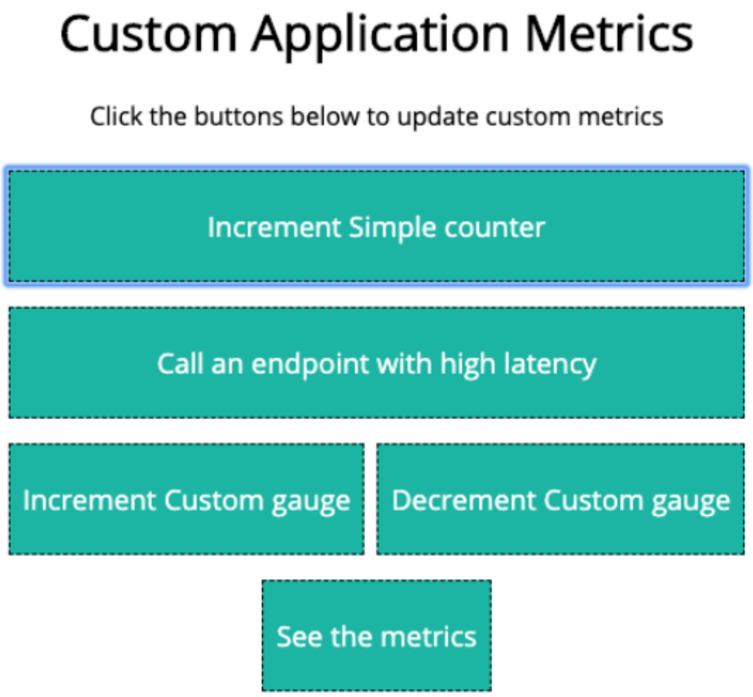
   ```
   cf push --random-route
   ```

5. After the `push` command finishes, locate the app URL in the `routes:` section of the output. See the following example output:

   ```
   Waiting for app to start...
           Uploaded droplet (60.5M)
           Uploading complete
           Cell 333e7fdf-806e-424d-b3a0-78967ecb6d28 stopping instance 6f3
   45835-8beb-48a5-b578-921f5de442c6

   name:               tutorial-example
   requested state:    started
   routes:             tutorial-example-random-route.cfapps.io
   last uploaded:      Wed 28 Aug 11:02:33 PDT 2019
   ```

6. In a browser, navigate to the app URL. The app UI looks like what you see in the following image:

   Custom Application Metrics

   Click the buttons below to update custom metrics

   It has the following buttons:

   - **Increment Custom gauge** and **Decrement custom gauge**:
     These buttons cause the custom metric to increase or decrease by a value of `1`.

You use these buttons later when you [Trigger Scaling](#trigger).

- **See the metrics**:
  This button opens `/actuator/prometheus` in your browser. You can use it to view values for `custom` and all the metrics produced by Micrometer. This page is important because Metric Registrar uses it to collect metrics.
- **Increment Simple counter** and **Call an endpoint with high latency**:
  You can ignore these buttons, as they are not used in this tutorial. To learn more about what they do, see the app code.

Register a custom metrics endpoint

When you want to your app to emit custom metrics, you register the app as a metric source with the *Metric Registrar*.

To register a custom metrics endpoint for the app:

1. Install the Metric Registrar CLI:

   ```
   cf install-plugin -r CF-Community "metric-registrar"
   ```

2. Register the metrics endpoint of the app. Because the app dependencies include the Micrometer Prometheus library, there is automatically a metrics endpoint at `/actuator/prometheus`. Run the following command and replace `APP-NAME` with the name of your app.

   ```
   cf register-metrics-endpoint APP-NAME /actuator/prometheus
   ```

3. Install the Log Cache CLI. Log Cache is a feature in TAS for VMs that lets you filter and query app logs. For more information, see Example Uses of the Log Cache CLI Plugin.

   ```
   cf install-plugin -r CF-Community "log-cache"
   ```

4. View the app metrics using the following command. The `--follow` flag appends output as metrics are emitted.

   ```
   cf tail APP-NAME --envelope-class metrics --follow
   ```

   The output looks similar to the following example. The `custom` metric displays in the output.

   ```
   Retrieving logs for app tutorial-example in org sandbox / space develop
   ment as example@user...

   2019-08-28T09:17:56.28-0700 [tutorial-example/1] GAUGE cpu:0.289158 per
   centage disk:135716864.000000 bytes disk_quota:1073741824.000000 bytes
   memory:399979315.000000 bytes memory_quota:2147483648.000000 bytes
   2019-08-28T09:17:56.50-0700 [tutorial-example/0] GAUGE custom:1.000000
   ```

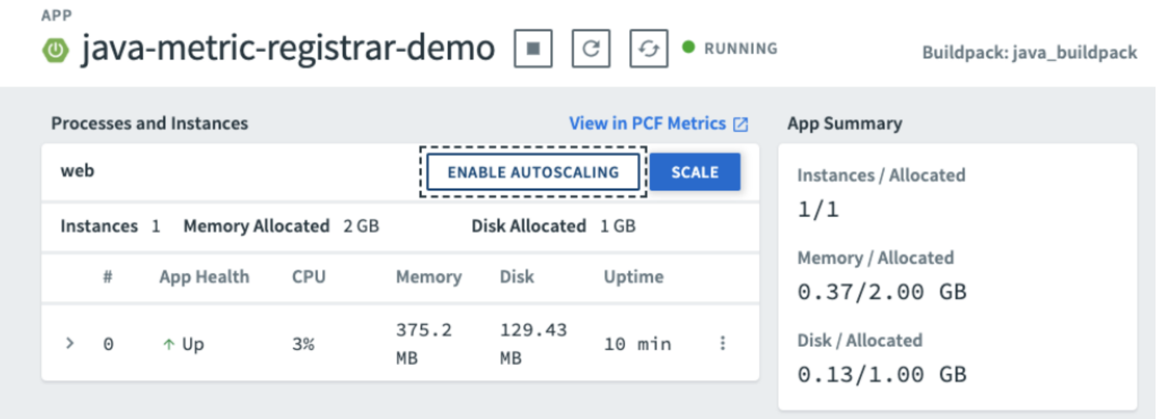
   Note: If you do not see output similar to the previous exampple, Metric Registrar might not be enabled in your Ops Manager installation. Contact your platform operator to confirm.

Create an autoscaling rule

The App Autoscaler component is integrated with Apps Manager. This is where you can create rules.

To create an autoscaling rule for the app and custom metric:

1. Navigate to the app in Apps Manager.
2. Click **Enable Autoscaling**.



Instances 1 Memory Allocated 2 GB Disk Allocated 1 GB

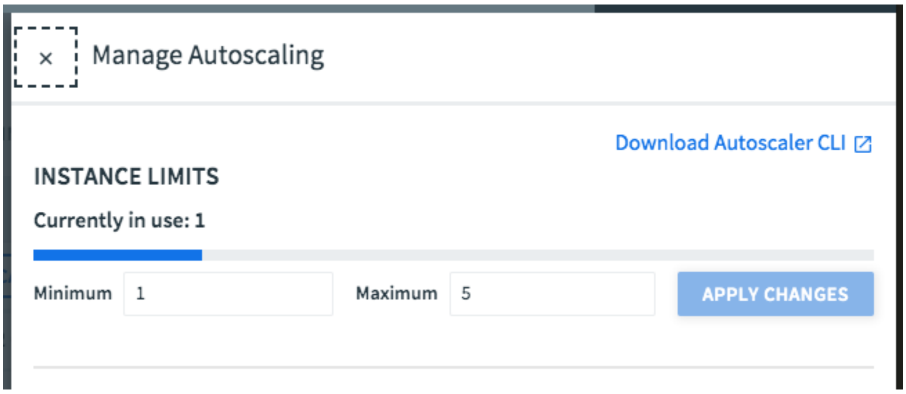
#	App Health	CPU	Memory	Disk	Uptime
0	↑ Up	3%	375.2 MB	129.43 MB	10 min

Instances / Allocated 1/1

Memory / Allocated 0.37/2.00 GB

Disk / Allocated 0.13/1.00 GB

3. Click **Manage Autoscaling**.
4. Modify the **Instance Limits**.
   App Autoscaler keeps instance counts within a range defined by minimum and maximum values, or *instance limits*.
    1. For **Minimum**, enter `1`.
    2. For **Maximum**, enter `5`.
    3. Click **Apply Changes**.

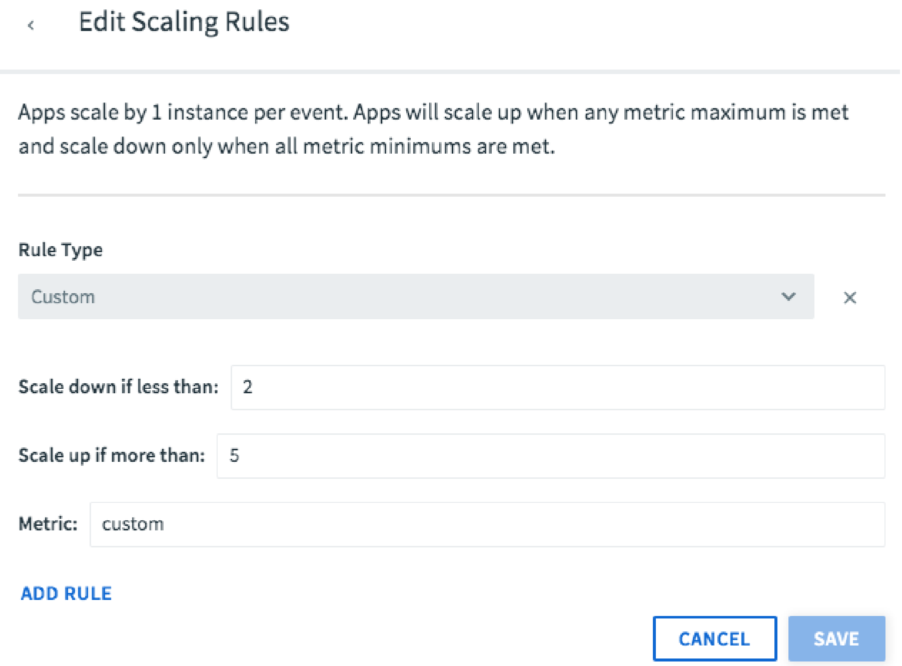


Currently in use: 1

Minimum 1 Maximum 5

5. Create an autoscaling rule.
   App Autoscaler increases or decreases instance counts based on how a current metric compares with configured **Scale up** and **Scale down** thresholds.
    1. In the **Scaling Rules** section, click **EDIT**.
    2. Click **ADD RULE**.

3. For **Rule Type**, select `Custom`.
4. For **Scale down if less than**, enter `2`. This and the following value are examples for the purposes of demonstrating the feature in this tutorial.
5. For **Scale up if more than**, enter `5`.
6. For **Metric**, enter `custom`. This is the name of the metric specified in the app code.
7. Click **Save**.



Edit Scaling Rules

Apps scale by 1 instance per event. Apps will scale up when any metric maximum is met and scale down only when all metric minimums are met.

Rule Type: Custom

Scale down if less than: 2

Scale up if more than: 5

Metric: custom

Trigger scaling

Now that you have pushed an app that emits a custom metric and configured autoscaling rules, you can trigger a scaling action. App Autoscaler scales the app when the custom metric goes above or below the threshold specified in the scaling rule.

To trigger scaling:

1. Navigate to the web UI of the app. Use the same URL from Push the Sample App.
2. Click **Increment Custom gauge** enough times to bring the custom metric over the threshold of `5` that you set in the scaling rule. You can check the value of the `custom` metric using the **See Metrics** button.
3. Monitor the app page in Apps Manager for about two minutes. App Autoscaler will begin to scale the app. It adds one instance at a time until it reaches the **Maximum** instance limit of `5`.

Next steps

Now that you have completed this tutorial, try emitting custom metrics and creating scaling rules with your own app. Review the resources listed in the Overview section to learn more. Once you have instrumented your app to emit custom metrics, you can follow the steps outlined in this tutorial to scale based on those metrics.

Using the App Autoscaler CLI for your TAS for VMs deployment

You can use the App Autoscaler Command-Line Interface (CLI) to control scaling apps in your TAS for VMs deployment. This article explains how.

The App Autoscaler automatically scales VMware Tanzu Application Service for VMs (TAS for VMs) apps in response to demand. The App Autoscaler CLI lets you control the App Autoscaler from your local command line by extending the Cloud Foundry command-line interface (cf CLI).

Install the App Autoscaler CLI Plugin

Before you can run App Autoscaler CLI commands on your local machine, you must install the App Autoscaler CLI plugin.

To install the plugin, do the following:

1. Download the plugin from VMware Tanzu Network.
2. To install the App Autoscaler CLI plugin, run the following command:

   ```
   cf install-plugin LOCATION-OF-PLUGIN
   ```

   Where `LOCATION-OF-PLUGIN` is the path to the binary file you downloaded from VMware Tanzu Network. For example:

   ```
   $ cf install-plugin ~/Downloads/autoscaler-for-pcf-cliplugin-macosx64-b
   inary-2.0.91
   ```

   For more information about installing cf CLI plugins, see [Installing a Plugin](../../cf-cli/use-cli-plugins.html#plugin-install).

Create and bind the autoscaling service

Before you can use the App Autoscaler, you must create an Autoscaling service and bind that service to your app. For more information, see Managing Service Instances with the cf CLI.

View apps

Run `cf autoscaling-apps` to view the apps that are bound to an autoscaler service instance in a space, their instance limits, and whether or not they are enabled.

```
$ cf autoscaling-apps

Presenting autoscaler apps in org my-org / space my-space as user
Name                  Guid                                      Enabled   Min
Instances   Max Instances   Scale Up Factor   Scale Down Factor
test-app              d0077dc5-34bd-42c7-b377-e0fcb14d67f3      true      1
```

```
4                1                  1
test-app-2            3f8c6e84-0b6e-4ec9-9335-0aad3979d672      false     10
40               5                  2
OK
```

Update Instance limits

Run `cf update-autoscaling-limits APP-NAME MIN-INSTANCE-LIMIT MAX-INSTANCE-LIMIT` to update the upper and lower app instance limits. The App Autoscaler does not attempt to scale beyond these limits. Replace `APP-NAME` with the name of your app. Replace `MIN-INSTANCE-LIMIT` with the minimum number of apps, and `MAX-INSTANCE-LIMIT` with the maximum number of apps.

```
$ cf update-autoscaling-limits test-app 10 40

Updated autoscaling instance limits for app test-app for org my-org / space m
y-space as user
OK
```

Update scaling factors

Run `cf update-autoscaling-factors APP-NAME SCALE-UP-FACTOR SCALE-DOWN-FACTOR` to update the app scaling factors. The App Autoscaler uses these factors to decide how many instances of your app to scale up or down. Replace `APP-NAME` with the name of your app. Replace `SCALE-UP-FACTOR` with the number of instances to scale up at a time, and `SCALE-DOWN-FACTOR` with the number of instances to scale down at a time.

```
$ cf update-autoscaling-factors test-app 3 2

Updated autoscaling factors for app hello-app in org my-org / space my-space
as user
OK
```

Enable autoscaling

Run `cf enable-autoscaling APP-NAME` to enable autoscaling on your app. Replace `APP-NAME` with the name of your app.

```
$ cf enable-autoscaling test-app-2

Enabled autoscaling for app test-app-2 for org my-org / space my-space as use
r
OK
```

> **Note:** By default, instance limits are set to `Min Instances:-1` and `Max Instances:-1`. To enable autoscaling, you must first update instance limits. See Update Instance Limits above.

Disable autoscaling

Run `cf disable-autoscaling APP-NAME` to disable autoscaling on your app. Replace `APP-NAME` with the name of your app.

```
$ cf disable-autoscaling test-app

Disabled autoscaling for app test-app for org my-org / space my-space as user
OK
```

View rules

Run `cf autoscaling-rules APP-NAME` to view the rules that the App Autoscaler uses to determine when to scale your app. Replace `APP-NAME` with the name of your app.

```
$ cf autoscaling-rules test-app

Presenting autoscaler rules for app test-app for org my-org / space my-space
as user
Rule Guid                               Rule Type         Rule Sub Type   Min
Threshold   Max Threshold
45870b7f-f5c9-403f-9150-e79314f62f06    cpu                               10
20
10a581c5-8fb4-48a2-b4de-8bc834aac146    http_throughput                   20
30
OK
```

Create a rule

Run `cf create-autoscaling-rule APP-NAME RULE-TYPE MIN-THRESHOLD MAX-THRESHOLD [--subtype SUBTYPE] [--metric METRIC] [--comparison-metric COMPARISON-METRIC]` to create a new autoscaling rule.

Replace the placeholders as follows:

- `APP-NAME` is the name of your app.
- `RULE-TYPE` is the type of your scaling rule.
- `MIN-THRESHOLD` is the minimum threshold for the metric.
- `MAX-THRESHOLD` is the maximum threshold for the metric.

You can use the following command options:

- `--metric`, `-m` is the metric for a Custom rule.
- `--comparison-metric`, `-c` is the comparison metric for a Compare rule.
- `--subtype`, `-s` is the rule subtype.

For example:

```
$ cf create-autoscaling-rule test-app http_latency 500 1000 -s avg_99th

Created autoscaler rule for app test-app for org my-org / space my-space as u
ser
Rule Guid                               Rule Type         Rule Sub Type   Min
Threshold   Max Threshold
a4a1292f-6f1d-486b-96f7-fc5b4bb9b27d    http_latency      avg_99th        500
1000
```

```
$ cf create-autoscaling-rule test-app custom 9380 9381 --metric jvm.classes.l
oaded
Created autoscaler rule for app test-app in org my-org / space my-space as us
er
OK
Guid                                    Type     Metric               Sub Typ
e   Min Threshold   Max Threshold
59365c9d-6fe7-4195-b8c1-0f5b07afd636    custom   jvm.classes.loaded
9380.00         9381.00
```

```
$ cf create-autoscaling-rule test-app compare .79 .8  --metric jvm.memory.use
d --comparison-metric jvm.memory.max
Created autoscaler rule for app test-app in org my-org / space my-space as us
er
OK
Guid                                    Type      Metric
Sub Type   Min Threshold   Max Threshold
93c4b831-0155-4771-8842-3247816e71df    compare   jvm.memory.used / jvm.memor
y.max              0.79            0.80
```

Valid rule types and subtypes

For a list of valid types and subtypes, see the following:

- type `custom`
    - metric `METRIC-NAME`
- type `CPU`
- type `memory`
- type `http_throughput`

    Note: VMware does not recommend using `http_throughput` as a scaling rule when logging volume is high in the system. For more information, see HTTP throughput based Autoscaling rules do not fire in the Knowledge Base.

- type `http_latency`
    - sub_type `avg_99th` or `avg_95th`

Note: `http_latency` requires a rule `subtype`.

Note: `http_latency` threshold units are in ms.
In general, the value for `MAX-THRESHOLD` should be at least twice the value for `MIN-THRESHOLD` to avoid excessive cycling. Latency is calculated from the Gorouter to the app and back to the Gorouter. Latency is not calculated between the user and the app.

- type `rabbitmq`
    - sub_type `YOUR-QUEUE-NAME`
- type `compare`
    - metric `METRIC-NAME`
    - comparison_metric `METRIC-NAME`

Delete a rule

Run `cf delete-autoscaling-rule APP-NAME RULE-GUID [--force]` to delete a single autoscaling rule. Replace `APP-NAME` with the name of your app, and replace `RULE-GUID` with the GUID.

```
$ cf delete-autoscaling-rule test-app 10a581c5-8fb4-48a2-b4de-8bc834aac146

Really delete rule 10a581c5-8fb4-48a2-b4de-8bc834aac146 for app test-app?> [y
N]:y
Deleted rule 10a581c5-8fb4-48a2-b4de-8bc834aac146 for autoscaler app test-app
for org my-org / space my-space as user
OK
```

Delete all rules

Run `cf delete-autoscaling-rules APP-NAME [--force]` to delete all autoscaling rules. Replace `APP-NAME` with the name of your app.

```
$ cf delete-autoscaling-rules test-app

Really delete ALL rules for app test-app?> [yN]:y
Deleted rules for autoscaler app test-app for org my-org / space my-space as
user
OK
```

View autoscaling events

Run `cf autoscaling-events APP-NAME` to view recent events related to autoscaling for your app. Replace `APP-NAME` with the name of your app.

```
$ cf autoscaling-events test-app
```

```
Time                  Description
2032-01-01T00:00:00Z  Scaled up from 3 to 4 instances. Current cpu of 20 is a
bove upper threshold of 10.
```

View App Autoscaler scheduled instance limit changes

Run `cf autoscaling-slcs APP-NAME` to view all scheduled instance limit changes. Replace `APP-NAME` with the name of your app.

For example:

```
$ cf autoscaling-slcs test-app

Guid                                      First Execution        Min Instance
s   Max Instances   Recurrence
f08f9803-6e5d-4519-abc3-fea640300d01      2018-06-12T22:00:00Z   0
1               Mo,Tu,We,Th,Fr
```

Create App Autoscaler scheduled instance limit change

Run `cf create-autoscaling-slc APP-NAME DATE-TIME MIN-INSTANCES MAX-INSTANCES [--recurrence RECURRENCE]` to create a new scheduled instance limit change.

Replace the placeholders as follows:

- `APP-NAME` is the name of your app.
- `DATE-TIME` is the date and time of the change.
- `MIN-INSTANCES` and `MAX-INSTANCES` are the minimum and maximum values of the range within which App Autoscaler can change the instance count for an app.
- `RECURRENCE` (optional) is the day of the week for which you want to repeat the change.

For example:

```
$ cf create-autoscaling-slc test-app 2018-06-14T15:00:00Z 1 2 --recurrence Sa

Created scheduled autoscaler instance limit change for app test-app in org my
-org / space my-space as user
OK
Guid                                      First Execution        Min Instance
s   Max Instances   Recurrence
7a19a8a2-e435-4c67-b038-cc4add8be686      2018-06-14T15:00:00Z   1
2               Sa
```

Note: App Autoscaler only supports times in UTC. App Autoscaler does not support setting alternate timezones or Daylight Saving Time

Delete App Autoscaler scheduled instance limit change

Run `cf delete-autoscaling-slc APP-NAME SLC-GUID [--force]` to delete a scheduled instance limit change. Replace `APP-NAME` with the name of your app and `SLC-GUID` with the GUID of your scheduled instance limit change.

For example:

```
$ cf delete-autoscaling-slc test-app d0077dc5-34bd-42c7-b377-e0fcb14d67f3

Really delete scheduled limit change d0077dc5-34bd-42c7-b377-e0fcb14d67f3 for
app test-app?> [yN]:y
Deleted scheduled limit change d0077dc5-34bd-42c7-b377-e0fcb14d67f3 for app t
est-app in org my-org / space my-space as user
OK
```

Configure autoscaling with a manifest

Run `cf configure-autoscaling APP-NAME MANIFEST-FILE-PATH` to use a service manifest to configure your rules, add instance limits, and set scheduled limit changes at the same time. Replace `APP-NAME` with the name of your app, and replace `MANIFEST-FILE-PATH` with the path and name of your App Autoscaler manifest.

An example manifest:

```
---
instance_limits:
  min: 1
  max: 2
scaling_factors:
  up: 3
  down: 2
rules:
- rule_type: "http_latency"
  rule_sub_type: "avg_99th"
  threshold:
    min: 500
    max: 1000
scheduled_limit_changes:
- recurrence: 10
  executes_at: "2032-01-01T00:00:00Z"
  instance_limits:
    min: 10
    max: 20
```

```
$ cf configure-autoscaling test-app autoscaler-manifest.yml

Setting autoscaler settings for app test-app for org my-org / space my-space
as user
OK
```

A `rules` block must be present in your App Autoscaler manifest. If your app does not require any rules changes, include an empty block:

```
---
instance_limits:
  min: 1
  max: 1
rules: []
scheduled_limit_changes:
- recurrence: 42
  executes_at: "2032-01-01T00:00:00Z"
  instance_limits:
    min: 0
    max: 0
```

A `scheduled_limit_changes` block must be present in your App Autoscaler manifest. If your app does not require any scheduled instance limit changes, include an empty block:

```
---
instance_limits:
  min: 1
  max: 2
rules:
- rule_type: "http_latency"
  rule_sub_type: "avg_99th"
  threshold:
     min: 500
     max: 1000
scheduled_limit_changes: []
```

Create a scheduled limit change

To create a scheduled limit change using the App Autoscaler manifest, you must understand how scheduled limit changes are constructed. App Autoscaler uses the `executes at` value in two ways:

1. App Autoscaler uses date and time to set the first (or only) occurrence of a scheduled limit change.
2. App Autoscaler then uses the time component to set the time of day in UTC for each recurrence. App Autoscaler uses this time of day for all subsequent scheduled limit changes.

When setting a recurrence schedule, the days of the week are bitmasked.

To instruct App Autoscaler when to execute a scheduled limit change, add together the bitmasked values for each of the day(s) of the week that you wish to trigger the scheduled limit change using this table.

Day	Su	Mo	Tu	We	Th	Fr	Sa
Value	64	32	16	8	4	2	1

Then set the `recurrence` field to that value.

Examples of recurrence specification

The following are examples of how to calculate a value for the `recurrence` field of a scheduled limit change:

- To schedule on weekdays, you add **(32+16+8+4+2)=62**. Set `recurrence: 62`.
- To schedule on weekends, you add **(64+1)=54**. Set `recurrence: 65`.
- To schedule every day, you add **(64+32+16+8+4+2+1)=127**. Set `recurrence: 127`.
- To schedule on Monday, Wednesday, and Friday, you add **(32+8+2)=42**. Set `recurrence: 42`.

The following is an example manifest to scale down every Friday at 8pm and back up every Monday at 4am assuming a UTC timezone:

```
---
instance_limits:
  min: 6
  max: 12
scaling_factors:
  up: 3
  down: 2
rules:
- rule_type: "http_latency"
  rule_sub_type: "avg_99th"
  threshold:
    min: 500
    max: 1000
scheduled_limit_changes:
- recurrence: 32
  executes_at: "2032-01-01T20:00:00Z"
  instance_limits:
    min: 1
    max: 3
- recurrence: 2
  executes_at: "2032-01-01T04:00:00Z"
  instance_limits:
    min: 6
    max: 12
```

App Autoscaler CLI known issues

The App Autoscaler CLI has the following known issues:

- The CLI returns an error message if you have more than one instance of the App Autoscaler service running in the same space.
  - To prevent this error, delete all but one App Autoscaler service instance from any space that the App Autoscaler service runs in.
- The CLI may output odd characters in Windows shells that do not support text color.
  - To prevent this error, run `SET CF_COLOR=false` in your Windows shell pane before you run App Autoscaler CLI commands.
  - Note that some Windows shells do not support the `CF_COLOR` setting.

Using the App Autoscaler API for your TAS for VMs deployment

You can use the App Autoscaler API as an interface to App Autoscaler to configure and control scaling apps in your TAS for VMs deployment.

You can also use the App Autoscaler Cloud Foundry command-line interface (cf CLI) plug-in, which offers the same features as the App Autoscaler API. See Using the App Autoscaler CLI.

API endpoints

The API uses the autoscale-api app that runs next to the autoscale app in the autoscaling space of the system org. This is the base URL of all the requests covered in the following sections.

If your system domain is `system-domain.com`, in most cases you can reach the autoscale app at `autoscale.sys.system-domain.com` and the API at `autoscale.sys.system-domain.com/api/v2`. Confirm that you can reach both the app and the API before you proceed.

Authentication

You must pass an access token to each API endpoint. The access token is expected as-is on the `Authorization` header of each request.

When you are logged in to your foundation with the cf CLI, get an access token by running `cf oauth-token`.

Note: The App Autoscaler API often returns a 404 error when a token is incorrect. If you a receiving a lot of 404 errors, check your access token.

Content type header

To make any PUT or POST requests, you must set the `Content-Type` header to `application/json`. If you do not set this, you might see the following error:

```
"Your token is invalid or the autoscaled application does not exist"
```

This error can appear regardless of the token you use.

Pagination

Most list calls use pagination. The App Autoscaler API uses a standard form of pagination that is very similar to how the Cloud Foundry API (CAPI) uses pagination.

The following is an example of a paginated response from the App Autoscaler API:

```
 {
  "pagination": {
    "total_results": 1,
    "total_pages": 1,
      "last": {
        "href": "https://autoscale.sys.example.com/api/v2/apps?space_guid=676e592a-a4e
8-43d9-8128-f583c0b45db8&page=1"
      },
      "next": null,
      "first": {
        "href": "https://autoscale.sys.example.com/api/v2/apps?space_guid=676e592a-a4e
8-43d9-8128-f583c0b45db8&page=1"
      },
      "previous": null
    },
    "resources": [
      ...list of objects...
    ]
}
```

Information

```
GET /api/v2/info
```

Parameters

None

Returns

A string denoting the App Autoscaler API version.

App bindings

This section lists operations for app bindings.

Apps that are bound to App Autoscaler are represented by the App Autoscaler API as apps.

App binding object

Name	Type	Description
created_at	string	The timestamp for when this object was created
enabled	Boolean	Enables or disables scaling by App Autoscaler for this app
guid	string	The app GUID from CAPI
instance_limits.max	integer	The maximum instance count that this app is allowed
instance_limits.min	integer	The minimum instance count that this app is allowed
scaling_factors.up	integer	The number of instances to increment by when scaling up
scaling_factors.down	integer	The number of instances to decrement by when scaling down
updated_at	string	The timestamp for when this object was last updated

Get All app bindings in a space

```
GET /api/v2/apps
```

Parameters

Parameter	In	Type	Required	Description
space_guid	query	string	true	The GUID of the space from which you want to get all app bindings
page	query	integer	false	The page of events to get

Returns

A paginated list of app bindings.

Get an app binding

```
GET /api/v2/apps/:app_guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to get the binding for

Returns

An app binding.

Update an app binding

```
PUT /api/v2/apps/:app_guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to update the binding for
enabled	body	Boolean	false	Enables or disables scaling by App Autoscaler for this app
instance_limits.max	body	integer	false	The maximum instance count that this app is allowed
instance_limits.min	body	integer	false	The minimum instance count that this app is allowed
scaling_factors.up	integer	The number of instances to increment by when scaling up		
scaling_factors.down	integer	The number of instances to decrement by when scaling down		

Returns

An app binding.

Events

This section lists operations to retrieve events.

Scaling events include all scaling decisions made by App Autoscaler and other relevant changes. For example, an app is enabled or disabled.

Event object

Name	Type	Description
created_at	string	The timestamp for when this object was created
description	string	A description of the event
scaling_factor	integer	The change in number of instances
updated_at	string	The timestamp for when this object was last updated

Get all events for an app

```
GET /api/v2/apps/:app_guid/events
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to retrieve events for
page	query	integer	false	The page of events to get

Returns

A paginated list of events.

Rules

This section lists operations for rules.

Rules are tied to individual apps, and no one app can have multiple rules of the same category.

Rule object

Name	Type	Description
created_at	string	The timestamp for when this object was created
comparison_metric	string	The divisor to compare metrics against
guid	string	The GUID for this particular object
metric	string	The metric on which scaling decisions are made
queue_name	string	The name of the queue to monitor for RabbitMQ rules
rule_type	string	The type of rule
rule_sub_type	string	The subtype of rule
threshold.max	float	The threshold beyond which the app is scaled up
threshold.min	float	The threshold below which the app is scaled down

Name	Type	Description
updated_at	string	The timestamp for when this object was last updated

Get all rules for an app

```
GET /api/v2/apps/:app_guid/rules
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to retrieve rules for
page	query	integer	false	The page of events to get

Returns

A paginated list of rules.

Create a rule

```
POST /api/v2/apps/:app_guid/rules
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to create a rule for
comparison_metric	body	string	false	The divisor to compare metrics against
metric	body	string	false	The metric on which scaling decisions are made
queue_name	body	string	false	The name of the queue to monitor for RabbitMQ rules
rule_type	body	string	true	The type of rule
rule_sub_type	body	string	false	The subtype of rule
threshold.max	body	float	true	The threshold beyond which the app is scaled up
threshold.min	body	float	true	The threshold below which the app is scaled down

Returns

A rule.

Replace all rules

```
PUT /api/v2/apps/:app_guid/rules
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to set rules for

Parameter	In	Type	Required	Description
array of rules	body	array	true	An array of rule objects. See Create a Rule.

Returns

A paginated list of rules.

Delete all rules

```
DELETE /api/v2/apps/:app_guid/rules
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to delete rules for

Returns

A 204 status code and empty body indicating that all rules were deleted.

Get a rule

```
GET /api/v2/apps/:app_guid/rules/:guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to get a rule for
guid	path	string	true	The GUID of the rule to get

Returns

A rule.

Delete a rule

```
DELETE /api/v2/apps/:app_guid/rules/:rule_guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to delete a rule for
rule_guid	path	string	true	The GUID of the rule to delete

Returns

A 204 status code and empty body indicating that the rule was deleted.

Scheduled limit changes

This section lists operations for scheduled limit changes (SLCs).

SLCs are jobs scheduled for the future that alter app status, such as enabling, disabling, or instance limits.

Scheduled limit change object

Name	Type	Description
created_at	string	The timestamp for when this object was created
enabled	Boolean	Whether or not the app is enabled or disabled
executes_at	string	The UTC timestamp indicating when this SLC is first executed
guid	string	The GUID of this object
instance_limits.max	integer	The maximum instance count that is set on the app
instance_limits.min	integer	The minimum instance count that is set on the app
recurrence	integer	A number that represents the days of the week that the change executes. Each bit in the number represents a day in the following order: sun, mon, tue, wed, thur, fri, sat. For example, repeat mon, wed, fri = 0101010 = 42.
updated_at	string	The timestamp for when this object was last updated

Get All scheduled limit changes for an app

```
GET /api/v2/apps/:app_guid/scheduled_limit_changes
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to retrieve SLCs for
page	query	integer	false	The page of SLCs to get

Returns

A paginated list of SLCs.

Create a scheduled limit change

```
POST /api/v2/apps/:app_guid/scheduled_limit_changes
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to create an SLC for

Parameter	In	Type	Required	Description
enabled	body	Boolean	true	Whether or not the app is enabled or disabled
executes_at	body	string	true	The UTC timestamp indicating when this SLC is first executed
instance_limits.max	body	integer	true	The maximum instance count that is set on the app
instance_limits.min	body	integer	true	The minimum instance count that is set on the app
recurrence	body	string	true	A number that represents the days of the week that the change executes. Each bit in the number represents a day in the following order: sun, mon, tue, wed, thur, fri, sat. For example, repeat mon, wed, fri = 0101010 = 42.

Returns

An SLC.

Replace all scheduled limit changes for an app

```
PUT /api/v2/apps/:app_guid/scheduled_limit_changes
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to replace SLCs for
array of SLCs	body	array	true	An array of SLC objects. See Create a Scheduled Limit Change.

Returns

A paginated list of SLCs.

Delete all scheduled limit changes for an app

```
DELETE /api/v2/apps/:app_guid/scheduled_limit_changes
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to replace SLCs for

Returns

A 204 status code and empty body indicating that the SLCs were deleted.

Get a scheduled limit change

```
GET /api/v2/apps/:app_guid/scheduled_limit_changes/:guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to get SLCs for
guid	path	string	true	The GUID of the SLC to retrieve

Returns

An SLC object.

Delete a scheduled limit change

```
DELETE /api/v2/apps/:app_guid/scheduled_limit_changes/:guid
```

Parameters

Parameter	In	Type	Required	Description
app_guid	path	string	true	The GUID of the app to delete SLCs for
guid	path	string	true	The GUID of the SLC to delete

Returns

A 204 status code and empty body indicating that the SLC was deleted.

Using the cf CLI

In this section:

- Installing the cf CLI
- Upgrading to cf CLI v7
- Upgrading to cf CLI v8
- Getting Started with the cf CLI
- Using the cf CLI with a Proxy Server
- Using the cf CLI with a Self-Signed Certificate
- Using cf CLI Plugins
- Developing cf CLI Plugins

Installing the cf CLI

The cf CLI is the official command line client for Cloud Foundry.

The procedures here describe how to install the cf CLI on your operating system. You can install the cf CLI with a package manager, an installer, or a compressed binary.

For more information about how to use the cf CLI, see Getting Started with cf CLI.

To learn when cf CLI updates are released and to download a new binary or installer, see Releases in the Cloud Foundry CLI repository on GitHub.

There are currently two major versions of the cf CLI: v7 and v8. See the README to decide which version to use.

Prerequisites

If you previously used the cf CLI Ruby gem, you must uninstall the gem before installing the cf CLI.

To uninstall the gem:

1. Run:

   ```
   gem uninstall cf
   ```

2. Verify that your Ruby environment manager uninstalled the gem by closing and reopening your terminal.

Install the cf CLI Using a package manager

These sections describe how to install the cf CLI using a package manager. You can install the cf CLI using a package manager on Mac OS X and Linux operating systems.

Linux installation

There are two ways to install the cf CLI using a package manager, depending on your Linux distribution.

To install the cf CLI on Debian and Ubuntu-based Linux distributions:

1. Add the Cloud Foundry Foundation public key and package repository to your system by running:

   ```
   wget -q -O - https://packages.cloudfoundry.org/debian/cli.cloudfoundry.org.key
   | sudo apt-key add -
   echo "deb https://packages.cloudfoundry.org/debian stable main" | sudo tee /et
   c/apt/sources.list.d/cloudfoundry-cli.list
   ```

2. Update your local package index by running:

   ```
   sudo apt-get update
   ```

3. To install **cf CLI v7**, run:

   ```
   sudo apt-get install cf7-cli
   ```

4. To install **cf CLI v8**, run:

   ```
   sudo apt-get install cf8-cli
   ```

To install the cf CLI on Enterprise Linux and Fedora RHEL6/CentOS6 and later distributions:

1. Configure the Cloud Foundry Foundation package repository by running:

   ```
   sudo wget -O /etc/yum.repos.d/cloudfoundry-cli.repo https://packages.cloudfound
   ```

```
ry.org/fedora/cloudfoundry-cli.repo
```

2. To install **cf CLI v7**, run:

   ```
   sudo yum install cf7-cli
   ```

3. To install **cf CLI v8**, run:

   ```
   sudo yum install cf8-cli
   ```

   This also downloads and adds the public key to your system.

Mac OS X installation

You can install the cf CLI on Mac OS X operating systems using the Homebrew package manager.

To install the cf CLI for Mac OS X using Homebrew:

1. Install Homebrew. For instructions, see Install Homebrew on the Homebrew website.
2. To install **cf CLI v7**, run:

   ```
   brew install cloudfoundry/tap/cf-cli@7
   ```

3. To install **cf CLI v8**, run:

   ```
   brew install cloudfoundry/tap/cf-cli@8
   ```

Install the cf CLI Using an Installer

This section describes how to install the cf CLI using an installer. You can install the cf CLI using an installer on Windows, Mac OS X, and Linux operating systems.

Linux Installation

This section describes how to install the cf CLI using an installer on the Debian, Ubuntu, and Red Hat Linux distributions.

To install the cf CLI for Linux using an installer, do the following:

1. Download the Linux installer.
   - **Debian or Ubuntu**: Paste the following URL in a browser to download the installer:

     ```
     https://cli.run.pivotal.io/stable?release=debian64&source=github
     ```

   - **Red Hat**: Paste the following URL in a browser to download the installer:

     ```
     https://cli.run.pivotal.io/stable?release=redhat64&source=github
     ```

2. Install the cf CLI using your system’s package manager.

   Note: The following commands may require `sudo`.

   - **Debian or Ubuntu**: Run the following command:

     ```
     dpkg -i path/to/cf-cli-*.deb && apt-get install -f
     ```

   - **Red Hat**: Run the following command:

     ```
     rpm -i path/to/cf-cli-*.rpm
     ```

Mac OS X Installation

This section describes how to install the cf CLI on Mac OS X operating systems using an installer.

To install the cf CLI for Mac OS X using an installer, do the following:

1. Paste the following URL in a browser to download the installer:

   ```
   https://cli.run.pivotal.io/stable?release=macosx64&source=github
   ```

2. Open the `.pkg` file.
3. In the installer wizard, click **Continue**.
4. Select an install destination and click **Continue**.
5. Click **Install**.

Windows Installation

This section describes how to install the cf CLI for use on the Windows command line. For instructions on how to install the cf CLI for use on the Windows Subsystem for Linux (WSL), also known as Bash, see Linux Installation above.

To install the cf CLI for Windows using an installer, do the following:

1. Paste the following URL in a browser to download the installer:

   ```
   https://cli.run.pivotal.io/stable?release=windows64&source=github
   ```

2. Unzip the file.
3. Right-click on the `cf_installer` executable and select **Run as Administrator**.
4. When prompted, click **Install**. Then, click **Finish**.

Install the cf CLI using a compressed binary

You can install the cf CLI using a compressed binary on Windows, Mac OS X, and Linux operating systems.

For more information about downloading and installing a compressed binary for **cf CLI v7**, see Installers and compressed binaries.

For more information about downloading and installing a compressed binary for **cf CLI v8**, see Installers and compressed binaries.

Verify installation

To verify the installation of the cf CLI:

1. Close and reopen the command prompt. Or, open a new tab in the command prompt.
2. Run:

   ```
   cf --help
   ```

If your installation was successful, the cf CLI help listing appears.

Uninstall the cf CLI

These sections describe how to uninstall the cf CLI. The method for uninstalling the cf CLI differs depending on the installation method.

Package manager

If you installed the cf CLI with a package manager, follow the instructions specific to your package manager.

Installer

If you installed the cf CLI with an installer, follow the procedure in this section that is specific to your operating system.

To uninstall the cf CLI on Mac OS X:

1. Delete the binary `/usr/local/bin/cf`.
2. Delete the directory `/usr/local/share/doc/cf-cli`.

To uninstall the cf CLI on Windows:

1. Navigate to the **Control Panel** and click **Programs and Features**.
2. Select **Cloud Foundry CLI VERSION**.
3. Click **Uninstall**.

Binary

To uninstall the cf CLI after installing it with a binary:

1. Navigate to the location where you copied the binary.
2. Delete the binary.

Upgrading to cf CLI v7

The main goal of Cloud Foundry CLI (cf CLI) v7 and CAPI V3 is to unlock new app developer workflows for users who require granular control of their apps and other advanced deployment strategies. For more information, see New Workflows Supported below. These workflows were previously limited by CAPI V2.

The cf CLI development team aims to provide:

- A seamless upgrade experience from cf CLI v6. Changes have been kept to a minimum. Where there are changes, the team has incorporated feedback from the community to simplify the cf CLI user experience.
- Details about breaking potential breaking changes and alternative workflows for scripting environments.

To understand the differences between specific commands, see Command Differences below.

For more information about CAPI V3, see the CAPI V3 documentation. For more information about CAPI V2, see the CAPI V2 documentation.

New workflows supported by cf CLI v7

cf CLI v7 uses CAPI V3, which offers developers more granular control over their apps. It enables new workflows by exposing packages, droplets, builds, and processes. CAPI V3 also includes new resources such as sidecars, manifests, deployments.

Some key new features available through the cf CLI v7 are:

- Rolling App Deployments: Push updates to apps without incurring downtime.
- Running cf push Sub-Step Commands: Exercise granular control over the cf push process. These commands break down the cf push process into sub-steps that can run independently.
- Pushing an App with Multiple Processes: Use a single command to push apps that run multiple processes. One example is a web app that has a UI process and a worker process.
- Pushing Apps with Sidecar Processes: Run additional processes in the same container as your app.
- Using Metadata: Add metadata to objects such as spaces and apps. This can help with operating, monitoring, and auditing.

Install cf CLI v7

To install cf CLI v7, see the README in the Cloud Foundry CLI repository on GitHub. It includes instructions for downloading the latest CAPI release candidate, which is what the cf CLI v7 beta is tested against.

Prerequisites

The cf CLI v7 requires cf-deployment v13.5.0 or later.

This version of cf-deployment contains CAPI release v1.95.0, which provides the CAPI V3 API v3.85.0.

For more information, see the CAPI release notes and the Cloud Controller release notes on GitHub.

If you target an older version of cf-deployment, cf CLI v7 presents a warning saying that the API version is less than the minimum supported. Not all commands run correctly. For example, `cf apps` does not work.

Command differences

These sections describe changes in commands from cf CLI v6 to cf CLI v7. They also provide important information for those who use the cf CLI in scripts.

For information about possible breaking changes, see the Table of Differences below. This table includes removed flag options, removed commands, and removed or changed argument requirements.

About scripting

If you have scripts that rely on the cf CLI, this section describes possible changes in cf CLI v7 which might affect scripts.

Some of these changes are:

- In cf CLI v7, if your scripts parse error text, output text errors are returned directly from CAPI. Where possible, cf CLI v7 no longer wraps errors it receives from the API.
- cf CLI v7 commands output errors and warnings to `stderr` rather than `stdout` to simplify debugging.
- Style changes including flavor text updates. For more information, see Colors in *CF CLI Style Guide* in the Cloud Foundry CLI repository on GitHub.
- Key-value and table column headers are displayed in lowercase.
- Single-quote resource names appear in error cases.

> ⚠ **Caution:** If the services attribute is declared at the top-level of the manifest.yml file, cf CLI v6 generates a warning, but in cf CLI v7, there is no warning and the app is still pushed.

Exit codes

cf CLI v7 attempts to consistently apply the principles of idempotency across all commands which require it. For more information, see General Principles in *CF CLI Style Guide* in the Cloud Foundry CLI repository on GitHub. Commands now exit `0` if the outcome a user expresses when running a specific command is unchanged after the command is executed. Examples include:

- Attempting to delete a resource which does not exist, such as a space. Commands such as `delete-route` and `delete-space` return `0` in those cases.
- If the `create-buildpack` command fails to create a buildpack, the command exits with `1` instead of `0`, which is the current cf CLI v6 behavior.

Table of differences

The table below summarizes how commands differ between cf CLI v7 and cf CLI v6.

Command	Changes

`cf add-network-policy`	- **[Removed flag]:** The flag `--destination-app` is deprecated. Instead, the destination app is the required second argument, using no flag.
`cf apps`	- **[Update]:** Displays information about `processes`. - **[Update]:** The `url` field is renamed to `routes`. - **[Update]:** Information about `instances`, `memory`, and `disk` is removed. - **[Update]:** Apps are listed alphabetically.
`cf bind-security-group`	- **[Update]:** `SPACE` is no longer an argument. To provide a space, use the `--space` flag.
`cf check-route`	- **[Update]:** `HOST` is no longer a required argument. - **[Update]:** No longer requires a backslash. - **[Added flag]:** Use `--hostname` to specify a hostname.
`cf create-buildpack`	- **[Removed flag]:** `--enable`. Creating a buildpack activates it by default. - **[Removed flag]:** `--disable`. You can not deactivate a buildpack upon creation. - **[Update]:** Creating a buildpack with position set to `0` is no longer valid.
`cf create-domain`	- **[Renamed]:** This command is renamed to `create-private-domain`.
`cf create-org`	- **[Update]:** `clients.read` scope (or `clients.admin`, or `zones.uaa.admin`) is now required when logged in using client credentials.
`cf create-quota`	- **[Renamed]:** This command is renamed to `create-org-quota`.
`cf create-space`	- **[Update]:** `clients.read` scope (or `clients.admin`, or `zones.uaa.admin`) is now required when logged in using client credentials.
`cf create-route`	- **[Update]:** `SPACE` is no longer a required argument. The command creates a route in the space you are targeting. - **[Removed flag]:** `--random-port`. This is now default behavior for routes with TCP domains if the `--port` flag is not provided.
`cf create-service-auth-token`	*This command is removed.*
`cf create-service-broker`	- **[Update]:** If the command does not successfully complete all phases, a service broker object may exist, which can then be updated or deleted.

`cf create-user`	- **[Added flag]:** `--password-prompt`. This option enhances security by removing the requirement to type your password on the command line.
`cf delete`	- **[Change in flag behavior]:** `-r` no longer deletes routes when the route is mapped to more than one app.
`cf delete-domain`	- **[Renamed]:** This command is renamed to `delete-private-domain`.
`cf delete-org`	- **[Update]:** The command fails if the org contains shared private domains.
`cf delete-quota`	- **[Renamed]:** This command is renamed to `delete-org-quota`.
`cf delete-service-auth-token`	*This command is removed.*
`cf domains`	- **[Update]:** The `status` column is renamed to `availability`. - **[Update]:** The table refers to private domains with `private` instead of `own`.
`cf files`	*This command is removed.*
`cf map-route`	- **[Removed flag]:** `--random-port`. In the cf CLI v7, when you map a TCP route to an app, a random port is assigned to the route by default. To specify a port for a TCP route, use the `--port` flag.
`cf marketplace`	- **[Renamed flag]:** The `-s` flag is renamed to `-e` for consistency with other commands. - **[Update]:** When the `-e` flag is specified, plan costs advertised by the service broker are displayed. - **[Update]:** When the `-e` flag is specified, and no service offering with that name is found, the exit code returned is 0. This is in contrast to the cf CLI v6, which returned exit code 1 in this case.
`cf migrate-service-instances`	*This command is removed.*
`cf packages`	- **[Update]:** Displays packages from most recent to least recent.

`cf push`	• **[Removed flag]:** `--route-path`. You can use the `routes` property in the manifest instead. • **[Removed flag]:** `-d` for domain. You can use the `routes` property in the manifest instead. • **[Removed flag]:** `--no-hostname`. You can use the `routes` property in the manifest instead. • **[Removed flag]:** `--hostname`. You can use the `routes` property in the manifest instead. • **[Added flag]:** `--strategy`. You can deploy an app without causing downtime using `cf push app_name --strategy rolling`. Exits when at least one instance of each process is healthy. • **[Added flag]:** `--no-wait`. When used, the command exits when the one instance one process becomes healthy. • **[Added flag]:** `--endpoint`. Required if you set health check type to `http` when pushing an app. • **[Updated flag]:** `--health-check-type none` is removed in favor of `--health-check-type process`. • **[Updated flag]:** `--no-route` no longer unbinds all existing routes associated with the app. • **[Updated flag]:** `-t` now has a long form `--app-start-timeout`. All short-form flags now have long-form equivalents.
`cf purge-service-offering`	• **[Removed flag]:** The `-p` flag is removed.
`cf quota`	• **[Renamed]:** This command is renamed to `org-quota`.
`cf quotas`	• **[Renamed]:** This command is renamed to `org-quotas`.
`cf remove-network-policy`	• **[Removed flag]:** The flag `--destination-app` is deprecated. Instead, the destination app is the required second argument, using no flag.
`cf rename-buildpack`	*This command is removed. Instead, use `--rename` flag with `cf update-buildpack`.*
`cf restart-app-instance`	• **[Added Flag]:** `--process`
`cf routes`	• **[Updated output]:** `port`, `type`, and `service` no longer appear in the table. • **[Renamed flag]:** `--orglevel` is now `--org-level`.
`cf run-task`	• **[Updated]:** `COMMAND` is no longer an argument. To specify a command, use the `--command` flag. • **[Added Flag]:** `--process`
`cf scale`	• **[Added flag]:** `--process`

`cf service-access`	- **[Update]:** When a service offering comes from a space-scoped service broker, the space and org are displayed.
`cf service-auth-tokens`	*This command is removed.*
`cf set-health-check`	- **[Added flag]:** `--process` - **[Added flag]:** `--invocation-timeout`
`cf set-quota`	- **[Renamed]:** This command is renamed to `set-org-quota`.
`cf set-running-environment-variable-group`	- **[Update]:** System environment variables can only be strings. This is enforced on the API.
`cf set-staging-environment-variable-group`	- **[Update]:** System environment variables can only be strings. This is enforced on the API.
`cf ssh`	- **[Added flag]:** `--process` - **[Added environment variable]:** `all_proxy`. Specifies a proxy server for all requests.
`cf start`	- **[Update]:** Stages an app to support `cf push app --no-start` use cases. If there is a new package, `start` stages and starts using the new package. If the app has been rolled back, `start` starts using the droplet you used to roll back your app. In the case of a droplet that is in a `FAILED` state, `start` ignores the failed droplet and restages the latest `READY` package to try to produce a healthy droplet. In cf CLI v6, `start` fails if the droplet is in a `FAILED` state.
`cf unshare-private-domain`	- **[Update]:** This command now provides a warning and requires confirmation before it proceeds.
`cf update-buildpack`	- **[Added flag]:** `--rename` - **[Change in flag behavior]:** `--unlock` and `--path` are now compatible.
`cf update-quota`	- **[Renamed]:** This command is renamed to `update-org-quota`.
`cf update-service-auth-token`	*This command is removed.*
`cf v3-COMMAND`	- **[Update]:** `v3` prefixes have been removed, since the commands now use CAPI V3 by default.
`cf apply-manifest`	- **[Update]:** If no flags are passed, the command defaults to using the manifest located in your `pwd`.
`cf v3-cancel-zdt-push`	- *This command is removed. Instead, use `cf cancel-deployment`.*

`cf v3-zdt-push`	- *This command is removed. Instead, use* `--strategy rolling` *flag with* `cf push`*.*
`cf buildpacks`	- The order of the columns are changed. The buildpack column header is renamed.

Upgrading to cf CLI v8

You can use Cloud Foundry Command Line Interface (cf CLI) v8 to interact with Cloud Foundry API (CAPI) V3. This article describes the major changes between cf CLI v7 and cf CLI v8.

The cf CLI development team aims to provide:

- A seamless upgrade experience from cf CLI v7. Changes are minimal. Where there are changes, the team has incorporated feedback from the community to simplify the cf CLI user experience.
- Details about breaking potential breaking changes and alternative workflows for scripting environments.

To understand the differences between specific commands, see Command differences below.

For more information about CAPI V3, see the CAPI V3 documentation.

For more information about cf CLI v8, see v8.0.0 in GitHub.

New workflows supported by cf CLI v8

Some key new features available through the cf CLI v8 are:

- **Asynchronous service operations**: All service-related operations are now asynchronous by default. This includes manipulating service keys and route bindings.

Install cf CLI v8

To install cf CLI v8, see the README in the Cloud Foundry CLI repository on GitHub. It includes instructions for downloading the latest CAPI release candidate, which is what the cf CLI v8 beta is tested against.

In cf CLI v8, Golang has been updated from v1.13 to v1.16. If you target a foundation that does not have a SAN, you might encounter errors because the common name field is deprecated in Golang v1.15 and later. For more information, see X.509 CommonName deprecation in the Golang v1.15 release notes.

Prerequisites

The cf CLI v8 requires cf-deployment v16.11.0 or later.

This version of cf-deployment contains CAPI release v1.109.0, which provides the CAPI V3 API v3.99.0.

For more information, see the cf CLI Versioning and Support Policy on GitHub.

Command differences

These sections describe changes in commands from cf CLI v7 to cf CLI v8. They also provide important information for those who use the cf CLI in scripts.

For information about possible breaking changes, see the Table of differences below. This table includes removed flag options, updated output, and removed or changed argument requirements.

Manifest differences

When you apply a manifest by running `cf push`, cf CLI v8 does not provide a manifest diff through the V3 manifest diff endpoint. This new endpoint supports version 1 manifests only. For more information, see Create a manifest diff for a space (experimental) in the CAPI documentation.

About scripting

If you have scripts that rely on the cf CLI, this section describes possible changes in cf CLI v8 that might affect scripts.

Some of these changes are:

- Style changes, including changes in the order or wording of the output.
- cf CLI v8 uses CAPI V3 to make requests related to services. CAPI V3 creates asynchronous jobs. If you want to continue to create jobs synchronously, use the new `--wait` flag.

Table of differences

The following table summarizes how commands differ between cf CLI v7 and cf CLI v8.

Command	Changes
`cf bind-service`	• **[Added flag]:** Use `--wait` to wait for the bind operation to complete.
`cf bind-route-service`	• **[Update]:** Bind route operation is async by default. • **[Added flag]:** Use `--wait` to wait for the bind operation to complete.
`cf create-service`	• **[Added flag]:** Use `--wait` to wait for the create operation to complete.
`cf create-service-key`	• **[Update]:** Create operation is async by default. • **[Added flag]:** Use `--wait` to wait for the create operation to complete.
`cf delete-service`	• **[Added flag]:** Use `--wait` to wait for the delete operation to complete.

`cf delete-service-key`	- **[Update]:** Delete operation is async by default. - **[Added flag]:** Use `--wait` to wait for the delete operation to complete.
`cf map-route`	- **[Added flag]:** Use `--destination-protocol` to use HTTP/2 protocol to communicate with apps. By default if destination-protocol is not set HTTP/1 protocol will be used for HTTP route.
`cf marketplace`	- **[Added flag]:** Use `--show-unavailable` to show plans that are not available for use.
`cf route`	- **[New]:** New command for viewing details about a route and its destinations.
`cf routes`	- **[Update]:** Added `service instance` column to output.
`cf service`	- **[Added flag]:** Use `--params` to retrieve and display the given service instances's parameters as JSON. All other output is suppressed. - **[Update]:** Displays information about `guid`, `type`, and `broker tags`. - **[Update]:** The `service` field is renamed to `offering`. - **[Update]:** The `service broker` field is renamed to `broker`. - **[Update]:** The `dashboard` field is renamed to `dashboard url`. - **[Update]:** Minor changes to the ordering and wording of each block of information.
`cf service-key`	- **[Update]:** Displays information about `last operation` and `message` as new columns.
`cf services`	- **[Added flag]:** Use `--no-apps` to not retrieve bound apps information. - **[Added flag]:** Use `--wait` to wait for the operation to complete.
`cf unbind-service`	- **[Added flag]:** Use `--wait` to wait for the unbind operation to complete.
`cf unbind-route-service`	- **[Update]:** Unbind route operation is async by default. - **[Added flag]:** Use `--wait` to wait for the unbind operation to complete.
`cf update-service`	- **[Added flag]:** Use `--wait` to wait for the update operation to complete. - **[Removed flag]:** `--upgrade`. Use new command `cf upgrade-service` to upgrade a plan.

`cf upgrade-service`	- **[Removed flag]:** `--force`. There is no longer user interaction required on this command.

Getting started with the cf CLI

The cf CLI is the official command line client for Cloud Foundry. You can use the cf CLI to manage apps, service instances, orgs, spaces, and users in your environment.

Prerequisite

To follow the procedures in this topic, you must download and install the latest version of the cf CLI v7 or v8. For more information, see Installing the Cloud Foundry command line interface.

Log in with the CLI

The `cf login` command uses the syntax described below to specify a target API endpoint, login credentials, an org, and a space.

The cf CLI prompts for credentials as needed. If you are a member of multiple orgs or spaces, `cf login` prompts you for which ones to log in to. Otherwise, it targets your org and space automatically.

To log in to the cf CLI:

1. Run:

   ```
   cf login -a API-URL -u USERNAME -p PASSWORD -o ORG -s SPACE
   ```

   Where:

   - `API-URL` is your API endpoint, the URL of the Cloud Controller in your TAS for VMs instance.
   - `USERNAME` is your username.
   - `PASSWORD` is your password. VMware discourages using the `-p` option, because it may record your password in your shell history.
   - `ORG` is the org where you want to deploy your apps.
   - `SPACE` is the space in the org where you want to deploy your apps.

When you successfully log in, you see output like the example below:

```
$ cf login -a https://api.example.com -u username@example.com -o example-org
-s development
API endpoint: https://api.example.com

Password>
Authenticating...
OK

Targeted org example-org
```

```
Targeted space development


API endpoint:   https://api.example.com
User:           username@example.com
Org:            example-org
Space:          development
```

Log in with the API

You can write a script to log in to the cf CLI. This allows you to avoid manually logging in to the cf CLI each time you use it.

To write a script to log in:

1. Target your API by running:

   ```
   cf api API-URL
   ```

   Where `API-URL` is your API endpoint, the URL of the Cloud Controller in your TAS for VMs instance.

   For more information, see api in the *Cloud Foundry CLI Reference Guide*.

2. Authenticate by running:

   ```
   cf auth USERNAME PASSWORD
   ```

   Where:

   - `USERNAME` is your username.
   - `PASSWORD` is your password. VMware discourages using the `-p` option, because it may record your password in your shell history.

   For more information about the `cf auth` command, see the Cloud Foundry CLI Reference Guide.

3. Target your org or space by running:

   ```
   cf target -o ORG -s SPACE
   ```

   Where:

   - `ORG` is the org you want to target.
   - `SPACE` is the space you want to target.

   For more information about the `cf target` command, see the Cloud Foundry CLI Reference Guide.

After you log in, the cf CLI saves a `config.json` file that contains your API endpoint, org, space values, and access token.

If you change these settings, the `config.json` file is updated accordingly.

By default, `config.json` is located in the `~/.cf` directory. You can relocate the `config.json` file using the `CF_HOME` environment variable.

Localize the CLI

The cf CLI translates terminal output into the language that you select. The default language is `en-US`.

The cf CLI supports these languages:

- Chinese (simplified): `zh-Hans`
- Chinese (traditional): `zh-Hant`
- English: `en-US`
- French: `fr-FR`
- German: `de-DE`
- Italian: `it-IT`
- Japanese: `ja-JP`
- Korean: `ko-KR`
- Portuguese (Brazil): `pt-BR`
- Spanish: `es-ES`

For more information, see config in the *Cloud Foundry CLI Reference Guide*.

Note: Localization with `cf config --locale` affects only messages that the cf CLI generates.

To set the language of the cf CLI:

1. Log in to the cf CLI:

   ```
   cf login
   ```

2. Run:

   ```
   cf config --locale LANGUAGE
   ```

   Where `LANGUAGE` is code of the language you want to set.

3. Confirm the language change by running:

   ```
   cf help
   ```

   The command returns output similar to the example below:

   ```
   $ cf help
   NOME:
   ```

```
   cf - Uma ferramenta de linha de comando para interagir com Cloud Fou
ndry

USO:
   cf [opções globais] comando [argumentos...] [opções de comando]

VERSÃO:
   6.14.1
   ...
```

Manage users and roles

The cf CLI includes commands that list users and assign roles in orgs and spaces.

List users

To list all users in an org or a space:

1. Log in to the cf CLI:

   ```
   cf login
   ```

2. Run one of these commands:

   - To list org users, run:

     ```
     cf org-users
     ```

   - To list space users, run:

     ```
     cf space-users
     ```

The command returns output similar to the example below:

```
$ cf org-users example-org
Getting users in org example-org as username@example.com...

ORG MANAGER
  username@example.com

BILLING MANAGER
  huey@example.com
  dewey@example.com

ORG AUDITOR
  louie@example.com
```

For more information, see org-users and space-users in the *Cloud Foundry CLI Reference Guide*.

Manage roles

You use the commands listed below to manage roles in the cf CLI. These commands require admin permissions and take username, org or space, and role as arguments:

- `cf set-org-role`
  For more information, see set-org-role in the *Cloud Foundry CLI Reference Guide*.
- `cf unset-org-role`
  For more information, see unset-org-role in the *Cloud Foundry CLI Reference Guide*.
- `cf set-space-role`
  For more information, see set-space-role in the *Cloud Foundry CLI Reference Guide*.

Manage roles

You use the commands listed below to manage roles in the cf CLI. These commands require admin permissions and take `username`, `org` or `space`, and `role` as arguments:

- `cf set-org-role`
  For more information, see the Cloud Foundry CLI Reference Guide.
- `cf unset-org-role`
  For more information, see the Cloud Foundry CLI Reference Guide.
- `cf set-space-role`
  For more information, see the Cloud Foundry CLI Reference Guide.
- `cf unset-space-role`
  For more information, see the Cloud Foundry CLI Reference Guide.

The available roles are:

- `OrgManager`
- `BillingManager`
- `OrgAuditor`
- `SpaceManager`
- `SpaceDeveloper`
- `SpaceAuditor`

For more information about user roles, see Orgs, Spaces, Roles, and Permissions.

The example below shows the cf CLI output for assigning the Org Manager role to a user within an org:

```
$ cf set-org-role huey@example.com example-org OrgManager

Assigning role OrgManager to user huey@example.com in org example-org as user
name@example.com...
OK
```

Note: If you are not an admin, you see this message when you try to run these commands: `error code: 10003, message: You are not authorized to perform the requested action`

Manage roles for users with identical usernames in multiple origins

If a username corresponds to multiple accounts from different user stores, such as both the internal UAA store and an external SAML or LDAP store, the `cf set-org-role` and the `cf unset-org-role` commands above return this error:

```
The user exists in multiple origins. Specify an origin for the requested user
from: ‘uaa’, ‘other’
```

To resolve this ambiguity, you can construct a `curl` command that uses the API to perform the desired role management function. For an example, see Associate Auditor with the Organization by Username in the Cloud Foundry API documentation.

Push an app

These sections describe how to use the `cf push` command to push a new app or sync changes to an existing app.

For more information, see push in the *Cloud Foundry CLI Reference Guide*.

Push a new app or push changes to an app

To push an app:

1. Log in to the cf CLI by running:

   ```
   cf login
   ```

2. Go to the directory of the app.
3. Push a new app or push changes to an app by running:

   ```
   cf push APP-NAME
   ```

   Where `APP-NAME` is the name of the app.

Push an app using a manifest

You can provide a path to a manifest file when you push an app. The manifest file may include information such as the name of the app, disk limit, and number of instances. You can use a manifest file rather than adding flags to the `cf push` command.

`cf push` locates the `manifest.yml` file in the current working directory by default. Or, you can provide a path to the manifest with the `-f` flag.

For more information about the `-f` flag, see push in the *Cloud Foundry CLI Reference Guide*.

> **Note:** When you provide an app name at the command line, `cf push` uses that app name whether or not there is a different app name in the manifest. If the manifest describes multiple apps, you can push a single app by providing its name at the command line; the cf CLI does not push the others. Use these behaviors for testing.

Push an app with a buildpack

You can specify a buildpack when you push an app with the `-b` flag. If you use the `-b` flag to specify a buildpack, the app remains permanently linked to that buildpack. To use the app with a different buildpack, you must delete the app and then push it again.

For more information about available buildpacks, see Buildpacks.

The example below pushes an app called `awesome-app` to the URL `http://awesome-app.example.com` and specifies the Ruby buildpack with the `-b` flag:

```
$ cf push awesome-app -b ruby_buildpack
Creating app awesome-app in org example-org / space development as username@e
xample.com...
OK

Creating route awesome-app.example.com...
OK
...

1 of 1 instances running

App started
...

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: awesome-app.example.com
last uploaded: Wed Jun 8 23:43:15 UTC 2016
stack: cflinuxfs3
buildpack: ruby_buildpack

     state     since                    cpu    memory     disk      details
#0   running   2016-06-08 04:44:07 PM   0.0%   0 of 1G    0 of 1G
```

Important: To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

Map a route to an app

You can provide a hostname for your app when you push the app. If you do not provide a hostname, the `cf push` command routes your app to a URL of the form `APP-NAME.DOMAIN`, where `APP-NAME` is the name of your app and `DOMAIN` is your default domain.

For information about mapping a route to your app, see Routes and domains.

To map a route to the app:

1. Log in to the cf CLI by running:

```
cf login
```

2. Map a route by running:

```
cf push APP-NAME --hostname APP-HOSTNAME
```

Where:

- `APP-NAME` is the name of the app.
- `APP-HOSTNAME` is the hostname of the app.

Manage user-provided service instances

These sections describe how to create or update a service instance.

Create a service instance

To create a new service instance, use the `cf create-user-provided-service` or `cf cups` commands. For more information, see create-user-provided-service in the *Cloud Foundry CLI Reference Guide*.

To create or update a user-provided service instance, you must supply basic parameters. For example, a database service might require a username, password, host, port, and database name.

You can provide these parameters in these ways:

- Interactively. For more information, see Supply Parameters Interactively.
- Non-interactively. For more information, see Supply Parameters Non-Interactively.
- With third-party log management software as described in RFC 6587. For more information, see Supply Parameters Through a Third Party and RFC 6587.

> **Note:** When used with third-party logging, the cf CLI sends data formatted according to RFC 5424. For more information, see [RFC 5424](http://tools.ietf.org/html/rfc5424).

Supply parameters interactively

To create a new service while supplying parameters interactively:

1. Log in to the cf CLI by running:

```
cf login
```

2. List parameters in a comma-separated list after the `-p` flag. Run:

```
cf cups SERVICE -p "PARAMETER, SECOND-PARAMETER, THIRD-PARAMETER"
```

Where:

- `SERVICE` is the name of the service you want to create.

  - PARAMETER, SECOND-PARAMETER, and THIRD-PARAMETER are parameters such as username, password, host, port, and database name.

Supply parameters non-interactively

To create a new service while supplying parameters non-interactively:

1. Log in to the cf CLI by running:

   ```
   cf login
   ```

2. Pass parameters and their values in as a JSON hash, bound by single quotes, after the `-p` tag. Run:

   ```
   cf cups SERVICE -p '{"host":"HOSTNAME", "port":"PORT"}'
   ```

   Where:

   - `SERVICE` is the name of the service you want to create.
   - `HOSTNAME` and `PORT` are service parameters.

Supply parameters through a third party

For specific log service instructions, see Streaming app logs to third-party services.

To create a service instance that sends data to a third party:

1. Log in to the cf CLI:

   ```
   cf login
   ```

2. Create a service instance that sends data to a third party by running:

   ```
   cf cups SERVICE -l THIRD-PARTY-DESTINATION-URL
   ```

   Where:

   - `SERVICE` is the name of the service you want to create.
   - `THIRD-PARTY-DESTINATION-URL` is the external URL of the third-party service.

Bind and unbind service instances

After you create a user-provided service instance, you can:

- Bind the service to an app with `cf bind-service`. For more information, see bind-service in the *Cloud Foundry CLI Reference Guide*.
- Unbind the service with `cf unbind-service`. For more information, see unbind-service in the *Cloud Foundry CLI Reference Guide*.
- Rename the service with `cf rename-service`. For more information, see rename-service in the *Cloud Foundry CLI Reference Guide*.

- Delete the service with `cf delete-service`. For more information, see delete-service in the *Cloud Foundry CLI Reference Guide*.

Update a service instance

To update one or more of the parameters for an existing user-provided service instance, use `cf update-user-provided-service` or `cf uups`.

For more information, see update-user-provided-service in the *Cloud Foundry CLI Reference Guide*.

> **Note:** The `cf uups` command does not update any parameter values that you do not supply.

Retrieve cf CLI return codes

The cf CLI uses exit codes, which help with scripting and confirming that a command has run successfully.

To view a cf CLI exit code:

1. Log in to the cf CLI by running:

   ```
   cf login
   ```

2. To check that the login was successful, run one of these commands, depending on your OS:

   - For Mac OS, run:

     ```
     echo $?
     ```

   - For Windows, run:

     ```
     echo %ERRORLEVEL%
     ```

If the command succeeds, the exit code is `0`.

View CLI help output

The `cf help` command lists the cf CLI commands and a brief description of each. For more information, see help in the *Cloud Foundry CLI Reference Guide*.

To list detailed help for any cf CLI command, add the `-h` flag to the command.

The example below shows detailed help output for the `cf delete` command:

```
$ cf delete -h
NAME:
   delete - Delete an app

USAGE:
```

```
   cf delete APP_NAME [-f -r]

ALIAS:
   d

OPTIONS:
   -f       Force deletion without confirmation
   -r       Delete any mapped routes (only deletes routes mapped to a single
app)
```

Using the cf CLI with a proxy server

If you have an HTTP or SOCKS5 proxy server on your network between a host running the cf CLI and your API endpoint, you must set `https_proxy` with the hostname or IP address of the proxy server. The `https_proxy` environment variable holds the hostname or IP address of your proxy server.

`https_proxy` is a standard environment variable. As with any environment variable, the specific steps you use to set it depends on your operating system.

Format of https_proxy

`https_proxy` is set with hostname or IP address of the proxy server in URL format, as in the example below:

```
https_proxy=http://proxy.example.com
```

If the proxy server requires a username and password, you must include the credentials, as in the example below:

```
https_proxy=http://username:password@proxy.example.com
```

If the proxy server uses a port other than 80, you must include the port number, as in the example below:

```
https_proxy=http://username:password@proxy.example.com:8080
```

If the proxy server is a SOCKS5 proxy, you must specify the SOCKS5 protocol in the URL, as in the example below:

```
https_proxy=socks5://socks_proxy.example.com
```

Note: The `cf ssh` command for cf CLI v7 does not work through a SOCKS5 proxy.

Set https_proxy in Mac OS or Linux

To set the `https_proxy` environment variable in Mac OS or Linux:

1. Use the command specific to your shell. For example, in bash, use the `export` command, as in the example below:

   ```
   export https_proxy=http://my.proxyserver.com:8080
   ```

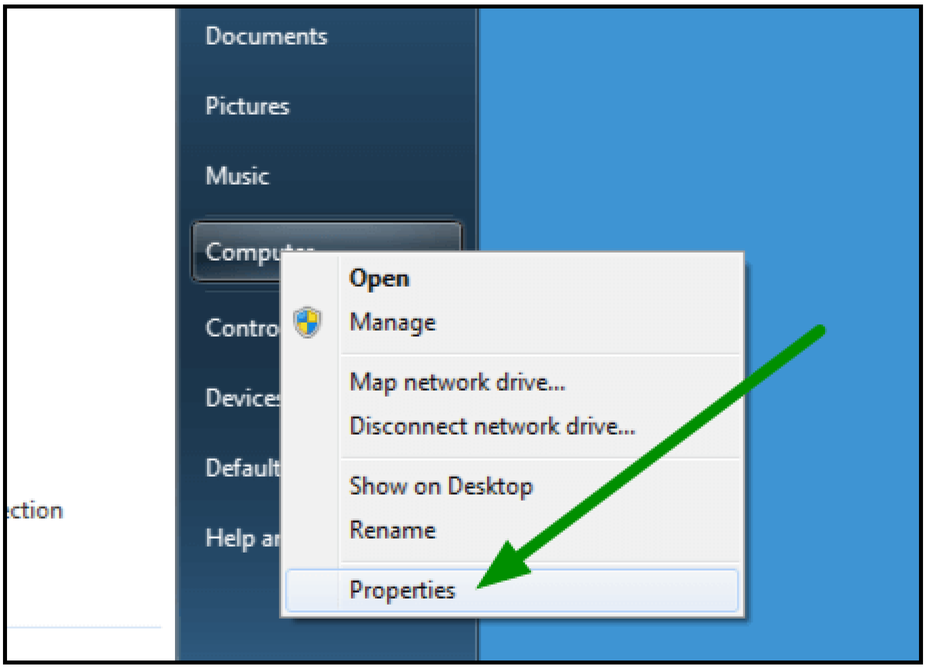
2. To make this change persistent, add the command to the appropriate profile file for the shell. For example, in bash, add a line like example below to your `.bash_profile` or `.bashrc` file:

   ```
   https_proxy=http://username:password@hostname:port
   export $https_proxy
   ```

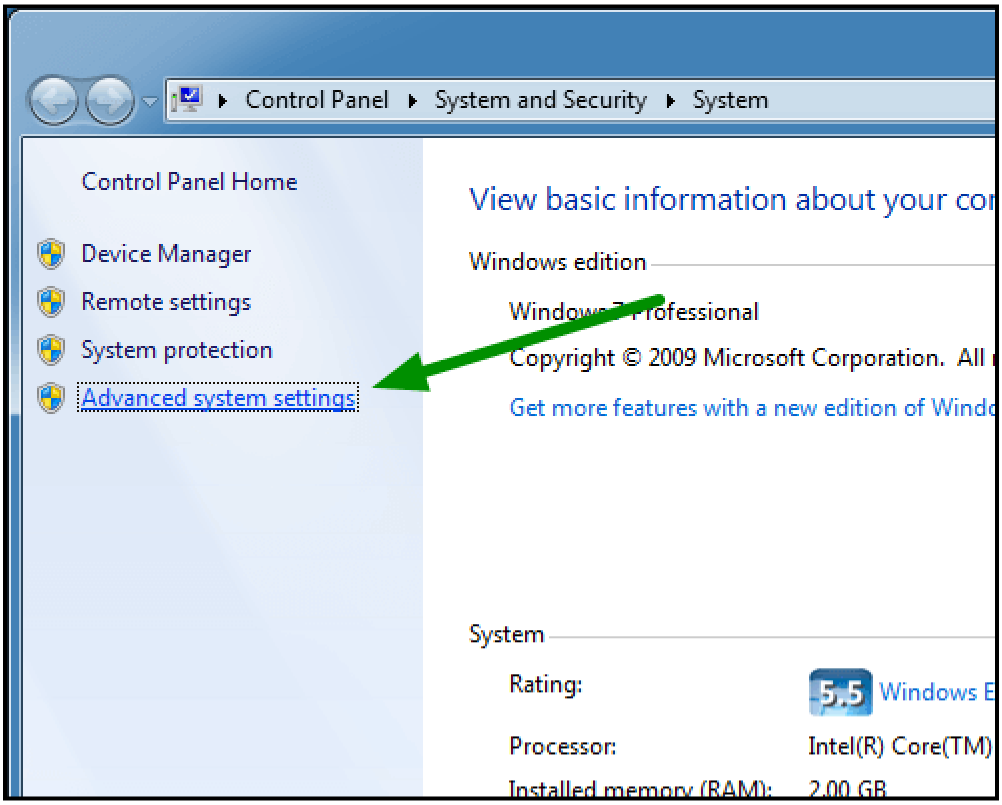
Set https_proxy in Windows

To set the `https_proxy` environment variable in Windows:

1. Open the **Start** menu.
2. Right-click **Computer** and select **Properties**.



3. In the left pane of the **System** window, click **Advanced system settings**.



Control Panel ▸ System and Security ▸ System

Control Panel Home

Device Manager

Remote settings

System protection

Advanced system settings

View basic information about your co

Windows edition

Windows 7 Professional

Copyright © 2009 Microsoft Corporation. All r

Get more features with a new edition of Windo

System

Rating: 5.5 Windows E

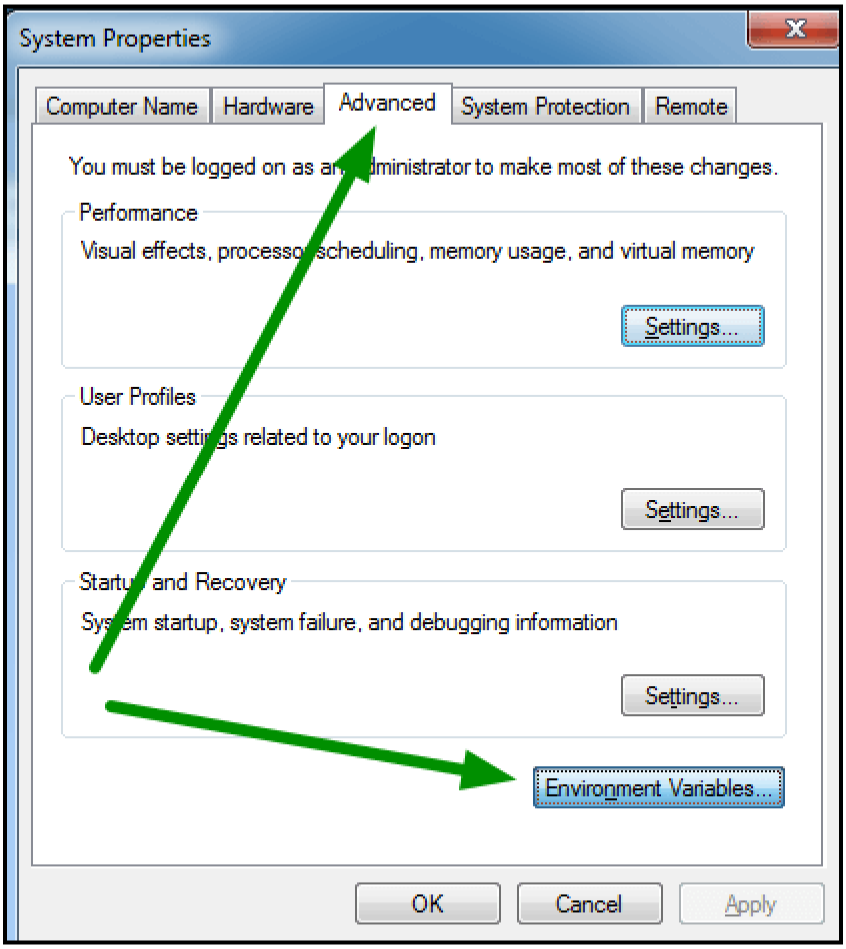
Processor: Intel(R) Core(TM)

Installed memory (RAM): 2.00 GB

4. In the **System Properties** window:

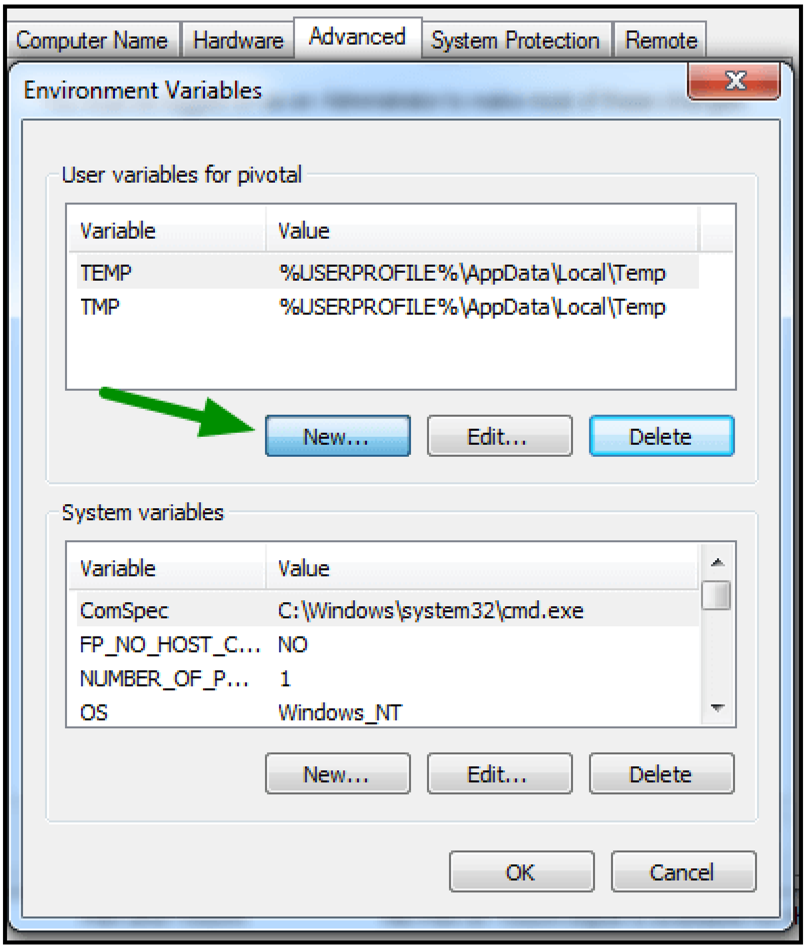
    1. Select **Advanced**.

    2. Click **Environment Variables**.



System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your logon
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

5. Under **User variables**, click **New**.



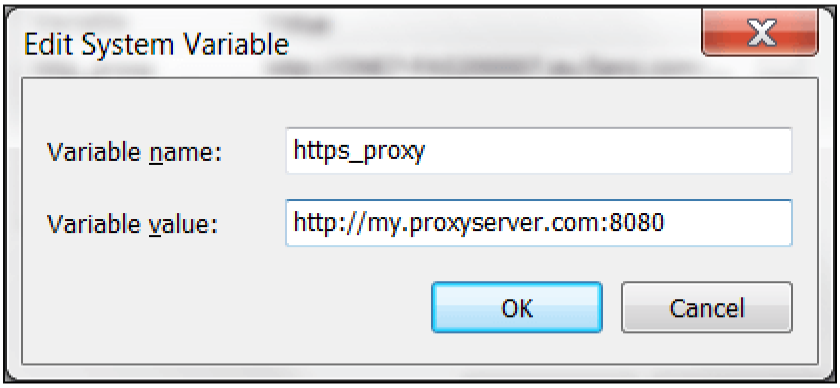
User variables for pivotal

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	1
OS	Windows_NT

6. For **Variable name**, enter `https_proxy`.
7. For **Variable value**, enter your proxy server information.



Variable name: https_proxy

Variable value: http://my.proxyserver.com:8080

8. Click **OK**.

Using the cf CLI with a self-signed certificate

This topic describes how developers can use the Cloud Foundry Command Line Interface (cf CLI) to communicate securely with a Cloud Foundry deployment with a self-signed certificate.

You can use the cf CLI to communicate securely with a Cloud Foundry deployment without specifying `--skip-ssl-validation` under these circumstances:

- The deployment uses a self-signed certificate.

- The deployment uses a certificate that is signed by a self-signed certificate authority (CA), or a certificate signed by a certificate that is signed by a self-signed CA.

Before following the procedure below, the developer must obtain either the self-signed certificate or the intermediate and CA certificate(s) used to sign the deployment's certificate. The developer can obtain these certificates from the Ops Manager operator.

Installing the certificate on physical machines

The certificates you must insert into your local truststore vary depending on the configuration of your deployment.

- If the deployment uses a self-signed certificate, you must insert the self-signed certificate into your local trust store.
- If the deployment uses a certificate that is signed by a self-signed CA, or a certificate signed by a certificate that is signed by a self-signed CA, you must insert the self-signed certificate and any intermediate certificates into your local trust store.

Installing the certificate on Mac OS X

To place the certificate file `server.crt` into your local truststore for Mac OS X:

1. Run:

   ```
   sudo security add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.key
   chain server.crt
   ```

Installing the certificate on Linux

To place the certificate file `server.crt` into your truststore for Linux:

1. Run one of the following commands, depending on your Linux distribution:
   - For Debian, Ubuntu, or Gentoo, run:

     ```
     cat server.crt >> /etc/ssl/certs/ca-certificates.crt
     ```

   - For Fedora or RHEL, run:

     ```
     cat server.crt >> /etc/pki/tls/certs/ca-bundle.crt
     ```

The examples above set the certificate permanently on your machine across all users and require `sudo` permissions. To set the certificate only in your current terminal or script, run one of these commands:

- Option 1:

  ```
  export SSL_CERT_FILE=PATH-TO-SERVER.crt
  ```

  Where `PATH-TO-SERVER.crt` is the filepath of the `server.crt` certificate file.

- Option 2:

```
export SSL_CERT_DIR=PATH-TO-SERVER-DIRECTORY
```

Where `PATH-TO-SERVER-DIRECTORY` is the directory of the `server.crt` certificate file.

Installing the certificate on Windows

To place the certificate file `server.crt` into your local truststore for Windows:

1. Right-click on the certificate file.
2. Click **Install Certificate**.
3. Choose to install the certificate as the **Current User** or **Local Machine**.
4. From the certification store list, select **Trusted Root Certification Authorities**.

Using a cf CLI plug-in

The Cloud Foundry CLI (cf CLI) includes plug-in capability. The plug-in allows you to add custom commands to the cf CLI.

You can install and use the plug-in that Cloud Foundry developers and third-party developers create. For a current list of community-supported plug-ins, see the Cloud Foundry Community CLI Plug-in page. For information about submitting your own plug-in, see the Cloud Foundry CLI Plugin Repository (CLIPR) repository on GitHub.

⚠ **Caution:** Plugins are not vetted in any way, including for security or functionality. Use plugins at your own risk.

The cf CLI identifies a plugin by its binary filename, its developer-defined plugin name, and the commands that the plugin provides. You use the binary filename only to install a plugin. You use the plugin name or a command for any other action.

Note: The cf CLI uses case-sensitive commands, but plugin management commands accept plugin and repository names irrespective of their casing.

Changing the plug-in directory

By default, the cf CLI stores plugins on your workstation in `$CF_HOME/.cf/plugins`, which defaults to `$HOME/.cf/plugins`.

To change the root directory of this path from `$CF_HOME`, you must set the `CF_PLUGIN_HOME` environment variable.

The cf CLI appends `.cf/plugins` to the `CF_PLUGIN_HOME` path that you specify and stores plug ins in that location. For example, if you set `CF_PLUGIN_HOME` to `/my-folder`, cf CLI stores plug-ins in `/my-folder/.cf/plugins`.

Installing a plug-in

To install a plugin:

1. Download a binary or the source code for a plugin from a trusted provider.

   > **Note:** The cf CLI requires a binary file compiled from source code written in Go. If you download source code, you must compile the code to create a binary.

2. Run:

   ```
   cf install-plugin BINARY-FILENAME
   ```

   Where `BINARY-FILENAME` is the path to and name of your binary file.

   > **Note:** You cannot install a plug in that has the same name or that uses the same command as an existing plugin. If you attempt to do so, you are prompted to uninstall the existing plugin.

   > **Note:** The cf CLI prohibits you from implementing any plugin that uses a native cf CLI command name or alias. For example, if you attempt to install a third-party plugin that includes the `cf push` command, the cf CLI halts the installation.

For more information, see install-plugin in the *Cloud Foundry CLI Reference Guide*.

Managing plug-ins and running plug-in commands

Use the contents of the `cf help` **CLI plugin management** and **Commands offered by installed plugins** sections to manage plugins and run plugin commands.

To manage plugins and run plugin commands:

1. List all installed plugins and all commands that the plugins provide by running:

   ```
   cf plugins
   ```

2. Execute a plugin command by running:

   ```
   cf PLUGIN-COMMAND
   ```

   Where `PLUGIN-COMMAND` is the plugin command you want to execute.

Checking for plug-in updates

To check all registered plugin repositories for newer versions of currently installed plugins:

1. Run:

   ```
   cf plugins --outdated
   ```

2. See the output of the above command, as in the example below:

```
$ cf plugins --outdated
Searching CF-Community, company-repo for newer versions of installed pl
ugins...
plugin          version   latest version
coffeemaker     1.1.2     1.2.0
Use 'cf install-plugin' to update a plugin to the latest version.
```

For more information about the `cf plugins` command, see cf plugins in the *Cloud Foundry CLI Reference Guide*.

Uninstalling a plug-in

To uninstall a plugin:

1. View the names of all installed plugins by running:

   ```
   cf plugins
   ```

2. Run:

   ```
   cf uninstall-plugin PLUGIN-NAME
   ```

   Where `PLUGIN-NAME` is the name of the plug-in you want to uninstall.

   > **Note:** You must use the name of the plug-in to uninstall it, not the binary filename.

For more information, see uninstall-plugin in the *Cloud Foundry CLI Reference Guide*.

Adding a plug-in repository

To add a plugin repository:

1. Run:

   ```
   cf add-plugin-repo REPOSITORY-NAME-URL
   ```

   Where `REPOSITORY-NAME-URL` is the URL of the plugin repository you want to add.

For more information, see add-plugin-repo in the *Cloud Foundry CLI Reference Guide*.

Viewing available plug-in repositories

To view your available plugin repositories:

1. Run:

   ```
   cf list-plugin-repos
   ```

For more information, see list-plugin-repos in the *Cloud Foundry CLI Reference Guide*.

Listing all plug-ins by repository

To show all plugins from all available repositories:

1. Run:

   ```
   cf repo-plugins
   ```

For more information, see repo-plugins in the *Cloud Foundry CLI Reference Guide*.

Troubleshooting

The cf CLI provides the error messages described below to help you troubleshoot installation and usage issues. Third-party plugins can provide their own error messages.

Permission denied

If you receive a `permission denied` error message, you lack required permissions to the plugin. You must have `read` and `execute` permissions to the plugin binary file.

Plug-in command collision

Plugin names and commands must be unique. The CLI displays an error message if you attempt to install a plugin with a non-unique name or command.

If the plugin has the same name or command as a currently installed plugin, you must first uninstall the existing plugin to install the new plugin.

If the plugin has a command with the same name as a native cf CLI command or alias, you cannot install the plugin.

Developing cf CLI plug-ins

You can create and install cf CLI plug-ins to provide custom commands. You can submit and share these plug-ins to the CF Community repository.

Prerequisite

To use plugins, you must use cf CLI v7 or later. For information about downloading, installing, and uninstalling the cf CLI, see Installing the Cloud Foundry Command Line Interface.

Creating a plug-in

To prepare to create a plugin:

1. Implement the predefined plug-in interface from plugin.go in the Cloud Foundry CLI repository on GitHub.
2. Clone the Cloud Foundry CLI repository from GitHub. To create a plugin, you need the basic GO plugin.

Initializing the plug-in

To initialize a plugin:

1. From within the `main()` method of your plugin, call:

   ```
   plugin.Start(new(MyPluginStruct))
   ```

   The `plugin.Start(...)` function requires a new reference to the `struct` that implements the defined interface.

Invoking cf CLI commands

To invoke cf CLI commands,

1. From within the `Run(...)` method of your plugin, call:

   ```
   cliConnection.CliCommand([]args)
   ```

The `Run(...)` method receives the `cliConnection` as its first argument. The `cliConnection.CliCommand([]args)` returns the output printed by the command and an error.

The output is returned as a slice of strings. The error occurs if the call to the cf CLI command fails.

For more information, see Plugin API in the Cloud Foundry CLI repository on GitHub.

Installing a plug-in

To install a plugin:

1. Run:

   ```
   cf install-plugin PATH-TO-PLUGIN-BINARY
   ```

   Where `PATH-TO-PLUGIN-BINARY` is the path to your plugin binary.

For more information about developing plugins, see plugin_examples in the Cloud Foundry CLI repository on GitHub.

Observability

In this section:

Logging and metrics in Cloud Foundry

This topic provides an overview of logging and metrics in VMware Tanzu Application Service for VMs (TAS for VMs). It includes information about logs and metrics sources and transport systems. It also lists products for viewing logs and metrics.

Accessing metrics

You must use a Cloud Foundry Command-Line Interface (cf CLI) plugin to access and view metrics directly from the command line. You can use either the Firehose plugin or the Log Cache plugin.

Accessing metrics using the Log Cache CLI Plug-in

To access metrics with the Log Cache plug-in:

1. Log in to the cf CLI by running:

   ```
   cf login -a API-URL -u USERNAME -p PASSWORD
   ```

   Where:

   - `API-URL` is your API endpoint.
   - `USERNAME` is your username.
   - `PASSWORD` is your password.

2. Install the Log Cache CLI plugin by running:

   ```
   cf install-plugin -r CF-Community "log-cache"
   ```

   For more information, see log-cache in *cf CLI Plugins* on the Cloud Foundry website.

3. Run:

   ```
   cf query 'METRIC-NAME{source_id="SOURCE-ID"}'
   ```

   Where:

   - `METRIC-NAME` is the name of the metric you want to view.
   - `SOURCE-ID` is the source ID of the component for which you want to view metrics.

   To find the source ID and metric name of the metric you want to view, see Key Performance Indicators and Key Capacity Scaling Indicators.

Accessing metrics using the Firehose Plug-in

To access metrics using the Firehose plugin:

1. Log in to the cf CLI by running:

   ```
   cf login -a API-URL -u USERNAME -p PASSWORD
   ```

   Where:

   - `API-URL` is your API endpoint.
   - `USERNAME` is your username.
   - `PASSWORD` is your password.

2. Install the Firehose cf CLI plugin by running:

   ```
   cf install-plugin -r CF-Community "Firehose Plugin"
   ```

   For more information, see Firehose Plugins in *cf CLI Plugins* on the Cloud Foundry website.

3. Run:

   ```
   cf nozzle -no-filter | grep SOURCE-ID | grep -i METRIC-NAME
   ```

   Where:

   - `METRIC-NAME` is the name of the metric you want to view.
   - `SOURCE-ID` is the source ID of the component for which you want to view metrics.

   For example:

   ```
   cf nozzle -no-filter | grep bbs | grep -i ConvergenceLRPDuration
   ```

   Because metrics are scraped at different intervals, it may take up to fifteen minutes for the Firehose to receive all metrics for the component. Re-run `cf nozzle` until you have received a complete set of metrics for a component. For more information, see Firehose Architecture in *Loggregator Architecture*.

   To find the source ID and metric name of the metric you want to view, see Key Performance Indicators and Key Capacity Scaling Indicators.

For more information about nozzles, see Nozzles in *Logs, Metrics, and Nozzles* in the Ops Manager Tile Developer Guide.

Sources for logs and metrics

There are two sources of TAS for VMs logs and metrics:

- TAS for VMs platform components, such as a Diego Cell, MySQL Server, or Cloud Controller
- Apps and app containers deployed on TAS for VMs

The table below describes the data included in logs and metrics from each source:

Source	Logs Data	Metrics Data

Platform components	Logs from TAS for VMs components	- Health metrics from BOSH-deployed VMs[1] - Platform metrics from TAS for VMs components. For example, Diego Cell capacity remaining and Gorouter throughput. - Metrics for any service tile that self-publishes to the Loggregator Firehose. For example, Redis and MySQL.
Apps and app containers	Logs from apps[2]	- Container metrics[3] - Custom app metrics[4]

[1] For more information about using the BOSH Health Monitor to collect health metrics on VMs, see Configuring a Monitoring System.

[2] For more information about app logging, see App Logging in TAS for VMs.

[3] For more information about container metrics, see Container Metrics.

[4] For more information about configuring an app to stream custom metrics to Loggregator, see Metric Registrar and Custom App Metrics.

Transport Systems for Logs and Metrics

The following transport systems deliver logs and metrics from their source to an observability product for viewing:

- **Loggregator:** Loggregator is the transport system for both logs and metrics on apps deployed on TAS for VMs, as well as metrics on TAS for VMs platform components. For more information about the Loggregator system, including Loggregator architecture and components, see Loggregator Architecture.
- **rsyslogd on TAS for VMs component VMs:** rsyslogd is the transport system for TAS for VMs component logs. Users can configure rsyslogd to transport component logs to a third-party syslog server.

The table below lists the transport system for logs and metrics on TAS for VMs platform components and apps:

Source	Logs Transport System	Metrics Transport System
Platform components	rsyslogd on TAS for VMs component VMs	Loggregator
Apps	Loggregator	Loggregator

Viewing Logs and Metrics

The table below lists the products and tools for viewing TAS for VMs logs and metrics:

Source	Products and Tools for Viewing Logs	Products and Tools for Viewing Metrics
Platform components	To view system logs from TAS for VMs components, configure rsyslogd to transport logs to a third-party product.	You can use the following products or tools to view platform component and VM metrics: • Loggregator Firehose CLI Plugin. See Installing the Loggregator Firehose Plugin for CLI. • Loggregator Log Cache CLI Plugin. See Cloud Foundry Community cf CLI Plugins. • Healthwatch. See Healthwatch for VMware Tanzu.
Apps	You can use the following products or tools to view app logs: • cf CLI cf logs command. See Cloud Foundry CLI Reference Guide. • Apps Manager. See Managing Apps and Service Instances Using Apps Manager. • Syslog forwarding. See Streaming App Logs to Log Management Services. • Loggregator Firehose CLI Plugin. See Installing the Loggregator Firehose Plugin for CLI. • Loggregator Log Cache CLI Plugin. See Cloud Foundry Community cf CLI Plugins.	You can use the following products or tools to view app metrics: • Loggregator Firehose CLI Plugin. See Installing the Loggregator Firehose Plugin for CLI. • Loggregator Log Cache CLI Plugin. See Cloud Foundry Community cf CLI Plugins. • App Metrics. See the App Metrics documentation.

Logging and Metrics architecture

Logging and metrics architecture includes components that transport logs and metrics from your TAS for VMs deployment to destinations such as the Cloud Foundry Command-Line Interface (cf CLI), monitoring tools, or internal system components.

Log and metric egress is the process by which logs and metrics are transported from the your deployment to destinations such as the cf CLI, monitoring tools, or other internal system components. These logs and metrics can either flow directly from the VMs that produce them to their consumers, or pass through the Loggregator Firehose.

Arrows are shown depicting the flow of information from VMs. One arrow points directly from VMs to Consumers. The other arrow points to the Firehose, which then sends to Consumers.

How Logs and Metrics Egress from VMs

Logs and metrics are sent from VMs via the following process:

A Forwarder Agent appears inside a square labeled VMs. Apps are shown sending to the Forwarder Agent, and Components are shown sending metrics to the the Forwarder Agent. Components also send logs to rsyslon, which then sends platform logs outside the system via syslog RFC5424. The Forwarder Agent sends to three downstream consumers. The Syslog Agent sends application logs via syslog RFC5424. The Metrics Agent exposes metrics via Prometheus endpoints. Finally, the Loggregator Agent sends metrics and application logs via syslog RFC5424.

1. Applications and components on the system emit logs and metrics.
2. Logs and metrics pass through two forwarders:
   - rsyslog, which sends platform logs in Syslog RFC5424 format.
   - The Forwarder Agent, which sends metrics and application logs to three agents (see below).
3. Metrics and application logs are sent through each of the following:
   - The Syslog Agent sends application logs in Syslog RFC5424 format to aggregate and application log destinations.
   - The Metrics Agent exposes metrics for Prometheus-style scraping.
   - The Loggregator Agent sends metrics and application logs to the Loggregator Firehose.

How the Firehose Forwards Logs and Metrics

The Loggregator Firehose sends logs and metrics through the V1 and V2 Firehose APIs, via the following process:

A Loggregator Agent appears inside a square labeled VMs. Loggregator Agents send to Dopplers, located on Doppler VMs. Dopplers, in turn, send to both Traffic Controllers and Reverse Log Proxies. V1 Firehose Consumers receive logs and metrics from Traffic Controllers, whereas V2 Firehose Consumers receive logs and metrics from Reverse Log Proxies. V2 Firehose Consumers can also receive logs and metrics from Reverse Log Proxy (RLP) Gateways.

1. The Loggregator Agent takes each log/metric and sends it to one downstream Doppler (distributing among a rondom subset of 5 Dopplers).
2. Dopplers then make a copy of log/metric for each consumer and send them to the Traffic Controllers and Reverse Log Proxies for distribution.
3. These two components forward logs and metrics differently:
   - Traffic Controllers receive Websocket connections from V1 Firehose Consumers, and send logs/metrics as V1 Envelopes.

     If any of these consumers start to fall behind, Traffic Controllers will log a slow consumer alert and disconnect that particular consumer.

   - Reverse Log Proxies receive gRPC connections from V2 Firehose Consumers, and send logs/metrics as V2 Envelopes.

    If a consumer falls behind, the envelopes will just be dropped.

    - Reverse Log Proxy Gateways (RLP Gateways) receive HTTP connections from V2 Firehose consumers, and send logs/metrics as JSON-encoded V2 Envelopes.

      Reverse Log Proxy Gateways connect to Reverse Log Proxies, rather than directly to Dopplers.

How Consumers Receive Logs and Metrics

Consumers of logs and metrics may receive them through a few different paths:

A Forwarder Agent appears inside a square labeled VMs. Apps are shown sending to the Forwarder Agent, and Components are shown sending metrics to the the Forwarder Agent. Components also send logs to rsyslon, which then sends platform logs outside the system via syslog RFC5424. The Forwarder Agent sends to three downstream consumers. The Syslog Agent sends application logs via syslog RFC5424. The Metrics Agent exposes metrics via Prometheus endpoints. Finally, the Loggregator Agent sends metrics and application logs via syslog RFC5424.

Log Cache provides a short-term snapshot of logs and metrics to the CF CLI and any CF Web UIs (such as Stratos or Tanzu Application Service Apps Manager). It receives logs and metrics either from the Firehose or directly from VMs via Syslog.

Observability integrations can consume logs and metrics via either:

- connecting to the Loggregator Firehose, or
- receiving logs in Syslog RFC 5424 format and scraping metrics from Prometheus endpoints.

Full Architecture References

In addition to the components in these architectures, you can also use a System Metrics Agents architecture on a Loggregator deployment to collect metrics from system components and expose them on Prometheus-scrapable endpoints.

Architecture reference diagrams

This section provides architecture diagrams that show the components that collect, store, and forward logs and metrics in your TAS for VMs deployment.

You can use these diagrams to:

- Understand the components that transport logs and metrics on your deployment.
- Diagnose performance issues related to logging and metrics.
- Make decisions about how to best scale the components or consumers described in the architecture.

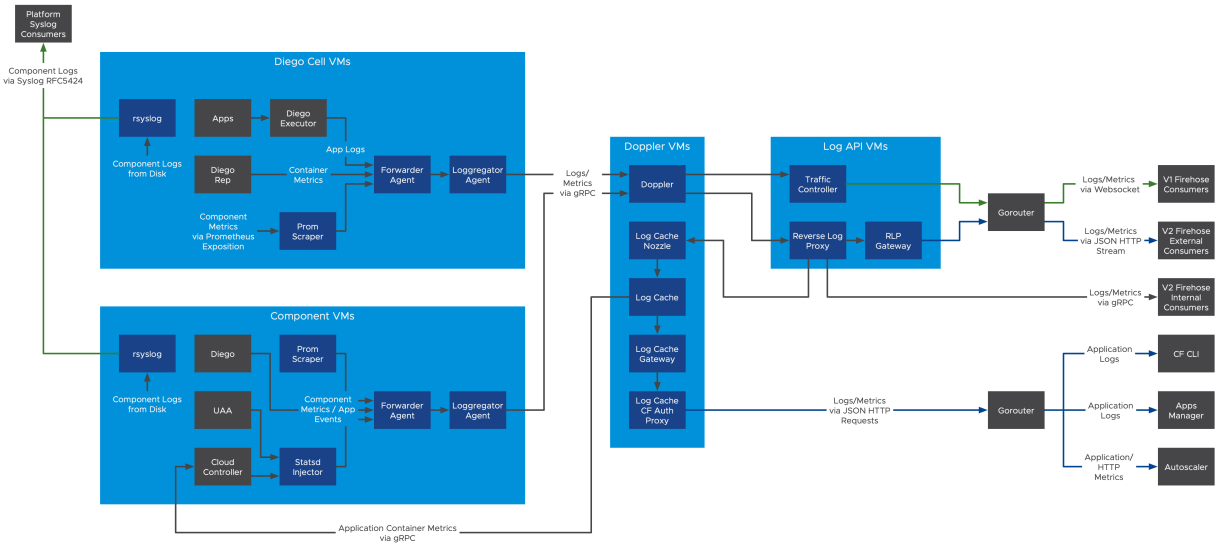
Firehose Architecture

- Loggregator Firehose architecture. See Loggregator Firehose Architecture.

- Shared-nothing architecture. See Shared-Nothing Architecture.
- System Metrics Agents architecture. See System Metrics Agents Architecture.

Loggregator Firehose Architecture

The following diagram shows how logs and metrics are transported from components and apps on your deployment to Loggregator Firehose consumers, such as nozzles, monitoring tools, or third-party software.



View a larger version of this image.

The following components are included in the Loggregator Firehose architecture, as shown in the diagram above:

- **Prom Scraper**: Prom Scrapers run on both component VMs and Diego Cell VMs. They aggregate metrics from components located on those VMs through Prometheus exposition. Prom Scrapers then forward the metrics to Forwarder Agents.
- **Statsd Injector**: Statsd Injectors run on component VMs. They receive metrics from components over the Statsd protocol. Statsd Injectors then forward the metrics to Forwarder Agents.
- **Forwarder Agent**: Forwarder Agents run on both component VMs and Diego Cell VMs. They receive logs and metrics from the apps and components located on those VMs. Forwarder Agents then forward the logs and metrics to Loggregator Agents and other agents.
- **Loggregator Agent**: Loggregator Agents run on both component VMs and Diego Cell VMs. They receive logs and metrics from the Forwarder Agents, then forward the logs and metrics from multiple Dopplers.
- **Doppler**: Dopplers receive logs and metrics from Loggregator Agents, store them in temporary buffers, and forward them to Traffic Controllers and Reverse Log Proxies.
- **Traffic Controller**: Traffic Controllers poll Doppler servers for logs and metrics, then translate these messages from the Doppler servers as necessary for external and legacy APIs. It services the Firehose Endpoint, also known as the V1 Firehose. The Firehose cf CLI

plugin allows you to view the output of the Firehose. For more information about the Firehose plugin, see Installing the Loggregator Firehose Plugin for cf CLI.

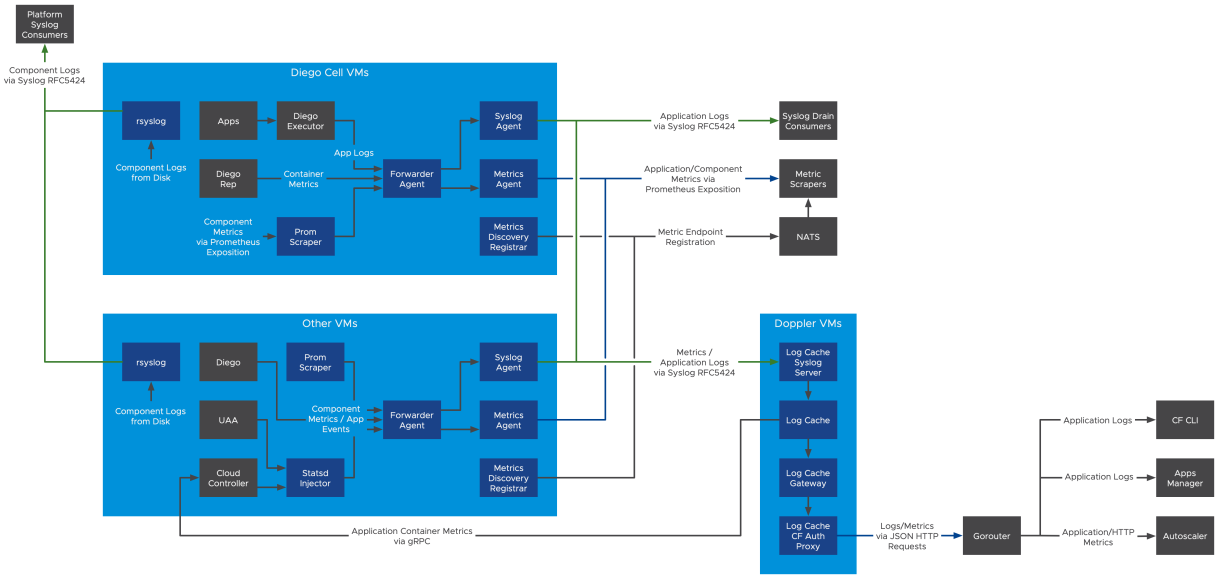
- **Reverse Log Proxy (V2 Firehose)**: Reverse Log Proxies (RLPs) collect logs and metrics from Dopplers and forward them to Log Cache and other consumers over gRPC. Operators can scale up the number of RLPs based on overall log volume.
- **Reverse Log Proxy Gateway (V2 Firehose)**: Reverse Log Proxies Gateways (RLP Gateways) collect logs and metrics from Reverse Log Proxies and forward them to consumers over HTTP. Collecting logs and metrics through the RLP Gateway has lower throughput compared to consuming from the Traffic Controller or Reverse Log Proxy.
- **Log Cache**: Log Cache allows you to view logs and metrics over a specified period of time. The Log Cache includes API endpoints and a CLI plug-in to query and filter logs and metrics. To download the Log Cache CLI plugin, see Cloud Foundry Plugins. The Log Cache API endpoints are available by default. For more information about using the Log Cache API, see Log Cache on GitHub.
- **rsyslog**: While not part of the firehose itself, rsyslog is responsible for delivering logs from platform components to outside consumers.

Shared-Nothing Architecture

Shared Nothing architecture

Similar to the Loggregator Firehose Architecture, the shared-nothing architecture allows you to forward logs and metrics from your deployment to external and internal consumers.

In contrast to the Loggregator Firehose architecture, logs and metrics to pass through fewer components in the shared-nothing architecture. For example, the shared-nothing architecture does not require the Forwarder Agent, Syslog Agent, or Metrics Agent.



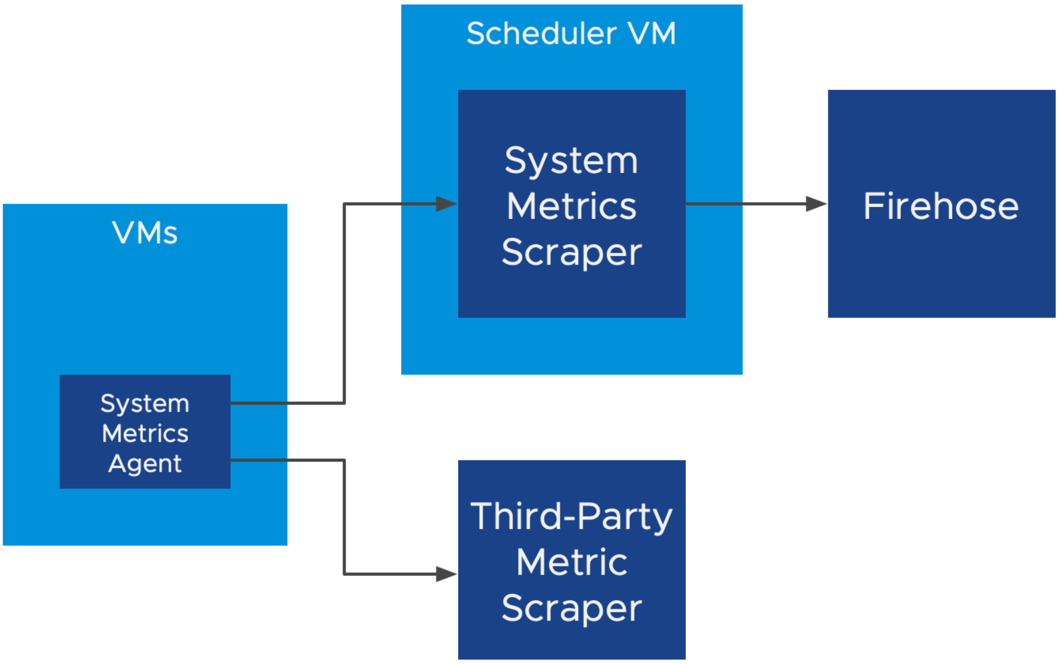
View a larger version of this image.

The following components are included in the shared-nothing Architecture, as shown in the diagram above:

- **Prom Scraper**: Prom Scrapers run on both Ops Manager component VMs and Diego Cell VMs. They aggregate metrics from Ops Manager components located on those VMs through Prometheus exposition. Prom Scrapers then forward those metrics to Forwarder Agents.
- **Statsd Injector**: Statsd Injectors run on Ops Manager component VMs. They receive metrics from Ops Manager components over the Statsd protocol. Statsd Injectors then forward those metrics to Forwarder Agents.
- **Forwarder Agent**: Forwarder Agents run on both Ops Manager component VMs and Diego Cell VMs. They receive logs and metrics from the applications and Ops Manager components located on those VMs. Forwarder Agents then forward the logs and metrics to Loggregator Agents and other agents.
- **Syslog Agent**: Syslog Agents run on Ops Manager component VMs and host VMs to collect and forward logs to configured syslog drains. This includes syslog drains for individual apps as well as aggregate drains for all apps in your foundation. You can specify the destination for logs as part of the individual syslog drain or in the TAS for VMs tile.
- **Syslog Binding Cache** (not pictured): Syslog Agents can overwhelm CAPI when querying for existing bindings. This component acts as a caching proxy between the Syslog Agents and CAPI.
- **Metrics Agents**: Metrics Agents receive metrics from Forwarder Agents and make them available to Metric Scrapers through Prometheus Exposition.
- **Metrics Discovery Registrars**: Metrics Discovery Registrars register each scrapeable endpoint with NATS for discovery by Metrics Scrapers.
- **Log Cache**: Log Cache allows you to view logs and metrics over a specified period of time. The Log Cache includes API endpoints and a CLI plug-in to query and filter logs and metrics. To download the Log Cache CLI plugin, see Cloud Foundry Plugins. The Log Cache API endpoints are available by default. For more information about using the Log Cache API, see Log Cache on GitHub.
- **Log Cache Syslog Server**: The Log Cache Syslog Server receives logs and metrics from Syslog Agents and sends them to Log Cache.

System Metrics Agents architecture

The following diagram shows the architecture of a deployment that uses System Metrics Agents to collect VM and system-level metrics.



View a larger version of this image.

The following describes the components of a Loggregator deployment that uses System Metrics Agents, as shown in the diagram above:

- **System Metrics Agent**: A standalone agent to provide VM system metrics using a Prometheus-scrapeable endpoint.
- **System Metrics Scraper**: The System Metrics Scraper forwards metrics from System Metrics Agents to Loggregator Agents over mTLS.

How components transport logs and metrics

This section provides detailed descriptions of how the components in the Loggregator Firehose architecture and in the shared-nothing architecture transport logs and metrics on your deployment.

It describes the transport of logs and metrics during the following phases:

- How components in the logging and metrics architectures collect and forward metrics from VMs on your deployment. For more information, see How logs and metrics egress from VMs.
- How components in a Loggregator Firehose architecture collect and forward logs and metrics. For more information, see How Loggregator Firehose forwards logs and Mmtrics.
- How the logging and metrics architectures expose logs and metrics to consumers. For more information, see How consumers receive logs and metrics.

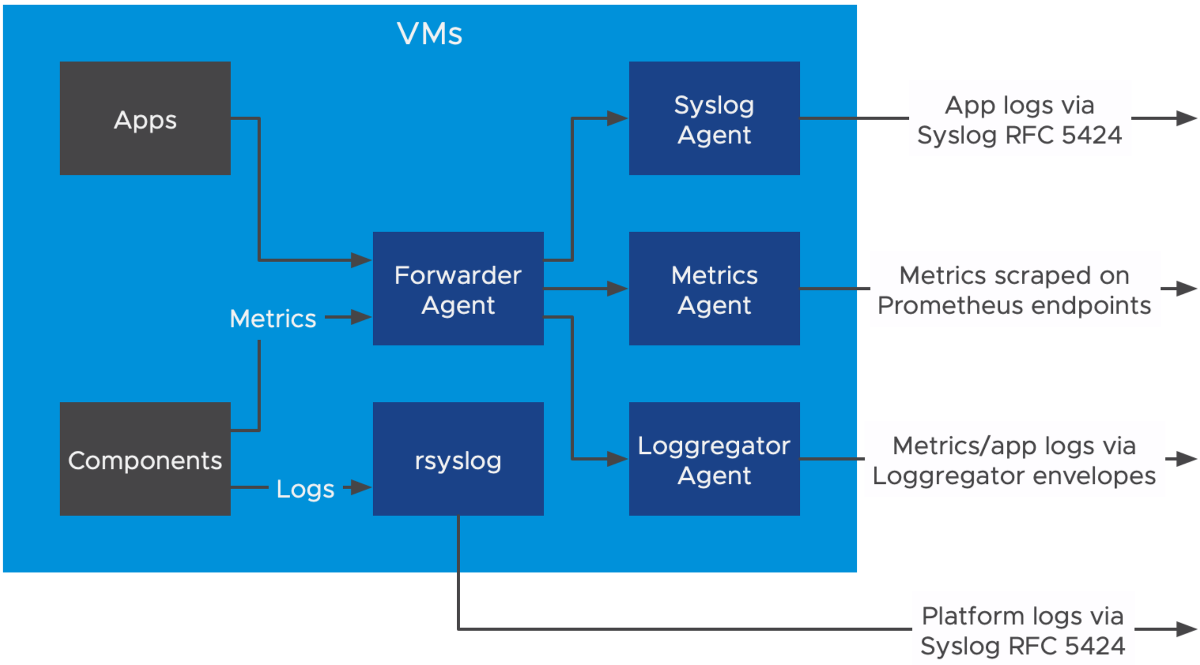
How logs and metrics egress from VMs

The following describes the transport of logs and metrics from VMs through system components and to a destination:

1. Apps and component VMs on your deployment emit logs and metrics.

2. Logs and metrics pass through two forwarders:
   - rsyslog. This sends logs from the component VMs in Syslog RFC 5424 format.
   - The Forwarder Agent. This sends metrics and app logs to the Syslog Agent, the Metrics Agent, and the Loggregator Agent.
3. From the Forwarder Agent, logs and metrics then pass through each of the following agents:
   - The Syslog Agent. This sends app logs in Syslog RFC 5424 format to aggregate and app log destinations.
   - The Metrics Agent. This exposes metrics for Prometheus-style scraping.
   - The Loggregator Agent. This sends metrics and app logs to the Loggregator Firehose. For more information about how logs and metrics flow through the Loggregator Firehose, see How the Loggregator Firehose Forwards Logs and Metrics below.

   For more information about how logs and metrics flow through the Loggregator Firehose, see How Loggregator Firehose forwards logs and metrics.


VMs
Apps
Components
Metrics
Logs
Forwarder Agent
rsyslog
Syslog Agent
Metrics Agent
Loggregator Agent
App logs via Syslog RFC 5424
Metrics scraped on Prometheus endpoints
Metrics/app logs via Loggregator envelopes
Platform logs via Syslog RFC 5424

How Loggregator Firehose forwards logs and metrics

For deployments that use a Loggregator Firehose architecture, the Loggregator Agent forwards logs and metrics emitted by apps and component VMs on your deployment to the Loggregator Firehose.

The following describes how the Loggregator Firehose sends logs and metrics through the V1 and V2 Firehose APIs:

1. The Loggregator Agent sends each log and metric to one Doppler. It distributes the logs and metrics among a random group of five Dopplers.

2. Dopplers make a copy of each log and metric for each consumer. The Dopplers then send the logs and metrics to Traffic Controllers and Reverse Log Proxies for distribution.
3. Traffic Controllers and Reverse Log Proxies distribute logs and metrics in the following ways:
    - Traffic Controllers receive WebSocket connections from V1 Firehose Consumers, and send logs and metrics as V1 Envelopes.

      If any of these consumers start to fall behind, Traffic Controllers log a slow consumer alert and disconnect that particular consumer.

    - Reverse Log Proxies receive gRPC connections from V2 Firehose Consumers and send logs and metrics as V2 Envelopes.

      If a consumer falls behind, the envelopes are dropped.

    - Reverse Log Proxy Gateways (RLP Gateways) receive HTTP connections from V2 Firehose consumers, and send logs and metrics as JSON-encoded V2 Envelopes.

      Reverse Log Proxy Gateways connect to Reverse Log Proxies, rather than directly to Dopplers.

How consumers receive logs and metrics

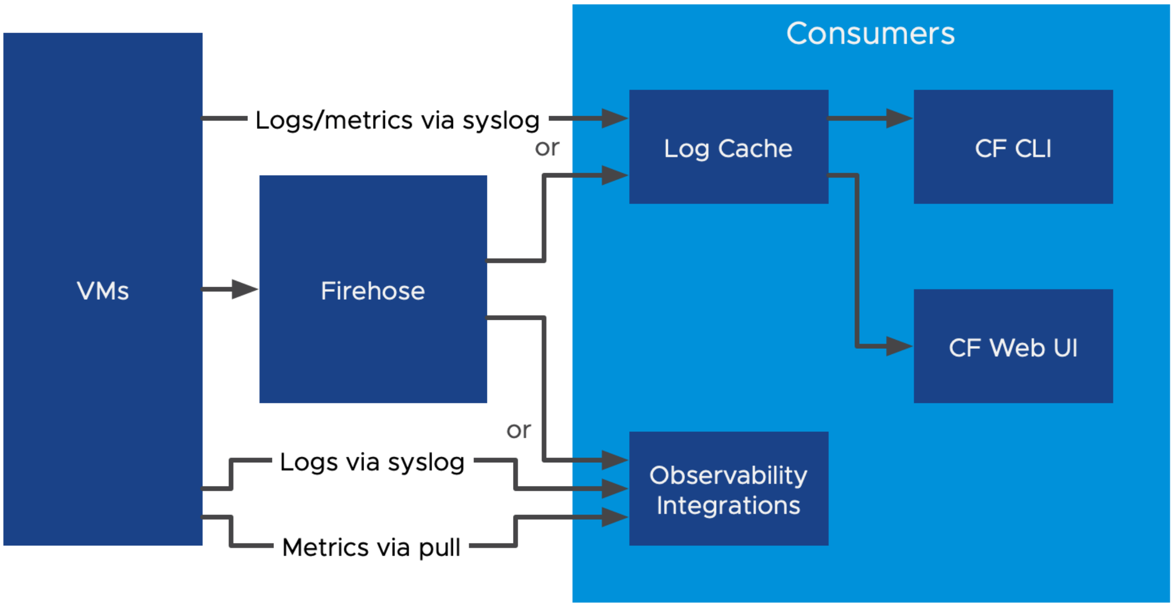
Consumers of logs and metrics include the Cloud Foundry Command Line Interface (cf CLI), Cloud Foundry web UIs such as Stratos, and any observability or monitoring product integrations.

The cf CLI and any Cloud Foundry web UIs can access logs and metrics through Log Cache. Log Cache receives logs and metrics from either the Firehose or directly from VMs through syslog. Log Cache provides short-term storage for logs and metrics where the cf CLI and web UIs can access them.

Integrations with observability or monitoring products receive logs and metrics through one of the following methods, depending on the logging and metrics architecture that your deployment uses:

- Connect to the Loggregator Firehose for access to both logs and metrics.
- Receive logs in syslog RFC 5424 format and scrape metrics from Prometheus endpoints.

The following diagram shows how consumers of logs and metrics receive logs and metrics from your deployment:



Related BOSH components

This section describes the components that forward BOSH-reported VM metrics to Loggregator. BOSH-reported VM metrics measure the health of BOSH-deployed VMs on which apps and components are deployed. Loggregator streams BOSH-reported VM metrics through the Firehose.

The following are the components that send BOSH-reported VM metrics to Loggregator:

- **BOSH Agent**: BOSH Agents are located on component VMs and Diego Cell VMs. They collect metrics, such as Diego Cell capacity remaining, from the VM and forward them to the BOSH Health Monitor.
- **BOSH Health Monitor**: The BOSH Health Monitor receives metrics from the BOSH Agents. It then forwards the metrics to a third-party service or to the BOSH System Metrics Forwarder.
- **BOSH System Metrics Plugin**: This plugin reads health events, such as VM heartbeats and alerts from the BOSH Health Monitor, and streams them to the BOSH System Metrics Server.
- **BOSH System Metrics Server**: The BOSH System Metrics Server streams metrics and heartbeat events to the BOSH System Metrics Forwarder over gRPC.
- **BOSH System Metrics Forwarder**: The BOSH System Metrics Forwarder is located on the Loggregator Traffic Controller. It forwards heartbeat events from the BOSH System Metrics Server as envelopes to Loggregator through a Loggregator Agent.

Log and Metric Agent architecture (Beta)

The Log and Metric Agent Architecture (Beta) contains a group of components that allow you to access all the same logs and metrics that you can through the Loggregator system.

Note: The Log and Metric Agent Architecture (Beta) does not currently integrate with Healthwatch or App Metrics.

The components of the Log and Metric Agent Architecture use a shared-nothing architecture that requires several fewer VMs than the Loggregator system.

The Log and Metric Agent Architecture includes components that collect, store, and forward logs and metrics in your Ops Manager deployment.

For more information about the components of Loggregator, see Loggregator Architecture.

Metric components

This section tells you about the components of the Log and Metric Agent Architecture that allow you to access metrics for your foundation.

These components allow you to access the same metrics available through the Loggregator Firehose with a pull based architecture. The Loggregator system uses a push-based model for forwarding metrics, in which all data is sent though the Firehose.

The following components of the Log and Metric Agent Architecture enable pull-based access to metrics:

- **Metrics Agent**:
  The Metrics Agent collects Loggregator V2 envelopes and makes them available on a Prometheus endpoint. The Metrics Agent performs a similar function to the Loggregator Agent in the Loggregator system.
- **Service Metrics Agent**:
  The Service Metrics Agent receives metrics from service instances on your Ops Manager deployment and makes them available on a Prometheus endpoint.
- **Metrics Discovery Registrar**:
  The Metrics Discovery Registrar publishes the location of the Prometheus endpoint defined by the Metrics Agent and Service Metrics Agent to NATs. This is helpful for configuring automation to scrape metric data from the endpoint. For more information about automating metric scraping, see Telegraf Reference Architecture (Beta).

Log components

This section tells you about the Log and Metric Agent Architecture components that allow you to access logs on your foundation.

These components are also a part of the Loggregator system. For more information about how these components function as part of the Loggregator system, see Loggregator Architecture.

The following components of the Log and Metric Agent Architecture enable access to logs:

- **Syslog Agent**:Syslog Agents run on Ops Manager component VMs and host VMs to collect and forward logs to configured syslog drains. This includes syslog drains for individual apps as well as aggregate drains for all apps in your foundation. You can specify the destination for logs as part of the individual syslog drain or in the TAS for VMs tile.

- **Aggregate Syslog Drain**:
  The aggregate syslog drain feature allows you to configure all Syslog Agents on your deployment to send logs to a single destination. You can use the aggregate syslog drain feature rather than the Loggregator Firehose to forward all logs for your deployment.
- **Log Cache**: Log Cache allows you to view logs and metrics over a specified period of time. The Log Cache includes API endpoints and a CLI plug-in to query and filter logs and metrics. To download the Log Cache CLI plugin, see Cloud Foundry Plugins. The Log Cache API endpoints are available by default. For more information about using the Log Cache API, see Log Cache on GitHub.

Logging

In this section:

- Configuring Logging in TAS for VMs
- App Logging in TAS for VMs
- Security Event Logging
- App Log Rate Limiting
- Customizing Platform Log Forwarding

Configuring Logging in TAS for VMs

This topic describes the types of logs that VMware Tanzu Application Service for VMs (TAS for VMs) generates. It also explains how to forward system logs to an external aggregator service, how to scale Loggregator component VMs to keep up with app log volume, and how to manage app traffic logging. For more information about Loggregator components, see Loggregator Architecture.

System Logs, App Logs, App Traffic Logs

TAS for VMs generates two types of logs, *system logs* from TAS for VMs components and *app logs* from hosted apps, as differentiated in the table below:

Log Type	Originate from	Follow format	Stream from	Can stream out to (configurable)	Visible to
System Logs	Platform components	Syslog standard	rsyslog agent	Component syslog drain	Operators
App Logs	Hosted apps	Format is up to the developer	Firehose[1]	External data platform, optionally via nozzles	Developers and Operators
		Converted to syslog standard	Syslog Agent	External syslog drain	

[1] The Loggregator Firehose also streams component metrics.

App traffic logs are system logs. When app containers communicate, or attempt to communicate, their host Diego Cells generate *app traffic logs*. App traffic logs are system logs, not app logs. These logs come from host Diego Cells, not apps, and they carry no information from within the app. App traffic logs only show app communication behavior, as detected from outside by the host Diego Cell.

Log Cache

Log Cache is a Loggregator feature that lets you filter and query app logs through a CLI plug-in or API endpoints. Cached app logs are available on demand; you do not need to stream them continuously.

Example Uses of the Log Cache CLI Plugin

To access cached logs with the Log Cache CLI plug-in, you must first download and install the plug-in.

To download the Log Cache CLI plug-in, see the cf CLI Plugins page on the Cloud Foundry website.

After you have installed the plug-in, the basic command to access cached app logs is:

```
cf tail OPTIONS APP-NAME-OR-ID
```

Where:

- `OPTIONS` are the flags you use to filter app logs.
- `APP-NAME-OR-ID` is the name or source ID of your app.

Some flags you can use Log Cache to filter app logs are:

- `--start-time`: Displays the start of the cache or the start of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `--end-time` to view logs within a time period.
- `--end-time`: Displays the end of the cache or the end of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `--start-time` to view logs within a time period.
- `--json`: Output logs in JSON.
- `--follow`: Append exported logs to `stdout`.

For more information on using the Log Cache CLI, see Log Cache CLI: Usage on GitHub.

Example Uses of the Log Cache API

The Log Cache API is hosted on TAS for VMs, and references your system domain to return responses. The root URL for API calls is `https://log-cache.SYSTEM-DOMAIN`, where `SYSTEM-DOMAIN` is your system domain.

The basic call to access and filter cached app logs is:

```
GET https://log-cache.SYSTEM-DOMAIN/v1/read/APP-ID
```

Where:

- `SYSTEM-DOMAIN` is your system domain.
- `APP-ID` is the source ID of your app.

Append the following parameters to your `GET` call to customize app logs:

- `start_time`: Displays the start of the cache or the start of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `end_time` to view logs within a time period.
- `end_time`: Displays the end of the cache or the end of a time period you specify. Results display as a UNIX timestamp, in nanoseconds. Pair with `start_time` to view logs within a time period.
- `envelope_types`: Filters by Envelope Type. The available filters are: `LOG`, `COUNTER`, `GAUGE`, `TIMER`, and `EVENT`. Set an envelope type filter to emit logs of only that type. Specify this parameter multiple times to include more types.
- `limit`: Sets a maximum number of envelopes to request. The max limit is 1000. This value defaults to 100.

More API parameters are available to customize retrieved app logs. For more information, see Log Cache: RESTful API Gateway.

Enable Syslog Forwarding

In the **System Logging** pane, you can configure system logging in TAS for VMs to forward log messages from TAS for VMs component VMs to an external service. VMware recommends forwarding logs to an external service for use in troubleshooting. If you do not fill these fields, platform logs are not forwarded but remain available on the component VMs and for download through Ops Manager.

> **Note:** This procedure explains how to configure system logging for TAS for VMs component VMs. To forward logs from Ops Manager tiles to an external service, you must also configure system logging in each tile. For more information about configuring system logging, see the documentation for the given tiles.

To configure the **System Logging** pane:

1. Select **System Logging**.
2. For **Address**, enter the hostname or IP address of the syslog server.
3. For **Port**, enter the port of the syslog server. The default port for a syslog server is 514.

   > **Note:** The host must be reachable from the TAS for VMs network and accept UDP or TCP connections. Ensure the syslog server listens on external interfaces.

4. For **Transport protocol**, select a transport protocol for log forwarding.
5. For **Encrypt syslog using TLS?**, select **Yes** to use TLS encryption when forwarding logs to a remote server.

    1. For **Permitted peer**, enter either the name or SHA1 fingerprint of the remote peer.
    2. For **Destination certificate**, enter the TLS CA certificate for the remote server.

6. Select the **Enable Cloud Controller security event logging** checkbox to include security events in the log stream. This feature logs all API requests, including the endpoint, user, source IP address, and request result, in the Common Event Format (CEF).

7. Enable the **Use TCP for file forwarding local transport** checkbox to transmit logs over TCP. This prevents log truncation, but may cause performance issues.

8. The **Do not forward debug logs** checkbox is enabled by default. To forward `DEBUG` syslog messages to an external service, deactivate the checkbox.

   > **Note:** Some TAS for VMs components generate a high volume of `DEBUG` syslog messages. Enabling the **Do not forward debug logs** checkbox prevents TAS for VMs components from forwarding the `DEBUG` syslog messages to external services. However, TAS for VMs still writes the messages to the local disk.

9. For **Custom rsyslog configuration**, enter a custom syslog rule. For more information about adding custom syslog rules, see Customizing Platform Log Forwarding.

10. For **Timestamp format for component logs**, select one of the following:

    > **Breaking Change:** If you change the timestamp format for logs, you might need to update any external monitoring configuration. For more information, see Timestamp Format for Component Logs Replaces Timestamp Format for Diego Logs in *TAS for VMs Breaking Changes*.

    - (Recommended) **Converge to human-readable RFC3339 format:** Every TAS for VMs component that supports RFC3339 timestamps uses this format in their logs. The RFC3339 format uses Coordinated Universal Time (UTC) and provides up to nine points of precision. For more information, see RFC3339. For a current list of components that use the RFC3339 timestamp format, see VMware Tanzu Application Service for VMs v2.10 Release Notes.

      For example:

      - 2019-11-21T22:16:18.750673404Z
      - 2019-11-21T22:16:18.750000000Z

    - **Maintain previous format:** TAS for VMs component logs use their previous timestamp format. VMware only recommends this option if you have scripts that require logs that use the previous timestamp format.

11. Configure how TAS for VMs emits app logs and app metrics for ingestion in your deployment. The options include:

    - Use existing Firehose integrations for app metric and app log ingestion.

- Preserve existing Firehose integrations for app metrics, but use an alternate method for app log ingestion.
- Deactivate all Firehose integrations and use alternate methods for both app log and app metric ingestion.

The following table provides the configuration procedures for each option. For more information about each field, see the Field Descriptions table below.

Option:	Configuration Procedure:
Use existing Firehose app log and metrics integrations	1. Select **Enable V1 Firehose**. 2. Select **Enable V2 Firehose**. 3. Deselect **Enable Log Cache Syslog Ingestion**. 4. Deselect **Disable logs in Firehose**. 5. (Optional) Configure **Aggregate log and metric drain destinations**.
Preserve existing Firehose integrations for app metrics, but use an alternate method for app log ingestion	⚠ **Caution:** Do not use this option if your deployment depends on partner log integrations. 1. Select **Enable V1 Firehose**. 2. Select **Enable V2 Firehose**. 3. Select **Enable Log Cache Syslog Ingestion**. 4. Select **Disable logs in Firehose**. 5. Configure **Aggregate log and metric drain destinations**.
Deactivate all Firehose integrations and use alternate methods for both app log and app metric ingestion	⚠ **Caution:** Do not use this option if your deployment depends on any of these: - Service tile metrics - Healthwatch or App Metrics - Partner log or metric integrations 1. Deselect **Enable V1 Firehose**. 2. Deselect **Enable V2 Firehose**. 3. Select **Enable Log Cache syslog ingestion**. 4. Deselect **Disable logs in Firehose**. 5. Configure **Aggregate log and metric drain destinations**. 6. Deactivate the Smoke Test errand. Otherwise, the TAS for VMs deployment fails. To deactivate errands, see Configure Errands.

Field Descriptions:

The following table provides more details on field values:

Field Name	Description
Enable V1 Firehose	Enabled by default. When enabled, app logs and app metrics flow to the Loggregator V1 Firehose.
Enable V2 Firehose	Enabled by default. When enabled, app logs and app metrics flow to the Loggregator V2 Firehose. If you deselect the checkbox To deactivate the V2 Firehose, you must also deactivate the V1 Firehose.
Enable Log Cache syslog ingestion	Deactivated by default. Configures Log Cache to ingest app logs and app metrics through the syslog server instead of the Reverse Log Proxy. If you deactivate the V1 Firehose, you must activate Log Cache syslog ingestion to receive service tile metrics.
Default loggregator drain metadata	Enabled by default. When enabled, TAS for VMs sends all metadata in app and aggregate syslog drains. Deactivating this option can reduce logging to external databases by up to 50 percent.
Disable logs in Firehose	Deselected by default. Prevents the Firehose from emitting app logs but still allows the Firehose to emit app metrics. Deactivating logs in Firehose helps reduce the load on TAS for VMs by allowing you to scale down Doppler and Traffic Controller VMs.
Aggregate log and metric drain destinations	Aggregate drains forward all app logs on your foundation to the endpoints that you provide in this field. Enter a comma-separated list of syslog endpoints for aggregate log drains. Specify the endpoints in the format: `syslog://HOSTNAME:PORT`. To use TLS for sending logs, specify `syslog-tls://HOSTNAME:PORT` or `https://HOSTNAME:PORT`. In TAS for VMs v2.10, aggregate drains no longer forward metrics by default. You can choose to forward app metrics and TAS for VMs component VM metrics by adding `?include-metrics-deprecated=true` to the endpoints. For example, `syslog://myhost:514?include-metrics-deprecated=true`. **Breaking Change:** If you are upgrading from TAS for VMs v2.9 and want the aggregate drain to continue forwarding app and component VM metrics, you must add `?include-metrics-deprecated=true` to the endpoint. For more information, see Aggregate Syslog Drains Contain Logs Only in the *TAS for VMs Breaking Changes*.

12. To send BOSH system metrics to your logging endpoint more or less frequently, change the value for **System metrics scrape interval**. The default value is `1m`, which sends BOSH system metrics to your logging endpoint once per minute. To send metrics more or less frequently, change the value. For example, enter `2m` to send metrics every two minutes or `10s` to send metrics every ten seconds. The minimum recommended value is five seconds.

13. Click **Save**.

To configure Ops Manager for system logging, see Settings Page in *Using the Ops Manager Interface*.

Include Container Metrics in Syslog Drains

Developers can monitor container metrics over the syslog protocol using the CF Drain CLI plugin. With the CF Drain CLI plug-in, you can use the Cloud Foundry Command Line Interface (cf CLI) tool to set the app container to deliver container metrics to a syslog drain. Developers can then monitor the app container based on those metrics.

For more information, see Including Container Metrics in Syslog Drains in *App Logging in TAS for VMs*.

Scale Loggregator

Apps constantly generate app logs and TAS for VMs platform components constantly generate component metrics. The Loggregator system combines these data streams and handles them as follows. For more information, see Loggregator Architecture.

- The Loggregator agent running on each component or app VM collects and sends this data out to Doppler components.
- Doppler components temporarily buffer the data before periodically forwarding it to the Traffic Controller. When the log and metrics data input to a Doppler exceeds its buffer size for a given interval, data can be lost.
- The Traffic Controller serves the aggregated data stream through the Firehose WebSocket endpoint.

Follow the instructions below to scale the Loggregator system. For guidance on monitoring and capacity planning, see Monitoring TAS for VMs.

Add Component VM Instances

To add component VM instances for Loggregator components:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **Resource Config**.
4. Increase the number in the **Instances** column of the component you want to scale. You can add instances for the following Loggregator components:
    - **Loggregator Traffic Controller**

      > **Note:** The Reverse Log Proxy (RLP) BOSH job is co-located on the Traffic Controller VM. If you want to scale Loggregator to handle more logs for syslog drains, you can add instances of the Traffic Controller. For more information, see Loggregator Architecture.

      > **Note:** The BOSH System Metrics Forwarder job is co-located on the Traffic Controller VM. If you want to scale Loggregator to handle more BOSH component metrics, you can add instances of the Traffic Controller. For more information, see Related BOSH Components in *Loggregator Architecture*.

    - **Doppler Server**

5. Click **Save**.
6. Return to the Ops Manager Installation Dashboard.
7. Click **Review Pending Changes**.
8. Click **Apply Changes**.

App Traffic Logging

App traffic logging generates logs when app containers communicate with each other directly, or attempt to communicate, as allowed by container-to-container networking (C2C) policies and App Security Groups (ASGs). For more information about C2C policies, see Container-to-Container Networking versus ASGs in *Container-to-Container Networking*. For more information about ASGs, see App Security Groups.

App traffic logging lets network security teams audit C2C traffic, by seeing allowed and denied packets, without needing access to the Cloud Controller or the apps themselves.

Enable App Traffic Logging

To enable app traffic logging:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select the **Networking** pane.
4. Enable the **Log traffic for all accepted and denied app packets** check box.
5. Click **Save**.
6. Return to the Ops Manager Installation Dashboard.
7. Click **Review Pending Changes**.
8. Click **Apply Changes**.

App Logging Behavior

App traffic logging generates log messages as follows:

- **TCP traffic:** Logs the first packet of every new TCP connection.
- **UDP traffic:** Logs UDP packets sent and received, up to a maximum per-second rate for each container. Set this rate limit in the **UDP logging interval** field in the **Networking** pane of the TAS for VMs tile. The default rate limit is `100`.
- **Packets denied:** Logs packets blocked by either a container-specific networking policy or by ASG rules applied across the space, org, or deployment. Logs packet denials up to a maximum per-second rate for each container, set in the **Denied logging interval** field in the **Networking** pane of the TAS for VMs tile. The default rate is `1`. For more information about container-specific networking policies, see Policies in *Container-to-Container Networking*.

App Traffic Log Format and Contents

App traffic logs are formatted as described in the silk-release Traffic logging documentation, following the `iptables-logger` format but without line breaks.

For example, the first part of an app traffic log line looks like: `{"timestamp": "2020-06-23T11:36:01.710452019Z", "source": "cfnetworking.iptables", "message": "cfnetworking.iptables.ingress-allowed", "log_level": 1, "data": { "destination": { "container_id": "d5978989-1401-49ff-46cd-33e5","app_guid": "bc6f229d-5e4a-4c41-a63f-e8795496c283",`.

Each log message includes:

- A timestamp, either in RFC3339 format or in epoch format. The format depends upon the configuration of the **Timestamp format for component logs** in the **System Logging** pane of the TAS for VMs tile
- The GUID for the source or destination app that sent or was designated to receive the packet
- The protocol of the communication, `TCP` or `UDP`
- GUIDs for the container, space, and org running the source or destination app
- IP addresses and ports for both source and destination apps
- A `message` field recording whether the packet was allowed or denied, with one of the following four possibilities:
  - `ASG allowed packet to exit source app container`
  - `C2C policy allowed packet to enter destination app container`
  - `ASG prevented packet from exiting source app container`
  - `C2C policy prevented packet from entering destination app container`
- Additional information described in the silk-release documentation

Denied Packet Causes

You can determine whether a denied-packet log resulted from a container networking policy or an ASG rule as follows:

- **Container networking policy:** Log `message` string includes `ingress-denied` and `packet direction` is `ingress`.
- **ASG rule:** Log `message` string includes `egress-denied` and `packet direction` is `egress`.

Global vs. ASG-Level App Traffic Logging

TAS for VMs supports two mechanisms for enabling app traffic logging. Enabling the **Log traffic for all accepted and denied app packets** check box in the **Networking** pane of the TAS for VMs tile enables app traffic logging globally for all ASGs and container policies. Setting the `log` property of an ASG to `true` enables app traffic logging at the individual ASG level. For more information about setting the `log` property of an ASG to `true`, see The Structure and Attributes of ASGs in *App Security Groups*.

Because these two mechanisms operate independently, TAS for VMs generates duplicate logs when app traffic logging is enabled globally and an ASG's `log` property is set to `true`. To avoid duplicate logs, VMware recommends setting the `log` property to `false` for all ASGs, or leaving it out entirely, when app traffic logging is enabled globally. For more information, see App Traffic Logging.

To focus on specific ASGs for log analysis, VMware recommends enabling app traffic logs globally and using a logging platform to filter traffic logs by ASG, rather than setting `log` at the individual ASG level.

App Logging in TAS for VMs

Loggregator, the VMware Tanzu Application Service for VMs (TAS for VMs) component responsible for logging, provides a stream of log output from your app and from TAS for VMs system components that interact with your app during updates and execution.

Overview

By default, Loggregator streams logs to your terminal. If you want to persist more than the limited amount of logging information that Loggregator can buffer, you can drain logs to a third-party log management service. For more information, see Third-Party Log Management Services.

TAS for VMs gathers and stores logs in a best-effort manner. If a client cannot consume log lines quickly enough, the Loggregator buffer may need to overwrite some lines before the client has consumed them. A syslog drain or a CLI tail can usually keep up with the flow of app logs.

Contents of a log line

Every log line contains four fields:

- Timestamp
- Log type (origin code)
- Channel: either `OUT`, for logs emitted on `stdout`, or `ERR`, for logs emitted on `stderr`
- Message

Loggregator assigns the timestamp when it receives log data. The log data is opaque to Loggregator, which only puts it in the message field of the log line. Apps or system components sending log data to Loggregator may include their own timestamps, which then appear in the message field.

Many TAS for VMs components include the option to use human-readable timestamps in log messages. To configure TAS for VMs to use human-readable timestamps when possible, see Configure System Logging in *Configuring TAS for VMs*. For more information about human-readable timestamps, see RFC 3339.

Origin codes distinguish the different log types. Origin codes from system components have three letters. The app origin code is `APP` followed by slash and a digit that indicates the app instance.

Many frameworks write to an app log that is separate from `stdout` and `stderr`. This is not supported by Loggregator. Your app must write to `stdout` or `stderr` for its logs to be included in the Loggregator stream. Check the buildpack your app uses to determine whether it automatically ensures that your app correctly writes logs to `stdout` and `stderr` only. Some buildpacks do this, and some do not.

Log types and their messages

Different types of logs have different message formats, as shown in the examples below. The digit appended to the code indicates the instance index: 0 is the first instance, 1 is the second, and so on.

API

Users make API calls to request changes in app state. Cloud Controller, the TAS for VMs component responsible for the API, logs the actions that Cloud Controller takes in response.

For example:

```
2016-06-14T14:10:05.36-0700 [API/0]      OUT Updated app with guid cdabc600-0b73-48e1-b7d2-26af2c63f933 ({"name"=>"spring-music", "instances"=>1, "memory"=>512, "environment_json"=>"PRIVATE DATA HIDDEN"})
```

STG

The Diego Cell or the Droplet Execution Agent emits STG logs when staging or restaging an app. These actions implement the desired state requested by the user. After the droplet has been uploaded, STG messages end and CELL messages begin. For STG, the instance index is almost always 0.

For example:

```
2016-06-14T14:10:27.91-0700 [STG/0]      OUT Staging...
```

RTR

The Gorouter emits RTR logs when it routes HTTP requests to the app. Gorouter messages include the app name followed by a Gorouter timestamp and selections from the HTTP request.

For example:

```
2016-06-14T10:51:32.51-0700 [RTR/1]      OUT www.example.com - [14/06/2016:17:51:32.459 +0000] "GET /user/ HTTP/1.1" 200 0 103455 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.123 Safari/537.30" 192.0.2.132:46359 x_forwarded_for:"198.51.100.120" x_forwarded_proto:"http" vcap_request_id:9365d216-623a-45cb-6ef6-eba495c19fa8 response_time:0.059468637 app_id:79cc58aa-3737-43ae-ac71-39a2843b5178
```

Zipkin trace logging

If Zipkin trace logging is enabled in TAS for VMs, then Gorouter access log messages contain Zipkin HTTP headers.

The following is an example access log message containing Zipkin headers:

```
2016-11-23T16:04:01.49-0800 [RTR/0]      OUT www.example.com - [24/11/2016:0
0:04:01.227 +0000] "GET / HTTP/1.1" 200 0 109 "-" "curl/7.43.0" 10.0.2.150:40
70 10.0.48.66:60815 x_forwarded_for:"198.51.100.120" x_forwarded_proto:"http"
vcap_request_id:87f9d899-c7a4-46cd-7b76-4ec35ce9921b response_time:0.26300096
6 app_id:8e5d6451-b369-4423-bce8-3a7a9e479dbb app_index:0 x_b3_traceid:"2d561
0bf5e0f7241" x_b3_spanid:"2d5610bf5e0f7241" x_b3_parentspanid:"-"
```

For more information about Zipkin tracing, see Zipkin Tracking in HTTP Headers in *HTTP Routing*.

LGR

Loggregator emits LGR to indicate problems with the logging process. Examples include `can't reach syslog drain url` and `dropped log messages due to high rate`.

APP

Every app emits logs according to choices by the developer.

For example:

```
2016-06-14T14:10:15.18-0700 [APP/0]      OUT Exit status 0
```

SSH

The Diego Cell emits SSH logs when a user accesses an app container through SSH by using the Cloud Foundry Command Line Interface (cf CLI) `cf ssh` command.

For example:

```
2016-06-14T14:16:11.49-0700 [SSH/0]      OUT Successful remote access by 192.
0.2.33:7856
```

For more information about the `cf ssh` command, see ssh in the Cloud Foundry CLI Reference Guide.

CELL

The Diego Cell emits CELL logs when it starts or stops the app. These actions implement the desired state requested by the user. The Diego Cell also emits messages when an app crashes.

For example:

```
2016-06-14T13:44:38.14-0700 [CELL/0]     OUT Successfully created container
```

The Diego Cell also emits a CELL log message when an app instance exceeds the `max_log_lines_per_second` limit, which is configured on the platform.

For example:

```
2020-01-13T16:12:25.86-0800 [APP/PROC/WEB/0] OUT app instance exceeded log ra
te limit (100 log-lines/sec) set by platform operator
```

This message only appears if the limit has been set by a platform operator. For more information, see Configure App Log Rate Limit (Beta) in *VMware Tanzu Application Service for VMs v2.10 Release Notes*.

Writing to the log from your app

Your app must write logs to `stderr` or `stdout`. Both are typically buffered, and you should flush the buffer before delivering the message to Loggregator.

You can write log messages to `stderr` or `stdout` synchronously. This approach is mainly used for debugging because it affects app performance.

Alternatively, you can write log messages to `stderr` or `stdout` synchronously. This approach is mainly used for debugging because it may affect app performance.

Including container metrics in syslog drains

By default, app logs are included in syslog drains. Syslog Agents forward logs to configured syslog drains and Loggregator. For more information about Syslog Agents, see Loggregator Architecture and Components in *Loggregator Architecture*.

Viewing logs in the command line interface

Use the cf CLI `cf logs` command to view logs. You can tail, dump, or filter log output. For more information about the `cf logs` command, see cf logs in the Cloud Foundry CLI Reference Guide.

Tailing logs

To stream Loggregator output to the terminal, run `cf logs APP-NAME` and replace `APP-NAME` with the name of your app.

For example:

```
$ cf logs spring-music
Connected, tailing logs for app spring-music in org example / space developme
nt as admin@example.com...

2016-06-14T15:16:12.70-0700 [RTR/4]      OUT www.example.com - [14/06/2016:2
2:16:12.582 +0000] "GET / HTTP/1.1" 200 0 103455 "-" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.26
61.102 Safari/537.36" 192.0.2.206:27743 x_forwarded_for:"203.0.113.222" x_for
warded_proto:"http" vcap_request_id:bd3e6ed1-5dd0-43ab-70ed-5d232b577b09 resp
onse_time:0.12050583 app_id:79bb58ab-3737-43be-ac70-39a2843b5177
2016-06-14T15:16:20.06-0700 [RTR/4]      OUT www.example.com - [14/06/2016:2
2:16:20.034 +0000] "GET /test/ HTTP/1.1" 200 0 6879 "http://www.example.com/"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/50.0.2661.102 Safari/537.36" 192.0.2.206:2228 x_forwarded_f
or:"203.0.113.222" x_forwarded_proto:"http" vcap_request_id:a31f0b1d-3827-4b8
f-57e3-6f42d189f025 response_time:0.033311281 app_id:79bb58aa-3747-43be-ac70-
39a3843b5178
2016-06-14T15:16:22.44-0700 [RTR/4]      OUT www.example.com - [14/06/2016:2
2:17:22.415 +0000] "GET /test5/ HTTP/1.1" 200 0 5461 "http://www.example.com/
test5" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36" 192.0.2.206:2228 x_forw
arded_for:"203.0.113.322" x_forwarded_proto:"http" vcap_request_id:5d6855a2-4
a79-4432-7927-de8215f5a2c7 response_time:0.029211609 app_id:79bb58aa-3737-43b
b-ac70-39a2943b5178
    ...
```

Use **Ctrl-C** (^C) to exit the real-time stream.

Dumping logs

To display all of the lines in the Loggregator buffer, run `cf logs APP-NAME --recent`, where `APP-NAME` is the name of your app.

Filtering logs

To view some subset of log output, run `cf logs APP-NAME` in conjunction with filtering commands of your choice. Replace `APP-NAME` with the name of your app. In the example below, `grep -v RTR` excludes all Gorouter logs:

```
$ cf logs spring-music --recent | grep -v RTR
2016-06-14T14:10:05.36-0700 [API/0]      OUT Updated app with guid cdabc604-0
b73-47e1-a7d5-24af2c63f723 ({"name"=>"spring-music", "instances"=>1, "memor
y"=>512, "environment_json"=>"PRIVATE DATA HIDDEN"})
2016-06-14T14:10:14.52-0700 [APP/0]      OUT - Gracefully stopping, waiting f
or requests to finish
2016-06-14T14:10:14.52-0700 [CELL/0]     OUT Exit status 0
2016-06-14T14:10:14.54-0700 [APP/0]      OUT === puma shutdown: 2016-06-14 2
1:10:14 +0000 ===
2016-06-14T14:10:14.54-0700 [APP/0]      OUT - Goodbye!
2016-06-14T14:10:14.56-0700 [CELL/0]     OUT Creating container
    ...
```

Log ordering

Ensuring log ordering in drains can be an important consideration for both operators and developers.

- Diego uses a nanosecond-based timestamp that can be ingested properly by Splunk. For more information, see TIME_FORMAT and subseconds in the Splunk documentation.
- The Elastic Stack can ingest the nanosecond timestamps but only supports millisecond precision, so timestamps are truncated. For more information, see the Date datatype in the Elasticsearch documentation and the Date type has not enough precision for the logging use case GitHub issue.

If you are developing a client that displays a stream of TAS for VMs logs to users, you can order the logs to improve the debugging experience for your user. The following are general tips for ordering logs:

- For CLIs, batch the logs and display the logs you have in that timeframe, sorted by timestamp.
- For web clients, use dynamic HTML to insert older logs into the sorting as they appear. This creates complete, ordered logs.
- Java app developers might want to convert stack traces into a single log entity. To simplify log ordering for Java apps, use the multi-line Java message workaround to convert your multi-line stack traces into a single log entity. For more information, see Multi-line Java message workaround.

  By modifying the Java log output, you can force your app to reformat stack trace messages, replacing newline characters with a token. Set your log parsing code to replace that token with newline characters again to display the logs properly in Kibana.

Security event logging

Learn how to activate and interpret security event logging for the Cloud Controller, the User Account and Authentication (UAA) server, and CredHub. You can use these logs to retrieve information about a subset of requests to the Cloud Controller, UAA server, and CredHub for security or compliance purposes.

Cloud Controller logging

The Cloud Controller logs security events to syslog. You must configure a syslog drain to forward your system logs to a log management service.

For more information, see Configuring System Logging in TAS for VMs.

Formatting log entries

Cloud Controller logs security events in the Common Event Format (CEF). CEF specifies the following format for log entries:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

Entries in the Cloud Controller log use the following format:

```
CEF:CEF_VERSION|cloud_foundry|cloud_controller_ng|CC_API_VERSION|
SIGNATURE_ID|NAME|SEVERITY|rt=TIMESTAMP suser=USERNAME suid=USER_GUID
cs1Label=userAuthenticationMechanism cs1=AUTH_MECHANISM
cs2Label=vcapRequestId cs2=VCAP_REQUEST_ID request=REQUEST
requestMethod=REQUEST_METHOD cs3Label=result cs3=RESULT
cs4Label=httpStatusCode cs4=HTTP_STATUS_CODE src=SOURCE_ADDRESS
dst=DESTINATION_ADDRESS cs5Label=xForwardedFor cs5=X_FORWARDED_FOR_HEADER
```

Refer to the following list for a description of the properties shown in the Cloud Controller log format:

- `CEF_VERSION`: The version of CEF used in the logs.
- `CC_API_VERSION`: The current Cloud Controller API version.

- `SIGNATURE_ID`: The method and path of the request. For example, `GET /v2/app:GUID`.
- `NAME`: The same as `SIGNATURE_ID`.
- `SEVERITY`: An integer that reflects the importance of the event.
- `TIMESTAMP`: The number of milliseconds since the Unix epoch.
- `USERNAME`: The name of the user who originated the request.
- `USER_GUID`: The GUID of the user who originated the request.
- `AUTH_MECHANISM`: The user authentication mechanism. This can be `oauth-access-token`, `basic-auth`, or `no-auth`.
- `VCAP_REQUEST_ID`: The VCAP request ID of the request.
- `REQUEST`: The request path and parameters. For example, `/v2/info?MY-PARAM=VALUE`.
- `REQUEST_METHOD`. The method of the request. For example, `GET`.
- `RESULT`: The meaning of the HTTP status code of the response. For example, `success`.
- `HTTP_STATUS_CODE`. The HTTP status code of the response. For example, `200`.
- `SOURCE_ADDRESS`: The IP address of the client who originated the request.
- `DESTINATION_ADDRESS`: The IP address of the Cloud Controller VM.
- `X_FORWARDED_FOR_HEADER`: The contents of the X-Forwarded-For header of the request. This is empty if the header is not present.

Examples of log entries

The following list provides several example requests with the corresponding Cloud Controller log entries.

- An anonymous `GET` request:

  ```
  CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/info|GET
  /v2/info|0|rt=1460690037402 suser= suid= request=/v2/info
  requestMethod=GET src=127.0.0.1 dst=192.0.2.1
  cs1Label=userAuthenticationMechanism cs1=no-auth cs2Label=vcapRequestId
  cs2=c4bac383-7cc9-4d9f-b1c0-1iq8c0baa000 cs3Label=result cs3=success
  cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor
  cs5=198.51.100.1
  ```

- A `GET` request with basic authentication:

  ```
  CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/syslog_drain_urls|GET
  /v2/syslog_drain_urls|0|rt=1460690165743 suser=bulk_api suid=
  request=/v2/syslog_drain_urls?batch_size\=1000 requestMethod=GET
  src=127.0.0.1 dst=192.0.2.1 cs1Label=userAuthenticationMechanism
  cs1=basic-auth cs2Label=vcapRequestId cs2=79187189-e810-33dd-6911-b5d015bbc999
  ::eat1234d-4004-4622-ad11-9iaai88e3ae9 cs3Label=result cs3=success
  cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor cs5=198.51.100.1
  ```

- A `GET` request with OAuth access token authentication:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/routes|GET
/v2/routes|0|rt=1460689904925 suser=admin suid=c7ca208f-8a9e-4aab-
92f5-28795f86d62a request=/v2/routes?inline-relations-depth\=1&q\=
host%3Adora%3Bdomain_guid%3B777-1o9f-5f5n-i888-o2025cb2dfc3
requestMethod=GET src=127.0.0.1 dst=192.0.2.1
cs1Label=userAuthenticationMechanism cs1=oauth-access-token
cs2Label=vcapRequestId cs2=79187189-990i-8930-52b2-9090b2c5poz0
::5a265621-b223-4520-afae-ab7d0ee7c75b cs3Label=result cs3=success
cs4Label=httpStatusCode cs4=200 cs5Label=xForwardedFor cs5=198.51.100.1
```

- A `GET` request that results in a 404 error:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|GET /v2/apps/7f310103-
39aa-4a8c-b92a-9ff8a6a2fa6b|GET /v2/apps/7f310103-39aa-4a8c-b92a-
9ff8a6a2fa6b|0|rt=1460691002394 suser=bob suid=a00i2026-55io-3983-
555o-40e611410aec request=/v2/apps/7f310103-39aa-4a8c-b92a-9ff8a6a2fa6b
requestMethod=GET src=127.0.0.1 dst=192.0.2.1
cs1Label=userAuthenticationMechanism cs1=oauth-access-token cs2Label=vcapReques
tId
cs2=49f21579-9eb5-4bdf-6e49-e77d2de647a2::9f8841e6-e04a-498b-b3ff-d59cfe7cb7ea
cs3Label=result cs3=clientError cs4Label=httpStatusCode cs4=404
cs5Label=xForwardedFor cs5=198.51.100.1
```

- A `POST` request that results in a 403 error:

```
CEF:0|cloud_foundry|cloud_controller_ng|2.54.0|POST /v2/apps|POST
/v2/apps|0|rt=1460691405564 suser=bob suid=4f9a33f9-fb13-4774-a708-
f60c939625cd request=/v2/apps?async\=true requestMethod=POST
src=127.0.0.1 dst=192.0.2.1 cs1Label=userAuthenticationMechanism
cs1=oauth-access-token cs2Label=vcapRequestId cs2=booc03111-9999-4003-88ab-
20i9r33333ou::5a4993fc-722f-48bc-aff4-99b2005i9bb5 cs3Label=result
cs3=clientError cs4Label=httpStatusCode cs4=403 cs5Label=xForwardedFor
cs5=198.51.100.1
```

UAA logging

UAA logs security events to a file located at `/var/vcap/sys/log/uaa/uaa.log` on the UAA virtual machine (VM). Because these logs are automatically rotated, you must configure a syslog drain to forward your system logs to a log management service.

For more information, see Configuring System Logging in TAS for VMs.

Log Events

UAA logs identify the following categories of events:

- Authorization and Password Events
- SCIM Administration Events
- Token Events
- Client Administration Events
- UAA Administration Events

To learn more about the names of the events included in these categories and the information they record in the UAA logs, see User Account and Authentication Service Audit Requirements.

Examples of log entries

The following sections provide several example requests with the corresponding UAA log entries.

Successful user authentication

```
Audit: TokenIssuedEvent ('["openid","scim.read","uaa.user",
"cloud_controller.read","password.write","cloud_controller.write",
"scim.write"]'): principal=a42026d6-5533-1884-eef2-838abcd0i3e3,
origin=[client=cf, user=bob], identityZoneId=[uaa]
```

- This entry records a `TokenIssuedEvent`.
- UAA issued a token associated with the scopes `"openid","scim.read","uaa.user","cloud_controller.read","password.write","cloud_controller.write","scim.write"` to the user `bob`.

Failed user authentication

```
Audit: UserAuthenticationFailure ('bob@example.com'):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[remoteAddress=198.51.100.1, clientId=cf],
identityZoneId=[uaa]
```

- This entry records a `UserAuthenticationFailure`.
- The user `bob@example.com` originating at `198.51.100.1` failed to authenticate.

Successful user creation

```
Audit: UserCreatedEvent ('["user_id=61965469-c821-
46b7-825f-630e12a51d6c","username=bob@example.com"]'):
principal=91220262-d901-44c0-825f-633i33b55d6c,
origin=[client=cf, user=admin, details=(198.51.100.1,
tokenType=bearertokenValue=<TOKEN>,
sub=20i03423-dd8e-33e1-938d-e9999e30f500,
iss=https://uaa.example.com/oauth/token)], identityZoneId=[uaa]
```

- This entry records a `UserCreatedEvent`.
- The `admin` user originating at `198.51.100.1` created a user named `bob@example.com`.

Successful user deletion

```
Audit: UserDeletedEvent ('["user_id=61965469-c821-
46b7-825f-630e12a51d6c","username=bob@example.com"]'):
principal=61965469-c821-46b7-825f-630e12a51d6c,
origin=[client=admin, details=(remoteAddress=198.51.100.1,
tokenType=bearertokenValue=<TOKEN>,
sub=admin, iss=https://uaa.example.com/oauth/token)], identityZoneId=[uaa]
```

- This entry records a `UserDeletedEvent`.
- The `admin` user originating at `198.51.100.1` deleted a user named `bob@example.com`.

CredHub logging

CredHub logs security events to a file located at `/var/vcap/sys/log/credhub/credhub_security_events.log` on the CredHub VM. Because these logs are automatically rotated, you must configure a syslog drain to forward your system logs to a log management service.

For more information, see Configuring System Logging in TAS for VMs.

Format for Log Entries

CredHub logs security events in the Common Event Format (CEF). CEF specifies the following format for log entries:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension
```

Entries in the CredHub log use the following format:

```
CEF:0|cloud_foundry|credhub|CREDHUB_SERVER_VERSION|
SIGNATURE_ID|NAME|0|rt=TIMESTAMP suser=USERNAME suid=USER_GUID
cs1Label=userAuthenticationMechanism cs1=AUTH_MECHANISM
request=REQUEST requestMethod=REQUEST_METHOD
cs3Label=versionUuid cs3=VERSION_UUID
cs4Label=httpStatusCode cs4=HTTP_STATUS_CODE src=SOURCE_ADDRESS dst=DESTINATION_ADDRESS
cs2Label=resourceName cs2=RESOURCE_NAME
cs5Label=resourceUuid cs5=RESOURCE_UUID deviceAction=OPERATION
cs6Label=requestDetails cs6=REQUEST_DETAILS
```

Refer to the following list for a description of the properties shown in the CredHub log.

- `CEF_VERSION`: The version of CEF used in the logs.
- `CREDHUB_SERVER_VERSION`: The current CredHub server version.
- `SIGNATURE_ID`: The method and path of the request. For example, `GET /v2/app:GUID`.
- `NAME`: The same as `SIGNATURE_ID`.
- `TIMESTAMP`: The number of milliseconds since the Unix epoch.
- `USERNAME`: The name of the user who originated the request, as defined by UAA. In the case of mTLS, no-auth, or not defined, this value is empty.
- `USER_GUID`: The “actor” identifier. For example, `mtls-app:GUID`. If there is no authenticated user, this value is empty.
- `AUTH_MECHANISM (cs1)`: The user authentication mechanism. This can be `oauth-access-token`, `mtls-auth`, or `no-auth`.
- `REQUEST`: The request path and parameters. For example, `/v2/info?MY-PARAM=VALUE`.
- `REQUEST_METHOD`: The method of the request. For example, `GET`.

- `HTTP_STATUS_CODE (cs4)`. The HTTP status code of the response. For example, `200`.
- `SOURCE_ADDRESS`: The IP address of the client who originated the request.
- `DESTINATION_ADDRESS`: The IP address of the CredHub VM.
- `RESOURCE_NAME (cs2)`: The credential path name. For example, `/my/awesome/credential`.
- `RESOURCE_UUID (cs5)`: The top-level credential UUID. This is not the credential version.
- `VERSION_UUID (cs3)`: The credential version UUID.
- `OPERATION (deviceAction)`: The device action. The possible operations include the following:
  - `get`
  - `set`
  - `generate`
  - `regenerate`
  - `bulk-regenerate`
  - `delete`
  - `find`
  - `get-permissions`
  - `add-permission`
  - `delete-permission`
  - `interpolate`
  - `info`
  - `version`
  - `health`
  - `key-usage`
  - `update-transitional-version`
- `REQUEST_DETAILS (cs6)`: A JSON blob that differs per request.

Note: The CredHub logs include descriptions for each custom label. For example, the logs include `cs2Label=resourceName` to define the `cs2` label. The value for `resourceName` appears in the log next to `cs2=/path/to/credential`.

Example Log Entries

The following sections provide several example requests with the corresponding CredHub log entries.

Accessing a credential

```
CEF:0|cloud_foundry|credhub|2.0.0-beta.28|GET /api/v1/data|
GET /api/v1/data|0|rt=1530901816757
suser=app:3c538393-d192-4e23-8c83-456654b3fa6c
suid=mtls-app:3c538393-d192-4e23-8c83-456654b3fa6c
cs1Label=userAuthenticationMechanism
cs1=mutual_tls request=/api/v1/data?path=0b353167-0d5a-48c7-5036-7eaa
requestMethod=GET
cs3Label=versionUuid cs3=null
cs4Label=httpStatusCode cs4=200 src=10.0.0.1 dst=credhub.service.cf.internal
cs2Label=resourceName cs2=null
cs5Label=resourceUuid cs5=null deviceAction=FIND
cs6Label=requestDetails
cs6={"nameLike":null,"path":"0b353167-0d5a-48c7-5036-7eaa","paths":null}
```

- A user authenticated to the CredHub instance using `mutual_tls` from `10.0.0.1`.
- The authenticated user accessed a CredHub credential.

Setting a credential

```
CEF:0|cloud_foundry|credhub|2.0.0-beta.28|PUT /api/v1/data|
PUT /api/v1/data|0|rt=1530901097921
suser=cc_service_key_client suid=uaa-client:cc_service_key_client
cs1Label=userAuthenticationMechanism
cs1=uaa request=/api/v1/data requestMethod=PUT
cs3Label=versionUuid cs3=32b3d5bf-463a-4045-a6b5-65c97e61059a
cs4Label=httpStatusCode cs4=200 src=10.0.0.1 dst=credhub.service.cf.internal
cs2Label=resourceName cs2=/1530901097842989110
cs5Label=resourceUuid cs5=ccda25c5-a40a-4512-b6f5-a42f8c3b5c4c deviceAction=SET
cs6Label=requestDetails
cs6={"name":"/1530901097842989110","type":"json","mode":null,
"additionalPermissions":[{"actor":"uaa-client:cc_service_key_client",
"allowedOperations":["READ","WRITE","DELETE","WRITE_ACL","READ_ACL"]}]}
```

- A user authenticated to the CredHub instance with UAA from `10.0.0.1`.
- The authenticated user set a CredHub credential.

Generating a credential

```
CEF:0|cloud_foundry|credhub|2.0.0-beta.28|POST /api/v1/data|
POST /api/v1/data|0|rt=1530901403532
suser=app:3c538393-d192-4e23-8c83-456654b3fa6c
suid=mtls-app:3c538393-d192-4e23-8c83-456654b3fa6c
cs1Label=userAuthenticationMechanism
cs1=mutual_tls request=/api/v1/data requestMethod=POST
cs3Label=versionUuid cs3=8c21b7b3-d4bf-4d8a-a0c5-64c8c207cb40
cs4Label=httpStatusCode cs4=200 src=10.0.0.1 dst=credhub.service.cf.internal
cs2Label=resourceName cs2=/0b353167-0d5a-48c7-5036-7eaa/2
cs5Label=resourceUuid cs5=1d55cff3-5264-434c-a127-0810f341cb2b deviceAction=GENERATE
cs6Label=requestDetails cs6={"name":"/my/awesome/credential","type":"password",
"mode":null,"additionalPermissions":[{"actor":"mtls-app:3c538393-d192-4e23-8c83-456654
b3fa6c",
"allowedOperations":["READ","WRITE","DELETE","WRITE_ACL","READ_ACL"]}]}
```

- A user authenticated to the CredHub instance using `mutual_tls` from `10.0.0.1`.

- The authenticated user generated a password credential named `/my/awesome/credential`.

Limiting your App Log Rate in Cloud Foundry

Learn about limiting your app log rate for apps in VMware Tanzu Application Service for VMs.

Log rate limiting limits the number of logs that can be sent to an app.

App log rate limiting is deactivated by default. VMware recommends activating this feature to prevent app instances from overloading the Loggregator Agent with logs, so the Loggregator Agent does not drop logs for other app instances on the same Diego Cell.

Using app log rate limiting can also do the following:

- Prevent apps from reporting inaccurate app metrics in the Cloud Foundry Command Line Interface (cf CLI), which can happen if Log Cache evicts metrics from its cache in order to store large volumes of logs.
- Limit the CPU usage of logging agents on the Diego Cell VM.

You can allow log rate limits on a per-app basis in bytes per second.

App Log Rate limiting in Bytes Per Second

In TAS for VMs, you can limit the number of bytes each app instance can generate per second.

You can configure app log rate limiting in bytes per second on a per-app basis through either the app manifest or the cf CLI. Additionally, you can enforce the log rate limit you configure for all apps that are deployed within a space or org by specifying the log rate limit in the quota plan for the space or org. For more information, see Creating and Modifying Quota Plans.

Determining the ideal App Log Rate Limit

The ideal app logging rate for a deployment depends on characteristics such as VM sizes and the number and type of apps in TAS for VMs. VMware recommends using at minimum the default limit of `100` lines/second.

When you enable app log rate limiting, Diego applies the rate limit to each app instance. For example, if there are five instances of an app running, Diego does not sum the logging rates of all five instances when determining if the rate limit has been exceeded. Instead, Diego evaluates the logging rate of each individual app instance and only limits instances that exceed the rate limit.

What happens when App Instances exceed the App Log Rate Limit

When an app instance exceeds the configured rate limit, Diego drops the app logs that exceed the per-second rate you configured through the **App log rate limit** section in the **App Containers** pane of the TAS for VMs tile. When this happens, you see a message indicating that logs are being dropped.

For more information about how Diego rate limits app logs, see the Go documentation.

How Diego Cells determine when an App Instance exceeds the App Log Rate Limit

The Diego Cell containing the app instance emits the `AppInstanceExceededLogRateLimitCount` counter metric when it exceeds the rate limit, similar to the following example:

```
origin:"rep" eventType:CounterEvent timestamp:1582582740243576212 deploymen
t:"cf" job:"diego-cell" index:"0e98fd00-47b2-4589-94f0-385f78b3a04d" ip:"10.
0.1.12" tags:<key:"instance_id" value:"0e98fd00-47b2-4589-94f0-385f78b3a04d"
> tags:<key:"source_id" value:"rep" > counterEvent:<name:"AppInstanceExceeded
LogRateLimitCount" delta:1 total:206 >
```

Each Diego Cell in a deployment has a unique `AppInstanceExceededLogRateLimitCount` counter. The `total` value of the counter is the sum total of all app instances on that Diego Cell that have exceeded the rate limit since the creation of the Diego Cell. When there are no app instances exceeding the rate limit, Diego Cells do not emit the `AppInstanceExceededLogRateLimitCount` metric.

For example, `app-instanceA` and `app-instanceB` are running on one Diego Cell, `app-instanceC` and `app-instanceD` are running on a second Diego Cell, and the current `total` for the `AppInstanceExceededLogRateLimitCount` is `125` on the first Diego Cell and `43` on the second Diego Cell. If `app-instanceD` exceeds the rate limit, the second Diego Cell emits the `AppInstanceExceededLogRateLimitCount` metric with a incremented `total` value of `44`. However, the first Diego Cell does not emit the `AppInstanceExceededLogRateLimitCount` metric, and the `total` value for the `AppInstanceExceededLogRateLimitCount` metric on the first Diego Cell is still `125`.

A Diego Cell emits the `AppInstanceExceededLogRateLimitCount` metric conditionally when an app instance on that Diego Cell begins to exceed the rate limit. For example, `app-instanceC` and `app-instanceD` are on the same Diego Cell. If `app-instanceC` exceeds the rate limit continually over a ten-minute period, and `app-instanceD` exceeds the rate limit during the first three minutes of each five-minute interval within that ten-minute period and then stops, the Diego Cell emits the `AppInstanceExceededLogRateLimitCount` metric three times within that ten-minute period.

Configuring an alert for the AppInstanceExceededLogRateLimitCount metric

If you are using a third-party log management service, you can configure an alert for when the aggregated sum of the `AppInstanceExceededLogRateLimitCount` metric across all the Diego Cells on TAS for VMs has been incremented more than a certain number of times or over a certain percentage in the last five or more minutes. When you configure this alert, consider the number of app instances running on TAS for VMs, the logging rate that you configured in TAS for VMs, your other TAS for VMs configuration settings, and so on.

For more information about third-party log management services, see Streaming App Logs to Log Management Services.

Identifying Apps that exceed the App Log Rate Limit

Diego also logs when a noisy app instance exceeds the rate limit set in TAS for VMs. A log message similar to the example below appears in the log stream for the noisy app:

```
2020-02-24T12:42:18.90-0800 [APP/PROC/WEB/0] OUT app instance exceeded log ra
te limit (100 log-lines/sec) set by platform operator
```

To identify which app instances are exceeding the app log rate limit:

> **Note:** The Firehose and Log Cache plugins were developed by the open-source Cloud Foundry community and are not supported by VMware.

1. Install the Firehose plugin by running:

   ```
   cf install-plugin 'Firehose Plugin'
   ```

2. Install the Log Cache plugin by running:

   ```
   cf install-plugin 'log-cache'
   ```

3. Filter your app log messages by running:

   ```
   cf nozzle -f LogMessage | grep "app instance exceeded log rate limit"
   ```

   The command returns all logs with log messages containing `"app instance exceeded log rate limit"`, similar to the following example:

   ```
   origin:"rep" eventType:LogMessage timestamp:1583859621886751670 deploym
   ent:"warp-drive" job:"diego-cell" index:"3a574bde-91df-48b8-ae21-1d6913
   da0908" ip:"10.0.1.33" tags:<key:"app_id" value:"34bcfafc-402b-4bb4-84d
   b-aea5401b79eb" > tags:<key:"app_name" value:"app-2" > tags:<key:"insta
   nce_id" value:"0" > tags:<key:"organization_id" value:"a30f39c2-4ff3-48
   a1-a869-a9ed21812a61" > tags:<key:"organization_name" value:"test" > ta
   gs:<key:"process_id" value:"34bcfafc-402b-4bb4-84db-aea5401b79eb" > tag
   s:<key:"process_instance_id" value:"92e2ee78-3a1d-41a6-4933-e47b" > tag
   s:<key:"process_type" value:"web" > tags:<key:"source_id" value:"34bcfa
   fc-402b-4bb4-84db-aea5401b79eb" > tags:<key:"space_id" value:"0e2d2d58-
   3ef5-43f3-b880-c8a30903a96b" > tags:<key:"space_name" value:"test-2" >
   logMessage:<message:"app instance exceeded log rate limit (100 log-line
   s/sec) set by platform operator" message_type:OUT timestamp:15838596218
   86751670 app_id:"34bcfafc-402b-4bb4-84db-aea5401b79eb" source_type:"AP
   P/PROC/WEB" source_instance:"0" >
   ```

   You can inspect these logs to identify the app instances that are exceeding the app log rate limit.

Customizing Platform Log Forwarding

You can configure VMware Tanzu Application Service for VMs (TAS for VMs) to forward logs to remote endpoints using the syslog protocol defined in RFC 5424. For more information about enabling log forwarding, see Configure System Logging in *Configuring TAS for VMs*.

TAS for VMs annotates forwarded messages with structured data. This structured data identifies the originating BOSH Director, deployment, instance group, availability zone, and instance ID. All logs forwarded from BOSH jobs have their PRI set to `14`, representing **Facility: user-level messages** and **Informational: informational messages**, as defined in RFC 5424 Section 6.2.1. This PRI value may not reflect the originally intended PRI of the log.

Logs forwarded from other sources, such as kernel logs, retain their original PRI value.

The table below describes the log line Structured Data:

Structured Data	Description
ENTERPRISE_NUMBER	TAS for VMs's private enterprise number, `47450`, as defined in RFC 5424 Section 7.2.2
DIRECTOR	The name of the BOSH Director managing the deployment.
DEPLOYMENT	The name of the BOSH deployment.
INSTANCE_GROUP	The name of the BOSH instance group.
AVAILABILITY_ZONE	The name of the BOSH availability zone.
ID	The BOSH GUID.

Log lines use the following format:

```
<$PRI>$VERSION $TIMESTAMP $HOST $APP_NAME $PROC_ID $MSG_ID
    [instance@ENTERPRISE_NUMBER director="$DIRECTOR" deployment="$DEPLOYMENT"
    group="$INSTANCE_GROUP" az="$AVAILABILITY_ZONE" id="$ID"] $MESSAGE
```

Example log messages:

```
<14>1 2017-01-25T13:25:03.18377Z 192.0.2.10 etcd - - [instance@47450
    director="test-env" deployment="cf-c42ae2c4dfb6f67b6c27" group="diego_database" az
="us-west1-a"
    id="83bd66e5-3fdf-44b7-bdd6-508deae7c786"] [INFO] the leader is
    [https://diego-database-0.etcd.service.cf.internal:4001]
<14>1 2017-01-25T13:25:03.184491Z 192.0.2.10 bbs - - [instance@47450
    director="test-env" deployment="cf-c42ae2c4dfb6f67b6c27" group="diego_database" az
="us-west1-a"
    id="83bd66e5-3fdf-44b7-bdd6-508deae7c786"]
    {"timestamp":"1485350702.539694548","source":"bbs","message":
    "bbs.request.start-actual-lrp.starting","log_level":1,"data":
    {"actual_lrp_instance_key":{"instance_guid":
    "271f71c7-4119-4490-619f-4f44694717c0","cell_id":
    "diego_cell-2-41f21178-d619-4976-901c-325bc2d0d11d"},"actual_lrp_key":
    {"process_guid":"1545ce89-01e6-4b8f-9cb1-5654a3ecae10-137e7eb4-12de-457d-
         8e3e-1258e5a74687","index":5,"domain":"cf-apps"},"method":"POST","net_info":
    {"address":"192.0.2.12","ports":[{"container_port":8080,"host_port":61532},
    {"container_port":2222,"host_port":61533}]},"request":
    "/v1/actual_lrps/start","session":"418.1"}}
```

Modify Which Logs TAS for VMs Forwards

When you enable log forwarding, TAS for VMs forwards all log lines written to the `/var/vcap/sys/log` directories on all Ops Manager virtual machines (VMs) to your configured External Syslog Aggregator endpoint by default.

To configure TAS for VMs to forward a subset of logs instead of forwarding all logs:

1. Go to the Ops Manager Installation Dashboard.
2. Click the TAS for VMs tile.
3. Select **System Logging**.
4. In the **Custom rsyslog configuration** field, enter a custom syslog rule. See the example custom syslog rules below.
5. Click **Save**.

The custom rsyslog rules shown below are written in RainerScript. The custom rules are inserted before the rule that forwards logs. The *stop* command, `stop`, prevents logs from reaching the forwarding rule. This filters out these logs.

Logs filtered out before forwarding remain on the local disk, where the BOSH job originally wrote them. These logs remain on the local disk only until BOSH Director recreates the VMs. You can access these logs from Ops Manager or through SSH.

Note: TAS for VMs requires a valid custom rule to forward logs. If your custom rule contains syntax errors, TAS for VMs forwards no logs.

Forward Only Logs From a Certain Job

This rule filters out logs unless they originate from the `uaa` job:

```
if ($app-name != "uaa") then stop
```

Exclude Logs With Certain Content

This rule filters out logs that contain “DEBUG” in the body.

```
if ($msg contains "DEBUG") then stop
```

Note: The above example contains "DEBUG" in the message body. Not all logs intended for debugging purposes contain the string "DEBUG" in the message body.

Monitoring

In this section:

- Monitoring TAS for VMs
- Selecting and Configuring a Monitoring System
- Identifying TAS for VMs Jobs Using vCenter

Monitoring TAS for VMs

You can monitor their VMware Tanzu Application Service for VMs (TAS for VMs) deployments using the instructions provided in these topics. For information about monitoring VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) deployments, see Logging and monitoring TKGI.

For more information about logging and metrics in TAS for VMs, see Overview of logging and metrics.

KPI and scaling changes from TAS for VMs v2.9 to TAS for VMs v2.12

There are no changes to key performance indicator (KPI) or scaling recommendations from VMware Tanzu Application Service for VMs (TAS for VMs) v2.9 to TAS for VMs v2.12.

Monitor services

For information about KPIs and metrics for services, see:

- **VMware Tanzu SQL [MySQL]:** Monitoring VMware Tanzu SQL [MySQL].
- **VMware Tanzu Gemfire:** Monitoring VMware Tanzu GemFire Service Instances.
- **Redis for VMware Tanzu:** Monitoring Redis for VMware Tanzu
- **VMware Tanzu RabbitMQ (pre-provisioned):** Monitoring and KPIs for Pre-Provisioned VMware Tanzu RabbitMQ
- **VMware Tanzu RabbitMQ (on-demand):** Monitoring and KPIs for On-Demand VMware Tanzu RabbitMQ

Selecting and configuring a monitoring system

Use this topic to help you make decisions for selecting and configuring a system to continuously monitor Ops Manager component performance and health.

Selecting a monitoring platform

VMware recommends using Healthwatch to monitor your deployment. Healthwatch is a service tile developed and supported by VMware and available on VMware Tanzu Network.

Many third-party systems can also be used to monitor an Ops Manager deployment.

Monitoring platform types

Monitoring platforms support two types of monitoring:

- A *dashboard* for active monitoring when you are at a keyboard and screen
- Automated *alerts* for when your attention is elsewhere

Some monitoring solutions offer both in one package. Others require putting the two pieces together.

Monitoring platforms

There are many monitoring options available, both open source and commercial products. Some commonly used platforms among Ops Manager customers include:

- Healthwatch by VMware
- VMware Partner Services available on VMware Tanzu Network:
  - AppDynamics
  - Datadog
  - Dynatrace
  - New Relic
  - WaveFront by VMware
- Other Commercial Services
  - VMware vRealize Operations (vROPS)
- Open-Source Tooling
  - Prometheus + Grafana
  - OpenTSDB

VMware Cloud Ops tools

The VMware Cloud Ops Team manages two types of deployments for internal VMware use: open-source Cloud Foundry, and Ops Manager.

For **Cloud Foundry**, VMware Cloud Ops uses several monitoring tools. The Datadog Config repository provides an example of how the VMware Cloud Ops team uses a customized Datadog dashboard to monitor the health of its open-source Cloud Foundry deployments.

To monitor its **Ops Manager** deployments, VMware Cloud Ops leverages a combination of Healthwatch and Google Stackdriver.

Key inputs for platform monitoring

BOSH VM and Ops Manager component health metrics

Most monitoring service tiles for Ops Manager come packaged with the Firehose nozzle necessary to extract the BOSH and Ops Manager metrics leveraged for platform monitoring. Nozzles are programs that consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and to forward it to other apps and services.

The nozzles gather the component logs and metrics streaming from the Loggregator Firehose endpoint. For more information about the Firehose, see Loggregator architecture.

As of Pivotal Platform v2.0, both BOSH VM Health metrics and Ops Manager component metrics stream through the Firehose by default.

- Ops Manager component metrics originate from the Metron agents on their source components, then travel through Dopplers to the Traffic Controller.
- The Traffic Controller aggregates both metrics and log messages system-wide from all Dopplers, and emits them from its Firehose endpoint.

For information about high-signal-value metrics and capacity scaling indicators in an Ops Manager deployment, see Key Performance Indicators and Key capacity scaling indicators.

Continuous functional smoke tests

Ops Manager includes smoke tests, which are functional unit and integration tests on all major system components. By default, whenever an operator upgrades to a new version of VMware Tanzu Application Service for VMs (TAS for VMs), these smoke tests run as a post-deploy errand.

VMware recommends additional higher-resolution monitoring by the execution of continuous smoke tests, or Service Level Indicator tests, that measure user-defined features and check them against expected levels.

- Healthwatch automatically executes these tests for TAS for VMs Service Level Indicators.
- The VMware Cloud Ops Cloud Foundry smoke tests repository offers additional testing examples.

For information about how to set up Concourse to generate custom component metrics, see Metrics in the Concourse documentation.

Warning and critical thresholds

To properly configure your monitoring dashboard and alerts, you must establish what thresholds can drive alerting and red/yellow/green dashboard behavior.

Some key metrics have more fixed thresholds, with similar threshold numbers numbers recommended across different foundations and use cases. These metrics tend to revolve around the health and performance of key components that can impact the performance of the entire system.

Other metrics of operational value are more dynamic in nature. This means that you must establish a baseline and yellow/red thresholds suitable for your system and its use cases. You can establish initial baselines by watching values of key metrics over time and noting what seems to be a good starting threshold level that divides acceptable and unacceptable system performance and health.

Continuous evolution

Effective platform monitoring requires continuous evolution.

After you establish initial baselines, VMware recommends that you continue to refine your metrics and tests to maintain the appropriate balance between early detection and reducing unnecessary alert fatigue. You should occasionally revisit the dynamic measures recommended in Key performance indicators and Key capacity scaling indicators to ensure they are still appropriate to the current system configuration and its usage patterns.

Identifying TAS for VMs Jobs Using vCenter

To effectively monitor, control, and manage the virtual machines making up your VMware Tanzu Application Service for VMs (TAS for VMs) deployment, you may need to identify which VM corresponds to a particular job in TAS for VMs. You can find the CID of a particular VM from Ops Manager by navigating to the **Status** tab in the TAS for VMs tile.

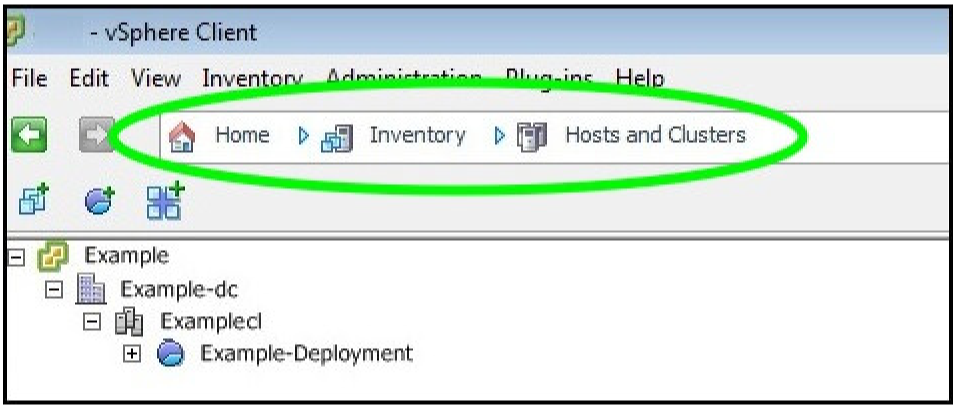
If you have deployed TAS for VMs to VMware vSphere, you can also identify which TAS for VMs job corresponds to which VM using the vCenter vSphere client.

Note: The CID shown in Ops Manager is the name of the machine in vCenter.

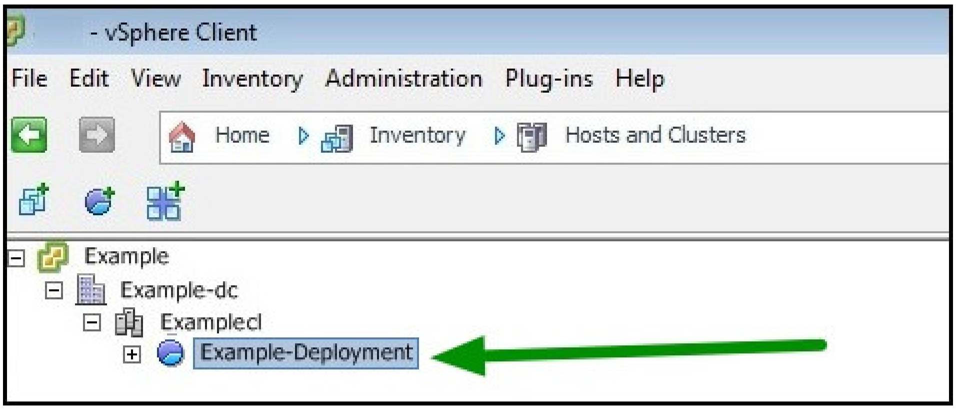
Identifying TAS for VMs Jobs Using vCenter

To identify TAS for VMs jobs using vCenter:

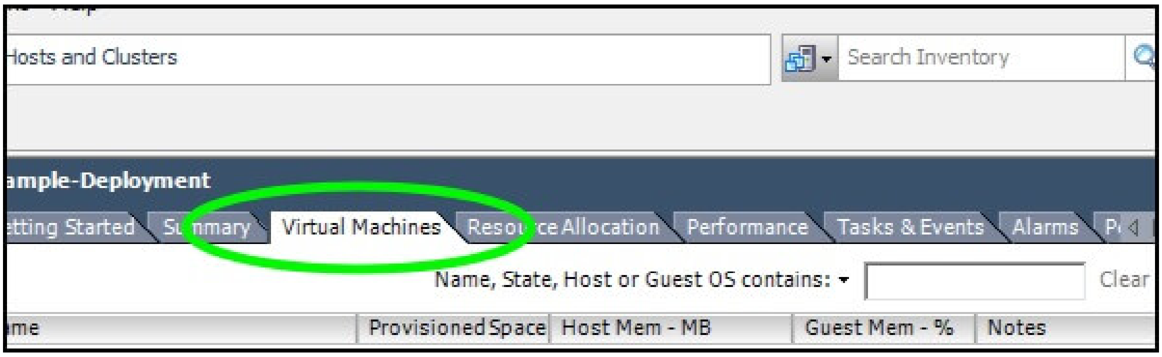
1. Start the vSphere client and log in to the vCenter Server system.
2. Select the **Inventory > Hosts and Clusters** view.



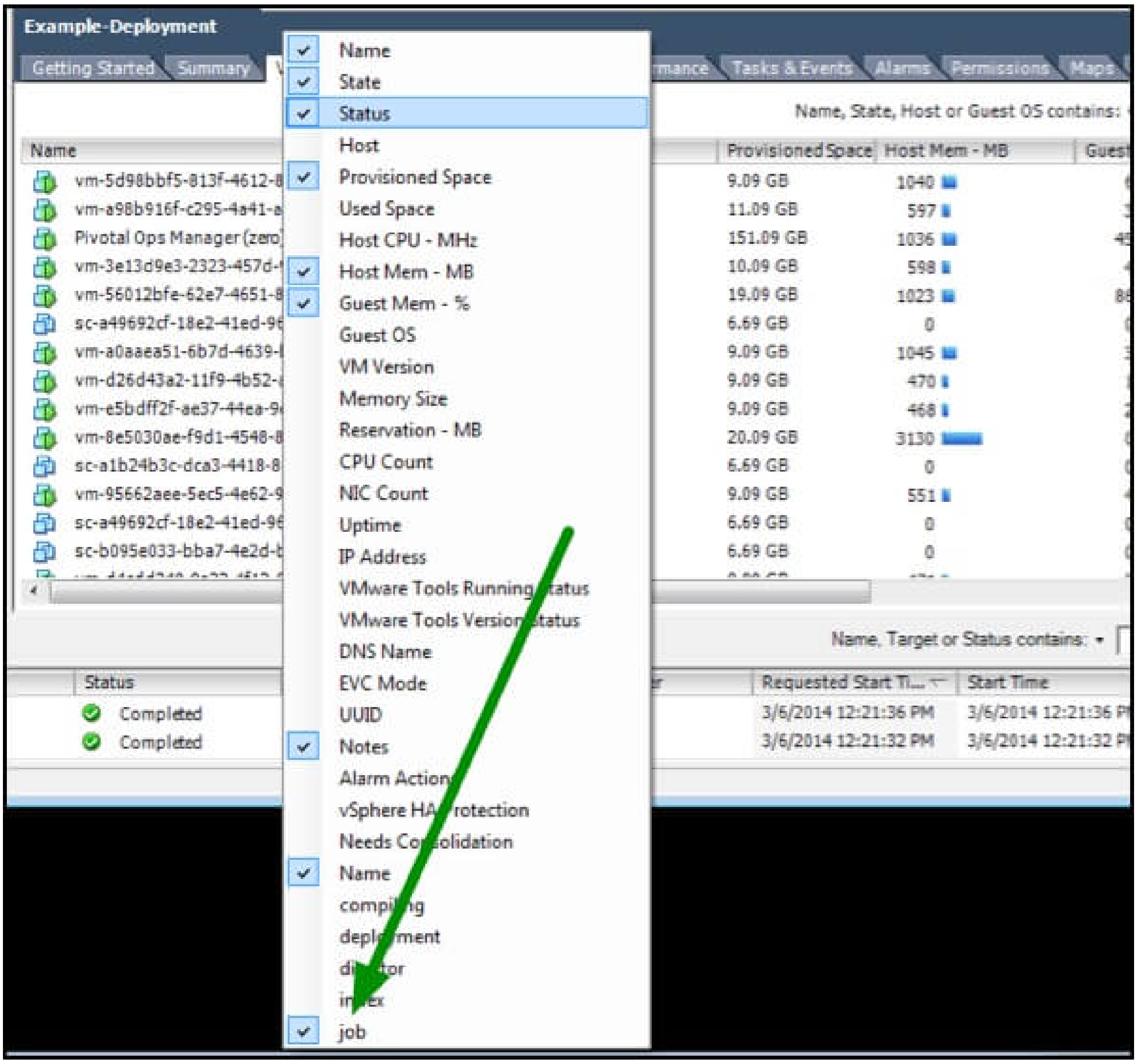
3. Select the Resource Pool containing your TAS for VMs deployment.



4. Select the **Virtual Machines** tab.



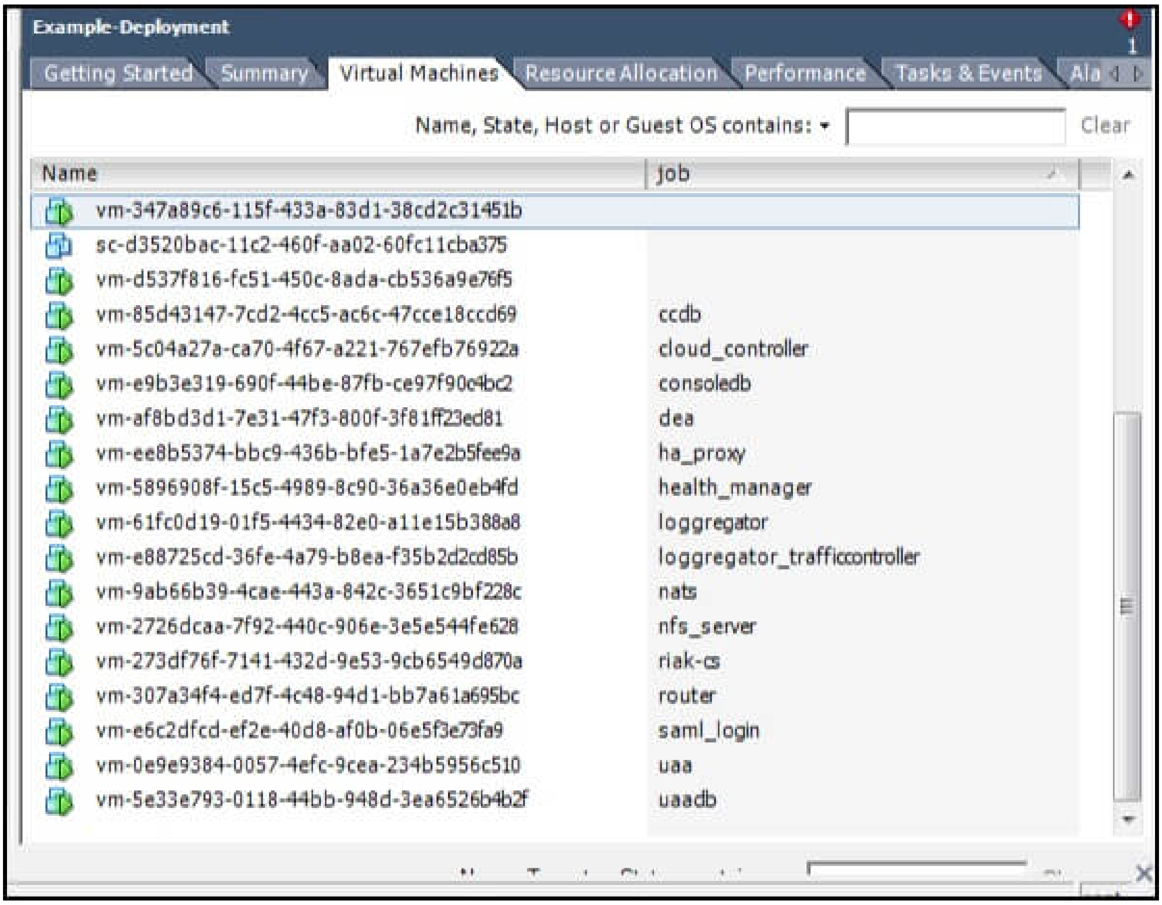
5. Right-click the column label heading and check **job**.



Name	Provisioned Space	Host Mem - MB
vm-5d98bbf5-813f-4612-8	9.09 GB	1040
vm-a98b916f-c295-4a41-a	11.09 GB	597
Pivotal Ops Manager (zero)	151.09 GB	1036
vm-3e13d9e3-2323-457d-	10.09 GB	598
vm-56012bfe-62e7-4651-8	19.09 GB	1023
sc-a49692cf-18e2-41ed-96	6.69 GB	0
vm-a0aaea51-6b7d-4639-	9.09 GB	1045
vm-d26d43a2-11f9-4b52-	9.09 GB	470
vm-e5bdff2f-ae37-44ea-9	9.09 GB	468
vm-8e5030ae-f9d1-4548-8	20.09 GB	3130
sc-a1b24b3c-dca3-4418-8	6.69 GB	0
vm-95662aee-5ec5-4e62-9	9.09 GB	551
sc-a49692cf-18e2-41ed-96	6.69 GB	0
sc-b095e033-bba7-4e2d-b	6.69 GB	0

Status	Requested Start Ti...	Start Time
Completed	3/6/2014 12:21:36 PM	3/6/2014 12:21:36 P
Completed	3/6/2014 12:21:32 PM	3/6/2014 12:21:32 P

6. The job column displays the TAS for VMs job associated with each virtual machine.



Name	job
vm-347a89c6-115f-433a-83d1-38cd2c31451b	
sc-d3520bac-11c2-460f-aa02-60fc11cba375	
vm-d537f816-fc51-450c-8ada-cb536a9e76f5	
vm-85d43147-7cd2-4cc5-ac6c-47cce18ccd69	ccdb
vm-5c04a27a-ca70-4f67-a221-767efb76922a	cloud_controller
vm-e9b3e319-690f-44be-87fb-ce97f90c4bc2	consoledb
vm-af8bd3d1-7e31-47f3-800f-3f81ff23ed81	dea
vm-ee8b5374-bbc9-436b-bfe5-1a7e2b5fee9a	ha_proxy
vm-5896908f-15c5-4989-8c90-36a36e0eb4fd	health_manager
vm-61fc0d19-01f5-4434-82e0-a11e15b388a8	loggregator
vm-e88725cd-36fe-4a79-b8ea-f35b2d2cd85b	loggregator_trafficcontroller
vm-9ab66b39-4cae-443a-842c-3651c9bf228c	nats
vm-2726dcaa-7f92-440c-906e-3e5e544fe628	nfs_server
vm-273df76f-7141-432d-9e53-9cb6549d870a	riak-cs
vm-307a34f4-ed7f-4c48-94d1-bb7a61a695bc	router
vm-e6c2dfcd-ef2e-40d8-af0b-06e5f3e73fa9	saml_login
vm-0e9e9384-0057-4efc-9cea-234b5956c510	uaa
vm-5e33e793-0118-44bb-948d-3ea6526b4b2f	uaadb

App Metrics

In this section:

- Container Metrics
- Metric Registrar and Custom App Metrics
- Using Metric Registrar
- Identifying the Source Deployment of Metrics

Container metrics

This topic tells you about the metrics that are emitted by all containers managed by VMware Tanzu Application Service for VMs (TAS for VMs) and its scheduling system, Diego.

App metrics include the container metrics, and any custom app metrics that developers create.

Diego Container metrics

Diego containers emit resource usage metrics for the app instance. Diego averages and emits each metric every 15 seconds.

The following table describes all Diego container metrics:

Metric	Description	Unit

`CpuPercentage`	CPU time used by AI as a percentage of a single CPU core. This is usually no greater than `100% * the number of vCPUs on the host Diego cell`, but it may be more due to discrepancies in measurement timing.	`float64`
`AbsoluteCPUEntitlement`	CPU time that the AI was entitled to use. At minimum, the CPU time that a Diego cell provides an AI is `min(app memory, 8 GB) * (Diego cell vCPUs/Diego cell memory) * 100%`. The platform operator can provide the Diego cell `vCPUs/memory` ratio to developers. If a Diego cell is not at capacity or if other workloads on it are idle, the cell can provide more than the minimum CPU.	`uint64`
`AbsoluteCPUUsage`	CPU time used by AI. `AbsoluteCPUUsage / AbsoluteCPUEntitlement` calculates a 0-100% range of AI usage per entitlement.	`uint64`
`MemoryBytes`	RAM memory used by AI, in MB.	`uint64`
`MemoryBytesQuota`	RAM memory available, in GB.	`uint64`
`DiskBytes`	Disk space used by AI, in MB.	`uint64`
`DiskBytesQuota`	Disk space available, in GB.	`uint64`
`ContainerAge`	Age of container, in nanoseconds.	`uint64`

Loggregator v1: Most of the container metrics are emitted in a `ContainerMetric` envelope. The `AbsoluteCPUEntitlement`, `AbsoluteCPUUsage`, and `ContainerAge` container metrics are emitted as separate `ValueMetric` envelopes.

Loggregator v2: All container metrics are emitted in gauge envelopes. The `AbsoluteCPUEntitlement`, `AbsoluteCPUUsage`, and `ContainerAge` container metrics are emitted in a separate envelope to the other container metrics.

Retrieve Container Metrics from the cf CLI

You can use Cloud Foundry command-line interface (cf CLI) commands to return container metrics.

Retrieving Container metrics from the cf CLI

You can retrieve container metrics using the Cloud Foundry Command Line Interface (cf CLI).

To retrieve CPU, memory, and disk metrics for all instances of an app, see Retrieve CPU, Memory, and Disk Metrics below.

To retrieve CPU entitlement metrics for all instances of an app, see Retrieve CPU Entitlement Metrics below.

To determine when an app has exceeded its CPU entitlement, see Monitor Apps That Exceed Their CPU Entitlement below.

Retrieving CPU, Memory, and Disk metrics

To retrieve CPU, memory, and disk metrics for all instances of an app:

1. Run the command `cf app APP-NAME`.

The command output lists Diego container metric values as follows:

Label in Output	Metrics listed, as described in Diego Container Metrics, above
`cpu`	`CpuPercentage`
`memory`	`MemoryBytes` of `MemoryBytesQuota`
`disk`	`DiskBytes` of `DiskBytesQuota`

For example:

Retrieving CPU Entitlement metrics

name: dora-example requested state: started routes: dora-example.bosh-lite.com last uploaded: Fri 05 Apr 10:41:21 PDT 2019 stack: cflinuxfs3 buildpacks: ruby

type: web instances: 1/1 memory usage: 256M state since cpu memory disk details

0 running 2019-04-05T17:41:31Z 0.4% 39.5M of 256M 89.9M of 1G

Retrieve CPU Entitlement Metrics (Experimental)

To see app instance `AbsoluteCPUEntitlement` metrics from the command line:

1. Install the CPU Entitlement Plugin from cpu-entitlement-plugin in GitHub.

Determining when Apps exceed their CPU Entitlement

You can use the Cloud Foundry CPU Overentitlement Plug-in to determine when an app has exceeded its CPU entitlement and might need to be scaled up.

```
$ cf cpu-entitlement dora-example
Note: This feature is experimental.
Showing CPU usage against entitlement for app dora-example in org example-org
/ space example-org-staging as dora@example.com ...

     avg usage   curr usage
#0   1.62%       1.66%
#1   2.93%       3.09%
#2   2.51%       2.62%
```

Important: To avoid security exposure, verify that you migrate your apps and custom buildpacks to use the `cflinuxfs4` stack based on Ubuntu 22.04 LTS (Jammy Jellyfish). The `cflinuxfs3` stack is based on Ubuntu 18.04 (Bionic Beaver), which reaches end of standard support in April 2023.

Metric Registrar and Custom App Metrics

This topic describes the Metric Registrar for VMware Tanzu Application Service for VMs (TAS for VMs), which lets developers define and use their own custom app metrics.

The topic also explains how to enable and configure the Metric Registrar from the TAS for VMs tile.

App metrics include custom app metrics that you can export with the Metric Registrar, plus container metrics emitted by all TAS for VMs containers. For information about container metrics, see Container Metrics.

Overview

The Metric Registrar allows app developers to export custom app metrics in a format that Loggregator can consume. App developers can then use the custom metrics to monitor apps with App Metrics and configure autoscaling rules with App Autoscaler.

App developers can export custom metrics to Loggregator by configuring their apps in one of the following ways:

- **Metrics Endpoint:** Publish and register a Prometheus Exposition metrics endpoint to an app. The Metric Registrar then polls this endpoint every 35 seconds and convert the metrics found in the response to Loggregator metrics.
- **Structured Log:** Modify your app to emit metrics using a specific JSON or DogStatsD format. The Metric Registrar then converts all matching log messages into Loggregator metrics or events.

For more information about installing the Metric Registrar Plugin and registering your app, see Emitting Custom App Metrics to the Metric Registrar.

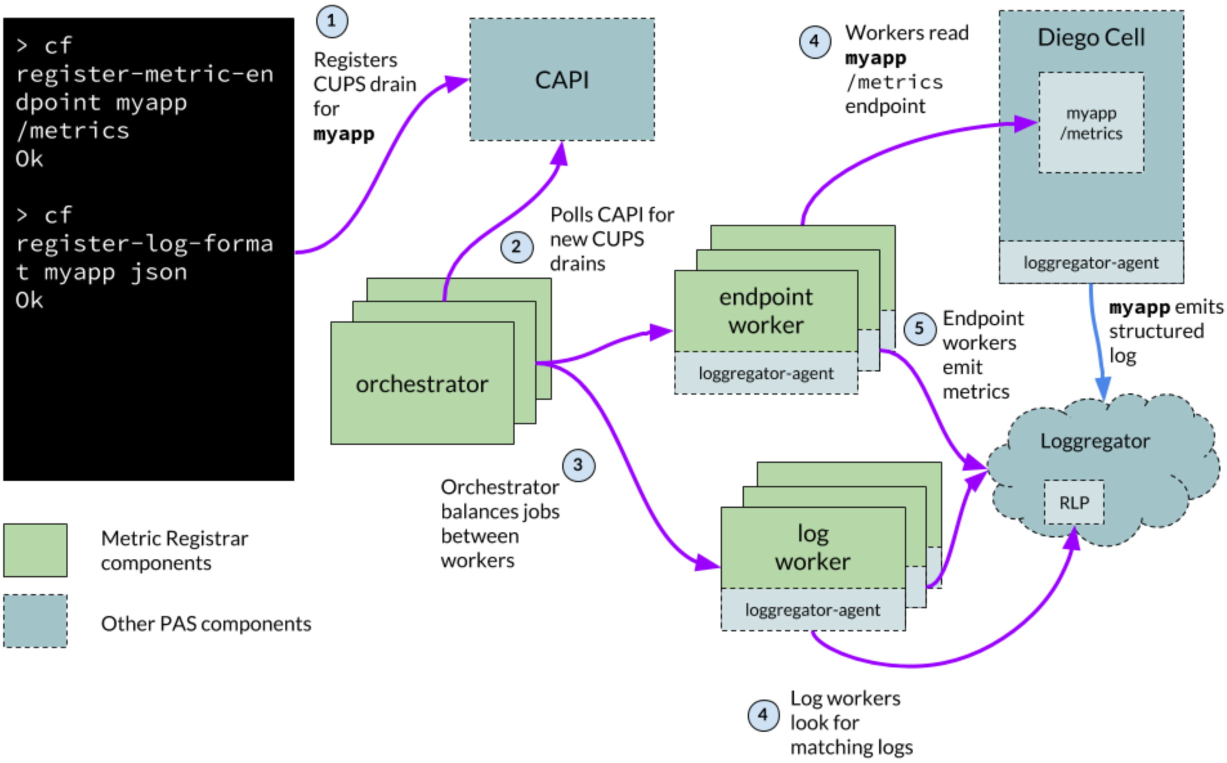
For more information about the components and products mentioned, see:

- Loggregator
- App Metrics
- App Autoscaler

Architecture

The following diagram illustrates how the Metric Registrar sends your custom app metrics to Loggregator. The components of the Metric Registrar are:

- The cf CLI plugin
- The `metric_registrar_endpoint_worker` and `metric_registrar_log_worker` jobs running on the Doppler VM of the TAS for VMs deployment
- The `metric_registrar_orchestrator` and `metric_registrar_smoke_test` jobs running on the Clock Global VM of the TAS for VMs deployment



```
> cf
register-metric-en
dpoint myapp
/metrics
Ok

> cf
register-log-forma
t myapp json
Ok
```

1
Registers CUPS drain for **myapp**
CAPI
2
Polls CAPI for new CUPS drains
orchestrator
3
Orchestrator balances jobs between workers
4
Workers read **myapp** /metrics endpoint
Diego Cell
myapp /metrics
loggregator-agent
endpoint worker
loggregator-agent
5
Endpoint workers emit metrics
myapp emits structured log
Loggregator
RLP
log worker
loggregator-agent
4
Log workers look for matching logs
Metric Registrar components
Other PAS components

Click the image for a larger representation.

Configure the Metric Registrar

Note: The Metric Registrar is bundled with TAS for VMs, and you configure it in the TAS for VMs tile. You do not install and configure Metric Registrar as a separate product tile.

In the Metric Registrar pane, you configure the Metric Registrar. Metric Registrar allows the conversion of structured logs into metrics. It also scrapes metrics endpoints and forwards the metrics to Loggregator. If enabled, VMware recommends also enabling the Metric Registrar Smoke Test errand.

Deactivate the Metric Registrar

The Metric Registrar is enabled by default.

To deactivate the Metric Registrar:

1. Select **Metric Registrar**.
2. Clear the **Enable Metric Registrar** checkbox.
3. Click **Save**.

Edit Default Scraping Interval

The scraping interval defines how often the Metric Registrar polls custom metric endpoints. The default is 35 seconds.

To edit the Metric Registrar scraping interval:

1. Select **Metric Registrar**.
2. Edit the **Endpoint scraping interval** field.
3. Click **Save**.

Add Blocked Tags

To prevent the Metric Registrar from consuming the value of a metric or event tag, you can add the tag to the **Blocked tags** field. For example, if you tag your metrics with a `customer_id`, you may want to add `customer_id` to the list of blocked tags.

By default, the following tags are blocked to prevent interference with other products like App Metrics that use and rely on such tags.

- `deployment`
- `job`
- `index`
- `id`

To prevent the Metric Registrar from consuming the value of a metric or event tag:

1. Select **Metric Registrar**.
2. Add the desired tags to the **Blocked tags** field in a comma-separated list.
3. Click **Save**.

Using Metric Registrar

This topic describes how to install the Metric Registrar CLI plugin and emit custom app metrics to the Metric Registrar by registering your app as a metric source.

For information about how the Metric Registrar processes custom app metrics and how to configure the Metric Registrar, see Metric Registrar and Custom App Metrics.

Overview

Registering your app as a metric source allows you to see your custom metrics in App Metrics and configure autoscaling rules with App Autoscaler.

For more information, see the App Metrics documentation and Scaling an App Using App Autoscaler.

Install the Plugin

To install the Metric Registrar CLI plugin:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI).

2. Run:

   ```
   cf install-plugin -r CF-Community "metric-registrar"
   ```

Register Your App

To register your app as a metric source, do one of the following:

- **Register a metrics endpoint:** Register a metrics endpoint for custom metrics to be parsed and emitted to Loggregator. For more information, see Register a Metrics Endpoint.
- **Register a structured log format:** Register a structured log format that can be emitted to Loggregator. For more information, see Register a Structured Log Format.

Note: If you are migrating from and manually send data to Metrics Forwarder, VMware recommends registering a structured log format. For more information, see Register a Structured Log Format below.

Register a Metrics Endpoint

The Metric Registrar supports custom metrics created with the open-source tool, Prometheus. Prometheus uses a text-based exposition format common in many open-source libraries. It also provides several out-of-the-box metrics for different programming languages.

For more information about Prometheus, see What is Prometheus? in the Prometheus documentation.

For examples of apps that use Prometheus to publish metrics to an endpoint, see metric-registrar-examples in GitHub.

Note: These endpoints are public by default. If you do not want to expose public metrics endpoints for your app, see Register a Structured Log Format below.

Note: Metrics endpoints must be served over HTTPS.

Prerequisites

Before registering a metrics endpoint, you must:

- For all Spring apps, update the `application.yml` file to include one or more Prometheus endpoints. For example:

  ```
  management:
    endpoints:
      web:
        exposure:
          include: "prometheus"
  ```

- For all Spring apps, update the security configuration file to permit access to the Prometheus endpoints. For an example, see metric-registrar-examples in GitHub.

- For all other apps, see Client libraries in the Prometheus documentation.

Register a Public Metrics Endpoint

To register a public metrics endpoint for an app:

1. Log in to the cf CLI.
2. For each Prometheus endpoint in your app, register the endpoint as a metric source by running:

   ```
   cf register-metrics-endpoint APP-NAME PATH-OR-URL --insecure
   ```

   Where:

   - `APP-NAME` is the name of the app.
   - `PATH-OR-URL` is the path to the Prometheus endpoint, or a URL without the scheme.

   For example, if the app name is `example` and the metrics endpoint is at `https://example.com/metrics`, the command would be:

   ```
   cf register-metrics-endpoint example /metrics --insecure
   ```

   Or, if pulling metrics from a different URL:

   ```
   cf register-metrics-endpoint example otherexample.com/metrics --insecure
   ```

   Note: When you specify a URL to a metrics endpoint, the scheme must be omitted and the URL must use HTTPS. When you specify a path to a metrics endpoint, the path must start with a `/` character.

   Note: Each metrics endpoint registered with Metric Registrar creates a service instance. If the generated service name is longer than 50 characters, it is encoded with a hash.

   Note: In Metric Registrar v1.3.1 and later, you must use the `--insecure` flag for the same values you provided in previous versions.

Verify a Metrics Endpoint

VMware recommends that you install the LogCache cf CLI plugin to view metrics.

To install and use the Log Cache cf CLI plugin:

1. To install Log Cache, run:

   ```
   cf install-plugin -r CF-Community "log-cache"
   ```

2. To view metrics for an app, run:

```
cf tail --envelope-class=metrics APP-NAME
```

Where `APP-NAME` is the name of your app.

For example, if you provided a custom metric called `users_per_cpu` as a gauge, you should see the following output:

```
2019-06-18T10:49:28.27-0600 [app-name/1] GAUGE cpu:0.154763 percentage
disk:14000128.000000 bytes disk_quota:33554432.000000 bytes memory:1019
0848.000000 bytes memory_quota:33554432.000000 bytes
2019-06-18T10:49:26.42-0600 [app-name/1] GAUGE users_per_cpu:557.500000
```

Where the first line is default container metrics and the second line is custom app metrics `users_per_cpu`.

> **Note:** The Metric Registrar produces the following benign error message at every polling interval: `[LGR/] ERR Invalid syslog drain URL: parse failure`

Register a Private Metrics Endpoint

To register a private metrics endpoint:

1. Set up a Prometheus endpoint that serves on any port other than 8080.
2. Register the endpoint by running:

   ```
   cf register-metrics-endpoint APP-NAME PATH --internal-port PORT
   ```

   Where:

   - `APP-NAME` is the name of the app.
   - `PATH` is the path to the Prometheus endpoint.
   - `PORT` is the port where the Prometheus endpoint is served.

Register a Structured Log Format

The Metric Registrar supports metrics emitted in JSON or DogStatsD formats. For more information about these formats, see JSON and DogStatsD below.

To register your app as a metric source:

1. Log in to the cf CLI.
2. Run:

   ```
   cf register-log-format APP-NAME FORMAT
   ```

   Where:

   - `APP-NAME` is the name of your app.

   - `FORMAT` is either `json` or `DogStatsD`.

3. In your app, log a structured `json` or `DogStatsD` message to represent the custom metric.

JSON

The table below shows the supported JSON format for event, gauge, and counter log types.

Type	Format

Events	`{`
	`   "type": "event",`
	`   "title": "title",`
	`   "body": "body",`
	`   "tags": {`
	`     "tag1": "tag value"`
	`   }`
	`}`

Gauges	`{`
	`   "type": "gauge",`
	`   "name": "some-counter",`
	`   "value": ,`
	`   "tags": {`
	`     "tag1": "tag value"`
	`   }`
	`}`

Counters	`{`
	`   "type": "counter",`
	`   "name": "some-counter",`
	`   "delta": ,`
	`   "tags": {`
	`     "tag1": "tag value"`
	`   }`
	`}`

DogStatsD Log Format

The table below shows the supported DogStatsD format for event, gauge, and counter log types. It also lists the supported fields. For more information about DogStatsD, see DogStatsD in the Datadog documentation.

Type	Format	Supported Fields

Events	`_e{title.length,text.length}:title|text|d:timestamp|`
	`h:hostname|p:priority|t:alert_type|#tag1,tag2`
		`title`
		`text`

<table>
<tr><td>Gauges</td><td><code>gauge.name:value|g|@sample_rate|#tag1:value,tag2</code></td><td><code>gauge.name</code>
<code>value</code></td></tr>
<tr><td>Counters</td><td><code>counter.name:value|c|@sample_rate|#tag1:value,tag2</code></td><td><code>counter.name</code>
<code>value</code></td></tr>
</table>

Register Using CUPS

Metric Registrar uses the syslog drain User Provided Service.

To create a Metric Registrar Service:

1. Create the service by running:

   ```
   cf create-user-provided-service SERVICE-NAME -l BINDING
   ```

   Where:

   - `SERVICE-NAME` is the name you choose for the service.
   - `BINDING` is the service binding. The binding can use the following schemes:
     - `secure-endpoint://`
     - `metrics-endpoint://`
     - `structured-format://`

2. Bind the provided service by running:

   ```
   cf bind-service APP-NAME SERVICE-NAME
   ```

   Where:

   - `APP-NAME` is the name of the app.
   - `SERVICE-NAME` is the service name you chose in the previous step.

The Metric Registrar supports metrics emitted in JSON or DogStatsD formats. For metrics endpoints, the rest of the URI specifies the host, port, and path of the metrics endpoint.

For example:

```
secure-endpoint://:8081/metrics
metrics-endpoint:///metrics
structured-format://json
```

For a secure endpoint, you must expose the additional port you are serving on using Diego.

For example:

```
cf curl /v2/apps/APP-GUID -X PUT -d '{"ports": [PORT1, PORT2, PORT3...]}'
```

Where:

- `APP-GUID` is the GUID of your app.

- `PORT1, PORT2, PORT3...` is a comma-separated list of the ports on which you want your app to receive traffic.

Identifying the Source Deployment of Metrics

This topic explains how to read app metrics metadata to identify which Ops Manager product generated the metrics, and to retrieve other information about the metrics source.

For example, you may want to distinguish between metrics coming from apps hosted by VMware Tanzu Application Service for VMs (TAS for VMs) itself and apps hosted by an Isolation Segment deployment.

Source Deployment from Tile GUID

Metrics identify their source deployment with a `deployment` tag followed by `cf-` prefixed to the GUID of the source Ops Manager tile. You can use this GUID to identify which tile deployed the Diego Cells generating the metrics.

Human-Friendly Metadata

TAS for VMs and other runtime tiles tag metrics with additional metadata to help users parse metrics coming from different deployments. Downstream monitoring products such as Healthwatch also use this metadata to display human-readable names.

The tags are:

- `product`: The value of this tag is always `VMware Tanzu Application Service` for the TAS for VMs tile. The tags for other products are: `Isolation Segment`, `Small Footprint VMware Tanzu Application Service`, and `VMware Tanzu Application Service for Windows`.
- `system_domain`: The value of this tag corresponds to what you set in the **System domain** field in the **Domains** pane of a given tile.
- `placement_tag`: The value of this tag is always `null` for TAS for VMs. However, for VMware Tanzu Application Service for VMs [Windows] (TAS for VMs [Windows]) and Isolation Segment tiles, you can configure this value using the **Segment name** field in the **App Containers** pane. An operator can display capacity and other relevant metrics using the `placement_tag` name. This makes it easier to reason about the importance of a given segment when issues arise.

These tags are properties of the Metron agent running on each VM in a deployment.

Performance and Scaling

In this section:

- Key Performance Indicators
- Key Capacity Scaling Indicators

Key performance indicators

The KPIs described here are provided for operators to give general guidance on monitoring a TAS for VMs deployment using platform component and system (BOSH) metrics. Although many metrics are emitted from the platform, the KPIs are high-signal-value metrics that can indicate emerging platform issues.

This alerting and response guidance has been shown to apply to most deployments. VMware recommends that operators continue to fine-tune the alert measures to their deployment by observing historical trends. VMware also recommends that operators expand beyond this guidance and create new, deployment-specific monitoring metrics, thresholds, and alerts based on learning from their deployments.

> **Note:** Thresholds noted as "dynamic" in the tables below indicate that while a metric is highly important to watch, the relative numbers to set threshold warnings at are specific to a given TAS for VMs deployment and its use cases. These dynamic thresholds should be occasionally revisited because the foundation and its usage continue to evolve. For more information, see Warning and Critical Thresholds in *Selecting and Configuring a Monitoring System*.

While the performance impact on TAS for VMs is considered when building new features, VMware does not perform discrete load and scale testing on a regular basis. It is impossible to test each configuration of TAS for VMs with every type of infrastructure and app workload. Therefore, VMware recommends using the following KPIs to identify when to scale your system.

For more information about accessing metrics used in these key performance indicators, see Overview of Logging and Metrics.

Diego auctioneer metrics

These sections describe Diego Auctioneer metrics.

Auctioneer app instance (AI) placement failures

auctioneer.AuctioneerLRPAuctionsFailed

Description	The number of Long Running Process (LRP) instances that the Auctioneer failed to place on Diego Cells. This metric is cumulative over the lifetime of the Auctioneer job. **Use:** This metric can indicate that TAS for VMs is out of container space or that there is a lack of resources within your environment. This indicator also increases when the LRP is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work. This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no app instances being scheduled. This error is most common due to capacity issues. For example, if Diego Cells do not have enough resources, or if Diego Cells are going back and forth between a healthy and unhealthy state. **Origin:** Firehose **Type:** Counter (Integer) **Frequency:** During each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning:** ≥ 0.5 **Red critical:** ≥ 1
Recommended response	1. To best determine the root cause, examine the Auctioneer logs. Depending on the specific error and resource constraint, you may also find a failure reason in the Cloud Controller (CC) API. 2. Investigate the health of your Diego Cells to determine if they are the resource type causing the problem. 3. Consider scaling additional Diego Cells using Ops Manager. 4. If scaling Diego Cells does not solve the problem, pull Diego Brain logs and BBS node logs and contact Support telling them that LRP auctions are failing.

Auctioneer time to fetch Diego Cell state

auctioneer.AuctioneerFetchStatesDuration

Description	Time in ns that the Auctioneer took to fetch state from all the Diego Cells when running its auction. **Use:** Indicates how the Diego Cells themselves are performing. Alerting on this metric helps alert that app staging requests to Diego may be failing. **Origin:** Firehose **Type:** Gauge, integer in ns **Frequency:** During event, during each auction
Recommended measurement	Maximum over the last 5 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning:** ≥ 2 s **Red critical:** ≥ 5 s

Recommended response	1. Check the health of the Diego Cells by reviewing the logs and looking for errors. 2. Review IaaS console metrics. 3. Inspect the Auctioneer logs to determine if one or more Diego Cells is taking significantly longer to fetch state than other Diego Cells. Relevant log lines have wording like `fetched Diego Cell state`. 4. Pull Diego Brain logs, Diego Cell logs, and Auctioneer logs and contact Support telling them that fetching Diego Cell states is taking too long.

Auctioneer app instance starts

	auctioneer.AuctioneerLRPAuctionsStarted
Description	The number of LRP instances that the Auctioneer successfully placed on Diego Cells. This metric is cumulative over the lifetime of the Auctioneer job. **Use:** Provides a sense of running system activity levels in your environment. Can also give you a sense of how many app instances have been started over time. The measurement VMware recommends, below, can help indicate a significant amount of container churn. However, for capacity planning purposes, it is more helpful to observe deltas over a long time window. This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no app instances being scheduled. **Origin:** Firehose **Type:** Counter (Integer) **Frequency:** During event, during each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	When observing a significant amount of container churn: 1. Look to eliminate explainable causes of temporary churn, such as a deployment or increased developer activity. 2. If container churn appears to continue over an extended period, pull logs from the Diego Brain and BBS node before contacting Support. When observing extended periods of high or low activity trends, scale TAS for VMs components up or down as needed.

Auctioneer task placement failures

	auctioneer.AuctioneerTaskAuctionsFailed

Description	The number of Tasks that the Auctioneer failed to place on Diego Cells. This metric is cumulative over the lifetime of the Auctioneer job. **Use:** Failing Task auctions indicate a lack of resources within your environment and that you likely need to scale. This indicator also increases when the Task is requesting an isolation segment, volume drivers, or a stack that is unavailable, either not deployed or lacking sufficient resources to accept the work. This metric is emitted on event, and therefore gaps in receipt of this metric can be normal during periods of no tasks being scheduled. This error is most common due to capacity issues. For example, if Diego Cells do not have enough resources, or if Diego Cells are going back and forth between a healthy and unhealthy state. **Origin:** Firehose **Type:** Counter (Float) **Frequency:** During event, during each auction
Recommended measurement	Per minute delta averaged over a 5-minute window
Recommended alert thresholds	**Yellow warning:** ≥ 0.5 **Red critical:** ≥ 1
Recommended response	1. In order to best determine the root cause, examine the Auctioneer logs. Depending on the specific error or resource constraint, you may also find a failure reason in the CC API. 2. Investigate the health of Diego Cells. 3. Consider scaling additional Diego Cells using Ops Manager. 4. If scaling Diego Cells does not solve the problem, pull Diego Brain logs and BBS logs for troubleshooting and contact Support for additional troubleshooting. Inform Support that Task auctions are failing.

Diego BBS metrics

These sections describe Diego BBS metrics.

BBS time to run LRP convergence

bbs.ConvergenceLRPDuration

Description	Time in ns that the BBS took to run its LRP convergence pass. **Use:** If the convergence run begins taking too long, apps or tasks may be crashing without restarting. This symptom can also indicate loss of connectivity to the BBS database. **Origin:** Firehose **Type:** Gauge (Integer in ns) **Frequency:** During event, every 30 seconds when LRP convergence runs, emission must be near-constant on a running deployment

Recommended measurement	Maximum over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning:** ≥ 10 s **Red critical:** ≥ 20 s
Recommended response	1. Check BBS logs for errors. 2. Try vertically scaling the BBS VM resources up. For example, add more CPUs or memory depending on its `system.cpu`/`system.memory` metrics. 3. Consider vertically scaling the TAS for VMs backing database, if `system.cpu` and `system.memory` metrics for the database instances are high. 4. If that does not solve the issue, pull the BBS logs and contact Support for additional troubleshooting.

BBS time to handle requests

	bbs.RequestLatency
Description	The maximum observed latency time over the past 60 seconds that the BBS took to handle requests across all its API endpoints. Diego now aggregates this metric to emit the maximum value observed over 60 seconds. **Use:** If this metric rises, the TAS for VMs API is slowing. Response to certain cf CLI commands is slow if request latency is high. **Origin:** Firehose **Type:** Gauge (Integer in ns) **Frequency:** 60 s
Recommended measurement	Average over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning:** ≥ 5 s **Red critical:** ≥ 10 s
Recommended response	1. Check CPU and memory statistics in Ops Manager. 2. Check BBS logs for faults and errors that can indicate issues with BBS. 3. Try scaling the BBS VM resources up. For example, add more CPUs and memory depending on its `system.cpu`/`system.memory` metrics. 4. Consider vertically scaling the TAS for VMs backing database, if `system.cpu` and `system.memory` metrics for the database instances are high. 5. If the above steps do not solve the issue, collect a sample of the Diego Cell logs from the BBS VMs and contact Support to troubleshoot further.

Cloud Controller and Diego in sync

	bbs.Domain.cf-apps

Description	Indicates if the `cf-apps` Domain is up-to-date, meaning that TAS for VMs app requests from Cloud Controller are synchronized to `bbs.LRPsDesired` (Diego-desired AIs) for execution. • `1` means `cf-apps` Domain is up-to-date • No data received means `cf-apps` Domain is not up-to-date **Use:** If the `cf-apps` Domain does not stay up-to-date, changes requested in the Cloud Controller are not guaranteed to propagate throughout the system. If the Cloud Controller and Diego are out of sync, then apps running could vary from those desired. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** 30 s
Recommended measurement	Value over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** ≠ 1 The threshold value VMware recommends represents a state where an up-to-date metric `1` has not been received for the entire 5-minute window.
Recommended response	1. Check the BBS and Clock Global (Cloud Controller clock) logs. 2. If the problem continues, pull the BBS logs and Clock Global (Cloud Controller clock) logs and contact Support to say that the `cf-apps` domain is not being kept fresh.

More app instances than expected

	bbs.LRPsExtra
Description	Total number of LRP instances that are no longer desired but still have a BBS record. When Diego wants to add more apps, the BBS sends a request to the Auctioneer to spin up additional LRPs. LRPsExtra is the total number of LRP instances that are no longer desired but still have a BBS record. **Use:** If Diego has more LRPs running than expected, there may be problems with the BBS. Deleting an app with many instances can temporarily spike this metric. However, a sustained spike in `bbs.LRPsExtra` is unusual and must be investigated. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 5 **Red critical:** ≥ 10

Recommended response	1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Support.

Fewer app instances than expected

	bbs.LRPsMissing
Description	Total number of LRP instances that are desired but have no record in the BBS. When Diego wants to add more apps, the BBS sends a request to the Auctioneer to spin up additional LRPs. LRPsMissing is the total number of LRP instances that are desired but have no BBS record. **Use:** If Diego has less LRP running than expected, there may be problems with the BBS. An app push with many instances can temporarily spike this metric. However, a sustained spike in `bbs.LRPsMissing` is unusual and must be investigated. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** 30 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 5 **Red critical:** ≥ 10
Recommended response	1. Review the BBS logs for proper operation or errors, looking for detailed error messages. 2. If the condition persists, pull the BBS logs and contact Support.

Crashed app instances

	bbs.CrashedActualLRPs
Description	Total number of LRP instances that have crashed. **Use:** Indicates how many instances in the deployment are in a crashed state. An increase in `bbs.CrashedActualLRPs` can indicate several problems, from a bad app with many instances associated, to a platform issue that is resulting in app crashes. Use this metric to help create a baseline for your deployment. After you have a baseline, you can create a deployment-specific alert to notify of a spike in crashes above the trend line. Tune alert values to your deployment. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** 30 s
Recommended measurement	Average over the last 5 minutes

Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. Look at the BBS logs for apps that are crashing and at the Diego Cell logs to see if the problem is with the apps themselves, rather than a platform issue. 2. Before contacting Support, pull the BBS logs and, if particular apps are the problem, pull the logs from their Diego Cells too.

Running app instances, rate of change

1hr average of bbs.LRPsRunning – prior 1hr average of bbs.LRPsRunning

Description	Rate of change in app instances being started or stopped on the platform. It is derived from `bbs.LRPsRunning` and represents the total number of LRP instances that are running on Diego Cells. **Use:** Delta reflects upward or downward trend for app instances started or stopped. Helps to provide a picture of the overall growth trend of the environment for capacity planning. You may want to alert on delta values outside of the expected range. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** During event, emission must be constant on a running deployment
Recommended measurement	derived=(1-hour average of `bbs.LRPsRunning` – prior 1-hour average of `bbs.LRPsRunning`)
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	Scale components as necessary.

BBS master elected

bbs.BBSMasterElected

Description	Indicates when there is a BBS master election. A BBS master election takes place when a BBS instance has taken over as the active instance. A value of `1` is emitted when the election takes place. **Use:** This metric emits when a redeployment of the BBS occurs. If this metric is emitted frequently outside of a deployment, this might be a signal of underlying problems that can be investigated. If the active BBS is continually changing, this can cause app push downtime. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** On event
Recommended measurement	*N/A*, the most effective visualization is as a stacked bar chart

Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** *N/A*
Recommended response	1. Check the BBS logs. 2. Check the BBS VM load average and instance group size. 3. Check the health of the connection to the backing SQL database and network latency.

Diego Cell metrics

These sections describe Diego Cell metrics.

Remaining memory available — Diego Cell memory chunks available

rep.CapacityRemainingMemory

Description	Remaining amount of memory in MiB available for this Diego Cell to allocate to containers. **Use:** Indicates the available Diego Cell memory. Insufficient Diego Cell memory can prevent pushing and scaling apps. The strongest operational value of this metric is to understand a deployment's average app size and monitor/alert on ensuring that at least some Cells have large enough capacity to accept standard app size pushes. For example, if pushing a 4 GB app, Diego would have trouble placing that app if there is no one Diego Cell with sufficient capacity of 4 GB or greater. As an example, VMware Cloud Ops uses a standard of 4 GB, and computes and monitors for the number of Diego Cells with at least 4 GB free. When the number of Diego Cells with at least 4 GB falls below a defined threshold, this is a scaling indicator alert to increase capacity. This *free chunk* count threshold must be tuned to the deployment size and the standard size of apps being pushed to the deployment. **Origin:** Firehose **Type:** Gauge (Integer in MiB) **Frequency:** 60 s

Recommended measurement	**For alerting**: 1. Determine the size of a standard app in your deployment. This is the suggested value to calculate *free chunks* of Remaining Memory by. 2. Create a script/tool that can iterate through each Diego Cell and do the following: 1. Pull the `rep.CapacityRemainingMemory` metric for each Diego Cell. 2. Divide the values received by 1000 to get the value in GB (if desired threshold is GB-based). 3. Compare recorded values to your minimum capacity threshold, and count the number of Diego Cells that have equal or greater than the desired amount of *free chunk* space. 3. Determine a desired scaling threshold based on the minimum amount of *free chunks* that are acceptable in this deployment given historical trends. 4. Set an alert to indicate the need to scale Diego Cell memory capacity when the value falls below the desired threshold number. **For visualization purposes**: Looking at this metric (`rep.CapacityRemainingMemory`) as a minimum value per Diego Cell has more informational value than alerting value. It can be an interesting heatmap visualization, showing average variance and density over time.
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. Assign more resources to the Diego Cells or assign more Diego Cells. 2. Scale additional Diego Cells using Ops Manager.
Alternative Metric	If you are using Healthwatch, VMware recommends the metric `healthwatch.Diego.AvailableFreeChunks`for this purpose. For more information, see Number of Available Free Chunks of Memory in *Healthwatch Metrics* in the Healthwatch documentation.

Remaining memory available — Overall remaining memory available

rep.CapacityRemainingMemory (Alternative Use)	
Description	Remaining amount of memory in MiB available for this Diego Cell to allocate to containers. **Use:** Can indicate low memory capacity overall in the platform. Low memory can prevent app scaling and new deployments. The overall sum of capacity can indicate that you need to scale the platform. Observing capacity consumption trends over time helps with capacity planning. **Origin:** Firehose **Type:** Gauge (Integer in MiB) **Frequency:** 60 s
Recommended measurement	Minimum over the last 5 minutes divided by 1024 (across all instances)

Recommended alert thresholds	**Yellow warning:** ≤ 64 GB **Red critical:** ≤ 32 GB
Recommended response	1. Assign more resources to the Diego Cells or assign more Diego Cells. 2. Scale additional Diego Cells using Ops Manager.
Alternative Metric	If you are using Healthwatch, VMware recommends the metric `healthwatch.Diego.AvailableFreeChunks`for this purpose. For more information, see Number of Available Free Chunks of Memory in *Healthwatch Metrics* in the Healthwatch documentation.

Remaining disk available — Diego Cell disk chunks available

rep.CapacityRemainingDisk	
Description	Remaining amount of disk in MiB available for this Diego Cell to allocate to containers. **Use:** Indicates the available Diego Cell disk. Insufficient free disk on Diego Cells prevents the staging or starting of apps or tasks, resulting in error messages like `ERR Failed to stage app: insufficient resources`. Because Diego fails to stage without at least 6 GB free, unreserved disk space on a given Diego Cell, the strongest operational value of this metric is to ensure that at least some Diego Cells have a large enough disk capacity to support the staging of apps and tasks. VMware recommends computing and monitoring for the number of Diego Cells with at least 6 GB Disk free. When the number of Diego Cells with at least 6 GB falls below a defined threshold, this is a scaling indicator alert to increase capacity. The alerting threshold value for the amount of *free chunks* of Disk must be tuned to the deployment size and the standard size of apps being pushed to the deployment. **Origin:** Firehose **Type:** Gauge (Integer in MiB) **Frequency:** 60 s

Recommended measurement	**For alerting:** 1. Because Diego fails to stage without at least 6 GB free, this is the suggested minimum value to calculate *free chunks* of Remaining Disk by. 2. Create a script/tool that can iterate through each Diego Cell and do the following: 1. Pull the `rep.CapacityRemainingDisk` metric for each Diego Cell. 2. Divide the values received by 1000 to get the value in GB (if desired threshold is GB-based). 3. Compare recorded values to your minimum capacity threshold, and count the number of Diego Cells that have equal or greater than the desired amount of *free chunk* space. 3. Determine a desired scaling threshold based on the minimum amount of *free chunks* that are acceptable in this deployment given historical trends. 4. Set an alert to indicate the need to scale Diego Cell disk capacity when the value falls below the desired threshold number. **For visualization purposes**: Looking at this metric (`rep.CapacityRemainingDisk`) as a minimum value per Diego Cell has more informational value than alerting value. It can be an interesting heatmap visualization, showing average variance and density over time.
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. Assign more resources to the Diego Cells or assign more Diego Cells. 2. Scale additional Diego Cells using Ops Manager.
Alternative Metric	If you are using Healthwatch, VMware recommends the metric `healthwatch.Diego.AvailableFreeChunks`for this purpose. For more information, see Number of Available Free Chunks of Memory in *Healthwatch Metrics* in the Healthwatch documentation.

Remaining disk available - Overall remaining disk available

rep.CapacityRemainingDisk
(Alternative Use)

Description	Remaining amount of disk in MiB available for this Diego Cell to allocate to containers. **Use:** Low disk capacity can prevent app scaling and new deployments. Because Diego staging Tasks can fail without at least 6 GB free, the red threshold VMware recommends is based on the minimum disk capacity across the deployment falling below 6 GB in the previous 5 minutes. It can also be meaningful to assess how many chunks of free disk space are above a given threshold, similar to `rep.CapacityRemainingMemory`. **Origin:** Firehose **Type:** Gauge (Integer in MiB) **Frequency:** 60 s
Recommended measurement	Minimum over the last 5 minutes divided by 1024 (across all instances)

Recommended alert thresholds	**Yellow warning:** ≤ 12 GB **Red critical:** ≤ 6 GB
Recommended response	1. Assign more resources to the Diego Cells or assign more Diego Cells. 2. Scale additional Diego Cells using Ops Manager.
Alternative Metric	If you are using Healthwatch, VMware recommends the metric `healthwatch.Diego.AvailableFreeChunks`for this purpose. For more information, see Number of Available Free Chunks of Memory in *Healthwatch Metrics* in the Healthwatch documentation.

Diego Cell rep time to sync

rep.RepBulkSyncDuration	
Description	Time in ns that the Diego Cell Rep took to sync the ActualLRPs that it claimed with its actual Garden containers. **Use:** Sync times that are too high can indicate issues with the BBS. **Origin:** Firehose **Type:** Gauge (Float in ns) **Frequency:** 30 s
Recommended measurement	Maximum over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning:** ≥ 5 s **Red critical:** ≥ 10 s
Recommended response	1. Investigate BBS logs for faults and errors. 2. If a particular Diego Cell or Diego Cells appear problematic, pull logs for the Diego Cells and the BBS logs before contacting Support.

Garden health check failed

rep.GardenHealthCheckFailed

Description	The Diego Cell periodically checks its health against the Garden back end. For Diego Cells, `0` means healthy, and `1` means unhealthy. **Use:** Set an alert for further investigation if multiple unhealthy Diego Cells are detected in the given time window. If one Diego Cell is impacted, it does not participate in auctions, but end-user impact is usually low. If multiple Diego Cells are impacted, this can indicate a larger problem with Diego, and can be considered a more critical investigation need. Suggested alert threshold based on multiple unhealthy Diego Cells in the given time window. Although end-user impact is usually low if only one Diego Cell is impacted, this must still be investigated. Particularly in a lower capacity environment, this situation could result in negative end-user impact if left unresolved. **Origin:** Firehose **Type:** Gauge (Float, 0-1) **Frequency:** 30 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** = 1 **Red critical:** > 1
Recommended response	1. Investigate Diego Cell servers for faults and errors. 2. If a particular Diego Cell or Diego Cells appear problematic: 1. Determine a time interval during which the metrics from the Diego Cell changed from healthy to unhealthy. 2. Pull the logs that the Diego Cell generated over that interval. The Diego Cell ID is the same as the BOSH instance ID. 3. Pull the BBS logs over that same time interval. 4. Contact Support. 3. As a last resort, if you cannot wait for Support, it sometimes helps to recreate the Diego Cell by running `bosh recreate`. For information about the `bosh recreate` command syntax, see Deployments in *Commands* in the BOSH documentation. ⚠ **Caution:** Recreating a Diego Cell destroys its logs. To enable a root cause analysis of the Diego Cell's problem, save out its logs before running `bosh recreate`.

Diego Locket metrics

These sections describe Diego Locket metrics.

Active Locks

	locket.ActiveLocks

Description	Total count of how many locks the system components are holding. **Use:** If the ActiveLocks count is not equal to the expected value, there is likely a problem with Diego. **Origin:** Firehose **Type:** Gauge **Frequency:** 60 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** ≠ 5
Recommended response	1. Run `monit status` to inspect for failing processes. 2. If there are no failing processes, then review the logs for the components using the Locket service: BBS, Auctioneer, TPS Watcher, Routing API, and Clock Global (Cloud Controller clock). Look for indications that only one of each component is active at a time. 3. Focus triage on the BBS first: ◦ A healthy BBS shows obvious activity around starting or claiming LRPs. ◦ An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer. ◦ Reference the BBS-level Locket metric `bbs.LockHeld`. A value of 0 indicates Locket issues at the BBS level. For more information, see Locks Held by BBS. 4. If the BBS appears healthy, then check the Auctioneer to ensure it is processing auction payloads. ◦ Recent logs for Auctioneer must show all but one of its instances are currently waiting on locks, and the active Auctioneer must show a record of when it last attempted to execute work. This attempt must correspond to app development activity, such as `cf push`. ◦ Reference the Auctioneer-level Locket metric `auctioneer.LockHeld`. A value of 0 indicates Locket issues at the Auctioneer level. For more information, see Locks Held by Auctioneer. 5. The TPS Watcher is primarily active when app instances crash. Therefore, if the TPS Watcher is suspected, review the most recent logs. 6. If you are unable to resolve on-going excessive active locks, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Support.

Locks Held by BBS

bbs.LockHeld

Description	Whether a BBS instance holds the expected BBS lock (in Locket). 1 means the active BBS server holds the lock, and 0 means the lock was lost. **Use:** This metric is complimentary to Active Locks, and it offers a BBS-level version of the Locket metrics. Although it is emitted per BBS instance, only 1 active lock is held by BBS. Therefore, the expected value is 1. The metric may occasionally be 0 when the BBS instances are performing a leader transition, but a prolonged value of 0 indicates an issue with BBS. **Origin:** Firehose **Type:** Gauge **Frequency:** Periodically
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** ≠ 1
Recommended response	1. Run `monit status` on the Diego database VM to check for failing processes. 2. If there are no failing processes, then review the logs for BBS. ◦ A healthy BBS shows obvious activity around starting or claiming LRPs. ◦ An unhealthy BBS leads to the Auctioneer showing minimal or no activity. The BBS sends work to the Auctioneer. 3. If you are unable to resolve the issue, pull logs from the Diego BBS, which include the Locket service component logs, and contact Support.

Locks held by auctioneer

	auctioneer.LockHeld
Description	Whether an Auctioneer instance holds the expected Auctioneer lock (in Locket). 1 means the active Auctioneer holds the lock, and 0 means the lock was lost. **Use:** This metric is complimentary to Active Locks, and it offers an Auctioneer-level version of the Locket metrics. Although it is emitted per Auctioneer instance, only 1 active lock is held by Auctioneer. Therefore, the expected value is 1. The metric may occasionally be 0 when the Auctioneer instances are performing a leader transition, but a prolonged value of 0 indicates an issue with Auctioneer. **Origin:** Firehose **Type:** Gauge **Frequency:** Periodically
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** ≠ 1

Recommended response	1. Run `monit status` on the Diego Database VM to check for failing processes. 2. If there are no failing processes, then review the logs for Auctioneer. ◦ Recent logs for Auctioneer must show all but one of its instances are currently waiting on locks, and the active Auctioneer must show a record of when it last attempted to execute work. This attempt must correspond to app development activity, such as `cf push`. 3. If you are unable to resolve the issue, pull logs from the Diego BBS and Auctioneer VMs, which includes the Locket service component logs, and contact Support.

Active presences

locket.ActivePresences

Description	Total count of active presences. Presences are defined as the registration records that the Diego Cells maintain to advertise themselves to the platform. **Use:** If the Active Presences count is far from the expected, there might be a problem with Diego. The number of active presences varies according to the number of Diego Cells deployed. Therefore, during purposeful scale adjustments to TAS for VMs, this alerting threshold can be adjusted. Establish an initial threshold by observing the historical trends for the deployment over a brief period of time, Increase the threshold as more Diego Cells are deployed. During a rolling deploy, this metric shows variance during the BOSH lifecycle when Diego Cells are evacuated and restarted. Tolerable variance is within the bounds of the BOSH maximum in-flight range for the instance group. **Origin:** Firehose **Type:** Gauge **Frequency:** 60 s
Recommended measurement	Maximum over the last 15 minutes
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. Ensure that the variance is not the result of an active rolling deploy. Also ensure that the alert threshold is appropriate to the number of Diego Cells in the current deployment. 2. Run `monit status` to inspect for failing processes. 3. If there are no failing processes, then review the logs for the components using the Locket service itself on Diego BBS instances. 4. If you are unable to resolve the problem, pull the logs from the Diego BBS, which include the Locket service component logs, and contact Support.

Diego Route Emitter metrics

These sections describe Diego Route Emitter metrics.

Route Emitter time to sync

	route_emitter.RouteEmitterSyncDuration
Description	Time in ns that the active Route Emitter took to perform its synchronization pass. **Use:** Increases in this metric indicate that the Route Emitter might have trouble maintaining an accurate routing table to broadcast to the Gorouters. Tune alerting values to your deployment based on historical data and adjust based on observations over time. The suggested starting point is ≥ 5 for the yellow threshold and ≥ 10 for the critical threshold. **Origin:** Firehose **Type:** Gauge (Float in ns) **Frequency:** 60 s
Recommended measurement	Maximum, per job, over the last 15 minutes divided by 1,000,000,000
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	If all or many jobs showing as impacted, there is likely an issue with Diego. 1. Investigate the Route Emitter and Diego BBS logs for errors. 2. Verify that app routes are functional by making a request to an app, pushing an app and pinging it, or if applicable, checking that your smoke tests have passed. If one or a few jobs showing as impacted, there is likely a connectivity issue and the impacted job must be investigated further.

TAS for VMs MySQL KPIs

These sections describe TAS for VMs MySQL KPIs.

When TAS for VMs uses an internal MySQL database, as configured in the **Databases** pane of the TAS for VMs tile, the database cluster generates KPIs as described below.

> **Note:** This section assumes you are using the **Internal Databases - MySQL - Percona XtraDB Cluster** option as your system database.

Server Availability

	/mysql/available

Description	If the MySQL Server is currently responding to requests. This indicates if the component is available. **Use**: If the server does not emit heartbeats, it is offline. **Origin**: Doppler/Firehose **Type**: boolean **Frequency**: 30 s
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	**Yellow warning**: N/A **Red critical**: < 1
Recommended response	Check the MySQL Server logs for errors. You can find the instance by targeting your MySQL deployment with BOSH and inspecting logs for the instance. For more information, see Failing Jobs and Unhealthy Instances. If your service plan is an highly available (HA) cluster, you can also run `mysql-diag` to check logs for errors.

Persistent Disk Used

/mysql/system/persistent_disk_used_percent

Description	The percentage of disk used on the persistent file system. **Use**: MySQL cannot function correctly if there is not sufficient free space on the file systems. Use these metrics to ensure that you have disks large enough for your user base. **Origin**: Doppler/Firehose **Type**: Percent **Frequency**: 30 s (default)
Recommended measurement	Maximum of persistent disk used of all of nodes
Recommended alert thresholds	**Single Node and Leader Follower:** • **Yellow warning:** > 25% • **Red critical:** > 30% **Highly Available Cluster:** • **Yellow warning:** > 80% • **Red critical:** > 90%
Recommended response	Upgrade the service instance to a plan with larger disk capacity. For Tanzu SQL for VMs v2.9 and later, if you set the `optimize_for_short_words` parameter to `true`, then see Troubleshooting VMware Tanzu SQL with MySQL for VMs before upgrading the service.

Ephemeral Disk Used

	/mysql/system/ephemeral_disk_used_percent
Description	The percentage of disk used on the ephemeral file system. **Use**: MySQL cannot function correctly if there is not sufficient free space on the file systems. Use these metrics to ensure that you have disks large enough for your user base. **Origin**: Doppler/Firehose **Type**: Percent **Frequency**: 30 s (default)
Recommended measurement	Maximum disk used of all nodes
Recommended alert thresholds	**Yellow warning**: > 80% **Red critical**: > 95%
Recommended response	Upgrade the service instance to a plan with larger disk capacity.

CPU Utilization Percent

	/mysql/performance/cpu_utilization_percent
Description	CPU time being consumed by the MySQL service. **Use**: A node that experiences context switching or high CPU usage becomes unresponsive. This also affects the ability of the node to report metrics. **Origin**: Doppler/Firehose **Type**: Percent **Frequency**: 30 s (default)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning:** > 80 **Red critical:** > 90
Recommended response	Determine what is using so much CPU. If it is from normal processes, update the service instance to use a plan with larger CPU capacity.

Connections

/mysql/variables/max_connections

/p.mysql/net/max_used_connections

Description	The maximum number of connections used over the maximum permitted number of simultaneous client connections. **Use**: If the number of connections drastically changes or if apps are unable to connect, there might be a network or app issue. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30 s
Recommended measurement	`max_used_connections` / `max_connections`
Recommended alert thresholds	**Yellow warning**: > 80 % **Red critical**: > 90 %
Recommended response	If this measurement mets or exceeds 80% with exponential growth, monitor app usage to ensure everything is working. When approaching 100% of max connections, apps may be experiencing times when they cannot connect to the database. The connections/second for a service instance vary based on app instances and app utilization.

Queries Delta

/mysql/performance/queries_delta

Description	The number of statements executed by the server over the last 30 seconds. **Use**: The server should always be processing some queries. If the server does not process any queries, the server is non-functional. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30 s
Recommended measurement	Average over last 2 minutes
Recommended alert thresholds	**Red critical**: 0
Recommended response	Investigate the MySQL server logs, such as the audit log, to understand why query rate changed and determine appropriate action.

Highly Available Cluster WSREP Ready

/mysql/galera/wsrep_ready

Description	Shows whether each cluster node can accept queries. Returns only 0 or 1. When this metric is 0, almost all queries to that node fail with the error: `ERROR 1047 (08501) Unknown Command` **Use:** Discover when nodes of a cluster have been unable to communicate and accept transactions. **Origin:** Doppler/Firehose **Type:** boolean **Frequency:** 30 s (default)
Recommended measurement	Average of values of each cluster node, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 1 **Red critical:** 0 (cluster is down)
Recommended response	• Run `mysql-diag` and check the MySQL Server logs for errors. • Ensure no infrastructure event is affecting intra-cluster communication. • Ensure that `wsrep_ready` is not set to off by using the query: `SHOW STATUS LIKE 'wsrep_ready';`.

Highly Available Cluster WSREP Cluster Size

/mysql/galera/wsrep_cluster_size

Description	The number of cluster nodes with which each node is communicating normally. **Use:** When running in a multi-node configuration, this metric indicates if each member of the cluster is communicating normally with all other nodes. **Origin:** Doppler/Firehose **Type:** count **Frequency:** 30 s (default)
Recommended measurement	(Average of the values of each node / cluster size), over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 3 (availability compromised) **Red critical:** < 1 (cluster unavailable)
Recommended response	Run `mysql-diag` and check the MySQL Server logs for errors.

Highly Available Cluster WSREP Cluster Status

/mysql/galera/wsrep_cluster_status

Description	Shows the primary status of the cluster component that the node is in. Values are: - `Primary`: Node has a quorum. - `Non-primary`: Node has lost a quorum. - `Disconnected`: Node is unable to connect to other nodes. **Use:** Any value other than `Primary` indicates that the node is part of a non-operational component. This occurs in cases of multiple membership changes that result in a loss of quorum. **Origin:** Doppler/Firehose **Type:** integer (see above) **Frequency:** 30 s (default)
Recommended measurement	Sum of each of the nodes, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** < 3 **Red critical:** < 1
Recommended response	- Check node status to ensure that they are all in working order and able to receive write-sets - Run `mysql-diag` and check the MySQL Server logs for errors

Gorouter metrics

These sections describe Gorouter metrics.

Router file descriptors

gorouter.file_descriptors

Description	The number of file descriptors currently used by the Gorouter job. **Use:** Indicates an impending issue with the Gorouter. Without proper mitigation, it is possible for an unresponsive app to eventually exhaust available Gorouter file descriptors and cause route starvation for other apps running on TAS for VMs. Under heavy load, this unmitigated situation can also result in the Gorouter losing its connection to NATS and all routes being pruned. While a drop in `gorouter.total_routes` or an increase in `gorouter.ms_since_last_registry_update` helps to surface that the issue might already be occurring, alerting on `gorouter.file_descriptors` indicates that such an issue is impending. The Gorouter limits the number of file descriptors to 100,000 per job. Once the limit is met, the Gorouter is unable to establish any new connections. To reduce the risk of DDoS attacks, VMware recommends doing one or both of the following: • Within TAS for VMs, set **Maximum connections per back end** to define how many requests can be routed to any particular app instance. This prevents a single app from using all Gorouter connections. The value specified can be determined by the operator based on the use cases for that foundation. • Add rate limiting at the load balancer level. **Origin:** Firehose **Type:** Gauge **Frequency:** 5 s
Recommended measurement	Maximum, per Gorouter job, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** 50,000 per job **Red critical:** 60,000 per job
Recommended response	1. Identify which app(s) are requesting excessive connections and resolve the impacting issues with these apps. 2. If the above mitigation steps have not already been taken, do so. 3. Consider adding more Gorouter VM resources to increase the number of available file descriptors.

Router exhausted connections

gorouter.backend_exhausted_conns

Description	The lifetime number of requests that have been rejected by the Gorouter VM due to the `Max Connections Per Backend` limit being reached across all tried back ends. The limit controls the number of concurrent TCP connections to any particular app instance and is configured within TAS for VMs. **Use:** Indicates that TAS for VMs is mitigating risk to other apps by self-protecting the platform against one or more unresponsive apps. Increases in this metric indicate the need to investigate and resolve issues with potentially unresponsive apps. A rapid rate of change upward is concerning and can be assessed further. **Origin:** Firehose **Type:** Counter (Integer) **Frequency:** 5 s
Recommended measurement	Maximum delta per minute, per Gorouter job, over a 5-minute window
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. If `gorouter.backend_exhausted_conns` spikes, first look to the Router Throughput metric `gorouter.total_requests` to determine if this measure is high or low in relation to normal bounds for this deployment. 2. If Router Throughput appears within normal bounds, it is likely that `gorouter.backend_exhausted_conns` is spiking due to an unresponsive app, possibly due to app code issues or underlying app dependency issues. To help determine the problematic app, look in access logs for repeated calls to one app. Then proceed to troubleshoot this app accordingly. 3. If Router Throughput also shows unusual spikes, the cause of the increase in `gorouter.backend_exhausted_conns` spikes is likely external to the platform. Unusual increases in load may be due to expected business events driving additional traffic to apps. Unexpected increases in load may indicate a DDoS attack risk.

Router throughput

gorouter.total_requests

Description	The lifetime number of requests completed by the Gorouter VM, emitted per Gorouter instance **Use:** The aggregation of these values across all Gorouters provide insight into the overall traffic flow of a deployment. Unusually high spikes, if not known to be associated with an expected increase in demand, could indicate a DDoS risk. For performance and capacity management, consider this metric a measure of router throughput per job, converting it to requests-per-second, by looking at the delta value of `gorouter.total_requests` and deriving back to 1s, or `gorouter.total_requests.delta)/5`, per Gorouter instance. This helps you see trends in the throughput rate that indicate a need to scale the Gorouter instances. Use the trends you observe to tune the threshold alerts for this metric. **Origin:** Firehose **Type:** Counter (Integer) **Frequency:** 5 s

Recommended measurement	Average over the last 5 minutes of the derived per second calculation
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	For optimizing the Gorouter, consider the requests-per-second derived metric in the context of router latency and Gorouter VM CPU utilization. From performance and load testing of the Gorouter, VMware has observed that at approximately 2500 simple requests per second, latency can begin to increase. This number changes based on your traffic profile and VM capabilities. To increase throughput and maintain low latency, scale the Gorouters either horizontally or vertically and ensure that the `system.cpu.user` metric for the Gorouter stays in the suggested range of 60-70% CPU Utilization. For more information about the `system.cpu.user` metric, see VM CPU Utilization.

Router handling latency

	gorouter.latency
Description	The time in milliseconds that represents the length of a request from the Gorouter's point of view. This timing starts when Gorouter recieves a request and stops when Gorouter finishes processing the response from the app. Long uploads, downloads, or app responses increase this time. This metric includes the request time to all back end endpoints, including both apps and routable system components like Cloud Controller and UAA. **Use:** Indicates the traffic profile of TAS for VMs. An alert value on this metric must be tuned to the specifics of the deployment and its underlying network considerations; a suggested starting point is 100 ms. **Origin:** Firehose **Type:** Gauge (Float in ms) **Frequency:** Emitted per Gorouter request, emission must be constant on a running deployment
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	Extended periods of high latency can point to several factors. The Gorouter latency measure includes network and back end latency impacts as well. 1. First inspect logs for network issues and indications of misbehaving back ends. 2. If it appears that the Gorouter needs to scale due to ongoing traffic congestion, do not scale on the latency metric alone. You can also look at the CPU utilization of the Gorouter VMs and keep it within a maximum 60-70% range. 3. Resolve high utilization by scaling the Gorouter. 4. Follow steps in the doc Troubleshooting Slow Requests in CF.

Time since last route register received

	gorouter.ms_since_last_registry_update
Description	Time in milliseconds since the last route register was received, emitted per Gorouter instance **Use:** Indicates if routes are not being registered to apps correctly. **Origin:** Firehose **Type:** Gauge (Float in ms) **Frequency:** 30 s
Recommended measurement	Maximum over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** > 30,000 This threshold is suitable for normal platform usage. It alerts if it has been at least 30 seconds since the Gorouter last received a message from an app.
Recommended response	1. Search the Gorouter and Route Emitter logs for connection issues to NATS. 2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 3. Look more broadly at the health of all VMs, particularly Diego-related VMs. 4. If problems persist, pull the Gorouter and Route Emitter logs and contact Support to say there are consistently long delays in route registry.

Router Error: 502 bad gateway

	gorouter.bad_gateways
Description	The lifetime number of bad gateways, or 502 responses, from the Gorouter itself, emitted per Gorouter instance. The Gorouter emits a 502 bad gateway error when it has a route in the routing table and, in attempting to make a connection to the back end, finds that the back end does not exist. **Use:** Indicates that route tables might be stale. Stale routing tables suggest an issue in the route register management plane, which indicates that something has likely changed with the locations of the containers. Always investigate unexpected increases in this metric. **Origin:** Firehose **Type:** Count (Integer, Lifetime) **Frequency:** 5 s
Recommended measurement	Maximum delta per minute over a 5-minute window
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic

Recommended response	1. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS. 2. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 3. Look broadly at the health of all VMs, particularly Diego-related VMs. 4. If problems persist, pull Gorouter and Route Emitter logs and contact Support to say there has been an unusual increase in Gorouter bad gateway responses.

Router Error: server error

	gorouter.responses.5xx
Description	The lifetime number of requests completed by the Gorouter VM for HTTP status family 5xx, server errors, emitted per Gorouter instance. **Use:** A repeatedly crashing app is often the cause of a big increase in 5xx responses. However, response issues from apps can also cause an increase in 5xx responses. Always investigate an unexpected increase in this metric. **Origin:** Firehose **Type:** Counter (Integer) **Frequency:** 5 s
Recommended measurement	Maximum delta per minute over a 5-minute window
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. Look for out-of-memory errors and other app-level errors. 2. As a temporary measure, ensure that the troublesome app is scaled to more than one instance.

Number of Gorouter routes registered

	gorouter.total_routes

Description	The current total number of routes registered with the Gorouter, emitted per Gorouter instance **Use:** The aggregation of these values across all Gorouters indicates uptake and gives a picture of the overall growth of the environment for capacity planning. VMware also recommends alerting on this metric if the number of routes falls outside of the normal range for your deployment. Dramatic decreases in this metric volume might indicate a problem with the route registration process, such as an app outage, or that something in the route register management plane has failed. If visualizing these metrics on a dashboard, `gorouter.total_routes` can be helpful for visualizing dramatic drops. However, for alerting purposes, the `gorouter.ms_since_last_registry_update` metric is more valuable for quicker identification of Gorouter issues. Alerting thresholds for `gorouter.total_routes` must focus on dramatic increases or decreases out of expected range. **Origin:** Firehose **Type:** Gauge (Float) **Frequency:** 30 s
Recommended measurement	5-minute average of the per second delta
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	1. For capacity needs, scale up or down the Gorouter VMs as necessary. 2. For significant drops in current total routes, see the `gorouter.ms_since_last_registry_update` metric for additional context. For more information, see Time Since Last Route Register Received. 3. Check the Gorouter and Route Emitter logs to see if they are experiencing issues when connecting to NATS. 4. Check the BOSH logs to see if the NATS, Gorouter, or Route Emitter VMs are failing. 5. Look broadly at the health of all VMs, particularly Diego-related VMs. 6. If problems persist, pull the Gorouter and Route Emitter logs and contact Support.

UAA metrics

UAA throughput

	uaa.requests.global.completed.count

Description	The lifetime number of requests completed by the UAA VM, emitted per UAA instance. This number includes health checks. **Use:** For capacity planning purposes, the aggregation of these values across all UAA instances can provide insight into the overall load that UAA is processing. VMware recommends alerting on unexpected spikes per UAA instance. Unusually high spikes, if they are not associated with an expected increase in demand, could indicate a DDoS risk and can be investigated. For performance and capacity management, look at the UAA Throughput metric as either a requests-completed-per-second or requests-completed-per-minute rate to determine the throughput per UAA instance. This helps you see trends in the throughput rate that might indicate a need to scale UAA instances. Use the trends you observe to tune the threshold alerts for this metric. From performance and load testing of UAA, VMware has observed that while UAA endpoints can have different throughput behavior, once throughput reaches its peak value per VM, it stays constant and latency increases. **Origin:** Firehose **Type:** Gauge (Integer), emitted value increments over the lifetime of the VM like a counter **Frequency:** 5 s
Recommended measurement	Average over the last 5 minutes of the derived requests-per-second or requests-per-minute rate, per instance
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	For optimizing UAA, consider this metric in the context of UAA request latency and UAA VM CPU utilization. To increase throughput and maintain low latency, scale the UAA VMs horizontally by editing the number of your **UAA** VM instances in the **Resource Config** pane of the TAS for VMs tile and ensure that the `system.cpu.user` metric for UAA is not sustained in the suggested range of 80-90% maximum CPU utilization. For more information, see UAA Request Latency and UAA VM CPU Utilization in *Key Capacity Scaling Indicators*.

UAA request latency

gorouter.latency.uaa

Description	Time in milliseconds that UAA took to process a request that the Gorouter sent to UAA endpoints. **Use:** Indicates how responsive UAA has been to requests sent from the Gorouter. Some operations might take longer to process, such as creating bulk users and groups. It is important to correlate latency observed with the endpoint and evaluate this data in the context of overall historical latency from that endpoint. Unusual spikes in latency could indicate the need to scale UAA VMs. This metric is emitted only for the routers serving the UAA system component and is not emitted per isolation segment even if you are using isolated routers. **Origin:** Firehose **Type:** Gauge (Float in ms) **Frequency:** Emitted per Gorouter request to UAA
Recommended measurement	Maximum, per job, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	Latency depends on the endpoint and operation being used. It is important to correlate the latency with the endpoint and evaluate this data in the context of the historical latency from that endpoint. 1. Inspect which endpoints requests are hitting. Use historical data to determine if the latency is unusual for that endpoint. For a list of UAA endpoints, see the UAA API documentation. 2. If it appears that UAA needs to be scaled due to ongoing traffic congestion, do not scale based on the latency metric alone. You must also ensure that the `system.cpu.user` metric for UAA stays in the suggested range of 80-90% maximum CPU utilization. 3. Resolve high utilization by scaling UAA VMs horizontally. To scale UAA, go to the **Resource Config** pane of the TAS for VMs tile and edit the number of your **UAA** VM instances.

UAA requests in flight

uaa.server.inflight.count

Description	The number of requests UAA is currently processing (in-flight requests), emitted per UAA instance. **Use:** Indicates how many concurrent requests are currently in flight for the UAA instance. Unusually high spikes, if they are not associated with an expected increase in demand, could indicate a DDoS risk. From performance and load testing of the UAA component, VMware has observed that the number of concurrent requests impacts throughput and latency. The UAA Requests In Flight metric helps you see trends in the request rate that might indicate the need to scale UAA instances. Use the trends you observe to tune the threshold alerts for this metric. **Origin:** Firehose **Type:** Gauge (Integer) **Frequency:** 5 s
Recommended measurement	Maximum, per job, over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** Dynamic **Red critical:** Dynamic
Recommended response	To increase throughput and maintain low latency when the number of in-flight requests is high, scale UAA VMs horizontally by editing the **UAA** VM field in the **Resource Config** pane of the TAS for VMs tile. Ensure that the `system.cpu.user` metric for UAA is not sustained in the suggested range of 80-90% maximum CPU utilization.

System (BOSH) metrics

These sections describe system metrics, or BOSH metrics.

> **Note:** BOSH system metrics appear in the Firehose in two different formats. The tables in this section list both formats. For more information, see Duplicate Metrics Appear in the Firehose in *VMware Tanzu Application Service for VMs v2.8 Release Notes.*

Virtual Machine (VM) health

	system.healthy **system_healthy**
Description	`1` means the system is healthy, and `0` means the system is not healthy. **Use:** This is the most important BOSH metric to monitor. It indicates if the VM emitting the metric is healthy. Review this metric for all VMs to estimate the overall health of the system. Multiple unhealthy VMs signals problems with the underlying IaaS layer. **Origin:** Firehose **Type:** Gauge (Float, 0-1) **Frequency:** 60 s

Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** *N/A* **Red critical:** < 1
Recommended response	Investigate TAS for VMs logs for the unhealthy components.

VM Memory Used

system.mem.percent
system_mem_percent

Description	System Memory — Percentage of memory used on the VM **Use:** Set an alert and investigate if the free RAM is low over an extended period. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 10 minutes
Recommended alert thresholds	***Doppler VMs Only*** **Yellow warning:** ≥ 90% **Red critical:** ≥ 95% ***Other VMs*** **Yellow warning:** ≥ 80% **Red critical:** ≥ 90%
Recommended response	The response depends on the job the metric is associated with. If appropriate, scale affected jobs out and monitor for improvement.

VM disk used

system.disk.system.percent
system_disk_system_percent

Description	System disk — Percentage of the system disk used on the VM **Use:** Set an alert to indicate when the system disk is almost full. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 80% **Red critical:** ≥ 90%

Recommended response	Investigate what is filling the jobs system partition. This partition must not typically fill because BOSH deploys jobs to use ephemeral and persistent disks.

VM ephemeral disk used

system.disk.ephemeral.percent
system_disk_ephemeral_percent

Description	Ephemeral disk — Percentage of the ephemeral disk used on the VM **Use:** Set an alert and investigate if the ephemeral disk usage is too high for a job over an extended period. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 80% **Red critical:** ≥ 90%
Recommended response	1. Run `bosh vms --details` to view jobs on affected deployments. 2. Determine cause of the data consumption, and, if appropriate, increase disk space or scale out the affected jobs.

VM persistent disk used

system.disk.persistent.percent
system_disk_persistent_percent

Description	Persistent disk — Percentage of persistent disk used on the VM **Use:** Set an alert and investigate further if the persistent disk usage for a job is too high over an extended period. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 30 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 80% **Red critical:** ≥ 90%
Recommended response	1. Run `bosh vms --details` to view jobs on affected deployments. 2. Determine cause of the data consumption, and, if appropriate, increase disk space or scale out affected jobs.

VM CPU Utilization

system.cpu.user
system_cpu_user

Description	CPU utilization — The percentage of CPU spent in user processes **Use:** Set an alert and investigate further if the CPU utilization is too high for a job. For monitoring Gorouter performance, CPU utilization of the Gorouter VM is the key capacity scaling indicator VMware recommends. For more information, see Router VM CPU Utilization in *Key Capacity Scaling Indicators*. **Origin:** Firehose **Type:** Gauge (%) **Frequency:** 60 s
Recommended measurement	Average over the last 5 minutes
Recommended alert thresholds	**Yellow warning:** ≥ 85% **Red critical:** ≥ 95%
Recommended response	1. Investigate the cause of the spike. 2. If the cause is a normal workload increase, then scale up the affected jobs.

Key capacity scaling indicators

VMware provides these indicators to operators as general guidance for capacity scaling. Each indicator is based on all platform metrics from all components.

This guidance is applicable to most TAS for VMs deployments. VMware recommends that operators fine-tune the suggested alert thresholds by observing historical trends for their deployments.

For more information about accessing metrics used in these key capacity scaling indicators, see Overview of Logging and Metrics.

Diego Cell capacity scaling indicators

There are three key capacity scaling indicators VMware recommends for a Diego Cell.

Diego Cell Memory Capacity

Description	The Diego Cell Memory Capacity indicator is the percentage of remaining memory your Diego Cells can allocate to containers. Divide the `CapacityRemainingMemory` metric with the `CapacityTotalMemory` to get this percentage. The metric `CapacityRemainingMemory` is the remaining memory, in MiB, available to a Diego Cell. The metric `CapacityTotalMemory` is the total memory, in MiB, available to a Diego Cell.

Source ID	`rep`
Metrics	`CapacityRemainingMemory` `CapacityTotalMemory`
Recommended thresholds	< average (35%) This threshold assumes you have three AZs.
How to scale	Deploy additional diego cells until the average free memory is 35%. This threshold assumes you have three AZs.
Additional details	**Type:** Gauge (%) **Frequency:** Emitted every 60 s **Applies to:** cf:diego_cells

	Diego Cell Disk Capacity
Description	The Diego Cell Disk Capacity indicator is the percentage of remaining disk capacity a given Diego Cell can allocate to containers. Divide the `CapacityRemainingDisk` metric by the `CapacityTotalDisk` metric to get this percentage. The metric `CapacityRemainingDisk` is the remaining amount of disk avaiable, in MiB, for this Diego Cell. The metric `CapacityTotalDisk` indicates the total amount of disk available, in MiB, for this Diego Cell.
Source ID	`rep`
Metrics	`CapacityRemainingDisk` `CapacityTotalDisk`
Recommended thresholds	< average (35%) This threshold assumes you have three AZs.
How to scale	Deploy additional diego cells until the average free memory is 35%. This threshold assumes you have three AZs.
Additional details	**Type:** Gauge (%) **Frequency:** Emitted every 60 s **Applies to:** cf:diego_cells

	Diego Cell Container Capacity
Description	The Diego Cell Container Capacity indicator is the percentage of containers remaining that a given Diego Cell can host. Divide the `CapacityRemainingContainers` metric by the `CapacityTotalContainers` metric to get this percentage. The metric `CapacityRemainingContainers` is the remaining number of containers. The metric `CapacityTotalContainer` is the total number of containers.
Source ID	`rep`
Metrics	`CapacityRemainingContainers` `CapacityTotalContainers`
Recommended thresholds	< average (35%) This threshold assumes you have three AZs.

How to scale	Deploy additional diego cells until the average free memory is 35%. This threshold assumes you have three AZs.
Additional details	**Type:** Gauge (%) **Frequency:** Emitted every 60 s **Applies to:** cf:diego_cells

Firehose performance scaling indicators

VMware recommends three key capacity scaling indicators for monitoring Firehose performance.

Log Transport Loss Rate

Description	The Log Transport Loss Rate indicator is the rate of messages dropped between the Dopplers and the Firehose. Divide the `dropped{direction=ingress}` metric by the `ingress` metric to get the loss rate. Metric `ingress` is the number of messages entering the Dopplers. `dropped` is the number of messages never delivered to the Firehose. For more information about Loggregator components, see Loggregator Architecture.
Source ID	`doppler`
Metrics	`dropped` `ingress`
Label	`{direction=ingress}` Dopplers emit two separate dropped metrics, one for ingress and one for egress. The envelopes have a `direction` label. For this indicator, use the metric with a `direction` tag with a value of `ingress`.
Recommended thresholds	**Scale indicator:** ≥ 0.01 If alerting: **Yellow warning:** ≥ 0.005 **Red critical:** ≥ 0.01 Excessive dropped messages can indicate the Dopplers or Traffic Controllers are not processing messages quickly enough.
How to scale	Scale up the number of Traffic Controller and Doppler instances. **Note:** At approximately 40 Doppler instances and 20 Traffic Controller instances, horizontal scaling is no longer useful for improving Firehose performance. To improve performance, add vertical scale to the existing Doppler and Traffic Controller instances by increasing CPU resources.
Additional details	**Type:** Gauge (float) **Frequency:** Base metrics are emitted every 5 s **Applies to:** cf:doppler

Doppler Message Ingress Capacity

Description	The Doppler Ingress counter is the number of messages ingressed by the Doppler instance. Divide the sum of the rate of `ingress` metrics across instances by the `current number of Doppler instances` to get this average.
Source ID	`doppler`
Metrics	`ingress`
Recommended thresholds	**Scale indicator:** ≥ 16,000 envelopes per second (or 1 million envelopes per minute)
How to scale	Increase the number of Doppler VMs in the **Resource Config** pane of the TAS for VMs tile.
Additional details	**Type:** Counter (float) **Frequency:** Emitted every 5 s **Applies to:** cf:doppler

Reverse Log Proxy Loss Rate

Description	The Reverse Log Proxy Loss Rate indicator is the rate of bound app logs dropped from the Reverse Log Proxies (RLP). Divide the `dropped` metric by the `ingress` metric to get this indicator. This loss rate is specific to the RLP and does not impact the Firehose loss rate.
Source ID	`rlp`
Metrics	`ingress` `dropped`
Recommended thresholds	**Scale indicator:** ≥ 0.1 If alerting: **Yellow warning:** ≥ 0.01 **Red critical:** ≥ 0.1 Excessive dropped messages can indicate that the RLP is overloaded and that the Traffic Controllers need to be scaled.
How to scale	Scale up the number of traffic controller instances to further balance log load.

Additional details	**Type:** Counter (Integer) **Frequency:** Emitted every 60 s **Applies to:** cf:loggregator

Firehose consumer scaling indicator

VMware recommends the following scaling indicator for monitoring the performance of consumers of the Firehose.

Slow Consumer Drops

Description	The Slow Consumer Drops indicator is the `slow_consumer` metric incremented for each connection the Firehose closes because a consumer could not keep up. This indicator shows how fast a Firehose consumer, such as a monitoring tool nozzle, is ingesting data. If this number is anomalous, it might result in the downstream monitoring tool not having all expected data, even though that data was successfully transported through the Firehose.
Source ID	`doppler_proxy`
Metrics	`slow_consumer`
Recommended thresholds	**Scale indicator:** VMware recommends scaling when the rate of Firehose Slow Consumer Drops is anomalous for a given environment.
How to scale	Scale up the number of nozzle instances. You can scale a nozzle using the subscription ID specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose evenly distributes data across all instances of the nozzle. For example, if you have two nozzle instances with the same subscription ID, the Firehose sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the Firehose sends one-third of the data to each instance. If you want to scale a nozzle, the number of nozzle instances must match the number of Traffic Controller instances.
Additional details	**Type:** Counter **Frequency:** Emitted every 5 s **Applies to:** cf:doppler

Reverse Log Proxy Egress Dropped Messages

Description	The Reverse Log Proxy Egress Dropped Messages indicator shows the number of messages dropped when consumers of the RLP, such as monitoring tool nozzles, ingest the exiting stream of logs and metrics too slowly. Within TAS for VMs, logs and metrics enter Loggregator for transport and then egress through the Reverse Log Proxy (RLP).

Source ID	`rlp`
Metrics	`dropped`
Label	`direction: egress`
Recommended thresholds	**Scale indicator:** Scale when the rate of `rlp.dropped, direction: egress` metrics is continuously increasing.
How to scale	Scale up the number of nozzle instances. The number of nozzle instances must match the number of Traffic Controller instances. You can scale a nozzle using the subscription ID specified when the nozzle connects to the RLP. If you use the same subscription ID on each nozzle instance, the RLP evenly distributes data across all instances of the nozzle. For example, if you have two nozzle instances with the same subscription ID, the RLP sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the RLP sends one-third of the data to each instance.
Additional details	**Type:** Counter **Frequency:** Emitted every 5 s **Applies to:** cf:loggregator_trafficcontroller

Doppler Egress Dropped Messages

Description	The Doppler Egress Dropped Messages indicator shows the number of messages that the Dopplers drop when consumers of the RLP, such as monitoring tool nozzles, ingest the exiting stream of logs and metrics too slowly. For more information about how the Dopplers transport logs and metrics through Loggregator, see Loggregator Architecture in *Loggregator Architecture*. **Note:** The `doppler.dropped` metric includes both `ingress` and `egress` directions. To differentiate between `ingress` and `egress`, refer to the `direction` tag on the metric.
Source ID	`doppler`
Metrics	`dropped` `egress`
Label	`direction: egress`

Recommended thresholds	**Scale indicator:** Scale when the rate of `doppler.dropped, direction: egress` metrics is continuously increasing.
How to scale	Scale up the number of nozzle instances. The number of nozzle instances must match the number of Traffic Controller instances. You can scale a nozzle using the subscription ID specified when the nozzle connects to the RLP. If you use the same subscription ID on each nozzle instance, the RLP evenly distributes data across all instances of the nozzle. For example, if you have two nozzle instances with the same subscription ID, the RLP sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the RLP sends one-third of the data to each instance.
Additional details	**Type:** Counter **Frequency:** Emitted every 5 s **Applies to:** cf:doppler

Syslog drain performance scaling indicators

There is a single key capacity scaling indicator VMware recommends for Syslog Drain performance.

Note: These Syslog Drain scaling indicators are only relevant if your deployment contains apps using the syslog drain binding feature.

Syslog Agent Loss Rate

Description	Divide the `loggregator.syslog_agent.dropped{direction:egress}` metric by the `loggregator.syslog_agent.ingress{scope:all_drains}` metric to get the rate of messages dropped as a percentage of total message traffic through Syslog Agents. The message traffic through Syslog Agents includes logs for bound apps. The loss rate of Syslog Agents indicates that the Syslog Drain consumer is ingesting logs from a Syslog Drain-bound app too slowly. The Syslog Agent loss rate does not affect the Firehose loss rate. Syslog Agents can still drop messages even though the Firehose does not drop messages.
Source ID	`syslog_agent`
Metrics	`dropped` `ingress`
Label	`direction:egress` `scope:all_drains`

Recommended thresholds	The scaling indicator VMware recommends is the minimum Syslog Agent loss rate per minute within a five-minute window. You can scale up if the loss rate is greater than `0.1` for five minutes or longer. **Scale indicator:** ≥ 0.1 If alerting: **Yellow warning:** ≥ 0.01 **Red critical:** ≥ 0.1
How to scale	Review the logs of the syslog server for intake issues and other performance issues. Scale the syslog server if necessary.
Additional details	**Type:** Counter (Integer) **Frequency:** Emitted every 60 s

Log cache scaling indicator

VMware recommends the following scaling indicator for monitoring the performance of log cache.

Log Cache Caching Duration

Description	The Log Cache Caching Duration indicator shows the age in milliseconds of the oldest data point stored in Log Cache. Log Cache stores all messages that are passed through the Firehose in an ephemeral in-memory store. The size of this store and the cache duration are dependent on the amount of memory available on the VM where Log Cache runs. Typically, Log Cache runs on the Doppler VM.
Source ID	`log_cache`
Metrics	`log_cache_cache_period`
Recommended thresholds	**Scale indicator:** Scale the VM on which Log Cache runs when the `log_cache_cache_period` metric drops below 900000 milliseconds.
How to scale	Scale up the number of Doppler VMs or choose a VM type for Doppler that provides more memory.
Additional details	**Type:** Gauge **Frequency:** Emitted every 15 s **Applies to:** cf:log-cache

Gorouter performance scaling indicator

There is one key capacity scaling indicator VMware recommends for Gorouter performance.

Note: The following metric appears in the Firehose in two different formats. The following table lists both formats.

Gorouter VM CPU Utilization

Description	The Gorouter VM CPU Utilization indicator shows how much of a Gorouter VM's CPU is being used. High CPU utilization of the Gorouter VMs can increase latency and cause requests per second to decrease.
Source ID	`cpu`
Metrics	`user`
Recommended thresholds	**Scale indicator:** ≥ 60% If alerting: **Yellow warning:** ≥ 60% **Red critical:** ≥ 70%
How to scale	Scale the Gorouters horizontally or vertically by editing the Router VM in the Resource Config pane of the TAS for VMs tile. At greater than 8 CPUs, vertical scaling is no longer beneficial for increasing throughput.
Additional details	**Type:** Gauge (float) **Frequency:** Emitted every 60 s **Applies to:** cf:router

UAA performance scaling indicator

There is one key capacity scaling indicator VMware recommends for UAA performance.

Note: The following metric appears in the Firehose in two different formats. The following table lists both formats.

UAA VM CPU Utilization

Description	The UAA VM CPU Utilization indicator shows how much of the UAA VM's CPU is used. High CPU utilization of the UAA VMs can cause requests per second to decrease.
Source ID	`cpu`
Metrics	`user`
Recommended thresholds	**Scale indicator:** ≥ 80% If alerting: **Yellow warning:** ≥ 80% **Red critical:** ≥ 90%

How to scale	Scale UAA horizontally or vertically. To scale UAA, go to the **Resource Config** pane of the TAS for VMs tile and edit the number of your **UAA** VM instances or change the VM type to a type that utilizes more CPU cores.
Additional details	**Type:** Gauge (float) **Frequency:** Emitted every 60 s **Applies to:** cf:uaa

NFS/WebDAV backed blobstore

There is one key capacity scaling indicator for external S3 external storage.

> **Note:** This metric is only relevant if your deployment does not use an external S3 repository for external storage with no capacity constraints.

> **Note:** The following metric appears in the Firehose in two different formats. The following table lists both formats.

External S3 External Storage

Description	The External S3 External Storage indicator shows the percentage of persistent disk used. *If applicable:* Monitor the percentage of persistent disk used on the VM for the NFS Server job. If you do not use an external S3 repository for external storage with no capacity constraints, you must monitor the TAS for VMs object store to push new app and buildpacks.
Source ID	`disk`
Metrics	`persistent.percent`
Recommended thresholds	≥ 75%
How to scale	Give your NFS Server additional persistent disk resources. If you use an internal NFS/WebDAV backed blobstore, consider scaling the persistent disk when it reaches 75% capacity.
Additional details	**Type:** Gauge (%) **Applies to:** cf:nfs_server

Reporting

In this section:

- Reporting App, Task, and Service Instance Usage
- Reporting Instance Usage with Apps Manager

Reporting App, Task, and Service Instance Usage

This topic describes how to use the Cloud Foundry Command Line Interface (cf CLI) to retrieve historical system- and org-level usage information about your apps, tasks, and service instances through the Cloud Controller and Usage Service APIs.

Usage reports are compiled from the `/v2/app_usage_events` endpoint. For more information, see List all App Usage Events in the Cloud Foundry API documentation.

You can also access usage information by using Apps Manager. For more information, see Reporting Instance Usage with Apps Manager.

To retrieve current app and event information from the Cloud Controller, see Retrieving App and Event Information in the open-source Cloud Foundry documentation.

Prerequisite

Before performing the procedures in this topic, ensure that your user is a member of the `cloud_controller.admin` or `usage_service.audit` group. For example, a UAAC admin can add a user to the `usage_service.audit` group with the following command:

```
uaac member add usage_service.audit USERNAME
```

Obtain System Usage Information

You can obtain the following system-wide usage information:

- App Usage
- Task Usage
- Service Usage
- All Orgs App Usage

App Usage

Use `curl` to make a request to `/system_report/app_usages` on the Usage Service endpoint to show system-wide app usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/app_usages" -k -v
\
-H "authorization: `cf oauth-token`"
{
"report_time": "2017-04-11 22:28:24 UTC",
"monthly_reports": [
    {
      "month": 4,
      "year": 2017,
      "average_app_instances": 17855.256153713308,
      "maximum_app_instances": 18145,
      "app_instance_hours": 4686533.080277303
    }
  ],
"yearly_reports": [
    {
      "year": 2017,
      "average_app_instances": 16184.9,
      "maximum_app_instances": 18145,
      "app_instance_hours": 39207433.0802773
    }
  ]
}
```

The table below describes the data generated by this report:

Value	Calculation Method
`app_instance_hours`	The total number of hours app instances have been running on the foundation for the month or year.
`average_app_instances`	The total `app_instance_hours` on the foundation for the month or year divided by the total hours for that time period. **Example:** For the month of January, 100 apps run for 300 hours each, generating a value of 30000 for `app_instance_hours`. When divided by 744, the total number of hours in January, you get a value of 40.3 for `average_app_instances`. **Notice:** That `average_app_instances` is calculated against the current running hours. This is most relevant when checking the report for the current month. If you look at the `report_time` in the example above, you'll see that the report was run on the 11th of April. Having a value of 4686533.080277303 for `app_instance_hours` when divided by 262.467, approx. the total number of hours for the first 11 days of April, you get an approx. value of 17855 for `average_app_instances`.
`maximum_app_instances`	The highest concurrent number of app instances running on the foundation for the month or year.

Task Usage

Use `curl` to make a request to `/system_report/task_usages` on the Usage Service endpoint to show system-wide task usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/task_usages" -k -v
\
-H "authorization: `cf oauth-token`"
{
"report_time": "2017-04-11T22:33:48.971Z",
"monthly_reports": [
    {
      "month": 4,
      "year": 2017,
      "total_task_runs": 235,
      "maximum_concurrent_tasks": 7,
      "task_hours": 43045.201944444445
    }
  ],
"yearly_reports": [
    {
      "year": 2017,
      "total_task_runs": 2894,
      "maximum_concurrent_tasks": 184,
      "task_hours": 45361.26694444445
    }
  ]
}
```

Service Usage

Use `curl` to make a request to `/system_report/service_usages` on the Usage Service endpoint to show system-wide service usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/system_report/service_usages" -k
-v \
-H "authorization: `cf oauth-token`"
{
  "report_time": "2017-05-11 18:29:14 UTC",
  "monthly_service_reports": [
    {
      "service_name": "fake-service-0507f1fd-2340-49a6-9d43-a347a5f5f6be",
      "service_guid": "177dcfde-cd51-4058-bd86-b98015c295f5",
      "usages": [
        {
          "month": 1,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 2,
          "year": 2017,
          "duration_in_hours": 0,
          "average_instances": 0,
          "maximum_instances": 0
        },
        {
          "month": 3,
          "year": 2017,
          "duration_in_hours": 4.182222222222227,
          "average_instances": 0,
          "maximum_instances": 2
        },
        {
          "month": 4,
          "year": 2017,
          "duration_in_hours": 2176.962222222186,
          "average_instances": 3,
          "maximum_instances": 7
        },
        {
          "month": 5,
          "year": 2017,
          "duration_in_hours": 385.61388888888854,
          "average_instances": 1.5,
          "maximum_instances": 3
        }
      ],
      "plans": [
        {
          "usages": [
            {
              "month": 1,
              "year": 2017,
              "duration_in_hours": 0,
              "average_instances": 0,
              "maximum_instances": 0
            },
            {
              "month": 2,
              "year": 2017,
              "duration_in_hours": 0,
              "average_instances": 0,
              "maximum_instances": 0
            },
            {
              "month": 3,
              "year": 2017,
              "duration_in_hours": 4.182222222222227,
              "average_instances": 0,
              "maximum_instances": 2
            },
            {
              "month": 4,
              "year": 2017,
              "duration_in_hours": 1465.6388888888941,
              "average_instances": 2,
              "maximum_instances": 5
            },
            {
              "month": 5,
              "year": 2017,
              "duration_in_hours": 385.61388888888854,
              "average_instances": 1.5,
              "maximum_instances": 3
            }
          ],
          "service_plan_name": "fake-plan",
          "service_plan_guid": "ac09f607-f4e5-4807-af16-e95856061bd7"
        }
      ]
```

```
    }
  ]
}
```

All Orgs Usage

The `app_usages` endpoint directly under your system domain returns system-wide app usage for all orgs that you have access to. The command works like the org-specific App Usage command, except you do not supply an org GUID. You can set `order_by` to either `total_instance_hours` or `megabyte_hours`.

The `order_by` ordering defaults to ascending, which lists orgs with the most usage at the end of the output. To change the sort ordering to descending, prepend the `order_by` argument with a `-`, as in `order_by=-megabyte_hours`.

Use `curl` to make a request to `/app_usages` on the Usage Service endpoint to show system-wide organization usage data:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/app_usages?start=2018-010-01&end
=2018-11-01&order_by=total_instance_hours" -k -v \
-H "authorization: `cf oauth-token`"
{
   "period_start":"2018-01-10T00:00:00Z",
   "period_end":"2018-11-01T23:59:59Z",
   "Organizations":[
      {
         "guid":"d5c9f947-df85-45af-abab-00a2174598f9",
         "total_instance_hours":"91535.6",
         "megabyte_hours":"1830712.2"
      },
      {
         "guid":"a9605fa4-99da-4932-84f5-3db8698c1fc1",
         "total_instance_hours":"99440.8",
         "megabyte_hours":"96698582.2"
      }
   ]
}
```

Obtain Org Usage Information

You can obtain the following org-specific usage information:

- App Usage
- Task Usage
- Service Usage

You must have the GUID of the org you want to obtain information about in order to perform the procedures in this section. To retrieve your org GUID, run the `cf org` command:

```
$ cf org YOUR-ORG --guid
```

App Usage

Use `curl` to make a request to `/app_usages` on the Usage Service endpoint to show app usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/YOUR-ORG-GUID/app_
usages?start=YYYY-MM-DD&end=YYYY-MM-DD" \
-k -v \
-H "authorization: `cf oauth-token`"

{
  "organization_guid": "88e485fa-2222-4993-8ef5-bac3711adfee",
  "period_start": "2019-01-16T00:00:00Z",
  "period_end": "2019-01-17T23:59:59Z",
  "app_usages": [
    {
      "space_guid": "d6988d4d-6175-4dee-b219-a6eabbc05514",
      "space_name": "development",
      "app_name": "static2",
      "app_guid": "1afc2b20-dd41-4ba4-8e6e-c894b5a2cb23",
      "parent_application_name": "static2",
      "parent_application_guid": "1afc2b20-dd41-4ba4-8e6e-c894b5a2cb23",
      "instance_count": 1,
      "memory_in_mb_per_instance": 1024,
      "duration_in_seconds": 172800
    },
    {
      "space_guid": "d6988d4d-6175-4dee-b219-a6eabbc05514",
      "space_name": "development",
      "app_name": "static",
      "app_guid": "54e767ff-4f83-45b1-bf06-46c501217845",
      "parent_application_name": "static",
      "parent_application_guid": "54e767ff-4f83-45b1-bf06-46c501217845",
      "instance_count": 1,
      "memory_in_mb_per_instance": 1024,
      "duration_in_seconds": 172800
    }
  ]
}
```

You may see results with `"instance_count": 0`. This happens when a v3 app is pushed but has processes that are not scaled up. The memory configuration and duration are still stored and can be useful for resource management. Apps with ‘0’ instances are not counted in the App Usages System Report above.

Task Usage

Use `curl` to make a request to `/task_usages` on the Usage Service endpoint to show task usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/YOUR-ORG-GUID/task
_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" \
-k -v \
-H "authorization: `cf oauth-token`"
{
     "organization_guid": "8f88362f-547c-4158-808b-4605468387d5",
     "period_start": "2014-01-01",
     "period_end": "2017-04-04",
     "spaces": {
       "e6445eb3-fdac-4049-bafc-94d1703d5e78": {
         "space_name": "smoketest",
         "task_summaries": [
           {
             "parent_application_guid": "04cd29d5-1f9e-4900-ac13-2e903f6582a
9",
             "parent_application_name": "task-dummy-app",
             "memory_in_mb_per_instance": 256,
             "task_count_for_range": 54084,
             "concurrent_task_count_for_range": 5,
             "total_duration_in_seconds_for_range": 37651415
           }
          ]
       },
       "b66665e4-873f-4482-acf1-89307ba9c6e4": {
         "space_name": "smoketest-experimental",
         "task_summaries": [
         {
             "parent_application_guid": "d941b689-4a27-44ec-91d3-1f97434dbed
9",
             "parent_application_name": "console-blue",
             "memory_in_mb_per_instance": 256,
             "task_count_for_range": 14,
             "concurrent_task_count_for_range": 2,
             "total_duration_in_seconds_for_range": 20583
           }
          ]
       }
     }
}
```

Note: In the `/task_usages` endpoint, `memory_in_mb_per_instance` is the memory of the task.

Service Usage

Use `curl` to make a request to `/service_usages` on the Usage Service endpoint to show service usage in an org. You must complete the placeholders in `start=YYYY-MM-DD&end=YYYY-MM-DD` to define a date range.

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/organizations/`cf org YOUR-ORG -
-guid`/service_usages?start=YYYY-MM-DD&end=YYYY-MM-DD" -k -v -H "authorizatio
n: `cf oauth-token`"
{
     "organization_guid": "8d83362f-587a-4148-806b-4407428887b5",
     "period_start": "2016-06-01T00:00:00Z",
     "period_end": "2016-06-13T23:59:59Z",
     "service_usages": [
        {
          "deleted": false,
          "duration_in_seconds": 1377982.52,
          "service_guid": "02802293-b769-44cc-807f-eee331ba9b2b",
          "service_instance_creation": "2016-01-20T18:48:16.000Z",
          "service_instance_deletion": null,
          "service_instance_guid": "b25b4481-19aa-4504-82c9-f303e7e9ed6e",
          "service_instance_name": "something-usage-db",
          "service_instance_type": "managed_service_instance",
          "service_name": "my-mysql-service",
          "service_plan_guid": "70915a70-7311-4f3e-ab0d-4a7dfd3ef2d9",
          "service_plan_name": "20gb",
          "space_guid": "e6445eb3-fdac-4049-bafc-94d1703d5e78",
          "space_name": "outer-space"
        }
    ]
}
```

Usage Availability

The app, task, and service instance usage data returned from the Usage Service is delayed to ensure accuracy. The `/usage_availability` endpoint can be used to retrieve the date which the Usage Service data is accurate to. The Usage Service is able to provide data for time periods up to and including this date.

Use the following curl command to retrieve the current usage availability timestamp:

```
curl "https://app-usage.YOUR-SYSTEM-DOMAIN/usage_availability" -k -v
{"date":"2019-01-01"}
```

Where `YOUR-SYSTEM-DOMAIN` is the system domain of your VMware Tanzu Application Service for VMs (TAS for VMs) environment.

Monitor Usage Service

VMware recommends monitoring usage service at regular intervals with a recurring CI task or third-party monitoring tool. Usage service runs asynchronous jobs such as fetching usage events from the Cloud Controller API.

Use the following curl command to check for failed jobs:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/heartbeat/failed_jobs" -k -v
Ok.
```

By default, the `error_threshold` is set to 1 and the command does the following:

- Succeeds with HTTP status `200` if the number of failed jobs is lower than the threshold
- Fails with HTTP status `500` if it exceeds the threshold

The following example specifies a threshold value:

```
$ curl "https://app-usage.YOUR-SYSTEM-DOMAIN/heartbeat/failed_jobs?error_threshold=5"
-k -v
Ok.
```

Usage Data Retention

Usage service retains all summarized monthly data and never deletes it. The endpoints listed below return data for all years:

- `/system_report/app_usages`
- `/system_report/task_usages`
- `/system_report/service_usages`

For examples of how to curl these endpoints, see Obtain System Usage Information.

Usage service deletes old records of granular data, such as usage per app, per day. By default, granular data is deleted after 365 days, so the following endpoints return data for the last 365 days. The retention period is configurable in the **Advanced Features** pane of the TAS for VMs tile.

The endpoints listed below return granular data:

- `/organizations/ORG-GUID/app_usages`
- `/organizations/ORG-GUID/task_usages`
- `/organizations/ORG-GUID/service_usages`
- `/app_usages`

Example: Autogenerated Database Usage Reports

For security compliance and record keeping, some Ops Manager customers create Concourse CI/CD pipelines that regularly run JavaScript scripts to do the following:

1. Call the `app-usage` APIs in the above sections of this topic, as well as the Security Event Logging topic, to collect data about service instances from an Ops Manager foundation. This data includes database instance information such as tile or database type, database creator, and date of creation or deletion.
2. Merge and format the data about services instances into a JSON output file and save it to an S3 bucket.
3. Input the file to another process that does the following:

1. Parses the data.
2. Eliminates duplicate entries from previous reports.
3. Sends the data to a database compliance system through APIs.

Reporting instance usage with Apps Manager

You can retrieve app, task, and service instance usage information using Apps Manager using the Accounting Report or the Usage Report for all apps in your VMware Tanzu Application Service for VMs (TAS for VMs).

You can also retrieve app, task, and service instance usage information using the Usage service API, or the Cloud Foundry API from the Cloud Foundry Command Line Interface (cf CLI). For more information, see Monitoring App, Task, and Service Instance Usage.

There are two ways to monitor app, task, and service instance usage from Apps Manager. The Accounting Report provides a summarized report, and the Usage Report provides a more detailed view of the data.

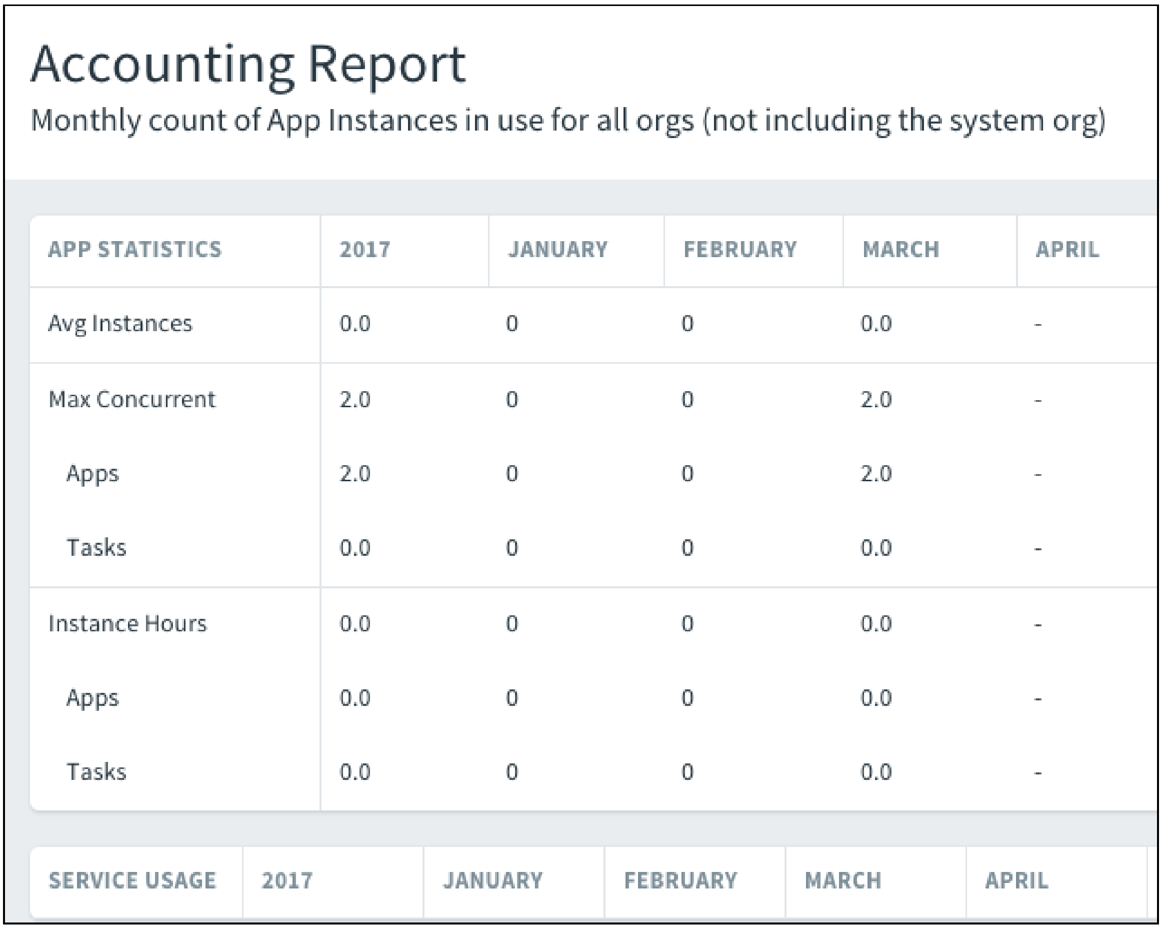
View the Accounting Report

The Accounting Report displays instance usage information for all orgs in your Ops Manager deployment except the **system** org. The Accounting Report provides a high-level overview of your usage by displaying your monthly count of app, task, and service instances.

To access the Accounting Report:

1. Log in to the Apps Manager as an admin. For more information, see Logging in to Apps Manager
2. Select **Accounting Report**.
3. In **App Statistics** and **Service Usage**, view the average and maximum instances in use per month.

   Max Concurrent displays the largest number of instances in use at a time over the indicated time period. The Accounting Report calculates these values from the `start`, `stop`, and `scale` app usage events in Cloud Controller.



Accounting Report

Monthly count of App Instances in use for all orgs (not including the system org)

APP STATISTICS	2017	JANUARY	FEBRUARY	MARCH	APRIL
Avg Instances	0.0	0	0	0.0	-
Max Concurrent	2.0	0	0	2.0	-
Apps	2.0	0	0	2.0	-
Tasks	0.0	0	0	0.0	-
Instance Hours	0.0	0	0	0.0	-
Apps	0.0	0	0	0.0	-
Tasks	0.0	0	0	0.0	-

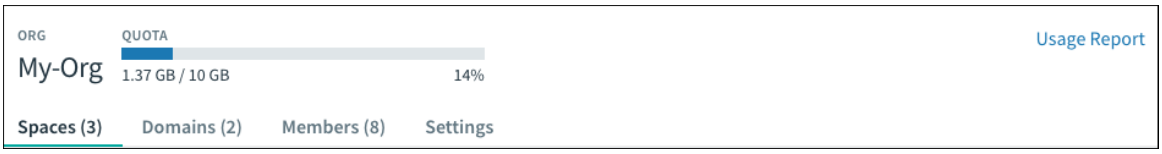
SERVICE USAGE	2017	JANUARY	FEBRUARY	MARCH	APRIL

View the Usage Report

The Usage Report provides a more granular view of your usage by detailing app, task, and service instance usage information for all spaces in a particular org, excluding the **system** org.

To access the Usage Report:

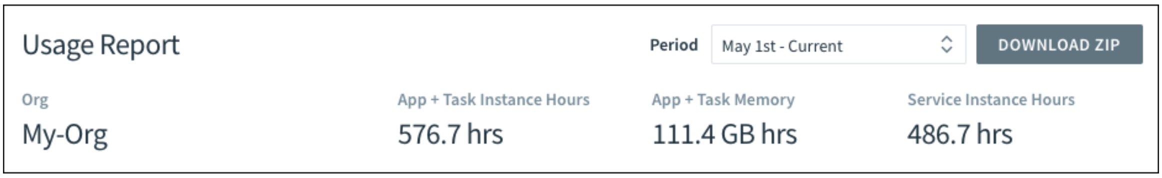
1. Log in to the Apps Manager as an admin, or as an account with the **Org Manager** or **Org Auditor** role. For more information about managing roles, see Managing User Accounts and Permissions Using the Apps Manager.
2. From the dropdown on the left, select the org for which you want to view a usage report.
3. Click **Usage Report** in the upper right.



ORG: My-Org

QUOTA: 1.37 GB / 10 GB 14%

The top of the Usage Report displays total **App + Task Instance Hours**, **App + Task Memory**, and **Service Instance Hours** by all spaces in the org.

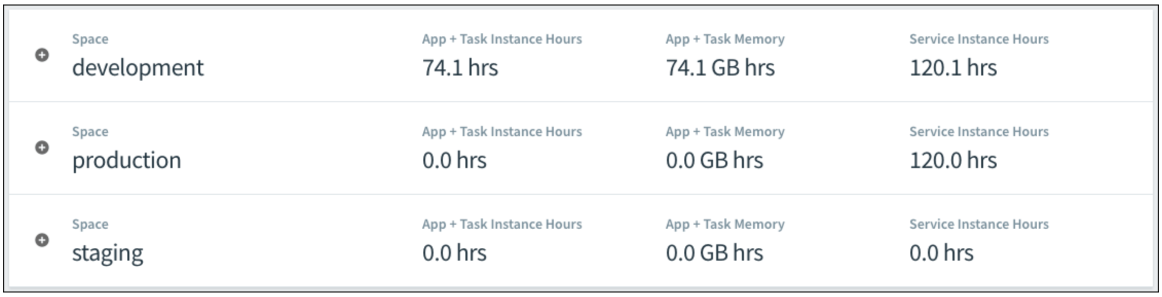


Usage Report

Period: May 1st - Current

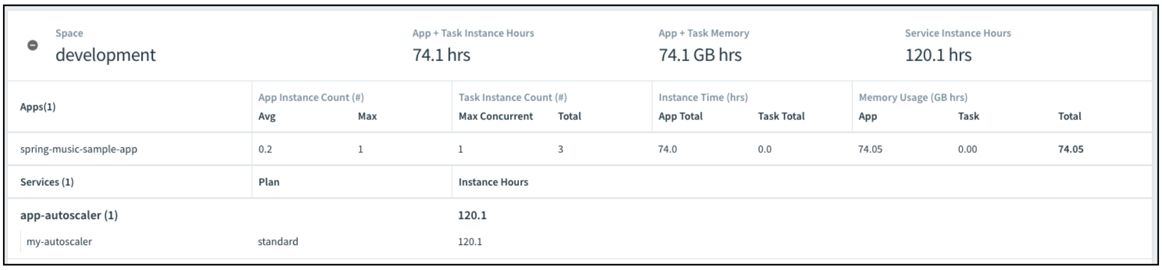
Org	App + Task Instance Hours	App + Task Memory	Service Instance Hours
My-Org	576.7 hrs	111.4 GB hrs	486.7 hrs

The report provides total usage information for each of your spaces.



Space	App + Task Instance Hours	App + Task Memory	Service Instance Hours
development	74.1 hrs	74.1 GB hrs	120.1 hrs
production	0.0 hrs	0.0 GB hrs	120.0 hrs
staging	0.0 hrs	0.0 GB hrs	0.0 hrs

To display more detailed information about app, task, and service instance usage in a space, click the name of the space for an expanded view.



Space	App + Task Instance Hours	App + Task Memory	Service Instance Hours
development	74.1 hrs	74.1 GB hrs	120.1 hrs

Apps(1)	App Instance Count (#)		Task Instance Count (#)		Instance Time (hrs)		Memory Usage (GB hrs)		
	Avg	Max	Max Concurrent	Total	App Total	Task Total	App	Task	Total
spring-music-sample-app	0.2	1	1	3	74.0	0.0	74.05	0.00	**74.05**

Services (1)	Plan	Instance Hours
app-autoscaler (1)		**120.1**
my-autoscaler	standard	120.1

Loggregator

In this section:

- Loggregator Guide for TAS for VMs Operators
- Deploying a Nozzle to the Loggregator Firehose
- Installing the Loggregator Plugin for cf CLI

Loggregator Guide for TAS for VMs Operators

This topic contains information for VMware Tanzu Application Service for VMs (TAS for VMs) deployment operators about how to configure the Loggregator system to avoid data loss with high volumes of logging and metrics data.

For more information about the Loggregator system, see Loggregator Architecture.

Loggregator message throughput and reliability

You can measure both the message throughput and message reliability rates of your Loggregator system.

Measuring message throughput

To measure the message throughput of the Loggregator system, you can monitor the total number of egress messages from all Metrons on your platform using the `metron.egress` metric.

If you do not use a monitoring platform, you can measure the overall message throughput of your Loggregator system. To measure the overall message throughput:

1. Log in to the Cloud Foundry Command Line Interface (cf CLI) with your admin credentials by running:

   ```
   cf login
   ```

2. Install the TAS for VMs Firehose plugin. For more information, see Install the Plugin in *Installing the Loggregator Firehose Plugin for cf CLI*.

3. Install Pipe Viewer by running:

   ```
   apt-get install pv
   ```

4. Run:

   ```
   cf nozzle -n | pv -l -i 10 -r > /dev/null
   ```

Measuring message reliability

To measure the message reliability rate of your Loggregator system, you can run black-box tests.

Scaling Loggregator

Most Loggregator configurations use preferred resource defaults. For more information about customizing these defaults and planning the capacity of your Loggregator system, see Key Capacity Scaling Indicators.

Scaling nozzles

Nozzles are programs that consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and to forward it to other apps and services. You can scale a nozzle using the subscription ID specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose evenly distributes data across all instances of the nozzle.

For example, if you have two nozzle instances with the same subscription ID, the Firehose sends half of the data to one nozzle instance and half to the other. Similarly, if you have three nozzle instances with the same subscription ID, the Firehose sends one-third of the data to each instance.

If you want to scale a nozzle, the number of nozzle instances should match the number of Traffic Controller instances:

Number of nozzle instances = Number of Traffic Controller instances

Stateless nozzles should handle scaling gracefully. If a nozzle buffers or caches the data, the nozzle author must test the results of scaling the number of nozzle instances up or down.

> **Note:** You can disable the Firehose. In place of the Firehose, you can configure an aggregate log and metric drain for your foundation. For more information, see Configure System Logging in *Configuring TAS for VMs*.

Slow nozzle alerts

The Traffic Controller alerts the nozzle programs if they consume events too slowly. The following metrics can be used to identify slow consumers:

- For v1 consumers: `doppler_proxy.slow_consumer` is incremented as consumers are disconnected for being slow.
- For v2 consumers: `doppler.dropped{direction="egress"}` or `rlp.dropped{direction="egress"}` are incremented when a v2 consumer fails to keep up.

For more information about the Traffic Controller, see Loggregator Architecture in *Loggregator Architecture*.

Forwarding logs to an external service

You can configure TAS for VMs to forward log data from apps to an external aggregator service. For information about how to bind apps to the external service and configure it to receive logs from TAS for VMs, see Streaming App Logs to Log Management Services.

Log message size constraints

When a Diego Cell emits app logs to Metron, Diego breaks up log messages greater than approximately 60 KiB into multiple envelopes.

Deploying a Nozzle to your Cloud Foundry Loggregator Firehose

Learn how to deploy a nozzle app to the VMware Tanzu Application Service for VMs (TAS for VMs) Loggregator Firehose.

For more information about nozzles and the Loggregator Firehose, see Loggregator Architecture. The TAS for VMs Loggregator team created an example nozzle app for use with this tutorial.

The following procedure deploys an example nozzle to the Firehose of a TAS for VMs installation deployed locally with BOSH Lite v2. For more information about BOSH Lite v2, see VirtualBox BOSH Lite v2 in the BOSH documentation.

To reduce the load on custom nozzles that you develop, you can request Firehose subscriptions that emit only metrics on an allow list. For examples, see `rlpreader` and `rlptypereader` in the Loggregator Tools repository and V2 Subscriptions in the Loggregator Release repository in GitHub.

> **Note:** You can disable the Firehose. In place of the Firehose, you can configure an aggregate log and metric drain for your foundation. For more information, see Configure System Logging in *Configuring TAS for VMs*.

Prerequisites

Before you deploy a nozzle to the Loggregator Firehose, you must have:

- BOSH CLI installed locally. For more information, see BOSH CLI in the BOSH documentation.
- Spiff installed locally and added to the load path of your shell. For more information, see the Spiff repository on GitHub.
- BOSH Lite v2 deployed locally using VirtualBox. For more information about BOSH Lite v2, see VirtualBox BOSH Lite v2 in the BOSH documentation.
- A working TAS for VMs deployment, including Loggregator, deployed with your local BOSH Lite v2. This deployment serves as the source of data. Use the `provision_cf` script included in the BOSH Lite v2 release.

> **Note:** Deploying TAS for VMs can take several hours depending on your internet bandwith, even when using the automated `provision_cf` script.

Step 1: Downloading Cloud Foundry BOSH manifest

To download the BOSH manifest:

1. Run `bosh deployments` to identify the name of your current BOSH deployment:

```
$ bosh deployments
+-------------+-------------+-----------------------------------------
--------+
| Name        | Release(s)   | Stemcell(s)
|
+-------------+-------------+-----------------------------------------
--------+
| cf-example  | cf-mysql/10  | bosh-vsphere-esxi-ubuntu-trusty-go_agen
t/2690.3 |
|             | cf/183.2     |
|
+-------------+-------------+-----------------------------------------
--------+
```

1. Run `bosh manifest -d MY-DEPLOYMENT MY-MANIFEST` to download and save each BOSH deployment manifest. You need this manifest to locate information about your databases. Repeat these instructions for each manifest. Replace `MY-DEPLOYMENT` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as `MY-MANIFEST`.

```
$ bosh manifest -d cf-example cf.yml
Deployment manifest saved to `cf.yml'
```

2. Place the .yml file in a secure location.

Step 2: Adding UAA client

You must authorize the example nozzle as a UAA client for your TAS for VMs deployment. To do this, add an entry for the example nozzle as a `client` for `uaa` under the `properties` key in your TAS for VMs deployment manifest YAML file. You must enter the example nozzle object in the correct location in the manifest, and with the correct indentation.

To add the nozzle as a UAA client for your deployment:

1. Open the deployment manifest in a text editor.
2. Locate the left-aligned `properties` key.
3. Under the `properties` key, locate `uaa` at the next level of indentation.
4. Under the `uaa` key, locate the `clients` key at the next level of indentation.
5. Enter properties for the `example-nozzle` at the next level of indentation, exactly as shown below. The `...` in the text below indicate other properties that might populate the manifest at each level in the hierarchy.

```
properties:
  ...
  uaa:
  ...
    clients:
    ...
      example-nozzle:
        access-token-validity: 1209600
        authorized-grant-types: client_credentials
        override: true
        secret: example-nozzle
        authorities: oauth.login,doppler.firehose
```

6. Save the deployment manifest file.

Step 3: Redeploying TAS for VMs

To redeploy TAS for VMs with BOSH, run:

```
bosh -e BOSH-ENVIRONMENT deploy
```

Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director.

For example:

```
$ bosh -e dev deploy
Acting as user 'admin' on deployment 'cf-warden' on 'Bosh Lite Director'
Getting deployment properties from director...
Detecting deployment changes
----------------------------
Releases
No changes
Compilation
No changes
Update
No changes
Resource pools
No changes
Disk pools
```

```
No changes
Networks
No changes
Jobs
No changes
Properties
uaa
clients
example-nozzle
    + access-token-validity: 1209600
    + authorized-grant-types: authorization_code,client_credentials,refresh_t
oken
    + override: true
    + secret: example-nozzle
    + scope: openid,oauth.approvals,doppler.firehose
    + authorities: oauth.login,doppler.firehose
Meta
No changes
Please review all changes carefully
Deploying
---------
Are you sure you want to deploy? (type 'yes' to continue):yes
```

Step 4: Cloning an example release

The TAS for VMs Loggregator team created an example nozzle app for use with this tutorial.

To clone the example nozzle release:

1. Clone the main release repository from the example-nozzle-release repository on GitHub by running:

   ```
   git clone https://github.com/cloudfoundry-incubator/example-nozzle-release.git
   ```

2. Go to the `example-nozzle-release` directory by running:

   ```
   cd example-nozzle-release
   ```

3. Update all of the included submodules by running:

   ```
    git submodule update --init --recursive
   ```

   For example:

   ```
   $ git submodule update --init --recursive
   Submodule 'src/github.com/cloudfoundry-incubator/example-nozzle' (git@g
   ithub.com:cloudfoundry-incubator/example-nozzle.git) registered for pat
   h 'src/github.com/cloudfoundry-incubator/example-nozzle'
   Submodule 'src/github.com/cloudfoundry-incubator/uaago' (git@github.co
   m:cloudfoundry-incubator/uaago.git) registered for path 'src/github.co
   m/cloudfoundry-incubator/uaago'
   ...
   ```

```
Cloning into 'src/github.com/cloudfoundry-incubator/example-nozzle'...
...
```

Step 5: Preparing Nozzle manifest

To prepare the nozzle deployment manifest:

1. In the `example-nozzle-release` directory, navigate to the `templates` directory by running:

   ```
   cd templates
   ```

   Within this directory, examine the two YAML files. `bosh-lite-stub.yml` contains the values used to populate the missing information in `template.yml`. By combining these two files, you can create a deployment manifest for the nozzle.

2. Create a `tmp` directory for the compiled manifest.

3. Use Spiff to compile a deployment manifest from the template and stub by running:

   ```
   spiff merge templates/template.yml templates/bosh-lite-stub.yml > tmp/manifest_
   bosh_lite.yml
   ```

   Save this manifest.

4. To identify the names of all deployments in the environment that you specify, run:

   ```
   bosh -e BOSH-ENVIRONMENT deployments
   ```

   Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director.

5. To obtain your BOSH Director UUID, run:

   ```
   bosh -e BOSH-ENVIRONMENT env --column=uuid
   ```

   Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director. For example:

   ```
   $ bosh -e dev env --column=uuid
   ```

6. In the compiled nozzle deployment manifest, locate the `director_uuid` property. Replace `PLACEHOLDER-DIRECTOR-UUID` with your BOSH Director UUID.

   ```
   compilation:
     cloud_properties:
       name: default
   network: example-nozzle-net
   reuse_compilation_vms: true
   workers: 1
   director_uuid: PLACEHOLDER-DIRECTOR-UUID
   ```

> **Note:** If you do not want to see the complete deployment procedure, run `scripts/make_manifest_spiff_bosh_lite` to automatically prepare the manifest.

Step 6: Creating Nozzle BOSH release

To create a nozzle BOSH release, run:

```
bosh -e BOSH-ENVIRONMENT create-release --name RELEASE-NAME
```

Where:

- `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director.
- `RELEASE-NAME` is `example-nozzle` to match the UAA client that you created in the TAS for VMs deployment manifest.

For example:

```
$ bosh -e dev create-release --name example-nozzle
Syncing blobs...
...
```

Step 7: Uploading Nozzle BOSH release

Upload the nozzle BOSH release that you created in Step 6: Create Nozzle BOSH Release above.

To upload the BOSH release, run:

```
bosh -e BOSH-ENVIRONMENT upload-release
```

Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director.

For example:

```
$ bosh -e dev upload-release
Acting as user 'admin' on 'Bosh Lite Director'

Copying packages
\----------------
example-nozzle
golang1.7

Copying jobs
\------------
example-nozzle

Generated   /var/folders/4n/qs1rjbmd1c5gfb78m3_06j6r0000gn/T/d20151009-71219-
17a5m49/d20151009-71219-rts928/release.tgz
Release size: 59.2M

Verifying release...
...
Release info
\------------
Name: nozzle-test
Version: 0+dev.2
```

```
Packages
- example-nozzle (b0944f95eb5a332e9be2adfb4db1bc88f9755894)
- golang1.7 (b68dc9557ef296cb21e577c31ba97e2584a5154b)

Jobs
- example-nozzle (112e01c6ee91e8b268a42239e58e8e18e0360f58)

License
- none

Uploading release
```

Step 8: Deploying Nozzle

To deploy the nozzle, run:

```
bosh -e BOSH-ENVIRONMENT deploy
```

Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director.

For example:

```
$ bosh -e dev deploy
Acting as user 'admin' on deployment 'example-nozzle-lite' on 'Bosh Lite Dire
ctor'
Getting deployment properties from director...
Unable to get properties list from director, trying without it...
Cannot get current deployment information from director, possibly a new deplo
yment
Please review all changes carefully

Deploying
\---------
Are you sure you want to deploy? (type 'yes' to continue):yes
```

Step 9: Viewing Nozzle output

The example nozzle outputs all of the data originating coming from the Firehose to its log files. To view this data, SSH into the `example-nozzle` VM and examine the logs.

To view nozzle output:

1. Access the nozzle VM at the IP address configured in the nozzle manifest template stub, `./templates/bosh-lite-stub.yml`, by running:

   ```
   bosh -e BOSH-ENVIRONMENT ssh
   ```

   Where `BOSH-ENVIRONMENT` is the alias that you set for your BOSH Director. For example:

```
$ bosh -e dev ssh example-nozzle
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.19.0-25-generic x86_64)
Documentation: https://help.ubuntu.com/
Last login: Wed Sep 23 21:29:50 2015 from 192.0.2.1
```

2. Use the `cat` command to output the `stdout` log file.

```
$ cat /var/vcap/sys/log/example-nozzle/example-nozzle.stdout.log
===== Streaming Firehose (will only succeed if you have admin credentia
ls)
origin:"bosh-system-metrics-forwarder" eventType:ValueMetric timestamp:
1541091851000000000 deployment:"cf-c42ae2c4dfb6f67b6c27" job:"loggregat
or_trafficcontroller" index:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" i
p:"" tags:>key:"id" value:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" > tag
s:>key:"source_id" value:"bosh-system-metrics-forwarder" > valueMetric:
>"system.swap.percent" value:0 unit:"Percent" >
origin:"bosh-system-metrics-forwarder" eventType:ValueMetric timestamp:
1541091851000000000 deployment:"cf-c42ae2c4dfb6f67b6c27" job:"loggregat
or_trafficcontroller" index:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" i
p:"" tags:>key:"id" value:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" > tag
s:>key:"source_id" value:"bosh-system-metrics-forwarder" > valueMetric:
>"system.swap.kb" value:0 unit:"Kb" >
origin:"bosh-system-metrics-forwarder" eventType:ValueMetric timestamp:
1541091851000000000 deployment:"cf-c42ae2c4dfb6f67b6c27" job:"loggregat
or_trafficcontroller" index:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" i
p:"" tags:>key:"id" value:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" > tag
s:>key:"source_id" value:"bosh-system-metrics-forwarder" > valueMetric:
>"system.disk.ephemeral.percent" value:3 unit:"Percent" >
origin:"bosh-system-metrics-forwarder" eventType:ValueMetric timestamp:
1541091851000000000 deployment:"cf-c42ae2c4dfb6f67b6c27" job:"loggregat
or_trafficcontroller" index:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" i
p:"" tags:>key:"id" value:"d1dffe15-5894-44de-b7f0-ad43161a0a7b" > tag
s:>key:"source_id" value:"bosh-system-metrics-forwarder" > valueMetric:
>"system.healthy" value:1 unit:"b" >
origin:"gorouter" eventType:ValueMetric timestamp:1541091851218590916 d
eployment:"cf-c56ab7c4dfb6f67b6c28" job:"router" index:"d5d1b5a4-2497-4
679-8d3b-66ffc978d829" ip:"10.0.4.13" tags:>key:"source_id" value:"goro
uter" > valueMetric:>"uptime" value:3.273478e+06 unit:"seconds" >
origin:"netmon" eventType:ValueMetric timestamp:1541091851234217334 dep
loyment:"cf-c56ab7c4dfb6f67b6c28" job:"diego_cell" index:"8007afda-3bff
-4856-857f-a47a43cbf994" ip:"10.0.4.18" tags:>key:"source_id" value:"ne
tmon" > valueMetric:>name:"numGoRoutines" value:13 unit:"count" >
origin:"netmon" eventType:ValueMetric timestamp:1541091851234129669 dep
loyment:"cf-c56ab7c4dfb6f67b6c28" job:"diego_cell" index:"8007afda-3bff
-4856-857f-a47a43cbf994" ip:"10.0.4.18" tags:>key:"source_id" value:"ne
tmon" > valueMetric:>"numCPUS" value:2 unit:"count" >
origin:"netmon" eventType:ValueMetric timestamp:1541091851234292367 dep
loyment:"cf-c56ab7c4dfb6f67b6c28" job:"diego_cell" index:"8007afda-3bff
-4856-857f-a47a43cbf994" ip:"10.0.4.18" tags:>key:"source_id" value:"ne
tmon" > valueMetric:>"memoryStats.numBytesAllocated" value:542328 uni
t:"count" >
```

```
origin:"netmon" eventType:ValueMetric timestamp:1541091851234279470 dep
loyment:"cf-c56ab7c4dfb6f67b6c28" job:"diego_cell" index:"8007afda-3bff
-4856-857f-a47a43cbf994" ip:"10.0.4.18" tags:>key:"source_id" value:"ne
tmon" > valueMetric:>"memoryStats.numBytesAllocatedStack" value:655360
unit:"count" >
...
```

Installing the Loggregator Firehose Plugin for cf CLI

Learn how to install the Loggregator Firehose Plug-in for the Cloud Foundry Command Line Interface (cf CLI).

The Loggregator Firehose plugin for the cf CLI allows VMware Tanzu Application Service for VMs (TAS for VMs) admins to access the output of the Loggregator Firehose. The output of the Firehose includes logs and metrics from apps deployed on TAS for VMs as well as metrics from TAS for VMs platform components. For more information about the Firehose, see the Loggregator Architecture section of the *Loggregator Architecture* topic.

For more information about using plugins with the cf CLI, see Using cf CLI Plugins.

Note: You can disable the Firehose. In place of the Firehose, you can configure an aggregate log and metric drain for your foundation. For more information, see Configure System Logging in *Configuring TAS for VMs*.

Prerequisites

Before you install the Loggregator Firehose plug-in, you need these prerequisites:

- Admin access to the TAS for VMs deployment that you want to monitor
- Cloud Foundry Command Line Interface (cf CLI) v6.12.2 or later

For information about downloading, installing, and uninstalling the cf CLI, see Installing the cf CLI.

Installing the Plug-in

To install the Loggregator Firehose plugin:

1. Add the CF Community plugin repository to your cf CLI plugins by running:

   ```
   cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
   ```

2. Install the Firehose plugin from the CF Community plugin repository by running:

   ```
   cf install-plugin -r CF-Community "Firehose Plugin"
   ```

Viewing the Firehose

To view the streaming output of the Firehose, which includes logging events and metrics from TAS for VMs system components, run:

```
cf nozzle --debug
```

Note: You must be logged in as a TAS for VMs admin to access the Firehose.

For more information about logging and metrics in TAS for VMs, see Loggregator Architecture.

Uninstalling the Plug-in

To uninstall the Loggregator Firehose plugin:

1. Run `cf plugins` to see a list of installed plugins.

   ```
   $ cf plugins
   Listing Installed Plugins...
   OK
   Plugin Name      Version    Command Name    Command Help
   FirehosePlugin   0.6.0      nozzle          Command to print out messages
   from the firehose
   ```

2. Uninstall the plugin by running:

   ```
   cf uninstall-plugin FirehosePlugin
   ```

Troubleshooting and Diagnostics

In this section:

Diagnosing Deployment Problems

This topic provides guidance for diagnosing issues encountered when installing products such as VMware Tanzu Application Service for VMs (TAS for VMs) with VMware Tanzu Operations Manager.

An important consideration when diagnosing issues is communication between VMs deployed by Ops Manager. Communication takes the form of messaging, routing, or both. If either of these go wrong, an installation can fail. For example, after installing TAS for VMs, one of the VMs must deploy a test app to the cloud during post-installation testing. The installation fails if the resulting traffic cannot be routed to the HA Proxy load balancer.

Viewing the Debug Endpoint

The debug endpoint is a web page that provides diagnostics information. If you have superuser privileges and can view the Ops Manager Installation Dashboard, you can access the debug endpoint.

To access the debug endpoint, open the following URL in a web browser:

```
https://OPS-MANAGER-FQDN/debug
```

Where `OPS-MANAGER-FQDN` is the fully-qualified domain name (FQDN) of your Ops Manager installation.

The debug endpoint offers three links:

- **Files** allows you to view the YAML files that Ops Manager uses to configure products that you install. The most important YAML file, `installation.yml`, provides networking settings and describes `microbosh`. In this case, `microbosh` is the VM whose BOSH Director component is used by Ops Manager to perform installations and updates of TAS for VMs and other products.

- **Components** describes the components in detail.
- **Rails log** shows errors thrown by the VM where the web app, such as a Rails app, is running, as recorded in the `production.log` file. To explore other logs, see Logging Tips.

Logging Tips

Identifying Where to Start

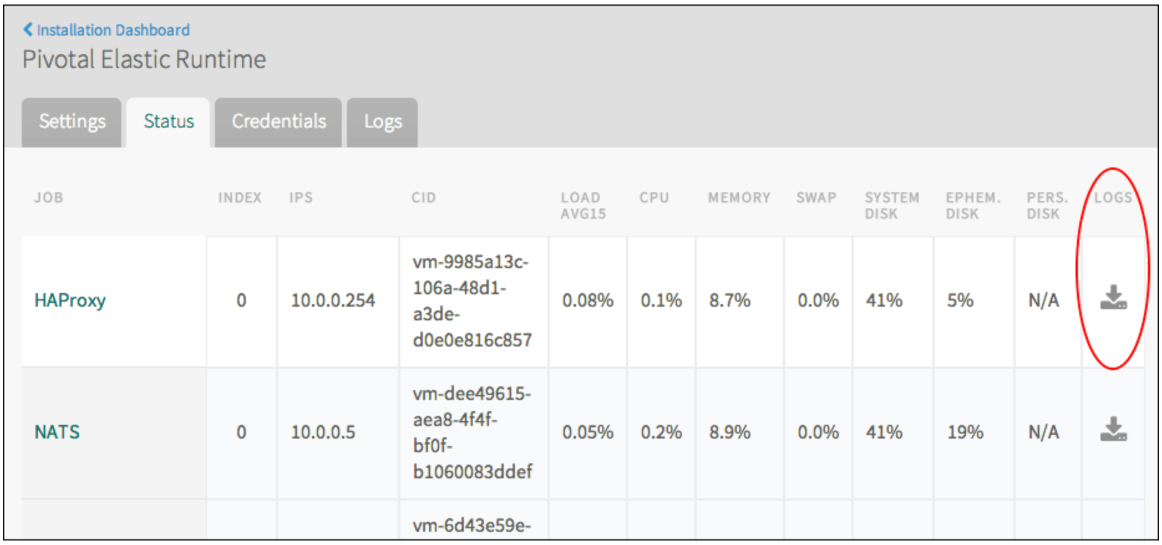
This section contains general tips for locating where a particular problem is called out in the log files. For guidance regarding specific logs, such as those for TAS for VMs components, see the sections below.

- Start with the largest and most recently updated files in the job log.
- Identify logs that contain `err` in the name.
- Scan the file contents for a “failed” or “error” string.

Viewing Logs for TAS for VMs Components

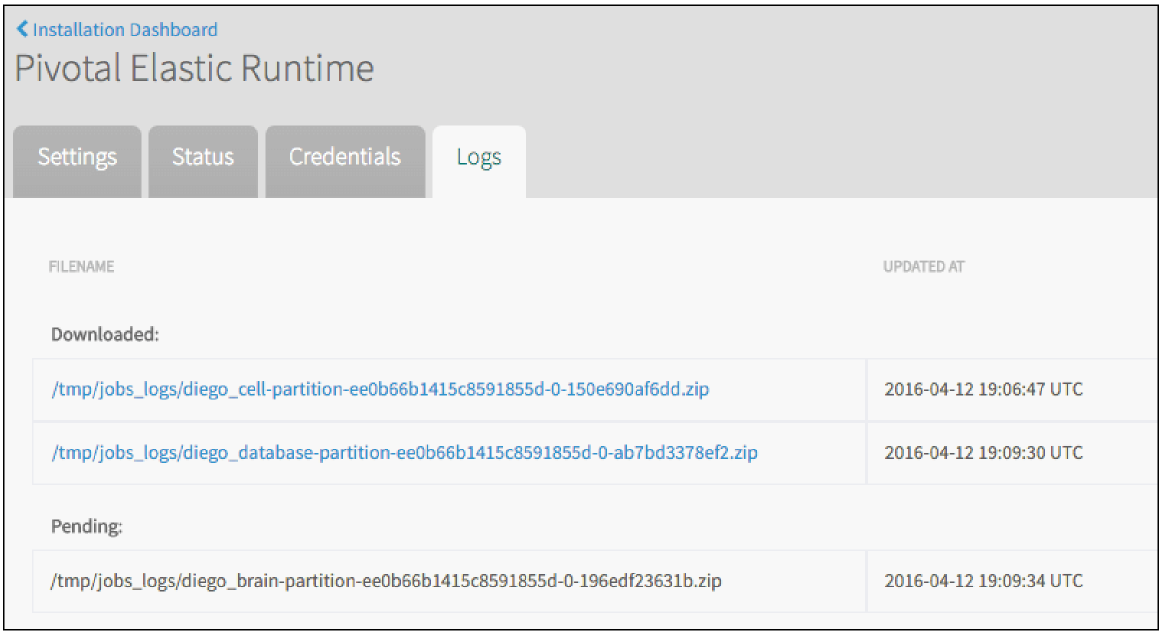
To troubleshoot specific TAS for VMs components by viewing their log files:

1. Go to your Ops Manager Installation Dashboard and click on the TAS for VMs tile.
2. Select the **Status** tab.
3. In the **Job** column, locate the component that you want to troubleshoot.
4. In the **Logs** column for the component, click the download icon.



JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
HAProxy	0	10.0.0.254	vm-9985a13c-106a-48d1-a3de-d0e0e816c857	0.08%	0.1%	8.7%	0.0%	41%	5%	N/A	⬇
NATS	0	10.0.0.5	vm-dee49615-aea8-4f4f-bf0f-b1060083ddef	0.05%	0.2%	8.9%	0.0%	41%	19%	N/A	⬇
			vm-6d43e59e-								

5. Select the **Logs** tab.



FILENAME	UPDATED AT
Downloaded:	
/tmp/jobs_logs/diego_cell-partition-ee0b66b1415c8591855d-0-150e690af6dd.zip	2016-04-12 19:06:47 UTC
/tmp/jobs_logs/diego_database-partition-ee0b66b1415c8591855d-0-ab7bd3378ef2.zip	2016-04-12 19:09:30 UTC
Pending:	
/tmp/jobs_logs/diego_brain-partition-ee0b66b1415c8591855d-0-196edf23631b.zip	2016-04-12 19:09:34 UTC

6. Once the ZIP file corresponding to the component moves to the **Downloaded** list, click the linked file path to download the ZIP file.
7. Once the download completes, unzip the file.

The contents of the log directory vary depending on which component you view. For example, the Diego Cell log directory contains subdirectories for the `metron_agent` `rep`, `monit`, and `garden` processes. To view the standard error stream for `garden`, download the Diego Cell logs and open `diego.0.job > garden > garden.stderr.log`.

Viewing Web App and BOSH Failure Logs in a Terminal Window

You can obtain diagnostic information by logging in to the VM where the BOSH Director job is running. To log in to the BOSH Director VM, you need:

- The IP address of the VM shown in the **Settings** tab of the BOSH Director tile.
- Your **import credentials**. Import credentials are the username and password used to import the `.ova` or `.ovf` file into your virtualization system.

To log in to the VM:

1. Open a terminal window.
2. To connect to the BOSH Director VM, run:

```
ssh IMPORT-USERNAME@PCF-VM-IP-ADDRESS
```

   Where:

   - `IMPORT-USERNAME` is the username you used to import the `.ova` or `.ovf` file into your virtualization system.
   - `VM-IP-ADDRESS` is the IP address of the BOSH Director installation VM.
3. Enter your import password when prompted.
4. Go to the home directory of the web app by running:

```
cd /home/tempest-web/tempest/web/
```

5. You are now in a position to explore whether things are as they can be within the web app. You can also verify that the `microbosh` component is successfully installed. A successful MicroBOSH installation is required to install TAS for VMs and any products like databases and messaging services.
6. Navigate to the BOSH installation log home directory by running:

```
cd /var/tempest/workspaces/default/deployments/micro
```

7. You may want to begin by running a tail command on the `current` log. Run:

```
cd /var/tempest/workspaces/default/deployments/micro
```

If you cannot resolve an issue by viewing configurations, exploring logs, or reviewing common problems, you can troubleshoot further by running BOSH diagnostic commands with the BOSH Command Line Interface (CLI).

Note: Do not manually modify the deployment manifest. Ops Manager overwrites manual changes to this manifest. In addition, manually changing the manifest may cause future deployments to fail.

Viewing the VMs in Your Deployment

To view the VMs in your deployment, follow the procedure specific to your IaaS.

Amazon Web Services (AWS)

To view the VMs in your AWS deployment:

1. Log in to the AWS Console.
2. Go to the EC2 Dashboard.
3. Click **Running Instances**.
4. Click the gear icon in the upper-right corner.
5. Select **job**, **deployment**, **director**, and **index**.
6. Click **Close**.

OpenStack

To view the VMs in your OpenStack deployment:

1. Install the novaclient from the python-novaclient repository on GitHub.
2. Point novaclient to your OpenStack installation and tenant by exporting the following environment variables:

```
export OS_AUTH_URL=YOUR_KEYSTONE_AUTH_ENDPOINT
export OS_TENANT_NAME=TENANT_NAME
export OS_USERNAME=USERNAME
export OS_PASSWORD=PASSWORD
```

3. List your VMs by running:

   ```
   nova list --fields metadata
   ```

vSphere

To view the VMs in your vSphere deployment:

1. Log in to vCenter.
2. Select **Hosts and Clusters**.
3. Select the top level object that contains your deployment. For example, select **Cluster**, **Datastore**, or **Resource Pool**.
4. In the top tab, click **Related Objects**.
5. Select **Virtual Machines**.
6. Right-click the **Table** heading and select **Show/Hide Columns**.
7. Select the **job**, **deployment**, **director**, and **index** boxes.

Viewing Apps Manager Logs in a Terminal Window

Apps Manager provides a graphical user interface to help manage organizations, users, apps, and spaces. For more information about Apps Manager, see Getting Started with Apps Manager

When troubleshooting Apps Manager performance, you can view the Apps Manager app logs. To view the Apps Manager app logs:

1. From a command line, log in to Ops Manager by running:

   ```
   cf login -a api.YOUR-SYSTEM-DOMAIN -u admin
   ```

   Where `YOUR-SYSTEM-DOMAIN` is your system domain.

   When prompted, enter your UAA Administrator credentials. To obtain these credentials, see **Credentials** tab in the TAS for VMs tile.

2. Target the `system` org and the `apps-manager` space by running:

   ```
   cf target -o system -s apps-manager
   ```

3. Tail the Apps Manager logs by running:

   ```
   cf logs apps-manager
   ```

Changing Logging Levels for Apps Manager

Apps Manager recognizes the `LOG_LEVEL` environment variable. The `LOG_LEVEL` environment variable allows you to filter the messages reported in Apps Manager log files by severity level. Apps Manager defines severity levels using the Ruby standard library Logger class.

By default, the Apps Manager `LOG_LEVEL` environment variable is set to `info`. The logs show more verbose messaging when you set the `LOG_LEVEL` to `debug`.

To change the Apps Manager `LOG_LEVEL` environment variable, run:

```
cf set-env apps-manager LOG_LEVEL LEVEL
```

Where `LEVEL` is the desired severity level.

You can set `LOG_LEVEL` to one of the six severity levels defined by the Ruby Logger class:

- **Level 5** `unknown`: An unknown message that should always be logged.
- **Level 4** `fatal`: An unhandleable error that results in a program crash.
- **Level 3** `error`: A handleable error condition.
- **Level 2** `warn`: A warning.
- **Level 1** `info`: General information about system operation.
- **Level 0** `debug`: Low-level information for developers.

Once set, Apps Manager log files only include messages at the set severity level and above. For example, if you set `LOG_LEVEL` to `fatal`, the log only includes `fatal` and `unknown` level messages.

Analyzing Disk Usage on Containers and Diego Cell VMs

To obtain disk usage statistics by Diego Cell VMs and containers, see Examining GrootFS Disk Usage.

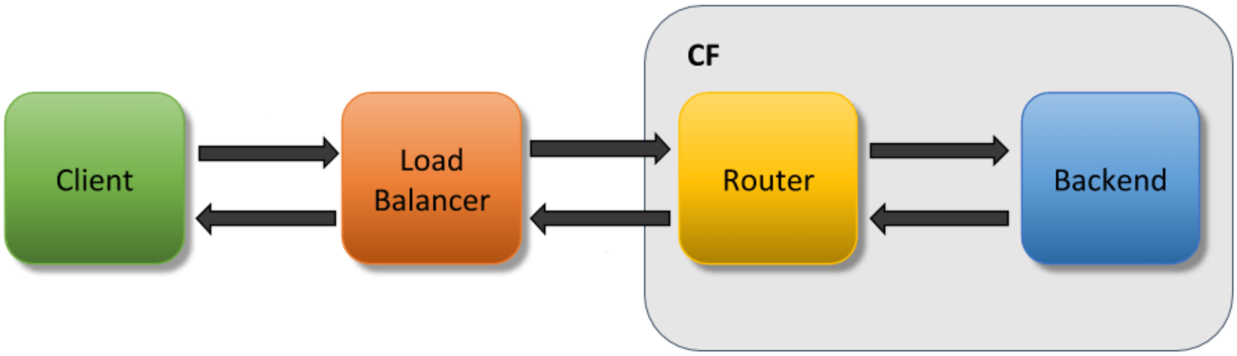
Troubleshooting slow requests in TAS for VMs

What part of the TAS for VMs request flow adds latency to your requests? Run the experiments in this article to find out.

TAS for VMs recommends running the procedures in this article in the order presented.

App requests typically transit the following components. Only the router (Gorouter) and app are within the scope of TAS for VMs. Operators can deploy the HAProxy load balancer that comes with TAS for VMs instead of, or in addition to, an infrastructure load balancer.

See the following diagram:



Any of the components in the diagram above can cause latency. Delays can also come from the network itself.

To troubleshoot slow requests and diagnose what might be causing latency to your app requests:

- Experiment 1: Measure Total Round-Trip App Requests
- Experiment 2: View Request Time in Access Logs
- Experiment 3: Duplicate Latency on Another Endpoint
- Experiment 4: Remove the Load Balancer from the Request Path
- Experiment 5: Remove Gorouter from the Request Path
- Experiment 6: Test the Network between the Router and the App

After you determine the cause of latency, see the following sections for more information:

- Use App Logs to Locate Delays in TAS for VMs
- Causes for Gorouter Latency
- Operations Recommendations

Experiment 1: Measure total round-trip app requests

To measure the total round-trip time for your deployed app that is experiencing latency, run:

```
time curl -v APP-ENDPOINT
```

Where `APP-ENDPOINT` is the URL endpoint for the deployed app.

For example:

```
$ time curl -v http://app1.app_domain.com

GET /hello HTTP/1.1
Host: app1.app_domain.com
User-Agent: curl/7.43.0
Accept: */*

HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
Date: Tue, 14 Dec 2019 00:31:32 GMT
Server: nginx
X-Content-Type-Options: nosniff
X-Vcap-Request-Id: c30fad28-4972-46eb-7da6-9d07dc79b109
Content-Length: 602
hello world!

real 2m0.707s
user 0m0.005s
sys  0m0.007s
```

The `real` time output shows that the request to `http://app1.app_domain.com` took approximately two minutes, round-trip. This seems like an unreasonably long time, so it makes sense to find out where the delay is occurring.

To narrow down the cause of latency, see the following table for information about the output that you see after running `time curl -v APP-ENDPOINT`:

Result	Explanation	Action
The output shows latency.	A component of your app request flow is causing a delay.	Continue with the rest of the experiments to try to narrow down which component is causing latency.
The output does not show latency.	Your app experiences inconsistent delays.	Try to find an endpoint that is consistently experiencing latency. If that is not possible, then continue with the following experiments, running each of them multiple times.
The output shows the error: `Could not resolve host: NONEXISTENT.com`	DNS failed to resolve.	Troubleshoot your DNS configuration.
The output of `curl` returns normally, but there is no value for `X-Vcap-Request-Id`.	The request from the load balancer did not reach TAS for VMs.	Troubleshoot your load balancer configuration.

Experiment 2: View request time in access logs

In TAS for VMs, delays can occur within Gorouter, within the app, or within the network between the two. If you suspect that you are experiencing latency, the most important logs are the access logs. The `cf logs` command streams log messages from Gorouter as well as from apps. This section describes how to find and understand the access log timestamps.

To view request time in access logs:

1. (Optional) Run `cf apps` to determine the name of the app.
2. Run:

   ```
   cf logs APP-NAME
   ```

   Where `APP-NAME` is the name of the app.
3. From another command line window, send a request to your app.
4. After your app returns a response, enter `Ctrl-C` to stop streaming `cf logs`.

   For example:

   ```
   $ cf logs app1

   2019-12-14T00:33:32.35-0800 [RTR/0] OUT app1.app_domain.com - [14/12/20
   19:00:31:32.348 +0000] "GET /hello HTTP/1.1" 200 0 60 "-" "HTTPClient/
   1.0 (2.7.1, ruby 2.3.3 (2019-11-21))" "10.0.4.207:20810" "10.0.48.67:61
   555" x_forwarded_for:"52.3.107.171" x_forwarded_proto:"http" vcap_reque
   st_id:"01144146-1e7a-4c77-77ab-49ae3e286fe9" response_time:120.00641734
   gorouter_time:0.000217 app_id:"13ee085e-bdf5-4a48-aaaf-e854a8a975df" ap
   p_index:"0" x_b3_traceid:"3595985e7c34536a" x_b3_spanid:"3595985e7c3453
   6a" x_b3_parentspanid:"-"
   2019-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 received request a
   t [14/12/2019:00:32:32.348 +0000] with "vcap_request_id": "01144146-1e7
   ```

```
a-4c77-77ab-49ae3e286fe9"
^C
```

In the example above, the first line contains the following timestamps:

- `14/12/2016:00:31:32.348`: Gorouter receives request
- `response_time:120.00641734`: Time measured from when Gorouter receives the request to when Gorouter finishes sending the response to the end user
- `gorouter_time:0.000217`: Gorouter response time, not including the time that Gorouter spent sending the request to the back end app or the time waiting for the response from the app.

The `response time` value shows that it took 120 seconds for Gorouter to send the request to the back end app, receive a response, and send that response to the end user. However, the `gorouter_time` value shows that the Gorouter took 0.000217 seconds to process the request and send the response back to the end user. This means that the bulk of the time was spent in TAS for VMs either within the app, or the network between the Gorouter and the app.

If `gorouter_time` also reported 120 seconds, it would indicate that the time was mostly spent either in the Gorouter itself, or in the network between the end user and the Gorouter, as relatively little time was spent sending the request to the app.

To narrow down the cause of latency, see the following table for information about the output that you see after running `cf logs APP-NAME`:

Result	Explanation	Action
The output shows latency within `gorouter_time`.	Gorouter, or the network between the end user and Gorouter, is causing the latency.	Continue with the rest of the experiments to try to narrow down which component is causing latency.
The output shows latency within `response_time` but not in `gorouter_time`.	The network between Gorouter and the app, or the app itself, is causing latency.	Jump to experiment 5, and continue with the rest of the experiments to try to narrow down which component is causing latency.
The output does not show latency within `response_time`.	A component before Gorouter is causing latency to the request before it reaches Gorouter.	Continue with the rest of the experiments to try to narrow down which component is causing latency.
The output does not show a log line.	Every incoming request should generate an access log message. If a request does not generate an access log message, it means Gorouter did not receive the request.	Troubleshoot your Gorouter configuration.

Experiment 3: Duplicate latency on another endpoint

The next step to debugging latency is finding an endpoint that consistently experiences delays. Use a test app that does not make any internal or external requests or database calls. For example, see dora on GitHub.

If you cannot push any apps to your foundation, find an API endpoint that does not make any external calls to use for the rest of the experiments. For example, use a health or information endpoint.

To duplicate latency on another endpoint:

1. Push an example app.
2. Measure a request's full round-trip time from the client and back as by running:

   ```
   time curl -v TEST-APP-ENDPOINT
   ```

   Where `TEST-APP-ENDPOINT` is the URL endpoint for the test app. While every network is different, this request should take less than 0.2 seconds.

See the following table for information about the output that you see after running `time curl -v TEST-APP-ENDPOINT`:

Result	Explanation	Action
The output shows latency.	One of the components in the app request path is causing latency.	Continue with the rest of the experiments, using the test app, to try to narrow down which component is causing latency.
The output does not show latency.	Something in your app configuration is causing latency.	Evaluate your app for recent changes. See below for more information.

If this experiment shows that something in your app is causing latency, use the following questions to start troubleshooting your app:

- Did you recently push any changes?
- Does your app make database calls?
  - If so, have your database queries changed?
- Does your app make requests to other microservices?
  - If so, is there a problem in a downstream app?
- Does your app log where it spends time? For more information, see Use App Logs to Locate Delays in TAS for VMs below.

Experiment 4: Remove the load balancer from request path

The next step is to remove the load balancer from the test path by sending the request directly to Gorouter. You can do this by accessing the network where Gorouter is deployed, sending the traffic directly to the Gorouter IP address, and adding the route in the host header.

To remove the load balancer from the request path:

1. Choose a router VM from your deployment and get its IP address. Record this value and use it as the `ROUTER-IP` when you run the command in a later step.
2. Run:

```
bosh ssh router/ROUTER-GUID
```

Where `ROUTER-GUID` is the unique identifier for the router VM. For more information about using this tool, see Advanced Troubleshooting with the BOSH CLI.

3. To determine the amount of time a request takes when it skips the load balancer, run:

```
time curl ROUTER-IP -H "Host: TEST-APP-ENDPOINT"
```

Where:

- `ROUTER-IP` is the router VM IP address that you located in the first step.
- `TEST-APP-ENDPOINT` is the URL endpoint for the test app.

See the following table for information about the output that you see after removing the load balancer from the app request path:

Result	Explanation	Action
The output shows latency.	The load balancer is not causing latency.	Continue with the rest of the experiments, using the test app, to try to narrow down which component is causing latency.
The output does not show latency.	A component before Gorouter is causing latency.	Look at your load balancer logs and logs for any other components that exist between the end client and Gorouter. Additionally, run these experiments with clients in different networks to help isolate where issues occur.

Experiment 5: Remove Gorouter from request path

The next step is to remove Gorouter from the request path. You can SSH into the router VM and send a request directly to the app.

To remove Gorouter from the app request path:

1. To retrieve the IP address and port number of the Diego Cell where your test app instance runs, run:

```
cf ssh TEST-APP -c "env | grep CF_INSTANCE_ADDR"
```

Where `TEST-APP` is the name of the test app.

For example:

```
$ cf ssh my-app -c "env | grep CF_INSTANCE_ADDR"
```

2. Choose any router VM from your deployment and SSH into it by running:

```
bosh ssh router/ROUTER-GUID
```

Where `ROUTER-GUID` is the unique identifier for the router VM. For more information about using this tool, see Advanced Troubleshooting with the BOSH CLI.

3. To determine the amount of time a request takes when it skips Gorouter, run `time curl CF_INSTANCE_ADDR`.

See the following table for information about the output that you see after removing Gorouter from the app request path:

Result	Explanation	Action
The output shows latency.	Gorouter is not causing latency.	Continue with the rest of the experiments, using the test app, to try to narrow down which component is causing latency.
The output does not show latency.	Gorouter is causing latency.	See Causes for Gorouter Latency and Operations Recommendations below.

Experiment 6: Test the network between the router and the app

The next step is to time how long it takes for your request to make it from the router VM to the Diego Cell where your app is deployed. You can do this by using `tcpdump` on both VMs.

To test the network between the router and the app:

1. Choose a router VM from your deployment and record its IP address. Use this value as the `ROUTER-IP` in later steps.
2. To get the IP address of the Diego Cell where your test app instance is running, run `cf ssh TEST-APP -c "env | grep CF_INSTANCE_IP"`, where `TEST-APP` is the name of the test app.
3. To get the port number of the Diego Cell where your test app instance is running, run `cf ssh TEST-APP -c "env | grep CF_INSTANCE_PORT"`, where `TEST-APP` is the name of the test app.
4. On the command line, locate the router VM that matches the `ROUTER-IP` value from the first step.
5. To SSH into the router VM, run `bosh ssh router/ROUTER-GUID`, where `ROUTER-GUID` is the unique identifier for the router VM. For more information about using this tool, see Advanced Troubleshooting with the BOSH CLI.
6. On the router VM, log in as root.
7. To capture all packets going to your app, run `tcpdump 'dst CF_INSTANCE_IP and dst port CF_INSTANCE_PORT'`.
8. In a second command line window, SSH into the Diego Cell that corresponds with `CF_INSTANCE_IP`. Run `bosh ssh digeo-cell/DIEGO-CELL-GUID`, where `DIEGO-CELL-GUID` is the unique identifier for the Diego Cell where your app is running.
9. On the Diego Cell, log in as root.
10. To capture all packets going to your app, run `tcpdump 'dst port CF_INSTANCE_PORT and src ROUTER-IP'`, where `ROUTER-IP` is the router VM IP address that you recorded in the first step.
11. In a third command line window, run `ssh ROUTER-IP`, where `ROUTER-IP` is the router VM IP address.
12. To make a request to your app, run `curl CF_INSTANCE_IP:CF_INSTANCE_PORT`.

13. Compare the first packet captured on the router VM with the first packet captured on the Diego Cell. The packets should match. Use the timestamps to determine how long it took the packet to traverse the network.

> **Note**: If you use a test app, this should be the only traffic to your app. If you are not using a test app and there is traffic to your app, then these `tcpdump` commands could result in many packet captures. If the `tcpdump` results are too verbose to track, you can write them to a `pcap` file and use `wireshark` to find the important packets. To write `tcpdump` commands to a file, use the `-w` flag. For example: `tcpdump -w router.pcap`.

See the following table for information about the output that you see after testing the network between the router and the app:

Result	Explanation	Action
The output shows latency.	Your network is causing latency.	Troubleshoot your network configuration.
The output does not show latency.	A component in your app request path is causing latency.	Repeat the diagnostic experiments to narrow down which component is causing latency. Follow the steps in Experiment 1: Measure Total Round-Trip App Requests and ensure that you are consistently experiencing latency.

Use app logs to locate delays in TAS for VMs

To gain a more detailed picture of where delays exist in your request path, augment the logging that your app generates. For example, call your logging library from the request handler to generate log lines when your app receives a request and finishes processing it:

```
2019-12-14T00:33:32.35-0800 [RTR/0] OUT app1.app_domain.com - [14/12/2019:00:
31:32.348 +0000] "GET /hello HTTP/1.1" 200 0 60 "-" "HTTPClient/1.0 (2.7.1, r
uby 2.3.3 (2019-11-21))" "10.0.4.207:20810" "10.0.48.67:61555" x_forwarded_fo
r:"52.3.107.171" x_forwarded_proto:"http" vcap_request_id:"01144146-1e7a-4c77
-77ab-49ae3e286fe9" response_time:120.00641734 app_id:"13ee085e-bdf5-4a48-aaa
f-e854a8a975df" app_index:"0" x_b3_traceid:"3595985e7c34536a" x_b3_spanid:"35
95985e7c34536a" x_b3_parentspanid:"-"
2019-12-14T00:32:32.35-0800 [APP/PROC/WEB/0]OUT app1 received request at [14/
12/2019:00:32:32.348 +0000] with "vcap_request_id": "01144146-1e7a-4c77-77ab-
49ae3e286fe9"
2019-12-14T00:32:32.50-0800 [APP/PROC/WEB/0]OUT app1 finished processing req
at [14/12/2019:00:32:32.500 +0000] with "vcap_request_id": "01144146-1e7a-4c7
7-77ab-49ae3e286fe9"
```

By comparing the router access log messages from Experiment 2: View Request Time in Access Logs with the new app logs above, you can construct the following timeline:

- `14/12/2016:00:31:32.348`: Gorouter receives request
- `2016-12-14T00:32:32.35`: App receives request
- `2016-12-14T00:32:32.50`: App finishes processing request

- `2016-12-14T00:33:32.35`: Gorouter finishes processing request

The timeline indicates that Gorouter took close to 60 seconds to send the request to the app and another 60 seconds to receive the response from the app. This suggests either of the following:

- A delay with Gorouter. See Causes for Gorouter Latency below.
- Network latency between Gorouter and the Diego Cells that host the app.

Causes for Gorouter latency

Two potential causes for Gorouter latency are:

- Routers are under heavy load from incoming client requests.
- Apps are taking a long time to process requests. This increases the number of concurrent threads held open by Gorouter, reducing capacity to handle requests for other apps.

Operations recommendations

- Monitor CPU load for Gorouters. At high CPU (70%+), latency increases. If the Gorouter CPU reaches this threshold, consider adding another Gorouter instance.
- Monitor latency of all routers using metrics from the Firehose. Do not monitor the average latency across all routers. Instead, monitor them individually on the same graph.
- Consider using Pingdom against an app on your TAS for VMs deployment to monitor latency and uptime. For more information, see the Pingdom website.
- Consider enabling access logs on your load balancer. To enable access logs, see your load balancer documentation. Just as you use Gorouter access log messages above to determine latency from Gorouter, you can compare load balancer logs to identify latency between the load balancer and Gorouter. You can also compare load balancer response times with the client response times to identify latency between client and load balancer.
- Deploy a nozzle to the Loggregator Firehose to track metrics for Gorouter. For more information, see Deploying a Nozzle to the Loggregator Firehose. Available metrics include:
  - CPU utilization
  - Latency
  - Requests per second

Troubleshooting TCP routing in TAS for VMs

Are your TCP routing issues related to DNS and load balancer misconfiguration, the TCP routing tier, or the routing subsystem? Take these steps to find out.

Rule out the app

If you are having TCP routing issues with an app, follow the procedure below to determine what to troubleshoot.

> **Note:** If you have mutual TLS app identity verification enabled, app containers accept incoming communication only from the Gorouter. This disables TCP routing. For more information, see TLS to Apps and Other Back End Services in *HTTP Routing*.

Prerequisites

This procedure requires that you have the following:

- An app with TCP routing issues.
- A TCP domain. See Routes and Domains for more information about creating a TCP domain.
- A simple HTTP web app that you can use to curl.

Procedure

1. Push a simple HTTP app using your TCP domain (cf CLI v6 only) by entering the following command:

   ```
   $ cf push MY-APP -d tcp.MY-DOMAIN --random-route
   ```

   For cf CLI v7 and later, the `-d` flag is not supported. Domain is set in the manifest, in the `routes` property.

   ```
   $ cf push MY-APP --random-route
   ```

2. Curl your app on the port generated for the route. For example, if the port is 1024:

   ```
   $ curl tcp.MY-DOMAIN:1024
   ```

3. If the curl request fails to reach the app, proceed to the next section: Rule Out DNS and the Load Balancer.
4. If the curl request to your simple app succeeds, curl the app you are having issues with.
5. If you cannot successfully curl your problem app, TCP routing is working correctly. There is an issue with the app you cannot successfully curl.

Rule out DNS and the load balancer

1. Curl the TCP router healthcheck endpoint:

   ```
   $ curl tcp.MY-DOMAIN:80/health -v
   ```

2. If you receive a `200 OK` response, proceed to the next section: Rule Out the Routing Subsystem.
3. If you do not receive a `200 OK`, your load balancer may not be configured to pass through the healthcheck port. Continue following this procedure to test your load balancer configuration.

4. Confirm that your TCP domain name resolves to your load balancer:

   ```
   $ dig tcp.MY-DOMAIN
   ...
   tcp.MY-DOMAIN. 300 IN  A 123.456.789.123
   ```

5. As an admin user, list the reservable ports configured for the `default-tcp` router group:

   ```
   $ cf curl /routing/v1/router_groups
   [
     {
       "guid": "d9b1db52-ea78-4bb9-7473-ec8e5d411b14",
       "name": "default-tcp",
       "type": "tcp",
       "reservable_ports": "1024-1123"
     }
   ]
   ```

6. Choose a port from the `reservable_ports` range and curl the TCP domain on this port. For example, if you chose port 1024:

   ```
   $ curl tcp.MY-DOMAIN:1024 -v
   ```

7. If you receive an immediate rejection, then the TCP router likely rejected the request because there is no route for this port.

8. If your connection times out, then you need to configure your load balancer to route all ports in `reservable_ports` to the TCP routers.

Rule out the routing subsystem

Send a direct request to the TCP router to confirm that it routes to your app.

1. SSH into your TCP router.

2. Curl the port of your route using the IP address of the router itself. For example, if the port reserved for your route is `1024`, and the IP address of the TCP router is `10.0.16.18`:

   ```
   $ curl 10.0.16.18:1024
   ```

3. If the curl is successful, then the load balancer either:

   - cannot reach the TCP routers, or
   - is not configured to route requests to the TCP routers from the `reservable_ports`

4. If you cannot reach the app by curling the TCP router directly, perform the following steps to confirm that your TCP route is in the routing table.

   1. Record the **Tcp Emitter Credentials** from the **UAA** row in the TAS for VMs **Credentials** tab. This OAuth client has permissions to read the routing table.

   2. Install the UAA CLI `uaac`:

```
$ gem install cf-uaac
```

3. Obtain a token for this OAuth client from UAA by providing the client secret:

   ```
   $ uaac token client get tcp_emitter
   Client secret:
   ```

4. Obtain an access_token:

   ```
   $ uaac context
   ```

5. Use the routing API to list TCP routes:

   ```
   $ curl api.MY-DOMAIN/routing/v1/tcp_routes -H "Authorization: bea
   rer TOKEN"
   [{"router_group_guid":"f3518f7d-d8a0-4279-4e89-c058040d0000",
   "backend_port":60000,"backend_ip":"10.244.00.0","port":60000,"mod
   ification_tag":{"guid":"d4cc3bbe-c838-4857-7360-19f034440000",
   "index":1},"ttl":120}]
   ```

6. In this output, each route mapping has the following:
   - **port**: your route port
   - **backend_ip**: an app instance mapped to the route
   - **backend_port**: the port number for the app instance mapped to the route
7. If your route port is not in the response, then the `tcp_emitter` may be unable to register TCP routes with the routing API. Look at the logs for `tcp_emitter` to see if there are any errors or failures.
8. If the route is in the response, but you were not able to curl the port on the TCP route directly, then the TCP router may be unable to reach the routing API. Look at the logs for `tcp_router` to see if there are any errors or failures.

Troubleshooting router error responses in TAS for VMs

Are your 502 errors the result of your infrastructure, TAS for VMs, or an app? This article helps you understand and debug these errors.

Outer error responses overview

In your deployment, 502 errors can come from any of the following:

- **Infrastructure**: issues with your load balancer or network can cause 502 errors.
- **Platform**: issues with Gorouter or Diego Cells can cause 502 errors. See the following sections:
  - Diagnose Gorouter Errors
  - Diagnose Stale Routes in Gorouter
  - Gorouter Error Classification Table

- **App**: issues with an app's load or configuration can cause 502 errors. See Diagnose App Errors.

If you are unsure of the source of 502 errors, see General Debugging Steps below.

General Debugging Steps

Some general debugging steps for any issue resulting in 502 errors are as follows:

1. Gather the Gorouter logs and Diego Cell logs at the time of the incident. To SSH into the router VM, see Advanced Troubleshooting with the BOSH CLI. To download the router VM logs from Ops Manager, see Monitoring VMs.
2. Review the logs and consider the following:
    1. Which errors are the Gorouters returning?
    2. Is Gorouter's routing table accurate? Are the endpoints for the route as expected? For more information, see Dynamic Routing Table in the Gorouter documentation on GitHub.
    3. Do the Diego Cell logs have anything interesting about unexpected app crashes or restarts?
    4. Is the app healthy and handling requests successfully? You can use request tracing headers to verify. For more information, see HTTP Headers for Zipkin Tracing in *HTTP Routing*.
3. Consider the following:
    - Does your load balancer log 502 errors, but Gorouter does not? This means that traffic is not reaching Gorouter.
    - Was there a recent platform change or upgrade that caused an increase in 502 errors?
    - Are there any suspicious metrics spiking? How is the CPU and memory utilization?

Log Formatting

Levels

The following table describes the log levels supported by Gorouter. The log level is set to `debug` and is not configurable.

Message	Description	Examples
`fatal`	Gorouter is unable to handle any requests due to a fatal error.	Gorouter cannot bind to its TCP port, a TAS for VMs component has published invalid data to Gorouter.
`error`	An unexpected error has occurred.	Gorouter failed to fetch token from UAA service.
`info`	An expected event has occurred.	Gorouter started or exited, Gorouter has begun to prune routes for stale droplets.
`debug`	A lower-level event has occurred.	Route registration, route unregistration.

Message Contents

This section section provides a sample Gorouter log entry and explanation of the contents.

```
[2017-02-01 22:54:08+0000]
{"log_level":0,"timestamp":1485989648.0895808,"message":"endpoint-
registered","source":"vcap.Gorouter.registry","data":{"uri":"0-*.login.bosh-
lite.com","backend":"10.123.0.134:8080","modification_tag":{"guid":"","index":0}}}
```

Property	Description
`log_level`	Logging level of the message
`timestamp`	Epoch time of the log
`message`	Content of the log entry
`source`	Gorouter function that initiated the log entry
`data`	Additional information that varies based on the message

Access Logs

This section provides details about Gorouter access logs.

Gorouter generates an access log in the following format when it receives a request:

```
<Request Host> - [<Start Date>] "<Request Method> <Request URL> <Request Protocol>"
<Status Code> <Bytes Received> <Bytes Sent> "<Referrer>" "<User-Agent>" <Remote Address>
<Backend Address> x_forwarded_for:"<X-Forwarded-For>" x_forwarded_proto:"<X-Forwarded-
Proto>" vcap_request_id:<X-Vcap-Request-ID> response_time:<Response Time> gorouter_time:
<Gorouter Time> app_id:<Application ID> app_index:<Application Index> x_cf_routererror:
<X-Cf-RouterError> <Extra Headers>
```

Gorouter access logs are also redirected to syslog.

See the list below for more information about the Gorouter access log fields:

- The following are optional fields: `Status Code`, `Response Time`, `Application ID`, `Application Index`, `X-Cf-RouterError`, and `Extra Headers`.
- If the access log lacks a `Status Code`, `Response Time`, `Application ID`, `Application Index`, or `X-Cf-RouterError`, the corresponding field shows `-`.
- `Response Time` is the total time it takes for the request to go through the Gorouter to the app and for the response to travel back through the Gorouter. This includes the time that the request spends traversing the network to the app and back again to the Gorouter. It also includes the time the app spends forming a response.
- `Gorouter Time` is the total time it takes for the request to go through the Gorouter initially plus the time it takes for the response to travel back through the Gorouter. This does not include the time the request spends traversing the network to the app. This also does not include the time the app spends forming a response.
- `X-Cf-RouterError` is populated if the Gorouter encounters an error. The returned values can help distinguish whether a non-2xx response code is due to an error in the Gorouter or

the back end. For more information on the possible errors, see the Diagnose App Errors section.

Diagnose Gorouter Errors

This section describes the basic structure of Gorouter logs and how to diagnose Gorouter errors.

Gorouter cannot connect to the app container

If Gorouter cannot connect to the app container, you might see this error in the `gorouter.log`:

```
[2018-07-05 17:59:10+0000] {"log_level":3,"timestamp":1530813550.92134,"message":
"backend-endpoint-failed","source":"vcap.gorouter","data":{"route-endpoint":
{"ApplicationId":"","Addr":"10.0.32.15:60099","Tags":null,"RouteServiceUrl":""},
"error":"dial tcp 10.0.32.15:60099: getsockopt: connection refused"}}
```

If TCP cannot make an initial connection to the backend, Gorouter retries TCP dial errors up to three times. If it still fails, Gorouter returns a 502 to the client and writes to the `access.log`.

Any of the following can cause connection errors between Gorouter and the app container:

- An app that is unresponsive, indicating an issue with the app.
- A stale route in Gorouter, indicating an issue with the platform. For more information, see Diagnose Stale Routes below.
- A corrupted app container, indicating a problem with the platform.

Gorouter Errors After Connecting

If Gorouter successfully dials the endpoint but an error occurs, you might see the following:

- `read: connection reset by peer` errors. When these errors occur, the Gorouter returns error to the client, marks backend ineligible, and does not retry another back end.

    - Only for idempotent requests and if route-integrity is enabled:

    If a TCP connection can be established (with Envoy), but the app itself does not send a complete http response, Gorouter retries the request on different app instances if any exist.

    A request is considered idempotent under the following circumstances (see also Gorouter source code):

    Request body is empty AND (Request method is one of “GET”, “HEAD”, “OPTIONS”, “TRACE” OR Request Header `X-Idempotency-Key` or `Idempotency-Key` is set)

    The most likely scenario for this to occur is during creation of a new app instance, while the app is not yet completely started, but Envoy is already accepting connections.

- TLS handshake errors. When these errors occur, the Gorouter retries up to three times. If it still fails, Gorouter can return a 502. These errors appear similar to the following in the `gorouter.log`, and a 502 error is logged in the `access.log`:

```
[2018-07-05 18:20:54+0000] {"log_level":3,"timestamp":1530814854.435983
4,"message":"
backend-endpoint-failed","source":"vcap.gorouter","data":{"route-endpoi
nt":
{"ApplicationId":"","Addr":"10.0.16.17:61002","Tags":null,"RouteService
Url":""},
"error":"x509:certificate is valid for 53079ca3-c4fe-4910-78b9-c1a6, no
t xxx"}}
```

- If a clients cancels a request before the server responds with headers, Gorouter returns a 499 error.

Diagnose Stale Routes in Gorouter

A stale route occurs when Gorouter contains out-of-date route information for a backend app. In nearly all cases, stale routes are self-correcting.

If TLS from Gorouter to apps and other backends is enabled, then when Gorouter detects that it is sending traffic to the wrong app, it prunes that backend app from its route table and terminates the connection. TLS from Gorouter to apps and other backends is always enabled in TAS for VMs v2.4.0 and later.

Causes of Stale Routes

When a route is unmapped or when an app container is deleted because the app is deleted or moved, a deregister message is sent to Gorouter. This message tells Gorouter to delete the route mapping to that container.

If Gorouter does not receive this deregister message, the route is now considered stale. Gorouter still attempts to send traffic to the app.

You are more likely to have stale routes when the following are true:

- SSL verification is not enabled.
- You are on one of the versions of TAS for VMs that does not send deregister messages multiple times. For the list of versions, see Enabling TLS from the Gorouter to application instances results in bad routes in PAS 2.3+ in the Knowledge Base.
- You unmapped a route to an app, but traffic to that route is still being sent to the app.

How to Locate Stale Routes

The following procedure helps you identify stale routes:

1. **Verify the state of the deployment**. Run `cf routes` for all spaces and ensure the route is only mapped to the intended apps. Sometimes there can be multiple routes using the same hostname and domain, but with different paths. If the domain is shared, check all orgs as well.
2. **Examine the Gorouter routes table**. It might be necessary to check multiple Gorouters, as it is possible that some received the proper deregister message and some did not.
    1. SSH to the VM where Gorouter is running.

    2. To print the entire Gorouter routes table, run: `/var/vcap/jobs/gorouter/bin/retrieve-local-routes | jq .`

    3. Find the entry for the suspected stale route. Note the values for `address` and `private_instance_id`.

3. **Cross-reference the Gorouter routes table entry with actual Long-Running Processes (LRPs)**:

    1. SSH onto the Diego Cell where the IP address matches the IP address that you found on the routes table entry.

    2. To get information about all of the actual LRPs, run: `cfdot actual-lrps | jq .'`

    3. Look through the actual LRPs to find the instance ID that you noted from the routes table. If that instance ID exists and the port in the route table does not exist in the ports section, then there is likely a stale route.

> **Note**: You might be tempted to use the CAPI endpoint `GET /v3/processes/:guid/stats` to find out information about the host and ports the app is using. However, it is an app developer endpoint and does not provide complete information for operators. Use the `cfdot` CLI on the Diego Cell to view the actual LRPs directly and all at once.

How to Fix Stale Routes

In most cases, you do not have to fix stale routes. Stale routes are normal in the routing table.

Gorouter fixes stale routes on its own when SSL verification is enabled. When Gorouter attempts to send traffic to a stale endpoint, it recognizes that the endpoint is stale. Gorouter prunes the stale route and then tries another endpoint. Stale routes are only a problem if an end user receives errors.

The following procedure helps you fix stale routes:

1. Ensure that SSL verification is enabled. For more information, see With TLS Enabled in *HTTP Routing*.

> **Note**: Using TLS to verify app identity depends on SSL verification. If you disable SSL verification, there is no way to avoid misrouting.

2. If there is a stale route, then restarting Gorouter fixes the immediate issue. If you restart all of the Gorouters and see the same issue for the exact same route, then the issue is not a stale route. The issue is that a bad route is continuing to be added to the route table no matter how many time Gorouter prunes it.

3. If Gorouter is continually missing deregister messages, it might be because either the NATS message bus or the Gorouters are overwhelmed. Look at the VM usage and consider scaling.

Gorouter error classification table

Use this table when you are debugging Gorouter errors. The table lists error types, status codes, and indicates if the Gorouter retries the errored request.

For each error, there is a `backend-endpoint-failure` log entry in `gorouter.log` and an error message in `gorouter.err.log`. Additionally, the `access.log` records the request status codes. For more information, see the Gorouter documentation on GitHub.

If the request is one that can be retried, Gorouter makes up to three attempts.

Error Type	Status Code	Can be re-tried?	Source of Issue	Evidence
Dial	502	Yes	App or Platform	Logs with error `dial tcp`
AttemptedTLSWithNonTLSBackend	525	Yes	Platform	Logs with error `tls: first record does not look like a TLS handshake` or `backend_tls_handshake_failed` metric increments
HostnameMismatch	503	Yes	Platform	Logs with error `x509: certificate is valid for < x >, not < y >` or `backend_invalid_id` metric increments
UntrustedCert	526	Yes	Platform	Logs with error prefix `x509: certificate signed by unknown authority` or `backend_invalid_tls_cert` metric increments
RemoteFailedCertCheck	496	Yes	Platform	Logs with error remote `error: tls: bad certificate`
ContextCancelled	499	No	Client/App	Logs with error `context canceled` This status code appears in logs only. It is never returned to clients as it occurs when the downstream client closes the connection before Gorouter responds.
RemoteHandshakeFailure	525	Yes	Platform	Logs with error remote `error: tls: handshake failure` and `backend_tls_handshake_failed` metric increments
ExpiredOrNotYetValidCertFailure	502	Yes	Platform	Logs with error `x509: certificate has expired or is not yet valid`. For example, this error can occur if the Diego Cell clock drifts.
unknown	502	No	App or Platform	If the error is not one of the types above, then `502` is the default response. Gorouter tracks `502` errors with the `gorouter.bad_gateways` metric. For more information, see Router Error: 502 Bad Gateway.

Diagnose App Errors

This section describes app-related 502 errors.

If 502 errors only occur in specific app instances and not all app instances on the platform, it is likely an app-related error. The app might be overloaded, unresponsive, or unable to connect to the database.

If all apps are experiencing 502 errors, then it could either be a platform issue, such as a misconfiguration, or an app issue, such as all apps being unable to connect to an upstream database.

If the Gorouter has back end keep-alive connections enabled, 502 errors can occur due to a race condition when the app instance keep-alive idle timeout is less than 90 seconds. For more information, see Gorouter Back End Keep-Alive Connections.

Note: Gorouter does not retry any error response returned by the app.

Gorouter specific response headers

In the case that Gorouter encounters an error connecting to an application backend, the `X-CF-RouterError` header is populated to help distinguish the origin of a non-2xx response code.

The value of the `X-Cf-Routererror` header can be one of the following:

Value	Description
invalid_cf_app_instance_header	The provided value for the `X-Cf-App-Instance` header does not match the required format of `APP_GUID:INSTANCE_ID`.
empty_host	The value for the `Host` header is empty, or the `Host` header is equivalent to the remote address. Some LB's optimistically set the `Host` header value with their IP address when there is no value present.
unknown_route	The desired route does not exist in the gorouter's route table.
no_endpoints	There is an entry in the route table for the desired route, but there are no healthy endpoints available.
Connection Limit Reached	The backends associated with the route have reached their max number of connections. The max connection number is set via the spec property `router.backends.max_conns`.
route_service_unsupported	Route services are not enabled or WebSockets requests are bound to route services. You can configure route services using the spec property `router.route_services_secret`. If the property is empty, route services are deactivated.
endpoint_failure	The registered endpoint for the desired route failed to handle the request.

Troubleshooting Ops Manager on GCP

This topic describes how to troubleshoot known issues when deploying Ops Manager on Google Cloud Platform (GCP).

Problems Connecting with Single Sign-On (SSO) for VMware Tanzu

Users may be unable to connect to applications running on Ops Manager using SSO for VMware Tanzu.

Explanation

SSO for VMware Tanzu does not support multi-subnets.

Solution

Ensure that you have configured only one subnet. See Preparing to Deploy Ops Manager on GCP Manually.

Uploading TAS for VMs Tile Causes Ops Manager Rails Application Crash

Uploading the VMware Tanzu Application Service for VMs (TAS for VMs) tile causes the Ops Manager Rails application to crash.

Explanation

In compressed format, the TAS for VMs tile is 5 GB in size. However, when uncompressed during installation, the TAS for VMs tile requires additional disk space that can exhaust the space allocated to the boot disk.

Solution

Ensure that the boot disk is allocated at least 50 GB of space. See Create the Ops Manager VM Instance in *Deploying Ops Manager on GCP Manually*.

TAS for VMs Deployment Fails - MySQL Monitor replication-canary Job

During installation of the TAS for VMs tile, the replication-canary job fails to start. The error reported in the installation log resembles the following:

```
Started updating job mysql_monitor > mysql_monitor/0
(48e7ec82-3cdf-41af-9d0f-90d1f12683c8) (canary). Failed: 'mysql_monitor/0
(48e7ec82-3cdf-41af-9d0f-90d1f12683c8)' is not running after update.
Review logs for failed jobs: replication-canary (00:05:13)

Error 400007: 'mysql_monitor/0 (48e7ec82-3cdf-41af-9d0f-90d1f12683c8)'
is not running after update.
Review logs for failed jobs: replication-canary
```

Explanation

This error can appear as a result of incorrect configuration of network traffic and missed communication between the Gorouter and a load balancer.

Solution

1. Make sure you have selected the **Forward SSL to TAS for VMs Router** option in your TAS for VMs Network Configuration.

2. Verify that you have configured the firewall rules properly and that TCP ports `80`, `443`, `2222`, and `8080` are accessible on your GCP load balancers. See Create Firewall Rules for the Network in *Preparing to Deploy Ops Manager on GCP Manually*.
3. Verify that you have configured the proper SSL certificates on your HTTP(S) load balancer in GCP. See Create HTTP Load Balancer in *Preparing to Deploy Ops Manager on GCP Manually*.
4. If necessary, re-upload a new certificate and update any required SSL Certificate and SSH Key fields in your TAS for VMs Network Configuration.

Insufficient External Database Permissions

Upgrade issues can be caused when the external database user used for the network policy DB is given insufficient permissions. To avoid this upgrade issue, ensure that the networkpolicyserver database user has the `ALL PRIVILEGES` permission.

Checking Ops Manager After a Power Failure on vSphere

This topic describes how to check the state of Ops Manager after a power failure in an on-premises vSphere installation.

If you have a procedure at your company for handling power failure scenarios and want to add steps for checking the state of Ops Manager, use this procedure as a template.

Overview

This section describes the process used by Ops Manager to recover from power failures and exceptions to that process.

Automatic Recovery Process

When power returns after a failure, vSphere and Ops Manager automatically do the following to recover your environment:

1. vSphere High Availability (HA) recovers VMs.
2. BOSH ensures that the processes on those VMs are healthy, with the exception of the Ops Manager VM and the BOSH VM itself. Ops Manager uses BOSH to deploy and manage its VMs. For more information, see the BOSH documentation.
3. The Diego runtime of VMware Tanzu Application Service for VMs (TAS for VMs) recovers apps that were running on the VMs. For more information, see Diego Components and Architecture .

Scenarios that Require Manual Intervention

Two scenarios exist that can require manual intervention when recovering your environment after a power failure:

- If TAS for VMs is configured to use a MySQL cluster instead of a single node, the cluster does not recover automatically.

- If you run Ops Manager v2.5.3 or earlier and encounter the following known issue in the BOSH Director: Monit inaccurately reports the health of UAA.

The procedure in this topic includes more detail about addressing these scenarios.

Checklist

Use the checklist in this section to ensure that Ops Manager is in a good state after a power failure.

This checklist assumes that your Ops Manager on vSphere installation is configured for vSphere HA and that you have the BOSH Resurrector enabled.

Phase	Component	Action
1	vSphere	Ensure vSphere is Running
2	Ops Manager	Ensure Ops Manager is Running
3	BOSH Director	Ensure BOSH Director is Running
4	BOSH Director	Ensure BOSH Resurrector Finished Recovering
5	TAS for VMs	Ensure VMs for TAS for VMs are Running. This may include manually recovering the MySQL cluster.
6	TAS for VMs	Ensure Apps Hosted on TAS for VMs are Running
7	Ops Manager Healthwatch	Check the Healthwatch Dashboard

Phase 1: Ensure vSphere is Running

Ensure that vSphere is running and has fully recovered from the power failure. Check your internal vSphere monitoring dashboard.

Phase 2: Ensure Ops Manager is Running

To ensure that Ops Manager is running, do the following:

1. Open vCenter and navigate to the resource pool that hosts your Ops Manager deployment.
2. Select the **Related Objects** > **Virtual Machines**.
3. Locate the VM with the name `OpsMan-VERSION`, for example `OpsMan-2.6`.
4. Review the **State** and **Status** columns for the Ops Manager VM. If Ops Manager is running, the columns show **Powered On** and **Normal**. If the columns do not show **Powered On** and **Normal**, restart the VM.

Phase 3: Ensure BOSH Director is Running

To ensure that BOSH Director is running, do the following:

1. In a browser, navigate to the Ops Manager UI and select the **BOSH Director for vSphere** tile.

> **Note**: If you do not know the URL of the Ops Manager VM, you can use the IP address that you obtain from vCenter.

2. Select **Status**.
3. In the **BOSH Director** row, locate and record the **CID**. The CID is the cloud ID and corresponds to the VM name in vSphere.
4. Go to the vCenter resource pool or cluster that hosts your Ops Manager deployment.
5. Select **Related Objects** > **Virtual Machines**.
6. Locate the VM with the name that corresponds to the **CID** value that you copied.
7. Review the **State** and **Status** columns for the VM. If the **State** column does not show **Powered On**, restart the VM.
8. If the **State** column shows that the VM is **Powered On** but the **Status** column does not does not show **Normal**, it may be due the following known issue: Monit inaccurately reports the health of UAA. To resolve this issue, do the following:
   1. SSH into the BOSH Director VM using the instructions in SSH into the BOSH Director VM.
   2. Run the following command to see that all processes are running:

      ```
      monit summary
      ```

   3. If the `uaa` process is not running, run the following command:

      ```
      monit restart UAA
      ```

Phase 4: Ensure BOSH Resurrector Finished Recovering

If enabled, the BOSH Resurrector re-creates any VMs in a problematic state after being recovered by vSphere HA.

To ensure BOSH Resurrector finished recovering, do the following:

1. Log in to the Ops Manager VM with SSH using the instructions in Log in to the Ops Manager VM with SSH.
2. Authenticate with the BOSH Director VM using the instructions in Authenticate with the BOSH Director VM.
3. Run the following command to see if there is any currently running or queued Resurrector activity:

   ```
   bosh tasks --all -d ''
   ```

   Review the task description for `scan` and `fix`. If no task are running, the BOSH Director has probably finished recovering. Run `bosh tasks --recent --all -d ''` to view finished tasks.

Phase 5: Ensure the VMs for TAS for VMs are Running

> **Note**: You can also apply the steps in this section to any Ops Manager services. To further ensure the health of Ops Manager services, use the Ops Manager Healthwatch dashboard and the documentation for each service.

To ensure that the VMs for TAS for VMs are running, do the following:

1. Run the following command to confirm that VMs are running:

   ```
   bosh vms
   ```

   BOSH lists VMs by deployment. The deployment with the `cf-` prefix is the TAS for VMs deployment.

2. If the `mysql` VM is not running, it is likely because it is a cluster and not a single node. Clusters require manual intervention after an outage. For instructions to confirm and recover the cluster, see Manually Recover MySQL (Clusters Only).

3. If any other VMs are not running, run the following command:

   ```
   bosh cck -d DEPLOYMENT
   ```

   This command scans for problems and provides options for recovering VMs. For more information, see IaaS Reconciliation in the BOSH documentation.

4. If you cannot get all VMs running, contact VMware Tanzu Support for assistance. Provide the following information:

   - You have started this checklist to recover from a power failure on vSphere
   - A list of failing VMs
   - Your Ops Manager version

Manually Recover MySQL (Clusters Only)

To manually recover MySQL, do the following:

1. In a browser, go to the Ops Manager UI and select the **VMware Tanzu Application Service for VMs** tile.
2. Select the **Resource Config** pane.
3. Review the **INSTANCES** column of the **MySQL Server** job. If the number of instances is greater than `1`, manually recover MySQL by following Recovering From MySQL Cluster Downtime.

Phase 7: Ensure Apps Hosted on TAS for VMs are Running

To ensure apps hosted on TAS for VMs are running, do the following:

1. Check the status of an app that you run on Ops Manager. Run any health checks that the app has or visit the URL of the app to verify that it is working.
2. Push an app to Ops Manager.

Phase 8: Check the Healthwatch Dashboard

You can use Ops Manager Healthwatch to further assess the state of Ops Manager. For more information, see Using Ops Manager Healthwatch.

Related Products

See the following links for product documentation related to VMware Tanzu Application Service for VMs (TAS for VMs):

- VMware Tanzu Operations Manager (Ops Manager) Documentation
- VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) Documentation
- Tanzu Kubernetes Grid (TKG) Documentation