VMware GemFire for TAS Documentation 1.14

VMware GemFire for Tanzu Application Service 1.14

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

https://docs.vmware.com/

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

Contents

VMware GemFire® for Tanzu™ Application Service v1.14

VMware GemFire for Tanzu Application Service is a high-performance, high-availability caching layer for VMware Tanzu Application Service for VMs (TAS for VMs). GemFire for TAS offers an in-memory key-value store. It delivers low-latency responses to a large number of concurrent data-access requests.

GemFire for TAS provides a service broker to create in-memory data clusters on demand. These clusters are dedicated to the TAS for VMs space and tuned for specific use cases defined by your service plan. Service operators can create multiple plans to support different use cases.

GemFire for TAS uses VMware GemFire. See VMware GemFire Java API Reference for links to the API for client access to data objects within VMware GemFire. GemFire for TAS supports writing JavaScript apps with the Node.js Client for VMware GemFire®

This documentation performs the following functions:

- Describes the features and architecture of GemFire for TAS
- Provides the VMware Tanzu Application Service for VMs operator with instructions for installing, configuring, and maintaining GemFire for TAS
- Provides app developers instructions for choosing a service plan and creating and deleting GemFire for TAS service instances
- Provides app developers instructions for binding apps

VMware GemFire for TAS Release Notes

This topic contains the release notes for VMware GemFire for TAS Release Notes.

v1.14.9

Release Date: May 31, 2023

Features included in this release:

- This version of VMware GemFire for Tanzu Application Service runs GemFire 9.15.6.
- This release supports VMware GemFire for Redis Apps v1.1.0.
- This release supports VMware Tanzu Application Service for VMs (TAS) versions 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 2.13 (partial), 3.0 (partial), and 4.0 (partial).

  Support for TAS 2.13, TAS 3.0, and TAS 4.0 is partially compatible. TAS 2.13, TAS 3.0, and TAS 4.0 contain a breaking change (described in the TAS 2.13 Release Notes, with the following recommended workaround:

  "Service instance metrics will not be not retrievable unless users deselect the **Enable Log Cache Syslog Ingestion** checkbox in the System Logging pane of the Tanzu Application Service for VMs tile in Ops Manager."

For VMware Tanzu Application Service for VMs and Ops Manager version compatibility, see Product Snapshot and Version Compatibility.

VMware GemFire for TAS v1.14.9 resolves the following issue:

- **GF4TAS-233:** Fixed a "Response exceeded maximum allowed length" error that occurred while running the `prime-cluster-for-pcc` errand by truncating its output.

v1.14.8

Release Date: April 27, 2023

Features included in this release:

- Support for Redis configuration options on service instances of VMware GemFire for Tanzu Application Service. For more information about these options, see Service Instance Configuration Parameters in *Using the VMware GemFire for Redis Apps Add-on.*
- This version of VMware GemFire for Tanzu Application Service runs GemFire 9.15.5.
- This release supports VMware Tanzu Application Service for VMs (TAS) versions 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 2.13 (partial), 3.0 (partial), and 4.0 (partial).

  Support for TAS 2.13, TAS 3.0, and TAS 4.0 is partially compatible. TAS 2.13, TAS 3.0, and TAS 4.0 contain a breaking change (described in the TAS 2.13 Release Notes, with the following recommended workaround:

"Service instance metrics will not be not retrievable unless users deselect the **Enable Log Cache Syslog Ingestion** checkbox in the System Logging pane of the Tanzu Application Service for VMs tile in Ops Manager."

For VMware Tanzu Application Service for VMs and Ops Manager version compatibility, see Product Snapshot and Version Compatibility.

v1.14.7

Release Date: February 22, 2023

Features included in this release:

- This version of VMware GemFire for Tanzu Application Service runs GemFire 9.15.4.
- This release supports VMware Tanzu Application Service for VMs (TAS) versions 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 2.13 (partial), and 3.0 (partial).

  Support for TAS 2.13 and TAS 3.0 is partially compatible. TAS 2.13 and TAS 3.0 contain a breaking change (described in the TAS 2.13 Release Notes, with the following recommended workaround:

  "Service instance metrics will not be not retrievable unless users deselect the **Enable Log Cache Syslog Ingestion** checkbox in the System Logging pane of the Tanzu Application Service for VMs tile in Ops Manager."

For VMware Tanzu Application Service for VMs and Ops Manager version compatibility, see Product Snapshot and Version Compatibility.

v1.14.6

Release Date: December 14, 2022

Features included in this release:

- Addition of the `PCC_CLUSTER-READ-ONLY` security role. This role has `CLUSTER:READ, DATA:READ` permissions.
- This version of VMware GemFire for Tanzu Application Service runs GemFire 9.15.3.
- This release supports VMware Tanzu Application Service for VMs (TAS) versions 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 2.13 (partial), and 3.0 (partial).

  Support for TAS 2.13 and TAS 3.0 is partially compatible. TAS 2.13 and TAS 3.0 contain a breaking change (described in the TAS 2.13 Release Notes, with the following recommended workaround:

  "Service instance metrics will not be not retrievable unless users deselect the **Enable Log Cache Syslog Ingestion** checkbox in the System Logging pane of the Tanzu Application Service for VMs tile in Ops Manager."

For VMware Tanzu Application Service for VMs and Ops Manager version compatibility, see Product Snapshot and Version Compatibility.

v1.14.5

Release Date: August 11, 2022

Features included in this release:

- VMware GemFire for TAS supports use of the VMware GemFire for Redis Apps add-on. For more information, see Using the VMware GemFire for Redis Apps Add-on.
- Support for Transport Layer Security (TLS) 1.3.
- Support for platform tags. For more information, see Configure Platform Tags in *Installing and Configuring VMware Tanzu GemFire*.
- Underlying JDK updated to 8u342.
- This version of VMware GemFire for Tanzu Application Service runs GemFire 9.15.1.
- Supports VMware Tanzu Application Service for VMs (TAS) versions 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, and 2.13 (partial).

  Support for TAS 2.13 is partially-compatible. TAS 2.13 contains a breaking change (as outlined in the TAS 2.13 release notes), with the following recommended workaround:

  “Service instance metrics will not be not retrievable unless users deselect the **Enable Log Cache Syslog Ingestion** checkbox in the System Logging pane of the Tanzu Application Service for VMs tile in Ops Manager.”

For VMware Tanzu Application Service for VMs and Ops Manager version compatibility, see Product Snapshot and Version Compatibility.

VMware GemFire for TAS v1.14.5 resolves the following issue:

- **GF4TAS-1:** Fixed an issue where certificates associated with a service instance were not deleted when the service instance was deleted.

Note: Jump upgrades to VMware GemFire for TAS v1.14.5 requires multiple steps. For example, if you are on 1.10.9, you must first JUMP upgrade to 1.14.3, then you can upgrade to 1.14.5.

---

v1.14.4

Release Date: March 18, 2022

Version 1.14.4 of VMware GemFire for TAS runs Apache Geode v1.14.4.

VMware GemFire for TAS v1.14.4 resolves the following issue:

GEODE-10093, GEM-3541: Fixed an issue in which the DeltaSession getAttribute method logs an NPE and returns an unserialized value when called on an attribute with a null value.

---

v1.14.3

Release Date: January 28, 2022

Version 1.14.3 of VMware GemFire for TAS runs Apache Geode v1.14.3.

VMware GemFire for TAS v1.14.3 resolves the following issues:

- **GEODE-9060, GEM-3488:** Cleaned up replicates list for the GII (Get Initial Image) provider to improve restarts.
- **GEODE-9905, GEM-3490:** Upgraded log4j to v2.17.1 to address CVE-2021-44832 and CVE-2021-45105.
- **GEM-3237:** Addresses CVE-2021-29153, deserialization of untrusted data vulnerability in JMX over RMI and REST APIs, by configuring process-wide serialization filtering when the system property `--J=-Dgeode.enableGlobalSerialFilter=true` is supplied.

---

v1.14.2

Release Date: December 16, 2021

Version 1.14.2 of VMware GemFire for TAS runs Apache Geode v1.14.2.

VMware GemFire for TAS v1.14.2 resolves the following issues:

- **GEODE-9898, GEM-3486:** Upgraded log4j to v2.16.0 to address CVE-2021-45046. For more information on these vulnerabilities and their impact on VMware products please see VMSA-2021-0028.

---

v1.14.1

Release Date: December 11, 2021

Version 1.14.1 of VMware GemFire for TAS runs Apache Geode v1.14.1.

VMware GemFire for TAS v1.14.1 resolves the following issues:

- **GEODE-9888, GEM-3478:** Upgraded log4j to address CVE-2021-44228.
- **GEODE-9825, GEM-3444:** Fixed network buffer handling problem (when TLS was deactivated, and cluster members used disparate `socket-buffer-size settings`) that could result in hangs.

---

v1.14.0

Release Date: September 28, 2021

Version 1.14.0 of VMware GemFire for TAS runs Apache Geode v1.14.0.

Features included in v1.14.0:

- *Jump upgrades* to v1.14 are supported from the following versions:
  - 1.13.4
  - 1.12.3
  - 1.11.3
  - 1.10.7

  Upgrading to GemFire for TAS 1.14 from any version other than those shown requires multiple steps. For example, if your current version is v1.10.5, you must first upgrade to v1.10.7, from which you can then jump upgrade to 1.14.

If your application connects via the Services Gateway feature, there are additional prerequisites.

For details on these and other upgrade issues, see Upgrading VMware GemFire.

- GemFire for TAS supports VMware Tanzu Application Service for VMs versions 2.7, 2.8, 2.9, 2.10, and 2.11.
- The *Advanced configuration* feature allows users to configure custom JVM options, GemFire properties, credentials, and Resource Manager properties. See Advanced Service Instance Configuration for more information.

Upgrade Notes

- Upgrades to current versions of VMware GemFire for TAS may require multiple steps. For example, to upgrade from v1.10.9 to v1.14.9, you must first jump upgrade to v1.14.3, then upgrade to v1.14.9.

  For more information about upgrading, see Upgrading VMware GemFire.
- GemFire Native Client applications must be upgraded to at least version 10.3.0 before upgrading to GemFire for TAS v1.14.9, due to a protocol alignment/client-server handshake fix in GemFire v9.15.

Product Snapshot and Version Compatibility

The following tables provide version and version-support information about VMware GemFire for Tanzu Application Service.

Version 1.14 Snapshots

Tables in this section are product snapshots for all of the v1.14 patch releases.

Version 1.14.9

Element	Details
Version	1.14.9
Release date	May 31, 2023
Software component version	VMware GemFire v9.15.6
Compatible GemFire for Redis Apps version	v1.1.0
Compatible VMware Tanzu Application Service for VMs versions	v4.0(partial), v3.0 (partial), v2.13 (partial), v2.12, v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager versions	v3.0, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version (Pivotal stemcell)	Ubuntu Xenial v621.543 or a more recent version
JDK shipped with VMware GemFire	1.8.0_342
Minimum Java buildpack version required for apps	v4.46 or a more recent version

Version 1.14.8

Element	Details
Version	1.14.8
Release date	April 27, 2023
Software component version	VMware GemFire v9.15.5
Compatible VMware Tanzu Application Service for VMs versions	v3.0 (partial), v2.13 (partial), v2.12, v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager versions	v3.0, v2.10

IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version (Pivotal stemcell)	Ubuntu Xenial v621.488 or a more recent version
JDK shipped with VMware GemFire	1.8.0_342
Minimum Java buildpack version required for apps	v4.46 or a more recent version

Version 1.14.7

Element	Details
Version	1.14.7
Release date	February 22, 2023
Software component version	VMware GemFire v9.15.4
Compatible VMware Tanzu Application Service for VMs versions	v3.0 (partial), v2.13 (partial), v2.12, v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager versions	v3.0, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version (Pivotal stemcell)	Ubuntu Xenial v621.401 or a more recent version
JDK shipped with VMware GemFire	1.8.0_342
Minimum Java buildpack version required for apps	v4.46 or a more recent version

Version 1.14.6

Element	Details
Version	1.14.6
Release date	December 14, 2022
Software component version	VMware GemFire v9.15.3
Compatible VMware Tanzu Application Service for VMs versions	v3.0 (partial), v2.13 (partial), v2.12, v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager versions	v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version (Pivotal stemcell)	Ubuntu Xenial v621.256 or a more recent version
JDK shipped with VMware GemFire	1.8.0_342
Minimum Java buildpack version required for apps	v4.46 or a more recent version

Version 1.14.5

Element	Details
Version	1.14.5
Release date	August 11, 2022
Software component version	VMware GemFire v9.15.1
Compatible VMware Tanzu Application Service for VMs versions	v2.13 (partial), v2.12, v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager versions	v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version (Pivotal stemcell)	Ubuntu Xenial v621.256 or a more recent version
JDK shipped with VMware GemFire	1.8.0_342
Minimum Java buildpack version required for apps	v4.46 or a more recent version

Version 1.14.4

Element	Details
Version	1.14.4
Release date	February 22, 2023
Software component version	VMware GemFire v1.14.4
Compatible VMware Tanzu Application Service for VMs version(s)	v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager version(s)	v2.11, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version	Xenial v621.211 or a more recent version
JDK shipped with VMware GemFire	1.8.0_242
Minimum Java buildpack version required for apps	v4.18

Version 1.14.3

Element	Details
Version	1.14.3
Release date	February 22, 2023
Software component version	VMware GemFire v1.14.3
Compatible VMware Tanzu Application Service for VMs version(s)	v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager version(s)	v2.11, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere

IPsec support	Yes
Required BOSH stemcell version	Xenial v621.198 or a more recent version
JDK shipped with VMware GemFire	1.8.0_242
Minimum Java buildpack version required for apps	v4.18

Version 1.14.2

Element	Details
Version	1.14.2
Release date	February 22, 2023
Software component version	VMware GemFire v1.14.2
Compatible VMware Tanzu Application Service for VMs version(s)	v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager version(s)	v2.11, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version	Xenial v621.183 or a more recent version
JDK shipped with VMware GemFire	1.8.0_242
Minimum Java buildpack version required for apps	v4.18

Version 1.14.1

Element	Details
Version	1.14.1
Release date	February 22, 2023
Software component version	VMware GemFire v1.14.1
Compatible VMware Tanzu Application Service for VMs version(s)	v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager version(s)	v2.11, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version	Xenial v621.183 or a more recent version
JDK shipped with VMware GemFire	1.8.0_242
Minimum Java buildpack version required for apps	v4.18

Version 1.14.0

Element	Details

Version	1.14.0
Release date	February 22, 2023
Software component version	VMware GemFire v1.14.0
Compatible VMware Tanzu Application Service for VMs version(s)	v2.11, v2.10, 2.9, 2.8, 2.7
Compatible Ops Manager version(s)	v2.11, v2.10
IaaS support	AWS, Azure, GCP, OpenStack, vSphere
IPsec support	Yes
Required BOSH stemcell version	Xenial v621.154 or a more recent version
JDK shipped with VMware GemFire	1.8.0_242
Minimum Java buildpack version required for apps	v4.18

Supported Server Versions by Release

The following table identifies the VMware GemFire or Apache Geode version incorporated into current and past versions of VMware GemFire for Tanzu Application Service.

VMware GemFire for TAS version	VMware GemFire version	Apache Geode version
1.14.9	v9.15.6	
1.14.8	v9.15.5	
1.14.7	v9.15.4	
1.14.6	v9.15.3	
1.14.5	v9.15.1	
1.14.4		v1.14.4
1.14.3		v1.14.3
1.14.2		v1.14.2
1.14.1*		v1.14.1
1.14.0*		v1.14.0
1.13.7		v1.13.8
1.13.6		v1.13.7
1.13.5		v1.13.6
1.13.4*		v1.13.5
1.13.3*		v1.13.4
1.13.2*		v1.13.3
1.13.1*		v1.13.2
1.13.0*		v1.13.1

1.13.0 Beta*		Build 312 for v1.13
1.12.4	9.10.13	
1.12.3	9.10.8	
1.12.2	9.10.6	
1.12.1	9.10.5	
1.12.0	9.10.2	
1.11.3	9.9.5	
1.11.2	9.9.3	
1.11.1	9.9.2	
1.11.0	9.9.1	
1.10.9	9.9.7	
1.10.8	9.9.6	
1.10.7	9.9.6	
1.10.6*	9.9.6	
1.10.5*	9.9.5	
1.10.4*	9.9.4	
1.10.3*	9.9.3	
1.10.2*	9.9.2	
1.10.1*	9.9.1	
1.10.0*	9.9.0	
1.9.2	9.8.6	
1.9.1	9.8.4	
1.9.0*	9.8.3	

* This version is not available on VMware Tanzu Network.

VMware GemFire for TAS and Other Services

This topic lists which service tiles offer on-demand and pre-provisioned service plans for use with VMware GemFire for TAS. These plans let developers provision service instances when they want.

These contrast with the older *pre-provisioned* service plans, which require operators to provision the service instances during installation and configuration through the service tile UI.

The following table lists which service tiles offer on-demand and pre-provisioned service plans:

Service tile	Standalone product related to the service	Supports on-demand	Supports pre-provisioned
RabbitMQ for VMware Tanzu [VMs]	Pivotal RabbitMQ	Yes	Yes. Only recommended for test environments.
Redis for VMware Tanzu	Redis	Yes	Yes (shared-VM plan). Recommended only for test environments.
MySQL for VMware Tanzu	MySQL	Yes	No
VMware GemFire	VMware GemFire	Yes	No

For services that offer both on-demand and pre-provisioned plans, you can choose the plan you want to use when configuring the tile.

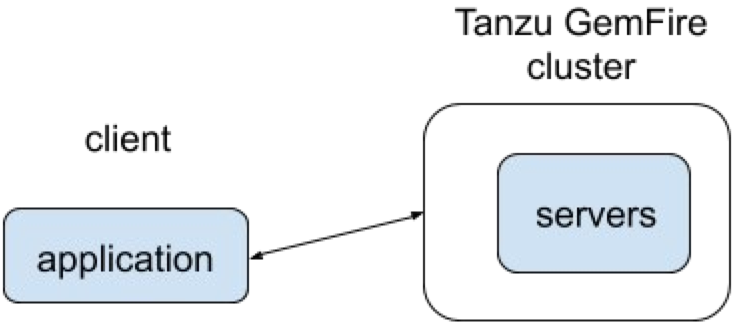
VMware GemFire for TAS Architecture

This topic introduces and describes VMware GemFire for TAS architecture.

VMware GemFire Basics

VMware GemFire is the data store within VMware GemFire for Tanzu Application Service. A GemFire for TAS service instance requires a small amount of administrative setup, and any app will use a limited portion of the API.

The GemFire for TAS architectural model is a client-server model. The clients are apps or microservices, and the servers are a set of cluster servers maintained by a GemFire for TAS service instance. The servers provide a low-latency, consistent, fault-tolerant data store within GemFire for TAS.



The cluster holds data in key/value pairs. Each pair is called an **entry**. Entries are logically grouped into sets called **regions**. A region is a map (or dictionary) data structure.

The app (client) uses GemFire for TAS as a cache. A cache lookup (read) is a get operation on a region. The cache operation of a cache write is a put operation on a region. The command-line interface, called gfsh, facilitates region administration. Use gfsh to create and destroy regions within the GemFire for TAS service instance.

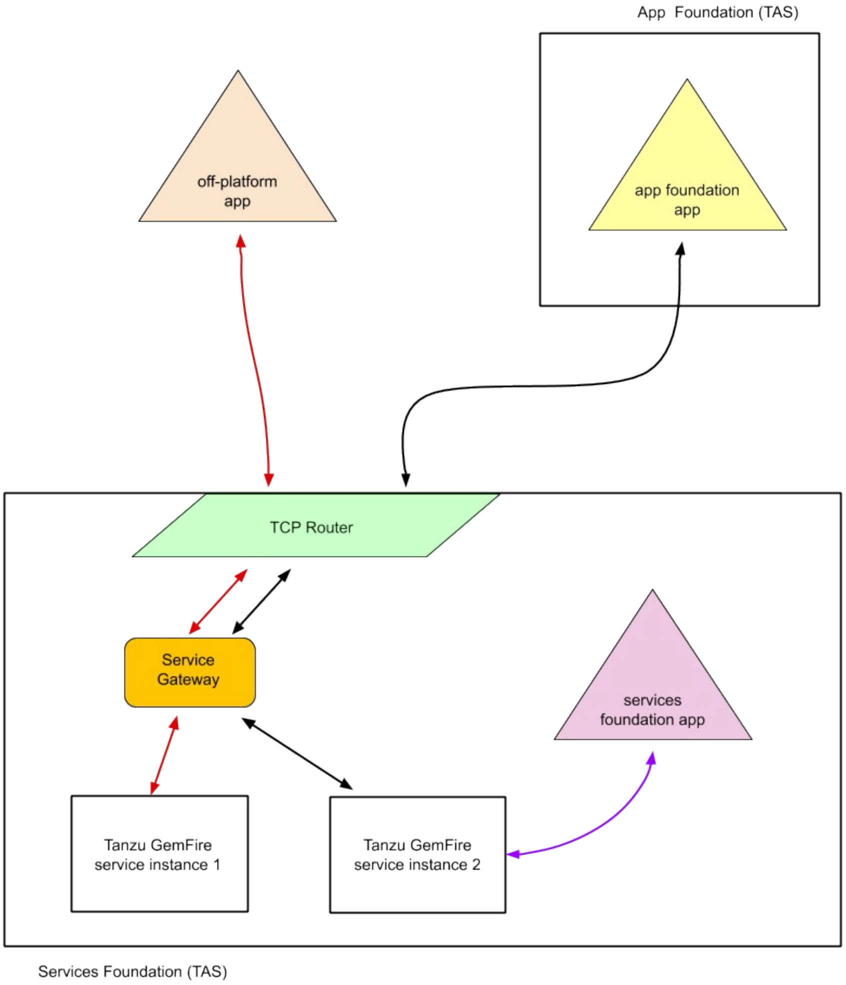
The App’s Location

A running app’s location affects how it communicates with the GemFire for TAS service instance. Apps may run in one of three locations:

- In the services foundation where the GemFire for TAS service instance runs. These apps require no extra communication support.
- In an app foundation, where a user-provided service instance proxies the communication to the GemFire for TAS service instance. Communication with the GemFire for TAS service

instance requires a service gateway.

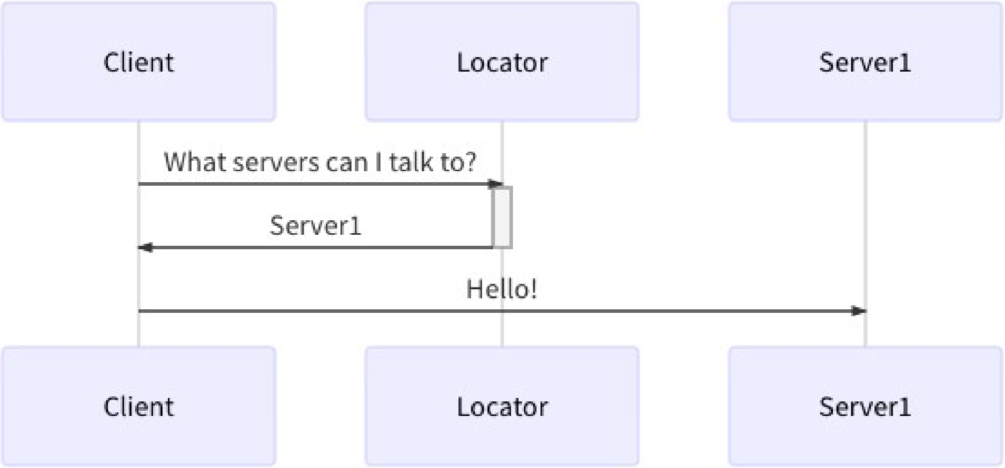
- Outside of any platform, where a platform is composed of any and all foundations. Communication with the GemFire for TAS service instance requires a service gateway.



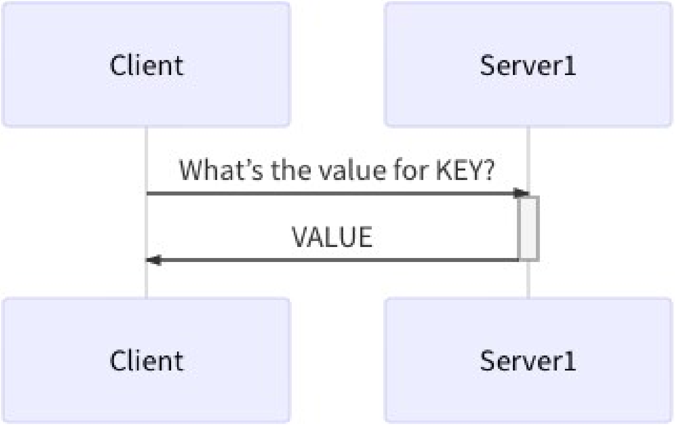
To use a service gateway, the operator must Configuring a Service Gateway. The developer must Provide Optional Parameters when creating the GemFire for TAS service instance. And, the app must be given and use properties that authenticate and authorize Communicating with a Service Instance through the service gateway.

Member Communication

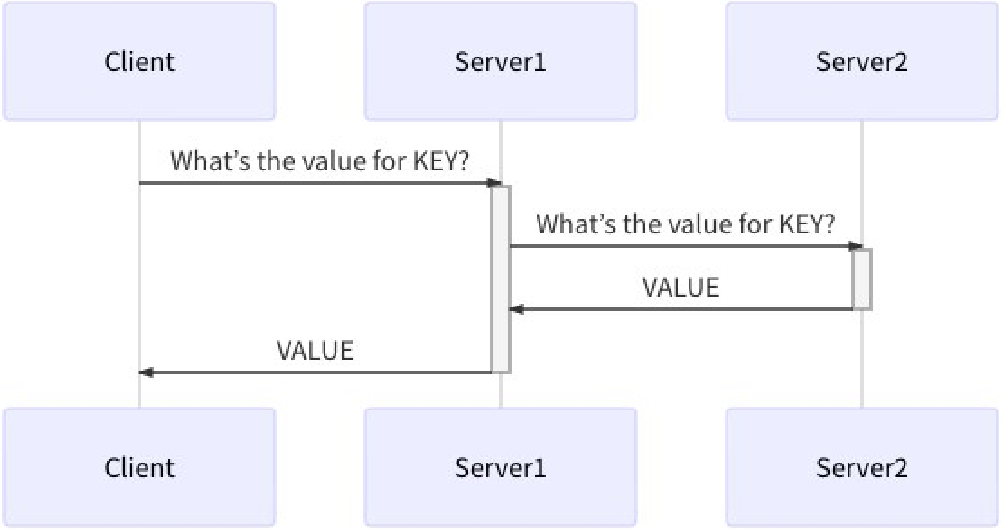
When a client connects to the cluster, it first connects to a locator. The locator replies with the IP address of a server for it to talk to. The client then connects to that server.



When the client wants to read or write data, it sends a request directly to the server.



If the server doesn't have the data locally, it fetches it from another server.



Recommended Usage and Limitations

This topic lists the topics in this documentation set that discuss recommended usage and limitations when using VMware GemFire for TAS.

- See Design Patterns for descriptions of the variety of design patterns that VMware GemFire for Tanzu Application Service supports.
- GemFire for TAS stores objects in key/value format, where the value can be any object.
- Using PDX objects as region entry keys is highly discouraged. See Using PDX Objects as Region Entry Keys for details.

Limitations

- See gfsh Command Restrictions for limitations on the use of gfsh commands.
- GemFire for TAS does not support scaling down the cluster.
- GemFire for TAS does not support plan migrations. The `-p` option to the `cf update-service` command is not supported.
- GemFire for TAS does not publish locator metrics for service instances created as Co-located Single VM plans or Co-located Multi VM plans.
- The Apache Geode Developer REST API is not available for data operations.
- Because VMware Tanzu Application Service for VMs applications are stateless, the *durable client* feature of Apache Geode should not be used.

Security

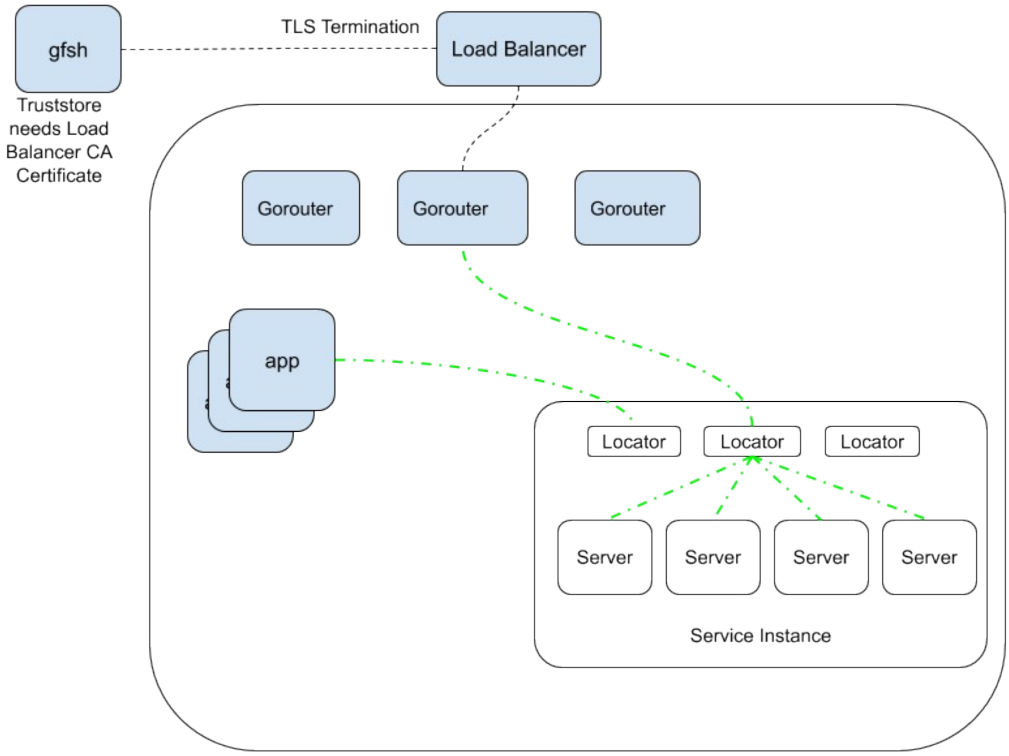
This topic describes security when using VMware GemFire for TAS.

The security measures implemented for a VMware Tanzu Application Service for VMs (TAS for VMs) foundation and for VMware GemFire for Tanzu Application Service service instances within that foundation attempt to reduce harm from agents with foundation access. For a general presentation on TAS for VMs security, see TAS for VMs Security in the VMware Tanzu Application Service for VMs documentation.

Transport-Layer Security (TLS) encryption prevents easy access to communication between components, and role-based authentication and authorization limits access to data.

TLS Encryption for the GemFire for TAS Service Instance

Without TLS encryption with and within the GemFire for TAS service instance, the diagram below identifies via green dotted-and-dashed lines the unencrypted, plaintext communication that a bad agent with TAS for VMs foundation access could listen to without TLS encryption.



Each of these unencrypted communication paths may be TLS-encrypted. Enabling TLS encryption implements a one-way authentication of apps, verifying the identity of cluster members.

You must also secure gfsh communication. Follow directions in Connect with gfsh over HTTPS.

Networking

To allow app access to the PCC network, create application security groups. Allow access on the following ports:

- 1099
- 8080
- 40404
- 55221

For more information, see Restricting App Access to Internal TAS for VMs Components in the VMware Tanzu Application Service for VMs documentation.

GemFire for TAS works with the IPsec Add-on for TAS for VMs. For more information, see IPsec for VMware Tanzu.

Security within the Cluster

The cluster within a GemFire for TAS service instance implements role-based authentication and authorizes cluster operations based upon the roles.

There are two sets of roles:

- One set has four roles for users that integrate an external authentication such as LDAP via User Account and Authentication (UAA). See Configuring UAA Roles for a description of the roles and the configuration that completes the integration.
- The other set of roles defaults when there is no external authentication integrated during the GemFire for TAS tile installation. The identifiers assigned for these roles are detailed in Create Service Keys. GemFire for TAS service instances are created with three default user roles for interacting with clusters:
    - **Cluster operator:** manages the VMware GemFire cluster and can access region data. Has the permissions `CLUSTER:MANAGE`, `CLUSTER:WRITE`, `CLUSTER:READ`, `DATA:MANAGE`, `DATA:WRITE`, and `DATA:READ`.
    - **Developer:** can access region data. Has the permissions `CLUSTER:READ`, `DATA:WRITE`, and `DATA:READ`.
    - **Gateway sender:** propagates region data to another GemFire for TAS service instance. Has the permission `DATA:WRITE`.

Which set is used for a GemFire for TAS service instance depends on the options chosen during GemFire for TAS tile installation.

All gfsh and JMX clients must authenticate as one of these user roles to access the cluster. To authorize operations, each user role is given predefined permissions for cluster operations. To accomplish a cluster operation, the user authenticates using one of the roles. Prior to initiating the requested operation, there is a verification that the authenticated user role has permission to do the operation. Read more about these permissions in Implementing Authorization.

Operator Guide

This document describes how a VMware Tanzu Application Service for VMs operator can install, configure, and maintain VMware GemFire for Tanzu Application Service.

In this topic:

Preparing for Installation

This topic lists the steps required to prepare to install VMware GemFire for TAS. Work through these items prior to VMware GemFire for Tanzu Application Service tile installation.

- The Networking for On-Demand Services section describes networking requirements for GemFire for TAS.
- GemFire for TAS requires TLS encryption for using gfsh and Pulse. Follow the instructions in Preparing for TLS to set up TLS encryption.
- For systems that implement an external authentication such as LDAP via User Account and Authentication (UAA), follow the instructions within Create a UAA Client. There are also additional configuration steps to set up the SSO system, as given within Configuring UAA

Roles.

Networking for On-Demand Services

This topic describes networking considerations for VMware GemFire for TAS.

- Service Network Requirement
    - Default Network and Service Network
    - Required Networking Rules for On-Demand Services
- VMware GemFire for TAS Instances Across WAN

Service Network Requirement

When you deploy VMware Tanzu Application Service for VMs (TAS for VMs), you must create a statically defined network to host the component VMs that make up the infrastructure. Components, such as Cloud Controller and UAA, run on this infrastructure network.

On-demand services might require you to host them on a separate network from the default network. You can also deploy on-demand services on a separate service networks to meet your own security requirements.

TAS for VMs supports dynamic networking. Operators can use dynamic networking with asynchronous service provisioning to define dynamically-provisioned service networks. For more information, see Default Network and Service Network below.

On-demand services are enabled by default on all networks. Operators can optionally create separate networks to host services in BOSH Director. Operators can select which network hosts on-demand service instances when they configure the tile for that service.

Default Network and Service Network

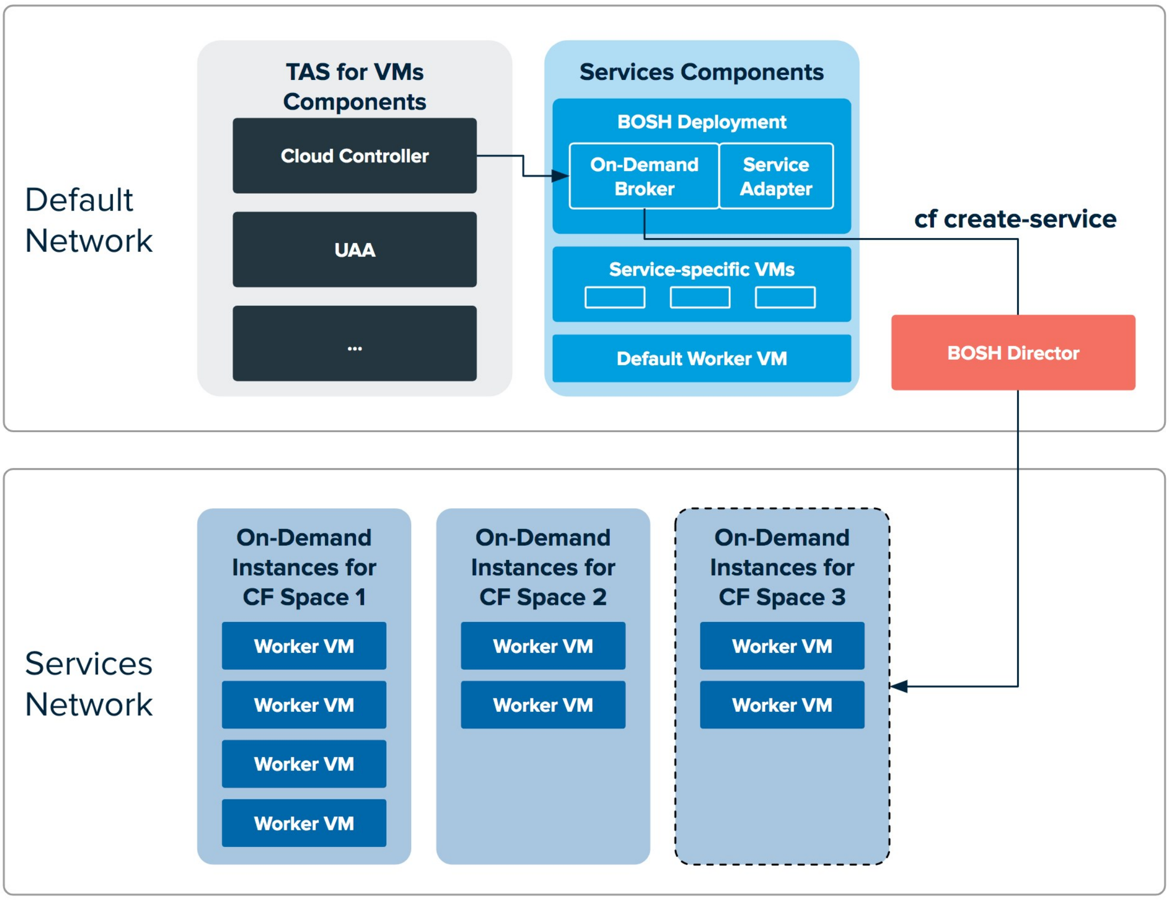
On-demand GemFire services use BOSH to dynamically deploy VMs and create single-tenant service instances in a dedicated network. On-demand services use the dynamically-provisioned service network to host single-tenant worker VMs. These worker VMs run as service instances within development spaces.

This on-demand architecture has the following advantages:

- Developers can provision IaaS resources for their services instances when the instances are created. This removes the need for operators to pre-provision a fixed amount of IaaS resources when they deploy the service broker.
- Service instances run on a dedicated VM and do not share VMs with unrelated processes. This removes the "noisy neighbor" problem, where an app monopolizes resources on a shared cluster.
- Single-tenant services can support regulatory compliances where sensitive data must be separated across different machines.

An on-demand service separates operations between the default network and the service network. Shared service components, such as executive controllers and databases, Cloud Controller, UAA, and other on-demand components, run on the default network. Worker pools deployed to specific spaces run on the service network.

The diagram below shows worker VMs in an on-demand service instance running on a separate services network, while other components run on the default network.



View a larger version of this image

Required Networking Rules for On-Demand Services

Before deploying a service tile that uses the on-demand service broker (ODB), you must create networking rules to enable components to communicate with ODB. For instructions for creating networking rules, see the documentation for your IaaS.

The following table lists key components and their responsibilities in the on-demand architecture.

Key Components	Component Responsibilities
BOSH Director	Creates and updates service instances as instructed by ODB.
BOSH Agent	Adds an agent on every VM that it deploys. The agent listens for instructions from the BOSH Director and executes those instructions. The agent receives job specifications from the BOSH Director and uses them to assign a role or job to the VM.
BOSH UAA	Issues OAuth2 tokens for clients to use when they act on behalf of BOSH users.
VMware Tanzu Application Service for VMs	Contains the apps that consume services.

ODB	Instructs BOSH to create and update services. Connects to services to create bindings.
Deployed service instance	Runs the given service. For example, a deployed GemFire service instance runs the GemFire service.

Regardless of the specific network layout, the operator must ensure network rules are set up so that connections are open as described in the table below.

This component...	Must communicate with...	Default TCP Port	Communication directions	Notes
GemFire for TAS cluster members	**GemFire for TAS cluster members**	49152-65535	Two-way	Inclusive range. GemFire for TAS servers and locators communicate with each other using UDP and TCP.
GemFire for TAS Service Instance 1	**GemFire for TAS Service Instance 2**	5000-5499	Two-way	Inclusive range. Gateway receivers and gateway senders communicate across WAN-separated service instances. Each GemFire for TAS service instance uses cluster defaults for the gateway receiver ports.
ODB	• **BOSH Director** • **BOSH UAA**	• 25555 (BOSH Director) • 8443 (UAA) • 8844 (CredHub)	One-way	The BOSH Director and BOSH UAA default ports are not configurable. The CredHub default port is configurable.
ODB	**VMware Tanzu Application Service for VMs**	8443	One-way	The default port is not configurable.
Errand VMs	• **VMware Tanzu Application Service for VMs** • **ODB**	• 443 • 8080	One-way	The default port is not configurable.
BOSH Agent	**BOSH Director**	4222	Two-way	The BOSH Agent runs on every VM in the system, including the BOSH Director VM. The BOSH Agent initiates the connection with the BOSH Director. The default port is not configurable.

Deployed apps on VMware Tanzu Application Service for VMs	**Deployed service instances**	- 55221 (VMware GemFire locators) - 40404 (VMware GemFire servers)	Two-way	These port numbers are not configurable.
Deployed apps on VMware Tanzu Application Service for VMs using GemFire for Redis Apps	**Deployed service instances**	6379	Two-way	There is no separate port for TLS. When TLS is enabled for the service instance this port will also be configured for TLS and all Redis communication (clients) must use TLS.
VMware Tanzu Application Service for VMs	- **ODB** - **Deployed service instances**	8080	One-way	PAS communicates with service instances because the Gorouter proxies gfsh requests to clusters.
GemFire for TAS	**GemFire for TAS**	1053	Two-way	Allows DNS resolution for clusters communicating across a WAN-connected system.

GemFire for TAS Instances Across WAN

GemFire for TAS service instances running within distinct VMware Tanzu Application Service for VMs foundations may communicate with each other across a WAN. In a topology such as this, the members within one service instance use their own private address space, as defined in RFC1918.

A VPN may be used to connect the private network spaces that lay across the WAN. The steps required to enable the connectivity by VPN are dependent on the IaaS providers.

The private address space for each service instance's network must be configured with non-overlapping CIDR blocks. Configure the network prior to creating service instances. Locate directions for creating a network on the appropriate IAAS provider in Platform Architecture and Planning Overview in the VMware Tanzu Ops Manager documentation.

Open port `1053` to allow DNS resolution of other WAN-connected clusters.

Preparing for Transport Layer Security (TLS)

This topic describes how to provide an existing Certificate Authority (CA) certificate to BOSH CredHub and how to generate a new CA certificate with BOSH CredHub.

> ⚠ **WARNING**: This procedure involves restarting all of the VMs in an existing VMware Tanzu Application Service for VMs (TAS for VMs) deployment in order to propagate a CA certificate. The operation can take a long time to complete.

Overview

Enabling TLS provisions VMware GemFire for Tanzu Application Service service instances with a certificate, so that apps, gfsh, and Pulse can establish encrypted connections with the GemFire for TAS service instance.

The certificate deployed on the GemFire for TAS service instance is a **server certificate**. The server certificate is generated by **CredHub**, a component designed for centralized credential management in TAS for VMs. CredHub is deployed on the same VM as the BOSH Director.

CredHub generates the server certificate using a **Certificate Authority (CA) certificate**. The CA certificate must be provided to CredHub by the operator or generated by CredHub.

Apps use the CA certificate to authenticate components of GemFire for TAS service instances. Apps that communicate with GemFire for TAS must have access to the CA certificate in order to validate that the server certificate can be trusted.

> ⚠ **WARNING**: An operator must rotate the CA certificate if it expires or if it becomes compromised. To rotate your CA certificate, see Managing Certificates. Do not attempt to rotate a CA certificate on your own. Contact Support and perform the procedure with their assistance.

Provide or Generate a CA Certificate

TLS authorization requires a credential generated by CredHub. You do not need to create a new User Account and Authentication (UAA) client for CredHub specifically to support TLS, as long as a UAA Client exists with `credhub.write` and `credhub.read` permissions. The client used in this section is one that was created during the Ops Manager installation process: the `ops_manager` client.

Perform the following steps to log in to CredHub and provide or generate a CA certificate:

1. From the Ops Manager VM, set the API target of the CredHub CLI to your CredHub server:

   ```
   credhub api https://BOSH-DIRECTOR:8844 --ca-cert=/var/tempest/workspaces/defaul
   t/root_ca_certificate
   ```

   Where `BOSH-DIRECTOR` is the IP address of the BOSH Director VM.

   For example:

   ```
   $ credhub api https://10.0.0.5:8844 --ca-cert=/var/tempest/workspaces/
   default/root_ca_certificate
   ```

2. Log in to CredHub:

   ```
   credhub login --client-name=CLIENT-NAME --client-secret=CLIENT-SECRET
   ```

   Where: * `CLIENT-NAME` is the client name, usually `ops_manager` or a CredHub-specific UAA client of your own creation. * `CLIENT-SECRET` is the client secret. Locate this under the Credentials tab of the Ops Manager tile with the name `Bosh Commandline Credentials`.

For example:

```
$ credhub login \
    --client-name=ops_manager \
    --client-secret=abcdefghijklm123456789
```

3. Use the CredHub CLI to check whether a services CA certificate already is present:

   ```
   credhub get --name="/services/tls_ca"
   ```

   If you already have a certificate at the `services/tls_ca` path, skip to step 5.

4. Use the CredHub CLI to generate a CA certificate or provide an existing one.

   > **Note**: Your deployment may have multiple CA certificates. VMware recommends that you use a dedicated CA certificate for services.

   - If you do not have a CA certificate, use the CredHub CLI to generate one:

     ```
     credhub generate \
     --name="/services/tls_ca" \
     --type="certificate" \
     --no-overwrite \
     --is-ca \
     --common-name="rootCA"
     ```

   - If you have an existing CA certificate that you want to use, create a new file named `root.pem` with the contents of the certificate. Then enter the following command, specifying the path to `root.pem` and the private key for the certificate:

     ```
     credhub set \
     --name="/services/tls_ca" \
     --type="certificate" \
     --certificate=./root.pem \
     --private=ERKSOSMFF...
     ```

5. Use the BOSH CLI v2 to extract the `certificate` portion from the CA certificate and print it:

   ```
   bosh interpolate <(credhub get --name=/services/tls_ca) \
   --path=/value/certificate
   ```

6. Record the output of the `bosh interpolate` command.

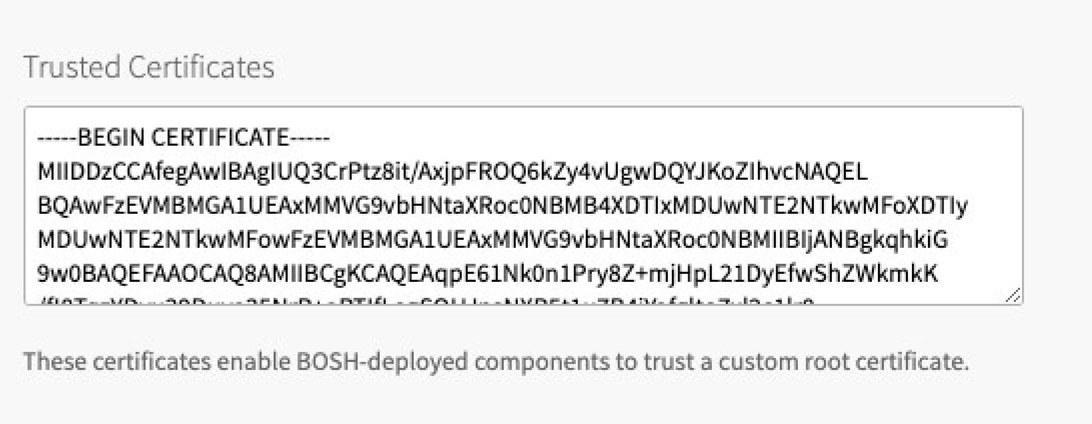
Add the CA Certificate to Ops Manager

Add the CA certificate to Ops Manager by adding it to the BOSH Director tile and the Tanzu Application Service tile.

Add the CA certificate to the BOSH Director tile:

1. Navigate to the Ops Manager **Installation Dashboard** and select the **Bosh Director** tile. Click **Security**.

2. Copy and paste the contents of the CA certificate into **Trusted Certificates**. If Trusted Certificates are already listed, append the new certificate to the existing content.

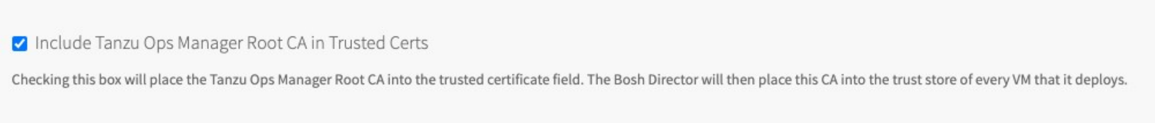


Trusted Certificates

```
-----BEGIN CERTIFICATE-----
MIIDDzCCAfegAwIBAgIUQ3CrPtz8it/AxjpFROQ6kZy4vUgwDQYJKoZIhvcNAQEL
BQAwFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMB4XDTIxMDUwNTE2NTkwMFoXDTIy
MDUwNTE2NTkwMFowFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqpE61Nk0n1Pry8Z+mjHpL21DyEfwShZWkmkK
```

These certificates enable BOSH-deployed components to trust a custom root certificate.

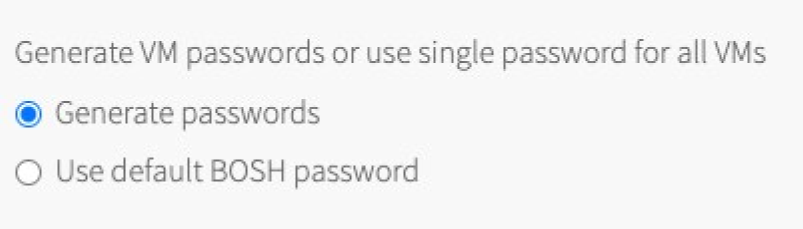
3. Check **Include Tanzu Ops Manager Root CA in Trusted Certs** to place the Tanzu Ops Manager Root CA into the trusted certificate field. The BOSH Director then places this CA into the truststore of every VM that it deploys.



☑ Include Tanzu Ops Manager Root CA in Trusted Certs

Checking this box will place the Tanzu Ops Manager Root CA into the trusted certificate field. The Bosh Director will then place this CA into the trust store of every VM that it deploys.

4. For **Generate VM passwords or use single password for all VMs**, select **Generate passwords** or **Use default BOSH password**, depending on your implementation needs.



Generate VM passwords or use single password for all VMs

● Generate passwords

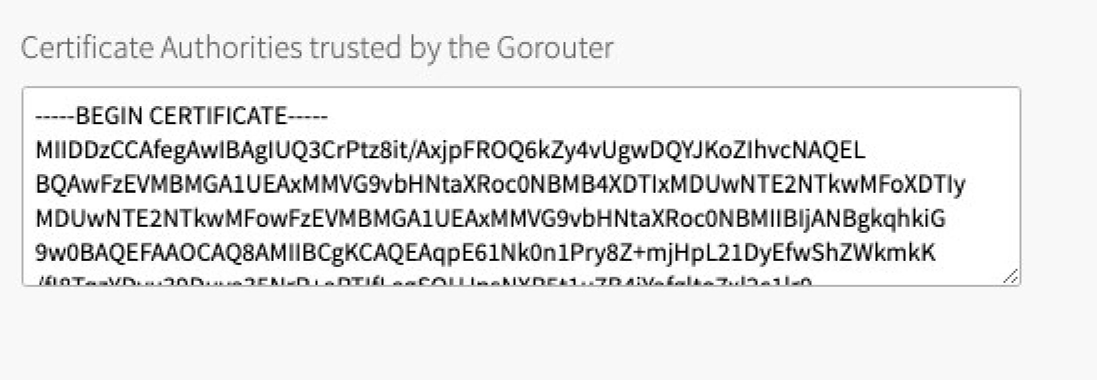
○ Use default BOSH password

5. Click **Save**.

Add the CA certificate to the Tanzu Application Service tile

1. The CA certificate must also be added for the Gorouter. Navigate to the TAS tile's **Settings** tab.

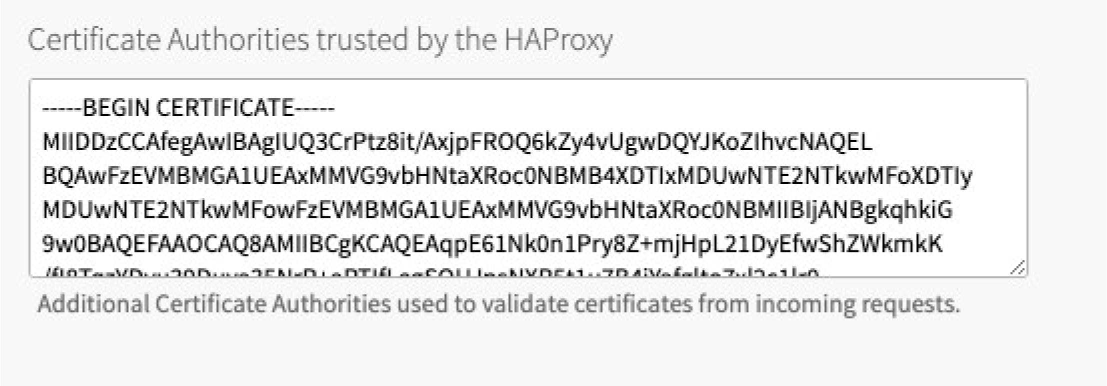
2. Click on **Networking**. Add the CA certificate to the box labeled **Certificate Authorities Trusted by Gorouter**.



Certificate Authorities trusted by the Gorouter

```
-----BEGIN CERTIFICATE-----
MIIDDzCCAfegAwIBAgIUQ3CrPtz8it/AxjpFROQ6kZy4vUgwDQYJKoZIhvcNAQEL
BQAwFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMB4XDTIxMDUwNTE2NTkwMFoXDTIy
MDUwNTE2NTkwMFowFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqpE61Nk0n1Pry8Z+mjHpL21DyEfwShZWkmkK
```

3. Scroll down and add the CA certificate to the box labeled **Certificate Authorities trusted by the HAProxy**.



Certificate Authorities trusted by the HAProxy

```
-----BEGIN CERTIFICATE-----
MIIDDzCCAfegAwIBAgIUQ3CrPtz8it/AxjpFROQ6kZy4vUgwDQYJKoZIhvcNAQEL
BQAwFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMB4XDTIxMDUwNTE2NTkwMFoXDTIy
MDUwNTE2NTkwMFowFzEVMBMGA1UEAxMMVG9vbHNtaXRoc0NBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqpE61Nk0n1Pry8Z+mjHpL21DyEfwShZWkmkK
```

Additional Certificate Authorities used to validate certificates from incoming requests.

4. Click **Save**.

Save Your Changes

1. Return to the Installation Dashboard.
2. Click **Review Pending Changes**. For more information, see Reviewing Pending Product Changes in the VMware Tanzu Ops Manager documentation.
3. Click **Apply Changes**.

Create a User Account and Authentication (UAA) Client

This topic explains how to create a User Account and Authentication (UAA) account for use with VMware GemFire for TAS.

Extra configuration is required when defining users that have specific security roles with an external authentication such as LDAP through User Account and Authentication (UAA).

Authenticating using an external authentication such as LDAP through UAA uses the UAA server. Access the UAA server through its command-line interface, UAAC.

When enabling the use of an external authentication such as LDAP through UAA during VMware GemFire for Tanzu Application Service tile configuration, you will specify a UAA client, as described in Security. This procedure creates that UAA client.

Create a UAA client for use in tile configuration with the command:

```
uaac client add cloudcache_gfsh --scope="PCC_*" --secret="THE-SECRET"
   --authorized_grant_types=password
```

Replace `THE-SECRET` with your chosen password (secret) for this UAA client. The remainder of the command is exactly as shown. The UAA client name is `cloudcache_gfsh`.

Installing and Configuring VMware GemFire for TAS

This topic explains how to install and configure VMware GemFire for TAS.

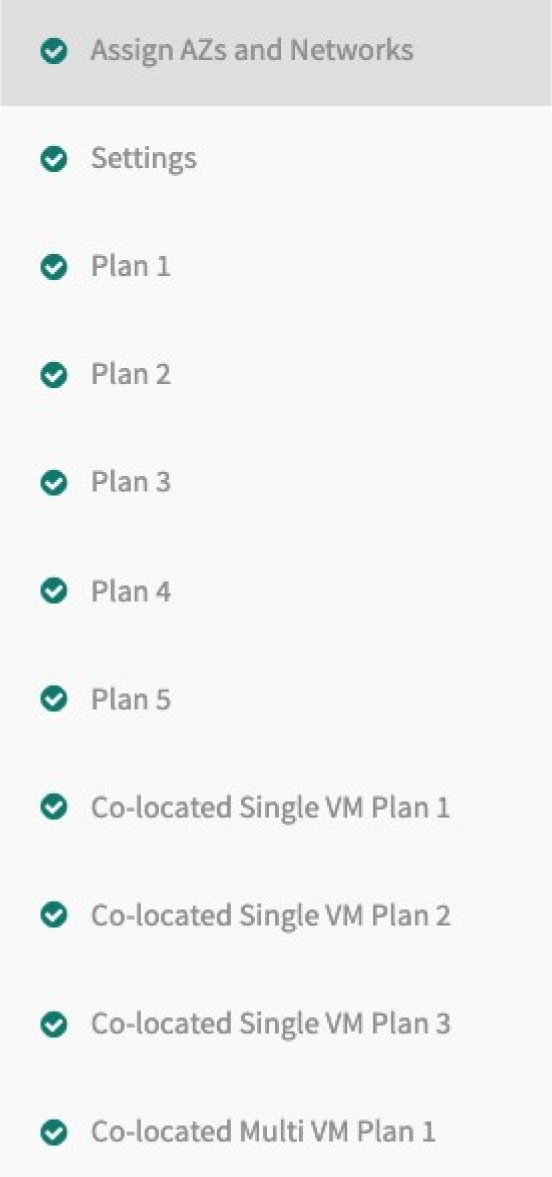
With an Ops Manager role (detailed in Configuring Role-Based Access Control (RBAC) in Ops Manager in the VMware Tanzu Ops Manager documentation) that has the proper permissions to install and configure, follow these steps to install GemFire for TAS on VMware Tanzu Application Service for VMs (TAS for VMs):

Procedure

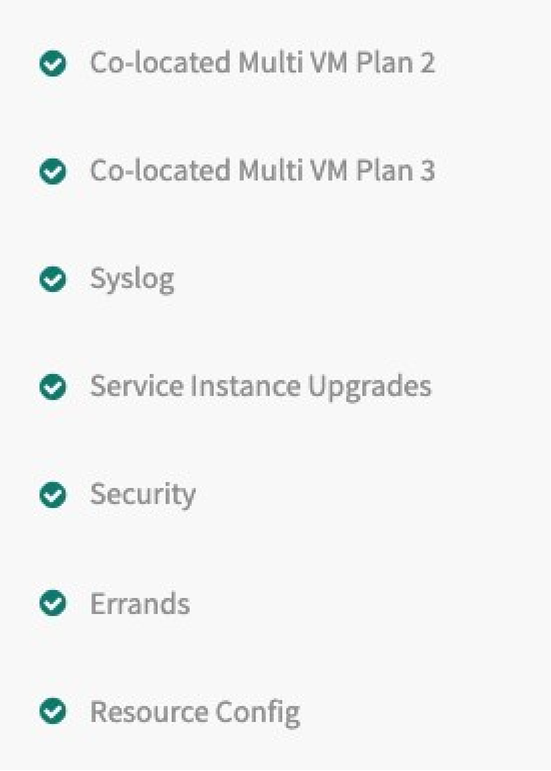
1. Download the tile from Tanzu Network.
2. Click **Import a Product** to import the tile into Ops Manager.
3. Click the **+** symbol next to the uploaded product description.
4. Click on the GemFire for TAS tile.
5. Complete all the configuration steps in the Configure Tile Properties section below.
6. Return to the Ops Manager Installation Dashboard. Click **Review Pending Changes**. For more information, see Reviewing Pending Product Changes in the VMware Tanzu Ops Manager documentation.
7. Click **Apply Changes** to complete the installation of the GemFire for TAS tile.

Configure Tile Properties

Configure the sections listed on the left side of the page.



Assign AZs and Networks
Settings
Plan 1
Plan 2
Plan 3
Plan 4
Plan 5
Co-located Single VM Plan 1
Co-located Single VM Plan 2
Co-located Single VM Plan 3
Co-located Multi VM Plan 1



✓ Co-located Multi VM Plan 2

✓ Co-located Multi VM Plan 3

✓ Syslog

✓ Service Instance Upgrades

✓ Security

✓ Errands

✓ Resource Config

As you complete a section, save it. A green check mark appears next to the section name. Each section name must show this green check mark before you can complete your installation.

- Assign AZs and Networks
- Smoke Test Plan Settings
- Allow Outbound Internet Access Settings
- Default Distributed System ID Settings
- Sharable Instances Settings
- Enable Creation of a Service Gateway
- Configure Service Plans
- Configure Platform Tags
- Configure a Co-located Multi VM Plan
- Configure a Co-located Single VM Plan
- Syslog
- Service Instance Upgrades
- Security
- Errands

Assign Availability Zones and Networks

To select AZs and networks for VMs used by GemFire for TAS, do the following:

1. Click **Assign AZs and Networks**.
2. Configure the fields on the **Assign AZs and Networks** pane as follows:

Field	Instructions
Place singleton jobs in	Select the region that you want for singleton VMs.
Balance other jobs in	Select the AZs that you want to use for distributing other cluster VMs. VMware recommends that you select all of them.
Network	Select your Tanzu Application Service (TAS) network.
Service Network	Select the network to be used for cluster VMs.

3. Click **Save**.

Smoke Test Plan Settings

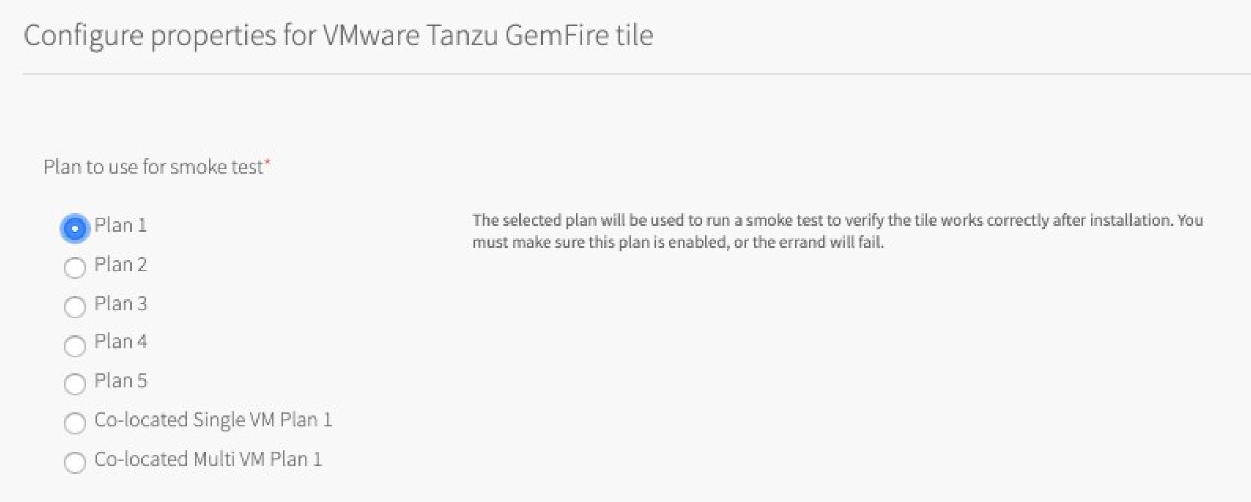
The smoke-tests errand that runs after tile installation. The errand verifies that your installation was successful. By default, the `smoke-test` errand runs on the `system` org and the `p-cloudcache-smoke-test` space.

> **Note**: Smoke tests will fail unless you enable global default application security groups (ASGs). You can enable global default ASGs by binding the ASG to the `system` org without specifying a space. To enable global default ASGs, use `cf bind-running-security-group`.

To select which plan to use for smoke tests, do the following:

1. Select a plan to use when the `smoke-tests` errand runs.
2. Ensure that the selected plan is enabled and configured. For information about configuring plans, see Configure Service Plans below. If the selected plan is not enabled, the `smoke-tests` errand fails.

VMware recommends that you use the smallest four-server plan for smoke tests. Because smoke tests create and later destroy this plan, using a very small plan reduces installation time.



Configure properties for VMware Tanzu GemFire tile

Plan to use for smoke test*

● Plan 1
○ Plan 2
○ Plan 3
○ Plan 4
○ Plan 5
○ Co-located Single VM Plan 1
○ Co-located Multi VM Plan 1

The selected plan will be used to run a smoke test to verify the tile works correctly after installation. You must make sure this plan is enabled, or the errand will fail.

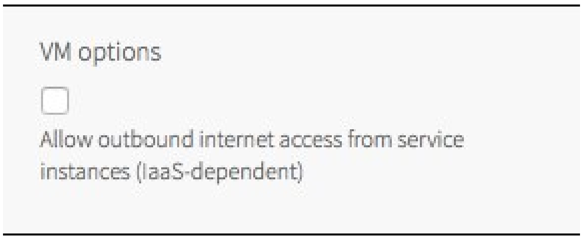
Allow Outbound Internet Access Settings

By default, outbound internet access is not allowed from service instances.

If BOSH is configured to use an external blob store, you must allow outbound internet access from service instances. Log forwarding and backups, which require external endpoints, might also require internet access.

To allow outbound internet access from service instance, do the following:

1. Select **Allow outbound internet access from service instances (IaaS-dependent)**.



VM options

□

Allow outbound internet access from service instances (IaaS-dependent)

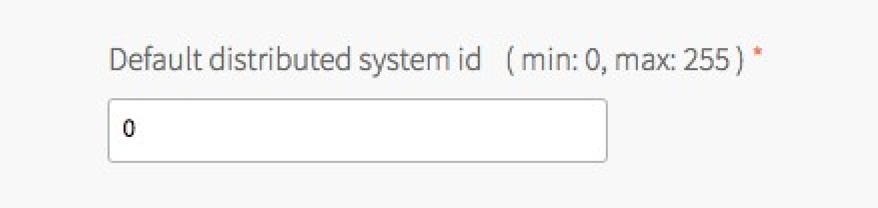
Note: Outbound network traffic rules also depend on your IaaS settings. Consult your network or IaaS administrator to ensure that your IaaS allows outbound traffic to the external networks that your require.

Default Distributed System ID Settings

Every service instance has an integer identifier called a **distributed system ID**. The ID defaults to the value `0`. Service instances that form a distributed system that communicate across a WAN must have distinct IDs. Those distinct ID values are set when creating the service instance.

To change the default distributed system ID value, do the following:

1. Replace the default value of `0` with your new default value. Acceptable values are integers greater than or equal to `0` and less than or equal to `255`.



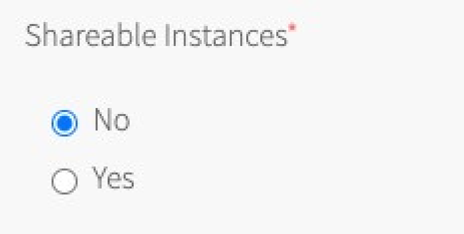
Default distributed system id (min: 0, max: 255) *

0

Sharable Instances Settings

To enable service-instance sharing for a GemFire for TAS service instance, do the following:

1. Set **Shareable Instances** to `Yes`.



Shareable Instances*

● No

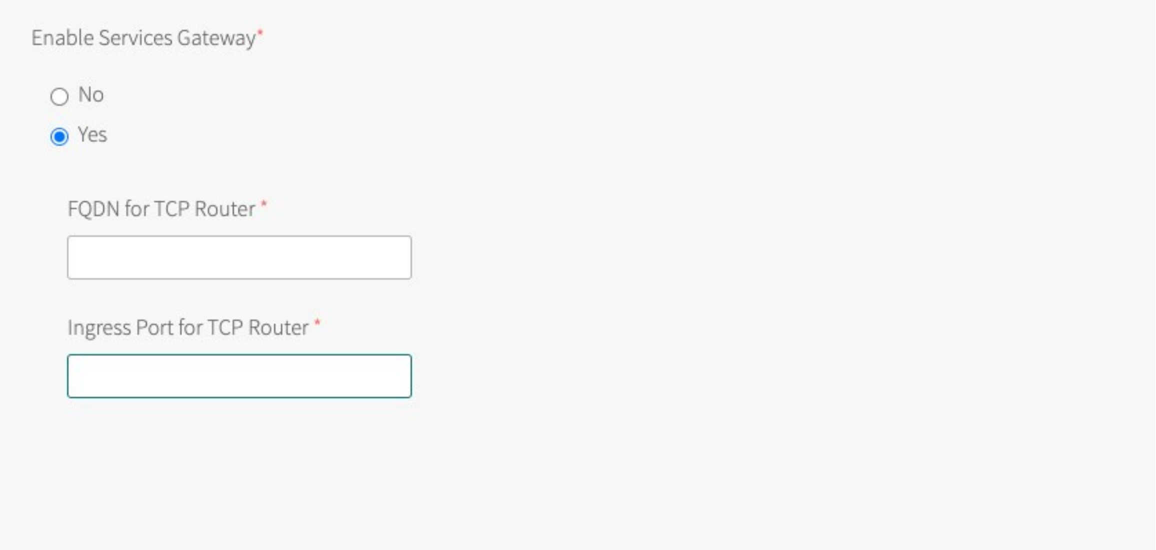
○ Yes

For more information about service-instance sharing, see Service-Instance Sharing

Enable Creation of a Service Gateway Setting

The service gateway permits apps running outside the services foundation (TAS) to communicate with the GemFire for TAS service instance. See The App's Location in *VMware GemFire Architecture* for an introduction to the service gateway.

1. If the communication between apps and the GemFire for TAS service instance require a service gateway, select the **Yes** radio button in the **Enable Service Gateway** setting. This permits the instantiation of the service gateway when a service instance is created and specifies a service gateway.
2. Enter a Fully Qualified Domain Name (FQDN) for the **FQDN for TCP Router** field. This was the FQDN that you used when you enabled TCP routing. You can also find the in the output of the `cf domains` command. From the output, select the `name` field with type `tcp`.
3. Set the **Ingress Port for TCP Router** field with an unused port that you noted when you enabled TCP routing. Alternatively, to find the range of ports from which to choose, in **Ops Manager** click **Tanzu Application Service**, then select **Networking**. The range of ports from which to choose is near the bottom of the page in a field named **TCP routing ports** under the heading **TCP routing**.



Enable Services Gateway*

○ No

● Yes

FQDN for TCP Router *

Ingress Port for TCP Router *

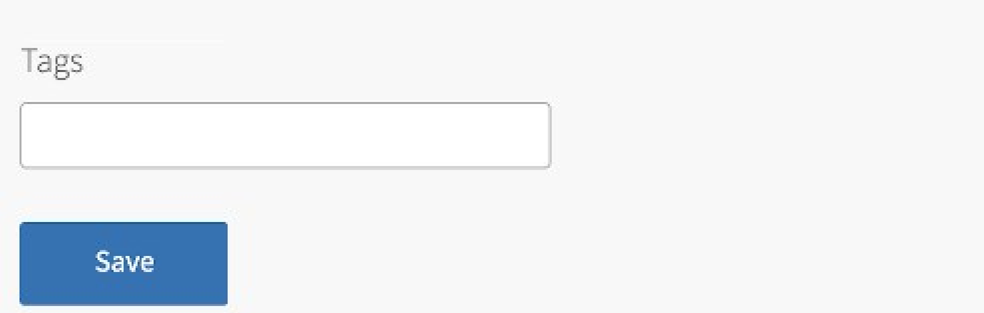
4. Click **Save**.

Configure Platform Tags

Platform tags are key-value pairs that are assigned to all service instance VM instances at the IaaS level. You can use these tags to organize and categorize your service instances. The accepted format for these key-value pairs depends on the underlying Cloud Provider. For example, Google Cloud Platform does not allow uppercase characters.

To use platform tags:

1. In the **Tags** text box, enter a comma-separated list of key-value pairs.



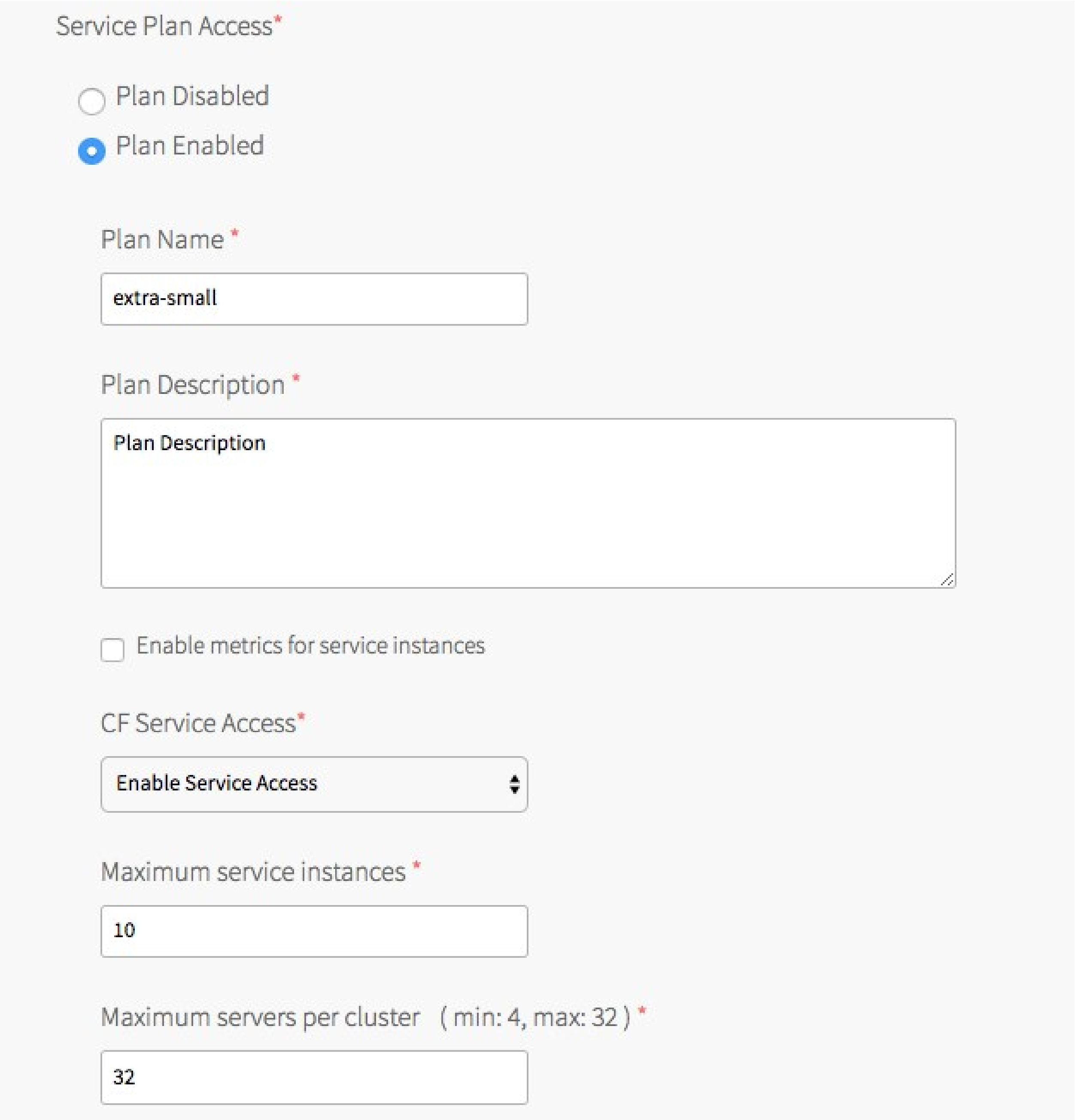
Tags

Save

2. Click **Save**.

Configure Service Plans

You will configure 11 individual plans by selecting each of the tabs. There are **Plan 1** through **Plan 5**, three plans that reside on a single VM (called **Co-located Single VM Plan**), and three plans that spread locators and servers among three VMs (called **Co-located Multi VM Plan**).



Service Plan Access*

○ Plan Disabled

● Plan Enabled

Plan Name *

extra-small

Plan Description *

Plan Description

□ Enable metrics for service instances

CF Service Access*

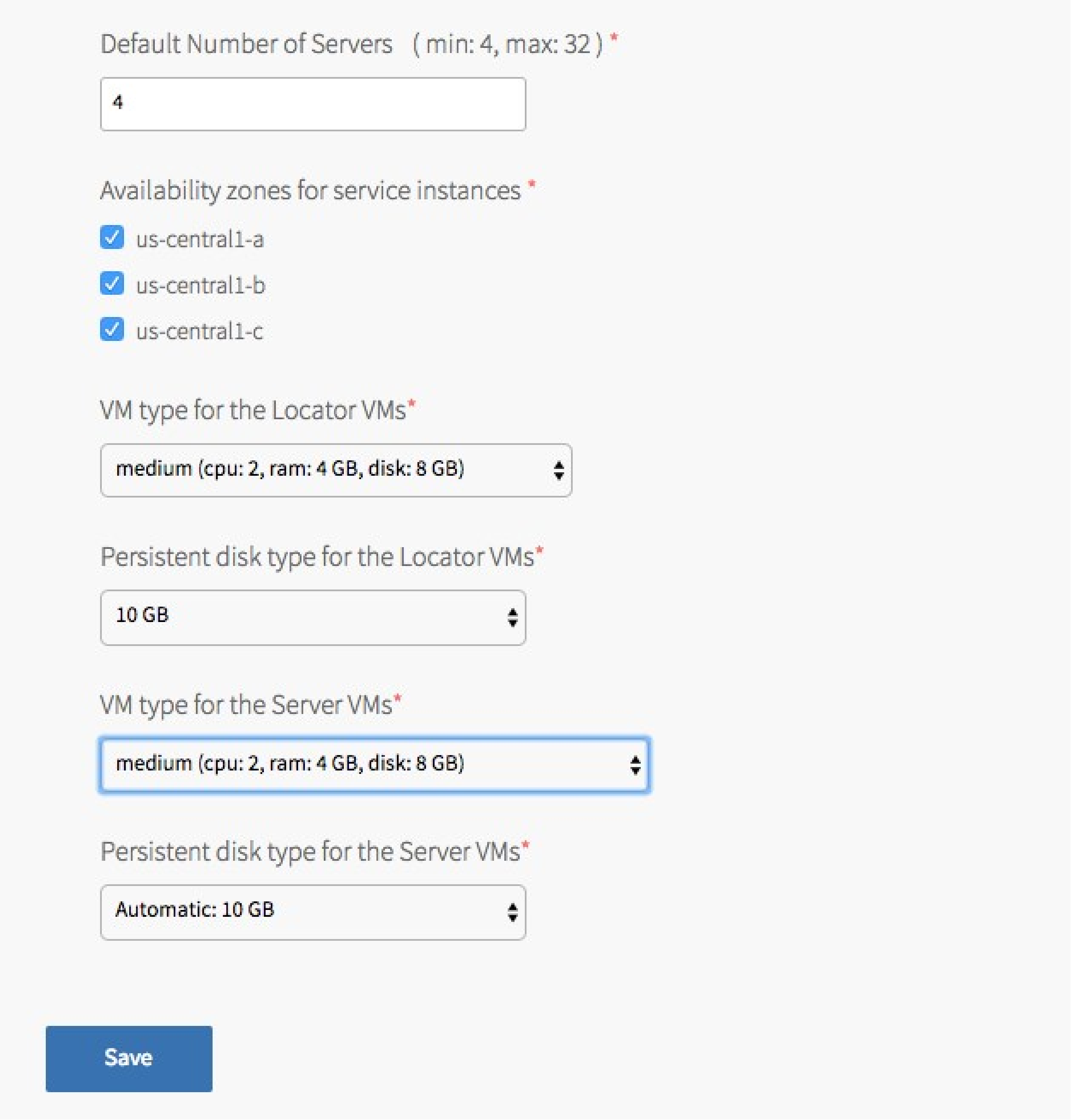
Enable Service Access

Maximum service instances *

10

Maximum servers per cluster (min: 4, max: 32) *

32



Default Number of Servers (min: 4, max: 32) *

4

Availability zones for service instances *

☑ us-central1-a

☑ us-central1-b

☑ us-central1-c

VM type for the Locator VMs*

medium (cpu: 2, ram: 4 GB, disk: 8 GB)

Persistent disk type for the Locator VMs*

10 GB

VM type for the Server VMs*

medium (cpu: 2, ram: 4 GB, disk: 8 GB)

Persistent disk type for the Server VMs*

Automatic: 10 GB

- The **Plan Enabled** option is selected by default. If you do not want to add this plan to the CF service catalog, select **Plan Disabled**. You must enable at least one plan.
- The **Plan Name** text field allows you to customize the name of the plan. This plan name is displayed to developers when they view the service in the Marketplace.
- The **Plan Description** text field allows you to supply a plan description. The description is displayed to developers when they view the service in the Marketplace.
- The **Enable metrics for service instances** checkbox enables metrics for service instances created using the plan. Once enabled, the metrics are sent to the Loggregator Firehose.
- The **CF Service Access** drop-down menu gives you the option to display or hide the service plan in the Marketplace.
    - **Enable Service Access** displays the service plan the Marketplace.
    - **Disable Service Access** makes the plan unavailable in the Marketplace. If you choose

this option, you cannot make the plan available at a later time.

  - **Leave Service Access Unchanged** makes the plan unavailable in the Marketplace by default, but allows you to make it available at a later time.

- The **Maximum service instances** field sets the maximum number of GemFire for TAS clusters that can exist simultaneously.
- When developers create or update a service instance, they can specify the number of servers in the cluster. The **Maximum servers per cluster** field allows operators to set an upper bound on the number of servers developers can request. If developers do not explicitly specify the number of servers in a service instance, a new cluster has the number of servers specified in the **Default Number of Servers** field.
- The **Availability zones for service instances** setting determines which AZs are used for a particular cluster. The members of a cluster are distributed evenly across AZs.

⚠ **Warning:** After you select AZs for your service network, you cannot add additional AZs. Adding additional AZs causes existing service instances to lose data on update.

The remaining fields control the VM type and persistent disk type for servers and locators. The total size of the cache is directly related to the number of servers and the amount of memory of the selected server VM type. VMware recommends the following configuration:

- For the **VM type for the Locator VMs** field, select a VM that has at least 2 CPUs, 1 GB of RAM and 4 GB of disk space.
- For the **Persistent disk type for the Locator VMs** field, select 10 GB or higher.
- For the **VM type for the Server VMs** field, select a VM that has at least 2 CPUs, 4 GB of RAM and 8 GB of disk space.
- For the **Persistent disk type for the server VMs** field, select 10 GB or higher.

For more information about VM memory allocation, see VM Memory Allocation.

When you finish configuring the plan, click **Save** to save your configuration options.

Configure a Co-located Multi VM Plan

Any of the Co-located Multi VM Plans use three VMs to implement a default plan with three locators and three servers. Each of the three VMs has one locator and one server. **Co-located Multi VM Plan 1** has a default plan name of `small-footprint`.

This type of plan is appropriate for production installations that can tolerate the diminished load-handling capacity and reduced resilience that would result from the loss of a VM. Installations needing resilience in the face of a VM loss should use a plan in which each GemFire for TAS cluster member resides within its own VM.

You can create a GemFire for TAS service instance using a Co-located Multi VM Plan with more servers than the default three servers. Follow the instructions in Create or Delete a Service Instance to increase the number of servers.

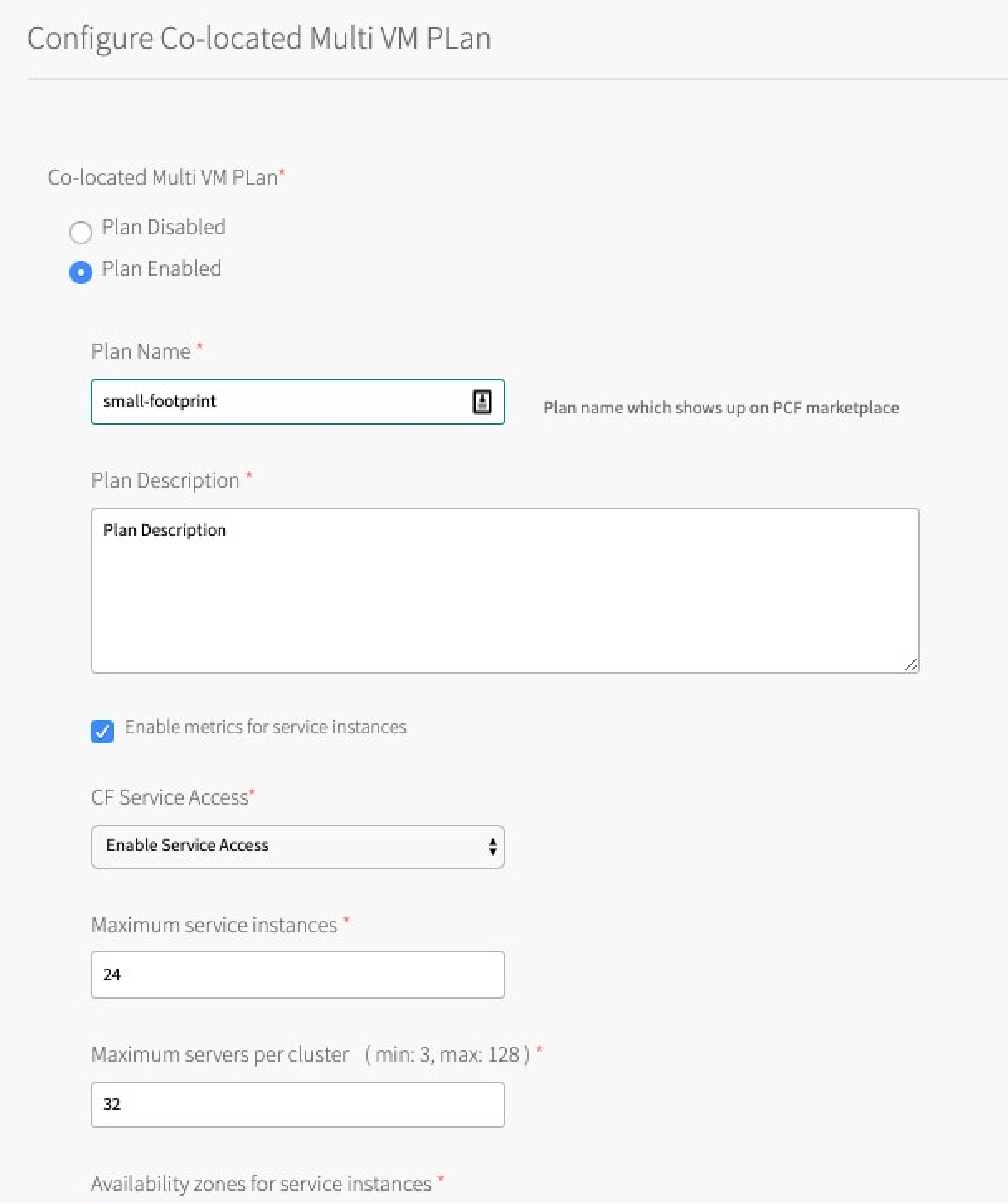
Scale up an existing service instance by increasing the number of servers, as specified in Updating a VMware GemFire for Tanzu Application Service Service Instance.

Whether creating a new service instance or updating an existing service instance to have more servers, each additional server will be within its own VM.

Scaling down the quantity of servers within an existing service instance is accomplished one server at a time to ensure redundancy is correctly maintained. To scale down, use the `cf update-service` command as described in Updating a VMware GemFire for Tanzu Application Service Service Instance to specify one server fewer than the total quantity of servers currently running.

An error message will be issued if you attempt to scale down by more than one server at a time, or if you attempt to reduce the number of servers below the minimum size of the default Co-located Multi VM Plan size of three servers.

To configure a Co-located Multi VM Plan, enable the plan with the Co-located Multi VM Plan tab.



Configure Co-located Multi VM PLan

Co-located Multi VM PLan*

○ Plan Disabled

● Plan Enabled

Plan Name *

small-footprint

Plan name which shows up on PCF marketplace

Plan Description *

Plan Description

☑ Enable metrics for service instances

CF Service Access*

Enable Service Access

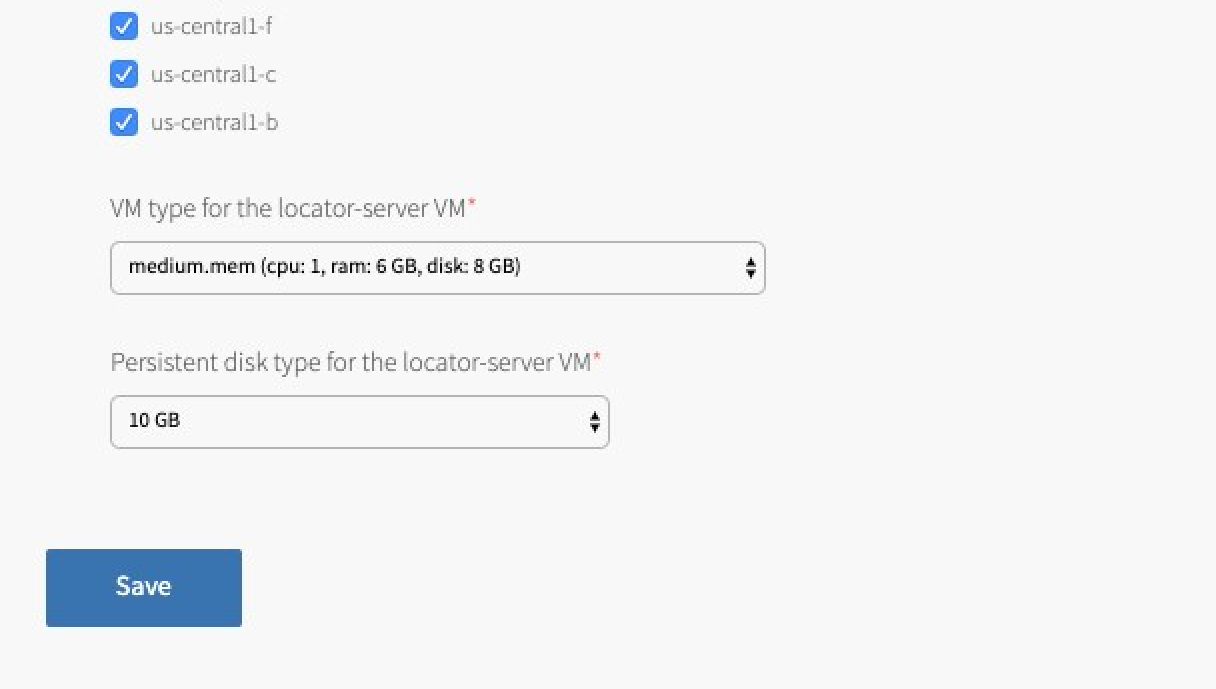
Maximum service instances *

24

Maximum servers per cluster (min: 3, max: 128) *

32

Availability zones for service instances *



☑ us-central1-f

☑ us-central1-c

☑ us-central1-b

VM type for the locator-server VM*

medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)

Persistent disk type for the locator-server VM*

10 GB

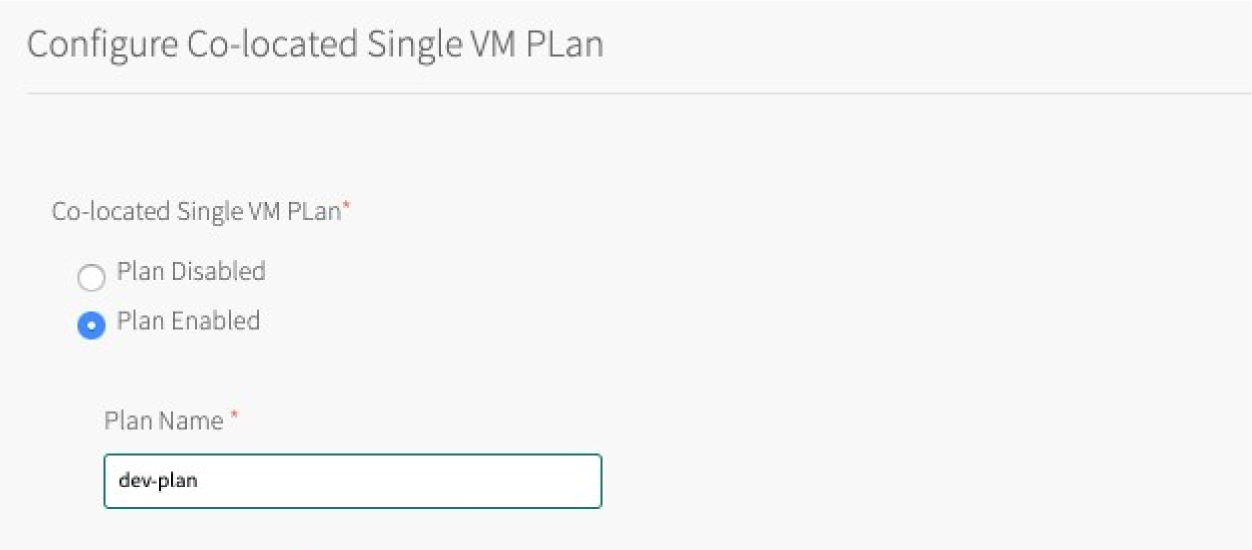
Configure a Co-located Single VM Plan

A Co-located Single VM Plan is a type of service plan. The plan provides a single locator and server, which are colocated within a single VM.

> ⚠ **Warning:** Data loss results when the single VM that this plan provides powers down or restarts.

The plan automatically creates a single region named `example_partition_region`. The region type is `PARTITION`, a partitioned region without redundancy or persistence of entries. You can create other regions as necessary with a Co-located Single VM Plan service instance.

There are three Co-located Single VM Plans to configure. The page for configuring this plan is similar to the page for configuring other service plans. To configure a Co-located Single VM Plan, input information in the fields and make selections from the options on each **Configure Co-located Single VM Plan** page.



Configure Co-located Single VM PLan

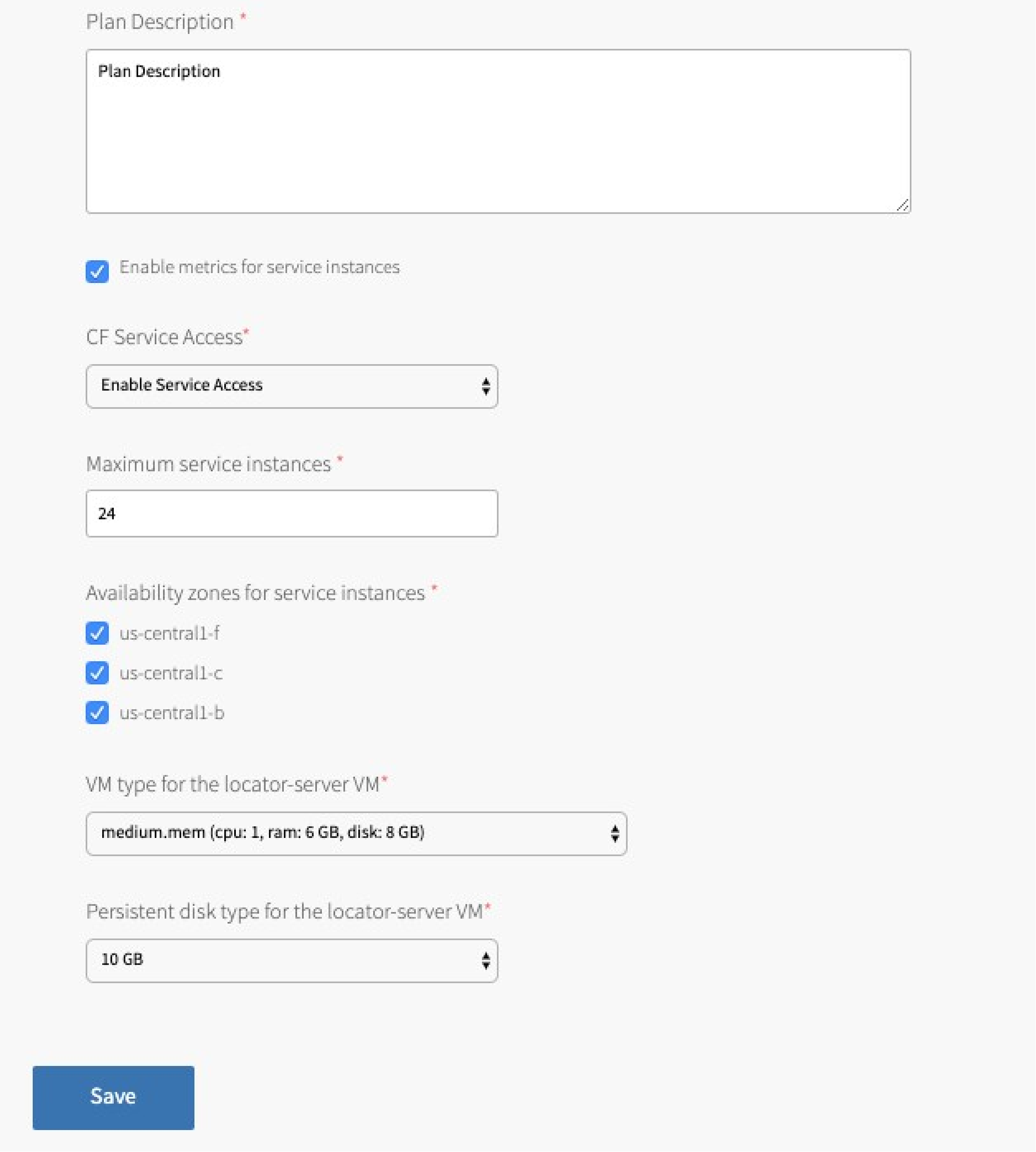
Co-located Single VM PLan*

○ Plan Disabled

● Plan Enabled

Plan Name *

dev-plan



Plan Description *

Plan Description

☑ Enable metrics for service instances

CF Service Access*

Enable Service Access

Maximum service instances *

24

Availability zones for service instances *

☑ us-central1-f

☑ us-central1-c

☑ us-central1-b

VM type for the locator-server VM*

medium.mem (cpu: 1, ram: 6 GB, disk: 8 GB)

Persistent disk type for the locator-server VM*

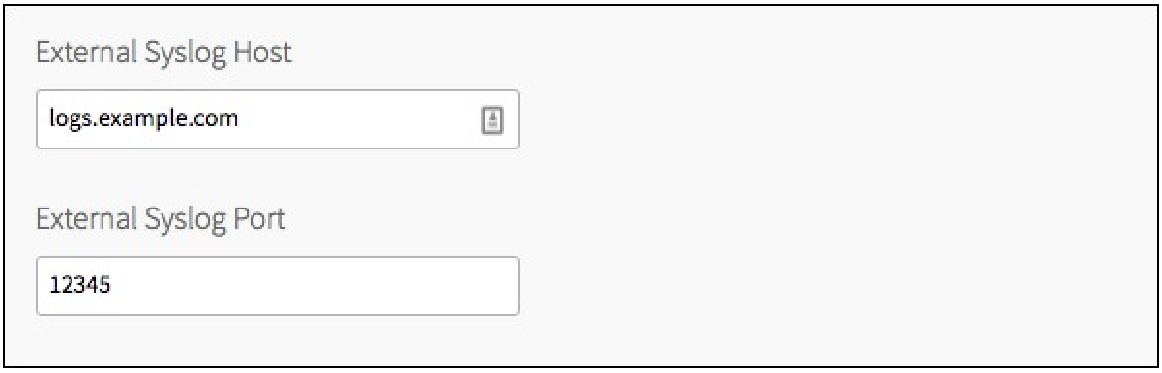
10 GB

Syslog

By default, syslog forwarding is not enabled in GemFire for TAS. However, GemFire for TAS supports forwarding syslog to an external log management service, for example, to Papertrail, Splunk, or your custom enterprise log sink. Use the broker logs for debugging problems creating, updating, and binding service instances.

To enable remote syslog for the service broker, do the following:

1. Click **Syslog**.



External Syslog Host

logs.example.com

External Syslog Port

12345

2. Configure the fields on the **Syslog** pane as follows:

Field	Instructions
Do you want to configure Syslog for VMware GemFire?	Select `Yes` to enable.
Address	Enter the address or host of the syslog server for sending logs, for example, `logs.example.com`.
Port	Enter the port of the syslog server for sending logs, for example, `29279`.
Transport Protocol	Select `TCP`, `UDP`, or `RELP`.
Send service instance logs to external	By default, only the broker logs are forwarded to your configured log management service. To forward server and locator logs from all service instances, select this option. This lets you monitor the health of the clusters, but generates a large volume of logs files. If you do not enable this option, only the broker logs which include information about service instance creation are forwarded, without logs about on-going cluster health.
Enable TLS	Select to enable secure log transmission through TLS. Without this, remote syslog sends unencrypted logs. VMware recommends that you enable TLS, as most syslog endpoints such as Papertrail and Logsearch require TLS.
Permitted Peer	This option is required if TLS is enabled. If several peer servers can respond to remote syslog connections, provide a regex, such as `*.example.com`.
SSL Certificate	Specifies the CA certificate for TLS communication. If the server certificate is not signed by a known authority, for example, an internal syslog server, provide the CA certificate of the log management service endpoint.

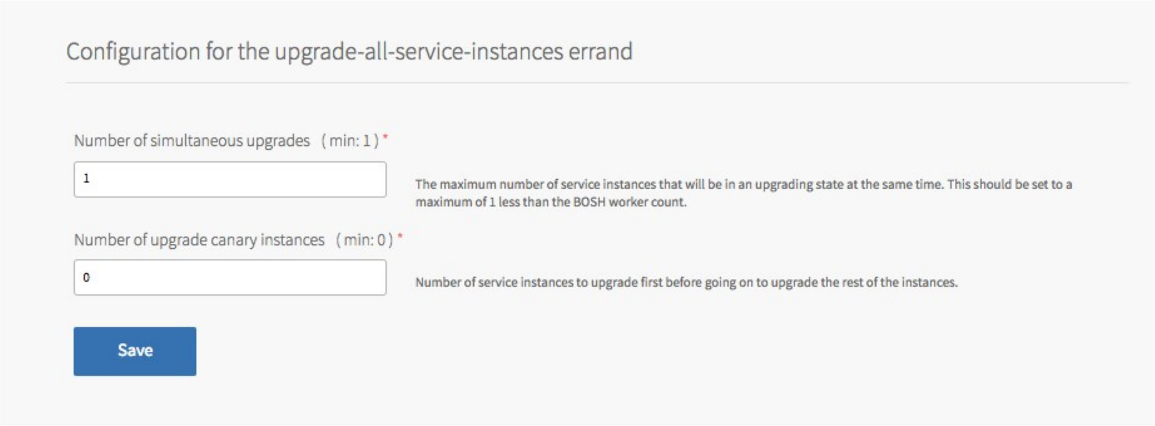
3. Click **Save**.

Service Instance Upgrades

You can upgrade a configurable number of service instances concurrently by entering a new value that is greater than `1` and less than the BOSH worker count for the **Number of simultaneous upgrades**.

1. Specify a set of service instances to act as canaries for the upgrade process by changing the **Number of upgrade canary instances** to a value greater than `0`. If all canary instances successfully upgrade, the remaining instances are upgraded. If any canary instance fails to

upgrade, the upgrade fails and no further instances are upgraded.



Configuration for the upgrade-all-service-instances errand

Number of simultaneous upgrades (min: 1) *

1

The maximum number of service instances that will be in an upgrading state at the same time. This should be set to a maximum of 1 less than the BOSH worker count.

Number of upgrade canary instances (min: 0) *

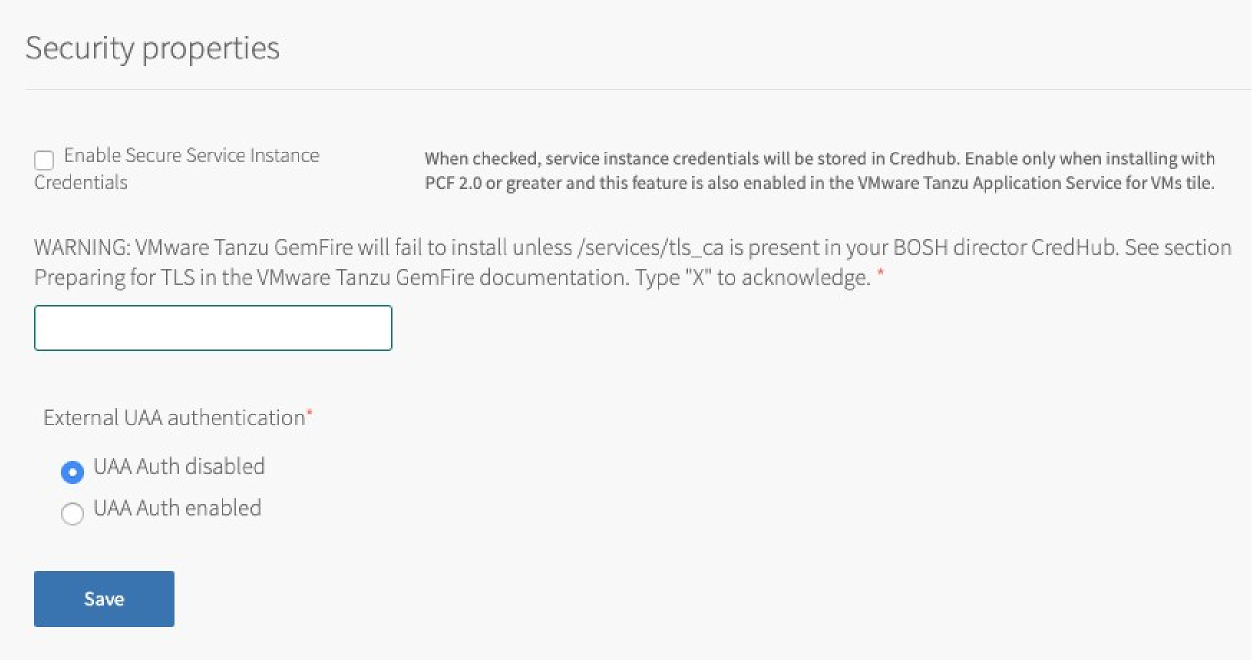
0

Number of service instances to upgrade first before going on to upgrade the rest of the instances.

2. Click **Save**.

Security

There are three configuration aspects to the Security settings. Each of the three must be considered and appropriately handled within these settings. Once the Security properties settings have been set, click **Save**. The Security properties Settings page appears as:



Security properties

□ Enable Secure Service Instance Credentials

When checked, service instance credentials will be stored in Credhub. Enable only when installing with PCF 2.0 or greater and this feature is also enabled in the VMware Tanzu Application Service for VMs tile.

WARNING: VMware Tanzu GemFire will fail to install unless /services/tls_ca is present in your BOSH director CredHub. See section Preparing for TLS in the VMware Tanzu GemFire documentation. Type "X" to acknowledge. *

External UAA authentication*

● UAA Auth disabled

○ UAA Auth enabled

The three items in this page to handle are:

1. The environment may be configured to more securely store service keys within CredHub, instead of within the cloud controller's data store. To enable this functionality, click on the box labeled **Enable Secure Service Instance Credentials** to enable use of CredHub. When service keys are stored within CredHub, output from the `cf env APP-NAME` command will not show credentials.
2. An `X` is required in the text box to promote the understanding that the environment must be set up to handle TLS encryption. For information about preparing the TAS for VMs environment, see Preparing for Transport Layer Security (TLS).
3. If User Account and Authentication (UAA) of GemFire for TAS users will use an external

system, enable this with the radio button labeled **UAA Auth enabled**. With UAA enabled, create a UAA client before doing this tile installation as explained in Create a User Account and Authentication (UAA) Client. Follow the instructions in Configuring User Account and Authentication (UAA) Roles to complete the configuration by setting up the user roles. With UAA enabled, two text boxes will appear. Fill the boxes with the UAA client name and that client's secret, which were set when the client was created.

Errands

By default, post-deploy and pre-delete errands always run. VMware recommends keeping these defaults. However, if necessary, you can change these defaults as follows.

For general information about errands in TAS for VMs, see Managing Errands in Ops Manager in the VMware Tanzu Ops Manager documentation.

1. Click **Errands**.
2. Change the setting for the errands.
3. Click **Save**.

VM Memory Allocation

This topic describes virtual machine memory allocation in VMware GemFire for TAS.

The service plans you choose determines how memory is allocated to the cluster processes. One consideration is how much memory is available in each VM.

Memory allocation depends on whether a plan is *colocated* or *non-colocated*:

- A *colocated plan* is one in which each VM hosts one locator and one server. For example, the default `small-footprint` plan comprises three locators and three servers hosted on three VMs.
- A *non-colocated plan* is one in which each locator or server runs in its own VM. For example, the default medium plan comprises three locators and four servers, each running in its own VM, for a total of seven VMs.

The following sections describe how VMware GemFire for Tanzu Application Service allocates memory based on the total memory of each VM.

Memory Allocation for Colocated Plans

- If total VM memory is 5GB or less, 512 MB is allocated for the locator Java heap; 90% of the remaining memory is allocated to the cluster server's Java heap, and the remainder is allocated to the OS.
- If total VM memory is greater than 5GB, 10% of total memory is allocated for the locator Java heap; 90% of the remaining memory is allocated to the cluster server's Java heap, and the remainder is allocated to the OS.

Server heap is reduced to accommodate the **Anti-Virus** and **File Integrity Monitoring (FIM)** add-ons, if they are present.

Memory Allocation for Non-colocated Plans

Under non-colocated plans, each VM hosts one cluster member, either a locator or a server.

- If total VM memory is 2GB or less, 1GB is allocated for the locator or server Java heap, and the remainder to the OS.
- If total VM memory is between 2G and 8GB, 2GB is allocated to the OS, and the remainder to the locator or server Java heap.
- If total VM total memory is greater than 8G, 4GB is allocated to the OS, and the remainder to locator or server Java heap.

Server heap is reduced to accommodate the **Pivotal Anti-Virus** and **File Integrity Monitoring (FIM)** TAS for VMs add-ons, if they are present.

Configuring User Account and Authentication (UAA) Roles

This topic describes how to configure User Account and Authentication (UAA) roles in VMware GemFire for TAS.

> ⚠ **Caution**
>
> The UAA roles are not the same as the default roles used by GemFire for TAS when external authentication has not been activated. For information about the default roles, see Security within the Cluster in *Security*.

Overview

VMware GemFire for TAS service instances include predefined security roles for use with UAA. Each role has specific permissions for cluster operations. Each user is assigned one or more of these roles.

When a user invokes a cluster operation using `gfsh`, the security manager for the VMware GemFire for TAS service verifies that at least one of the user’s security roles has the permissions required to perform the cluster operation.

The cluster within a GemFire for TAS service instance implements role-based authentication and authorizes cluster operations based upon the roles. Two sets of roles exist:

- **UAA Roles**: A set of roles for GemFire for TAS instances that integrate with external authentication like LDAP.
- **Default Roles**: The default roles used by GemFire for TAS when no external authentication was integrated during the GemFire for TAS tile installation.

This topic describes how to configure **UAA roles**. For information about the default roles, see Security within the Cluster in *Security*.

Prerequisites

1. Before installing VMware GemFire for TAS and configuring UAA roles, you must create a

UAA client. For more information, see Create a User Account and Authentication (UAA) Client. Record the UAA client name and client secret.

2. Before configuring UAA roles, you must activate UAA Authorization within the VMware GemFire tile:

    1. In the Ops Manager Installation Dashboard, open the VMware GemFire tile.
    2. Open the Security pane.
    3. Select the **UAA Auth enable** radio button and input the UAA client name and client secret when prompted.

For more information about configuring the VMware GemFire tile, see Installing and Configuring VMware GemFire for TAS.

VMware GemFire for TAS Predefined UAA Security Roles

VMware GemFire for TAS predefined security roles for use with UAA and permissions:

Security Role Name	Permissions	Description

PCC_ADMIN	`CLUSTER:MANAGE`
	`CLUSTER:WRITE`
	`CLUSTER:READ`
	`DATA:MANAGE`
	`DATA:WRITE`
	`DATA:READ`
		All permissions required to manage the cluster and access region data.

PCC_OPERATOR	`CLUSTER:MANAGE`
	`CLUSTER:WRITE`
	`CLUSTER:READ`
		All permissions required to manage the cluster. Cannot access region data.

PCC_DATA-ACCESS	`CLUSTER:READ`
	`DATA:MANAGE`
	`DATA:WRITE`
	`DATA:READ`
		All permissions required to access region data. Cannot manage the cluster.

PCC_CLUSTER-READ-ONLY	`CLUSTER:READ` `DATA:READ`	Can view cluster and region data. Cannot manage the cluster or manipulate region data.
PCC_READ-ONLY	`DATA:READ`	Can view region data. Cannot manage the cluster or manipulate region data.

Configure the Roles

Before configuring the UAA roles, you must configure VMware GemFire for TAS to use UAA. For more information, see Prerequisites.

Configure the UAA server and your external authentication system, such as LDAP, with the Space-specific roles as follows:

1. In a terminal window, log in to the Cloud Foundry CLI and your Org. For example:

   ```
   cf login
   cf target -o NAME-OF-ORG
   ```

   Where `NAME-OF-ORG` is the name of your Org.

2. Retrieve and record the GUID of the Space that will host your VMware GemFire for TAS service instance using the command below. You use this GUID to create Space-specific groups within your Enterprise SSO system in the next step.

   ```
   cf space --guid NAME-OF-SPACE
   ```

   Where `NAME-OF-SPACE` is the name of the Space that will host your VMware GemFire for TAS service instance.

   The form of the output GUID will be similar to this example:

   ```
   03badc2a-4243-4251-84b5-c9bfba276f04
   ```

3. Create Space-specific groups for each of the UAA roles within your Enterprise SSO system. The name of each group is the name of the UAA role followed by an underscore character, followed by the GUID of the Space that you recorded in the previous step.

   Using the Space GUID in the example above, the names of the groups would be: * **PCC_ADMIN** group: `PCC_ADMIN_03badc2a-4243-4251-84b5-c9bfba276f04` * **PCC_OPERATOR** group: `PCC_OPERATOR_03badc2a-4243-4251-84b5-c9bfba276f04` * **PCC_DATA-ACCESS** group: `PCC_DATA-ACCESS_03badc2a-4243-4251-84b5-c9bfba276f04` * **PCC_CLUSTER-READ-ONLY** group: `PCC_CLUSTER-READ-ONLY_03badc2a-4243-4251-84b5-c9bfba276f04` * **PCC_READ-ONLY** group: `PCC_READ-ONLY_03badc2a-4243-4251-84b5-c9bfba276f04`

4. Assign users to these Space-specific groups within your Enterprise SSO system.

5. In a terminal window, use the UAA Command Line Interface (UAAC) to log in as `admin client` to your UAA server.

6. Use the UAAC to add each group name to the UAA server by running the following command for each group:

   ```
   uaac group add ROLE_SPACEGUID
   ```

   Where `ROLE_SPACEGUID` is a group name that you created in a previous step.

   For example, using the group name from above, the following commands add the groups to the UAA server:

   ```
   $ uaac group add PCC_ADMIN_03badc2a-4243-4251-84b5-c9bfba276f04
   ```

```
$ uaac group add PCC_OPERATOR_03badc2a-4243-4251-84b5-c9bfba276f04
$ uaac group add PCC_DATA-ACCESS_03badc2a-4243-4251-84b5-c9bfba276f04
$ uaac group add PCC_CLUSTER-READ-ONLY_03badc2a-4243-4251-84b5-c9bfba2
76f04
$ uaac group add PCC_READ-ONLY_03badc2a-4243-4251-84b5-c9bfba276f04
```

7. Use the UAAC to map each group name to the UAA server by running the `uaac group map` command. For example, for LDAP:

   ```
   uaac group map --name ROLE_SPACEGUID "GROUP-DISTINGUISHED-NAME"
   ```

   Where: * `ROLE_SPACEGUID` is a group name that you created in a previous step. * `GROUP-DISTINGUISHED-NAME` is the LDAP distinguished name of a Space-specific group that you created in a previous step.

   For example:

   ```
   $ uaac group map --name PCC_DATA-ACCESS_03badc2a-4243-4251-84b5-c9bfba
   276f04 "CN=PCC_DATA-ACCESS_03badc2a-4243-4251-84b5-c9bfba276f04,OU=Gro
   ups,DC=pivotal,DC=io"
   ```

   For more information about the `uaac group map` command, see Grant Admin Permissions to an External Group (SAML, LDAP, or OIDC) in *Creating and Managing Users with the UAA CLI (UAAC)* in the VMware Tanzu Application Service for VMs product documentation.

Setting Service Instance Quotas

This topic explains how to set service instance quotas in VMware GemFire for TAS.

- Create Global-level Quotas
- Create Plan-level Quotas
- Create and Set Org-level Quotas
- Create and Set Space-level Quotas
- View Current Org and Space-level Quotas
- Monitor Quota Use and Service Instance Count
- Calculate Resource Costs for On-Demand Plans
  - Calculate Maximum Resource Cost Per On-Demand Plan
  - Calculate Maximum Resource Cost for All On-Demand Plans
  - Calculate Actual Resource Cost of all On-Demand Plans

On-demand provisioning is intended to accelerate app development by eliminating the need for development teams to request and wait for operators to create a service instance. However, to control costs, operations teams and administrators must ensure responsible use of resources.

There are several ways to control the provisioning of on-demand service instances by setting various **quotas** at these levels:

- Global

- Plan
- Org
- Space

After you set quotas, you can:

- View Current Org and Space-level Quotas
- Monitor Quota Use and Service Instance Count
- Calculate Resource Costs for On-Demand Plans

Create Global-level Quotas

Each on-demand service has a separate service broker. A global quota at the service level sets the maximum number of service instances that can be created by a given service broker. If a service has more than one plan, then the number of service instances for all plans combined cannot exceed the global quota for the service.

The operator sets a global quota for each service tile independently. For example, if you have two service tiles, you must set a separate global service quota for each of them.

When the global quota is reached for a service, no more instances of that service can be created unless the quota is increased, or some instances of that service are deleted.

Create Plan-level Quotas

A service may offer one or more plans. You can set a separate quota per plan so that instances of that plan cannot exceed the plan quota. For a service with multiple plans, the total number of instances created for all plans combined cannot exceed the global quota for the service.

When the plan quota is reached, no more instances of that plan can be created unless the plan quota is increased or some instances of that plan are deleted.

Create and Set Org-level Quotas

An org-level quota applies to all on-demand services and sets the maximum number of service instances an organization can create within their foundation. For example, if you set your org-level quota to 100, developers can create up to 100 service instances in that org using any combination of on-demand services.

When this quota is met, no more service instances of any kind can be created in the org unless the quota is increased or some service instances are deleted.

To create and set an org-level quota, do the following:

1. Run this command to create a quota for service instances at the org level:

   ```
   cf create-quota QUOTA-NAME -m TOTAL-MEMORY -i INSTANCE-MEMORY -r ROUTES -s SERVICE-INSTANCES --allow-paid-service-plans
   ```

   Where:

   - `QUOTA-NAME` is a name for this quota.

    - `TOTAL-MEMORY` is the maximum memory used by all service instances combined.
    - `INSTANCE-MEMORY` is the maximum memory used by any single service instance.
    - `ROUTES` is the maximum number of routes allowed for all service instances combined.
    - `SERVICE-INSTANCES` is the maximum number of service instances allowed for the org.

   For example:

   ```
   $ cf create-quota myquota -m 1024mb -i 16gb -r 30 -s 50 --allow-paid-s
   ervice-plans
   ```

2. Associate the quota you created above with a specific org by running the following command:

   ```
   cf set-quota ORG-NAME QUOTA-NAME
   ```

   For example:

   ```
   $ cf set-quota dev_org myquota
   ```

For more information about managing org-level quotas, see Creating and Modifying Quota Plans in the Tanzu Application Service for VMs product documentation.

Create and Set Space-level Quotas

A space-level service quota applies to all on-demand services and sets the maximum number of service instances that can be created within a given space in a foundation. For example, if you set your space-level quota to 100, developers can create up to 100 service instances in that space using any combination of on-demand services.

When this quota is met, no more service instances of any kind can be created in the space unless the quota is updated or some service instances are deleted.

To create and set a space-level quota, do the following:

1. Run the following command to create the quota:

   ```
   cf create-space-quota QUOTA-NAME -m TOTAL-MEMORY -i INSTANCE-MEMORY -r ROUTES -
   s SERVICE-INSTANCES --allow-paid-service-plans
   ```

   Where:

    - `QUOTA-NAME` is a name to use for this quota.
    - `TOTAL-MEMORY` is the maximum memory used by all service instances, combined.
    - `INSTANCE-MEMORY` is the maximum memory used by any single service instance.
    - `ROUTES` is the maximum number of routes allowed for all service instances, combined.
    - `SERVICE-INSTANCES` is the maximum number of service instances allowed for the org.

   For example:

   ```
   $ cf create-space-quota myspacequota -m 1024mb -i 16gb -r 30 -s 50 --a
   ```

```
llow-paid-service-plans
```

2. Associate the quota that you created above with a specific space by running the following command:

   ```
   cf set-space-quota SPACE-NAME QUOTA-NAME
   ```

   For example:

   ```
   $ cf set-space-quota myspace myspacequota
   ```

For more information about managing org-level quotas, see Creating and Modifying Quota Plans in the Tanzu Application Service for VMs product documentation.

View Current Org and Space-level Quotas

To view **org** quotas, run the following command.

```
cf org ORG-NAME
```

To view **space** quotas, run the following command:

```
cf space SPACE-NAME
```

For more information about managing org-level quotas, see Creating and Modifying Quota Plans in the Tanzu Application Service for VMs product documentation.

Monitor Quota Use and Service Instance Count

Service-level and plan-level quota use, and total number of service instances, are available through the on-demand broker metrics emitted to Loggregator. These metrics are listed below:

Metric Name	Description
`on-demand-broker/SERVICE-NAME/quota_remaining`	Quota remaining for all instances across all plans
`on-demand-broker/SERVICE-NAME/PLAN-NAME/quota_remaining`	Quota remaining for a specific plan
`on-demand-broker/SERVICE-NAME/total_instances`	Total instances created across all plans
`on-demand-broker/SERVICE-NAME/PLAN-NAME/total_instances`	Total instances created for a specific plan

Note: Quota metrics are not emitted if no quota has been set.

You can also view service instance usage information in Apps Manager. For more information, see Reporting Instance Usage with Apps Manager in the Tanzu Application Service for VMs product documentation.

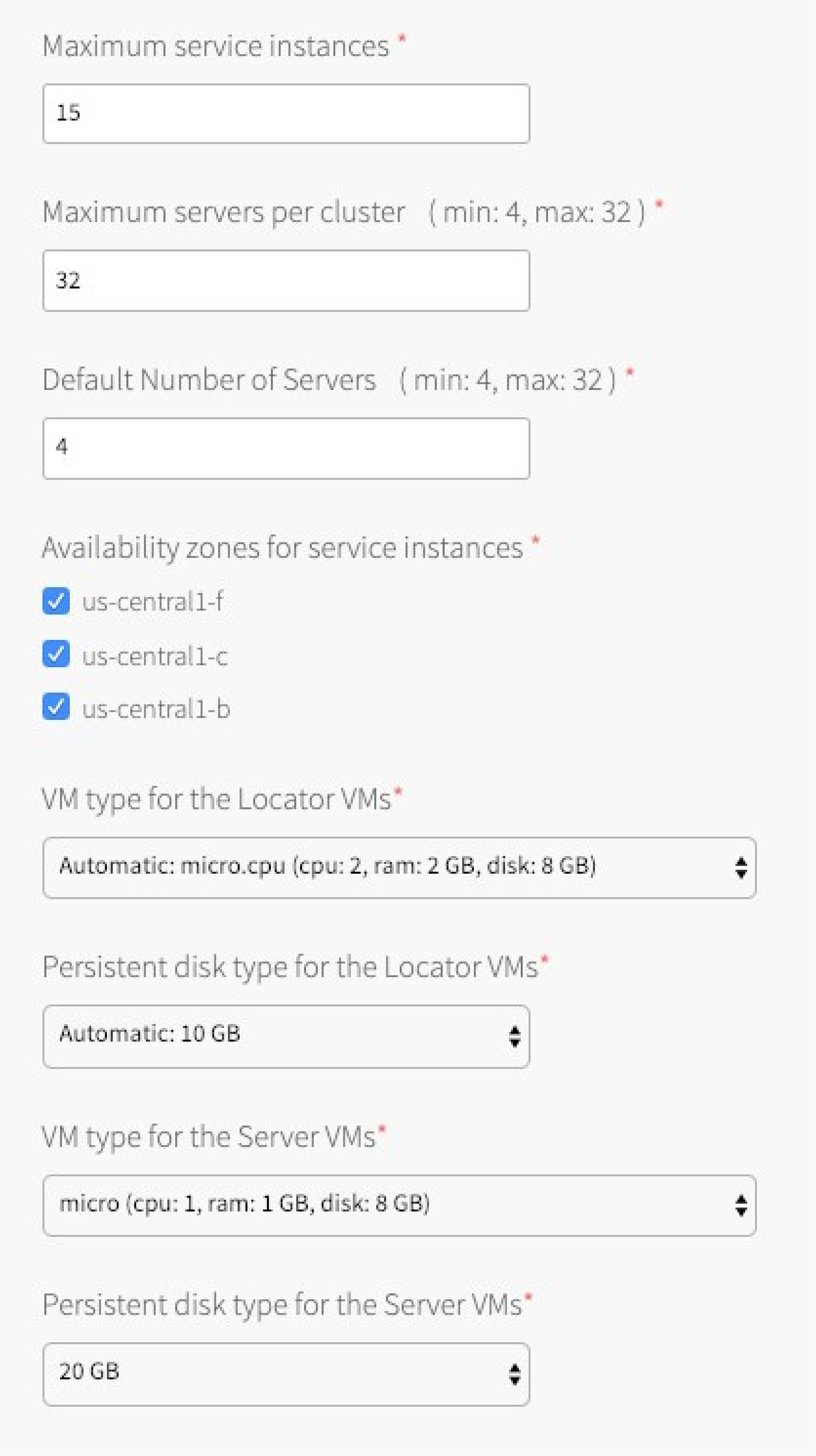
Calculate Resource Costs for On-Demand Plans

On-demand plans use dedicated VMs, disks, and various other resources from an IaaS, such as AWS. To calculate maximum resource cost for plans individually or combined, multiply the quota by the cost of the resources selected in the plan configurations. The specific costs depend on your IaaS.

To view configurations for your VMware GemFire on-demand plan, do the following:

1. Navigate to **Ops Manager Installation Dashboard** > **VMware GemFire** > **Settings**.

The image below shows an example that includes the VM type and persistent disk selected for the server VMs, as well as the quota for this plan.



Maximum service instances *

15

Maximum servers per cluster (min: 4, max: 32) *

32

Default Number of Servers (min: 4, max: 32) *

4

Availability zones for service instances *

☑ us-central1-f

☑ us-central1-c

☑ us-central1-b

VM type for the Locator VMs*

Automatic: micro.cpu (cpu: 2, ram: 2 GB, disk: 8 GB)

Persistent disk type for the Locator VMs*

Automatic: 10 GB

VM type for the Server VMs*

micro (cpu: 1, ram: 1 GB, disk: 8 GB)

Persistent disk type for the Server VMs*

20 GB

Note: Although operators can limit on-demand instances with plan quotas and a global quota, as described in the above topics, IaaS resource usage still varies based on the number of on-demand instances provisioned.

Calculate Maximum Resource Cost Per On-Demand Plan

To calculate the maximum cost of VMs and persistent disk for each plan, do the following calculation:

plan quota x cost of selected resources

For example, if you selected the options in the above image, you have selected a VM type **micro** and a persistent disk type **20 GB**, and the plan quota is **15**. The VM and persistent disk types have an associated cost for the IaaS that you are using. Therefore, to calculate the maximum cost of resources for this plan, multiply the cost of the resources selected by the plan quota:

(15 x cost of micro VM type) + (15 x cost of 20 GB persistent disk) = max cost per plan

Calculate Maximum Resource Cost for All On-Demand Plans

To calculate the maximum cost for all plans combined, add together the maximum costs for each plan. This assumes that the sum of your individual plan quotas is less than the global quota.

For example:

(plan1 quota x plan1 resource cost) + (plan2 quota x plan2 resource cost) = max cost for all plans

Calculate Actual Resource Cost of all On-Demand Plans

To calculate the current actual resource cost across all of your on-demand plans:

1. Find the number of instances currently provisioned for each active plan by looking at the `total_instance` metric for that plan.
2. Multiply the `total_instance` count for each plan by that plan's resource costs. Record the costs for each plan.
3. Add up the costs noted in Step 2 to get your total current resource costs.

For example:

(plan1 total_instances x plan1 resource cost) + (plan2 total_instances x plan2 resource cost) = current cost for all plans

Upgrading VMware GemFire for TAS

This topic describes upgrading VMware GemFire for TAS.

You can upgrade directly to VMware GemFire for Tanzu Application Service version v1.15 from the latest patch release of versions GemFire for TAS 1.10 or later. This "jump upgrade" is a two-step process:

1. Verify that your current version of GemFire for TAS is up-to-date.
2. Follow the upgrade procedure.

Verify that Your Current Version is Up-to-date

Verify that your current version of GemFire for TAS has been updated to its latest patch release. If you have not installed the latest patch, then do so.

The latest patch releases are:

GemFire for TAS Version (Major.Minor)	Latest Patch Release
1.14	1.14.4
1.13	1.13.7
1.12	1.12.4
1.11	1.11.3
1.10	1.10.9

Upgrading to GemFire for TAS 1.14 from any version other than those shown requires multiple steps. For example, if your current version is v1.10.5, you must first upgrade to the latest patch of 1.10 shown in the table, from which you can then jump upgrade to 1.14.

In addition, be sure to check the following section for any version-specific upgrade considerations.

Version-specific Upgrade Notes

- Each GemFire for TAS release is compatible with two VMware Tanzu Application Service for VMs (TAS for VMs) and Ops Manager versions. Incorporate those upgrades to TAS for VMs and Ops Manager in your upgrade process as required to maintain compatibility.
- If your application connects via the Services Gateway feature, which was introduced in v1.13.0, you must first upgrade to the latest GemFire for TAS v1.13 patch before upgrading further. Then upgrade to 1.14, unbind, rebind and restart your application.
- To upgrade to v1.9 from earlier releases, you must upgrade minor releases sequential order. For example, VMware GemFire for Tanzu Application Service v1.7 must be upgraded to GemFire for TAS v1.8 prior to upgrading to GemFire for TAS v1.9.
- VMware GemFire for TAS version 1.13.1 included an Apache Geode v1.13.2 performance improvement that increased the defaults for maximum number of pooled message processor threads and maximum partitioned region message processor threads.

  If you are upgrading from a version of VMware GemFire for TAS earlier than version 1.13.1, and your system had been hitting the old, lower default maximums, then upgrading to v1.15 may cause increased use of system resources as the system is no longer constrained to the old values. If your system depends on constraining system resources to the old default values, you can set these properties explicitly using DistributionManager.MAX_THREADS and DistributionManager.MAX_PR_THREADS, respectively.

  The following table shows the old and new maximum values.

System Resource	Old Default Value	New Default Value
DistributionManager.MAX_THREADS	100	1000

System Resource	Old Default Value	New Default Value
DistributionManager.MAX_PR_THREADS	The greater of (CPUs * 4) or 16	The greater of (CPUs * 32) or 200

Upgrade Procedure

Follow these steps to upgrade GemFire for TAS:

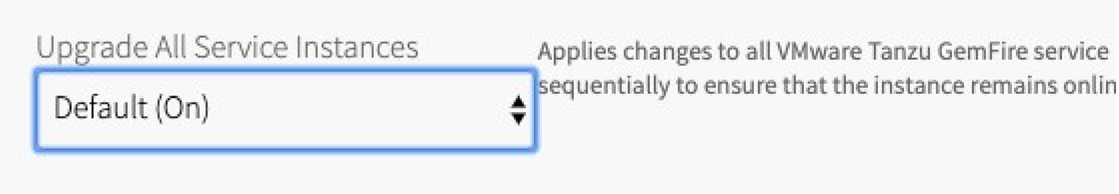
1. Download the new version of the tile from VMware Tanzu Network.
2. Upload the product to Ops Manager.
3. Click **Add** next to the uploaded product.
4. Click on the GemFire for TAS tile and configure the upgrade options.
    - To try the upgrade on a small number of service instances first, set the quantity of canary service instances as described in Service Instance Upgrades.
    - Set the number of instances that are to be upgraded in parallel as described in Service Instance Upgrades.
    - Under the **Errands** section, choose the **Default (On)** value for the **Upgrade All Service Instances** post-deploy errand. **Save** the change.
5. Click **Review Pending Changes**. For more information, see Reviewing Pending Product Changes in the Ops Manager product documentation.
6. Click **Apply Changes**.

Enable Individual Service Instance Upgrades

The default upgrade path upgrades all service instances as a result of a tile upgrade. This operation can take a lengthy amount of time. To expedite upgrades, an operator can permit developers to upgrade their own service instances once the tile has been upgraded.

An operator enables individual service instance upgrades during tile installation. This feature requires TAS for VMs/Ops Manager 2.7 or a higher version and works for upgrading from GemFire for TAS 1.9.0 to a higher version.

Within the GemFire for TAS tile, in the **Errands** section, the default for the **Upgrade All Service Instances** errand, which upgrades all service instances, appears as:

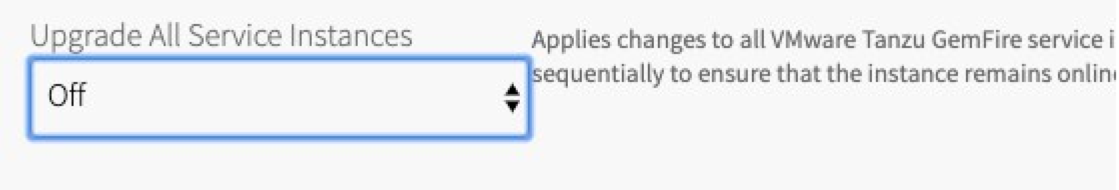


Upgrade All Service Instances

Default (On)

Applies changes to all VMware Tanzu GemFire service sequentially to ensure that the instance remains onlin

To change the state of this errand such that individual service instance upgrades are enabled, choose **Off** for this errand:



Upgrade All Service Instances

Off

Applies changes to all VMware Tanzu GemFire service i
sequentially to ensure that the instance remains onlin

Click **Save**.

Once individual service instance upgrades are enabled, the developer upgrades an individual service instance following the instructions in Upgrade a Single Service Instance.

Updating VMware GemFire for TAS Plans

This topic explains how to update VMware GemFire for TAS plans.

Follow the steps below to update plans in Ops Manager.

1. Click on the VMware GemFire for Tanzu Application Service tile.
2. Click on the plan you want to update under the **Information** section.
3. Edit the fields with the changes you want to make to the plan.
4. Click **Save** button on the bottom of the page.
5. Click on **Ops Manager** to navigate to the **Installation Dashboard**.
6. Click **Review Pending Changes**. For more information, see Reviewing Pending Product Changes in the VMware Tanzu Ops Manager documentation.
7. Click **Apply Changes**.

Plan changes are not applied to existing services instances until you run the `upgrade-all-service-instances` BOSH errand. You must use the BOSH CLI to run this errand. Until you run this errand, developers cannot update service instances.

Changes to fields that can be overridden by optional parameters, for example `num_servers` or `new_size_percentage`, change the default value of these instance properties, but do not affect existing service instances.

If you change the allowed limits of an optional parameter, for example the maximum number of servers per cluster, existing service instances in violation of the new limits are not modified.

When existing instances are upgraded, all plan changes are applied to them. Upgrades and updates to service instances can cause a rolling restart of cluster servers. Be aware that the rebalancing of data to maintain redundancy may impact the performance of the remainder of the servers within the service instance.

⚠ **WARNING:** Data loss may result from the restart of a cluster. See Restarting a Cluster for the conditions under which data loss occurs.

Uninstalling VMware GemFire for TAS

This topic explains how to uninstall VMware GemFire for TAS.

To uninstall VMware GemFire for Tanzu Application Service, follow the steps from below from the

Installation Dashboard:

1. Click the trash can icon in the bottom-right-hand corner of the tile.
2. Click **Review Pending Changes**. For more information, see Reviewing Pending Product Changes in the VMware Tanzu Ops Manager documentation.
3. Click **Apply Changes**.

Backing Up and Restoring Service Instances

This topic explains backing up and restoring service instances for use with VMware GemFire for TAS.

Both disaster recovery and system validation depend upon backups. A VMware GemFire for Tanzu Application Service service instance backup consists of the configuration for a GemFire for TAS service instance together with all region data that has been persisted to disk. A GemFire for TAS service instance backup may be used to restore the service on a new, but unconfigured GemFire for TAS service instance.

Before You Begin

> ⚠ **WARNING**: Consider each of these important items when preparing to back up and restore a GemFire for TAS service instance. Since recovery depends on a correct implementation, address each item during planning.

- The backup consists of data from regions that have been persisted to disk. All non-persistent region data will be lost. A persistent region is one that writes its region entries to disk.
- The backup and restoration process does not restore WAN replication configuration. A GemFire for TAS service instance that replicates data via WAN may be backed up. A restore of that GemFire for TAS service instance will contain all the persistent data. That GemFire for TAS service instance will be able to communicate with other WAN-connected service instances after further configuration.
- The disk size on the jumpbox VM must be large enough to hold the backup artifacts. The quantity of disk space needs to exceed the sum total of all disk space used within the GemFire for TAS service instance to be backed up. An approximation of that quantity of disk space may be calculated by looking at the fully populated GemFire for TAS service instance. Sum the disk space (in `/var/vcap/store`) used by each locator and server.
- The GemFire for TAS service instance needs to be quiescent when the backup is taken. An active cluster could have disk writes in progress, leading to a backup for which region data may or may not have been written to disk.
- The fresh GemFire for TAS service instance that is to be used for a restore must have same quantity of locators and servers as the GemFire for TAS service instance had when its backup was taken. The fresh GemFire for TAS service instance must have sufficient resources to hold the data that is in the backup.
- Do not configure the fresh GemFire for TAS service instance that is to be used for a restore after creation. It must be empty. Do not create any regions or start any gateway senders. The restore will create all regions, both persistent and not persistent. The non-persistent regions

will be empty.

- Service keys are not part of the backup. Service keys will need to be created anew on a restored service instance.

Backing Up a GemFire for TAS Service Instance

1. Optional: Back up the VMware Tanzu Application Service for VMs environment. For instructions, see Backing Up and Restoring Deployments in the VMware Tanzu Ops Manager documentation.
2. Optional: Compact the disk stores of the cluster to be backed up. Use the gfsh `compact disk-store` command after connecting to the cluster with a role that is authorized with the `CLUSTER:MANAGE` operation permission. See the documentation for gfsh `compact disk-store`.
3. Optional: Acquire a region statistic that may be used for a minimal validation upon using this backup for subsequent restoration. Use the gfsh `describe region` command on each persistent region, after connecting to the cluster with a role that is authorized with the `CLUSTER:READ` operation permission.

   ```
   describe region --name=REGION-NAME
   ```

   For each persistent region (the `data-policy` will contain the string `PERSISTENT`), record the `size` of region. This size is the quantity of entries in the region. Assuming that activity on the region is quiescent, when this backup is used in a restoration, the quantity of region entries in the restored region forms a minimal validation of the region.
4. SSH to the jumpbox. Assuming that the Ops Manager VM is being used as the jumpbox, follow directions in Log in to the Ops Manager VM with SSH in *Advanced Troubleshooting with the BOSH CLI* in the VMware Tanzu Ops Manager documentation.
5. Determine all needed credentials and parameters for the command that makes the backup:
   - `BOSH-DIRECTOR-IP`: Obtain the BOSH Director IP address by following the instructions in Retrieve BOSH Director Address in *Backing Up Deployments with BBR* in the VMware Tanzu Ops Manager documentation..
   - `SERVICE-INSTANCE-DEPLOYMENT-NAME`: Obtain the deployment name by following the instructions at Acquire the Deployment Name.
   - `BOSH-CLIENT`, `BOSH-CLIENT-PASSWORD`, and `PATH-TO-BOSH-SERVER-CERTIFICATE`: To learn all three of these values, open the Credentials tab of the BOSH Director tile in Ops Manager. Find and open `Bosh Commandline Credentials`. The path is the path to the root Certificate Authority (CA) certificate, and its value will be `/var/tempest/workspaces/default/root_ca_certificate` if Ops Manager and the jumpbox are on the same VM.
6. Make the backup. From the jumpbox, issue this command, substituting values acquired in the previous step:

   ```
   BOSH_CLIENT_SECRET=BOSH-CLIENT-PASSWORD \
   bbr deployment \
   --target BOSH-DIRECTOR-IP \
   --username BOSH-CLIENT \
   ```

```
--ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
--deployment SERVICE-INSTANCE-DEPLOYMENT-NAME \
backup
```

`BOSH_CLIENT_SECRET` is an environment variable that is set only for the duration of the command.

The backup will be within a directory on the jumpbox named by the deployment name and a timestamp.

7. Copy the backup to a permanent home for archiving. VMware recommends compressing and encrypting the files. Disaster recovery plans often recommend archiving multiple copies of each backup.

Restoring a GemFire for TAS Service Instance

1. Use SSH to connect to the jumpbox. Assuming that the Ops Manager VM is being used as the jumpbox, follow directions in Log in to the Ops Manager VM with SSH in *Advanced Troubleshooting with the BOSH CLI* in the VMware Tanzu Ops Manager documentation.
2. Transfer the archived backup to the jumpbox. Expand if compressed, and decrypt if encrypted.
3. Make a new, but empty service instance. See directions in Creating a VMware GemFire for Tanzu Application Service Service Instance. If the service instance to be restored was connected to a WAN, be sure to create the new instance with the same `distributed_system_id` as the old one.
4. Determine all needed credentials and parameters for the command that does the restore by following step 5 instructions from making the backup.
5. Do the restore. From the jumpbox, issue this command, substituting values acquired in the previous step:

   ```
   $ BOSH_CLIENT_SECRET=BOSH-CLIENT-PASSWORD \
   bbr deployment \
   --target BOSH-DIRECTOR-IP \
   --username BOSH-CLIENT \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
   --deployment SERVICE-INSTANCE-DEPLOYMENT-NAME \
   restore --artifact-path PATH-TO-SERVICE-INSTANCE-ARTIFACT
   ```

   `PATH-TO-SERVICE-INSTANCE-ARTIFACT` is the path to the artifact for the instance that you are currently restoring.
6. Create a new service key, as the service key was not an artifact that the backup created. For a stand-alone service instance (one that is not part of a WAN), follow the directions at Create a Service Key. For a WAN-connected service instance, see Restoring a WAN Connection.
7. If the region size was captured prior to making the backup, in order to do a minimal validation of the restored cluster, use the gfsh `describe region` command on each persistent region, after connecting to the cluster with a role that is authorized with the `CLUSTER:READ` operation permission.

```
describe region --name=REGION-NAME
```

For each persistent region, the `size` of the restored region should be the same as the value captured when the backup was made.

8. Rebind or restage all apps.

Configuring a Service Gateway

This topic describes how to configure a service gateway for use with VMware GemFire for TAS.

A service gateway enables communication from an app to the GemFire for TAS service instance. The running App's Location determines whether a service gateway is required.

The operator needs to complete two items to enable a service gateway:

- With a VMware Tanzu Application Service for VMs (TAS for VMs) version 2.10 deployment, follow the directions to Enabling and Configuring TCP Routing in the VMware TAS for VMs documentation. Note the range entered for the **TCP routing ports**. One port from this range will be required to configure the GemFire for TAS tile.
- Configure the GemFire for TAS tile to enable the service gateway as described in *Settings: Enable Creation of a Service Gateway* in Installing and Configuring VMware GemFire.

Enabling Service-Instance Sharing

This topic explains how to enable VMware GemFire for TAS service instance sharing.

The VMware GemFire for Tanzu Application Service implementation of instance sharing permits read-only access to a GemFire for TAS service instance within one space by an app within another space. A Cloud Foundry `admin` operator, or one with administrative privileges, may enable service-instance sharing for GemFire for TAS service instances.

Enable service-instance sharing across a foundation with:

```
$ cf enable-feature-flag service_instance_sharing
```

You must configure GemFire for TAS service-instance sharing as detailed in Service-Instance Sharing.

Rotating Certificates

This topic specifies the procedure to follow to check expiration dates and rotate certificates when using VMware GemFire for TAS.

To rotate the Services TLS CA and its leaf certificates, for service instances that are *not* connected by WAN gateways, use one of the following procedures:

- If you are running Ops Manager 2.10, follow the procedure in Rotate the Services TLS CA and Its Leaf Certificates in *Advanced Certificate Rotation with CredHub Maestro* in the VMware Tanzu Ops Manager 2.10 documentation.

> ⚠ **Warning**: For apps that connect to the VMware GemFire cluster using TLS, at

each step in the procedure that rebinds and restages apps, recreate the truststore following the directions in Create a Truststore) in *Accessing a Service Instance*prior to rebinding. Spring Boot Data Geode (SBDG) apps that set GemFire properties `ssl-use-default-context=true` and `ssl-endpoint-identification-enabled=false` do not need to recreate a truststore.

- If you are running Ops Manager 2.9, follow the procedure in Rotate the Services TLS CA and Its Leaf Certificates in *Advanced Certificate Rotation with CredHub Maestro* in the VMware Tanzu Ops Manager 2.9 documentation.

  ⚠ **Warning**: For apps that connect to the VMware GemFire cluster using TLS, at each step in the procedure that rebinds and restages apps, recreate the truststore following the directions in Create a Truststore) in *Accessing a Service Instance*prior to rebinding. Spring Boot Data Geode (SBDG) apps that set GemFire properties `ssl-use-default-context=true` and `ssl-endpoint-identification-enabled=false` do not need to recreate a truststore.

WAN-Connected Service Instances

Follow this procedure for rotating the certificates of foundations, when service instances connected via WAN gateways reside on distinct foundations.

The procedure assumes two foundations have service instances, one called foundation A and the other called foundation B. The procedure may be followed by installations that have more than two foundations. To expand the procedure to cover more foundations, in each place where foundation A and foundation B are identified, also apply the directions for a third foundation C, a fourth foundation D, and so on.

The procedure is broken into parts. Do the parts in the order listed. The parts are:

- Generate New TLS CA Certificates
- Collect All the Certificates
- Add All the Certificates to the Tiles
- Apply Changes for the First Time
- Set New Services TLS CA Certificate
- Apply Changes for the Second Time
- Remove the Old Services TLS CA Certificates
- Apply Changes for the Third Time

Generate New TLS CA Certificates

Follow these steps twice to obtain a new TLS CA certificate, once to generate a new TLS CA certificate for foundation A, and once to generate a new TLS CA certificate for foundation B.

1. Check if CredHub has a new temporary certificate from a previous rotation attempt. Run:

```
credhub get -n /services/new_ca
```

> **Note**: This procedure assumes that `/services/new_ca` is the CredHub location used when generating the temporary certificate. If you used a different location, specify that location instead.

2. If an older temporary certificate already exists, delete that certificate by running:

   ```
   credhub delete -n /services/new_ca
   ```

3. Do one of the following:

   - **If you get your certificates from your CA**: If you generate certificates from your own private or public CA, obtain a new certificate from that CA. You add this to the Ops Manager settings in the next high-level step.

   - **If you use self-signed certificates**: Generate a new self-signed certificate with a new name or path in CredHub:

     ```
     credhub generate \
     --name="/services/new_ca" \
     --type="certificate" \
     --no-overwrite \
     --is-ca \
     --duration=1825 \
     --common-name="opsmgr-services-tls-ca"
     ```

     Where: - `name` is the CredHub path or name of the new certificate. You can use `/services/new_ca` or substitute a different name as long as you use the value consistently in the rest of the procedure. - `common-name` is the common name of the generated certificate. You can use `opsmgr-services-tls-ca` or substitute a different common name as long as you use the value consistently in the rest of the procedure. - `duration` is set to `1825`, which is a recommended value of five years. If you do not specify a duration, the default value is `365` (one year).

   - **If you use an intermediate certificate signed by a root CA in CredHub**: Generate a new certificate signed by the CredHub root CA by running:

     ```
     credhub generate \
     --name="/services/new_ca" \
     --type="certificate" \
     --no-overwrite \
     --is-ca \
     --duration=1825 \
     --common-name="opsmgr-services-tls-ca" \
     --ca=/PATH-TO-ROOT-CA
     ```

     Where: - `name` is the CredHub path or name of the new certificate. You can use `/services/new_ca` or substitute a different name as long as you use the value consistently in the rest of the procedure. - `common-name` is the common name of the generated certificate. You can use `opsmgr-services-tls-ca` or substitute a different common name as long as you use the value consistently in the rest of the procedure. - `duration` is set to `1825`, which is a recommended value of five years. If you do not specify a duration, the default value is `365` (one year). - `ca` is the path to the root CA for the new certificate. Replace `PATH-TO-ROOT-CA` with the appropriate path for the root CA.

Collect All the Certificates

Each of foundation A and foundation B has a current certificate and the newly generated one from the previous step, totalling four certificates across two foundations.

1. On foundation A, log in to CredHub.
2. On foundation A, get the *current* Services TLS CA certificate by capturing the certificate in the output of:

   ```
   credhub get --name=/services/tls_ca -k ca
   ```

3. On foundation A, get the *new* Services TLS CA certificate either from a pre-existing file, or from your new CredHub location by capturing the certificate in the output of:

   ```
   credhub get --name=/services/new_ca -k ca
   ```

4. On foundation B, log in to CredHub.
5. On foundation B, get the *current* Services TLS CA certificate by capturing the certificate in the output of:

   ```
   credhub get --name=/services/tls_ca -k ca
   ```

6. On foundation B, get the *new* Services TLS CA certificate either from a pre-existing file, or from your new CredHub location by capturing the certificate in the output of:

   ```
   credhub get --name=/services/new_ca -k ca
   ```

Add All the Certificates to the Tiles

This part adds all the certificates (current and new) to the tiles. For a distributed system with two foundations, the four certificates are:

- The current TLS CA certificate for foundation A (called `/services/tls_ca`)
- The current TLS CA certificate for foundation B (also called `/services/tls_ca`)
- The new TLS CA certificate for foundation A (called `/services/new_ca`)
- The new TLS CA certificate for foundation B (also called `/services/new_ca`)
- On foundation A, paste all certificates (ordering of the certificates does not matter) into the **BOSH Director > Security > Trusted Certificates** field. Click **Save**.
- On foundation A, paste all certificates (ordering of the certificates does not matter) into two fields of the VMware Tanzu Application Service tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. Click **Save**.
- On foundation A, paste all certificates (ordering of the certificates does not matter) into two

fields of the Isolation Segment tile, if the environment has this optional tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. Click **Save**.

- On foundation B, paste all certificates (ordering of the certificates does not matter) into the **BOSH Director > Security > Trusted Certificates** field. Click **Save**.
- On foundation B, paste all certificates (ordering of the certificates does not matter) into two fields of the VMware Tanzu Application Service tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. Click **Save**.
- On foundation B, paste all certificates (ordering of the certificates does not matter) into two fields of the Isolation Segment tile, if the environment has this optional tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. Click **Save**.

Apply Changes for the First Time

⚠ **Warning**: This procedure involves redeploying all of the VMs in your Ops Manager deployment to apply a CA certificate. The operation can take a long time to complete.

You may apply the changes to both foundation A and foundation B at the same time.

Follow these steps twice, once to apply the changes to foundation A, and once to apply the changes to foundation B.

1. Navigate back to the **Installation Dashboard**.
2. Click **Review Pending Changes**.
3. Ensure that all product tiles, including TAS for VMs, TAS for VMs [Windows], Isolation Segment, and partner tiles, are selected.
4. Identify which on-demand service tiles are using the Services TLS CA certificate. Run:

   ```
   credhub curl -p /api/v1/certificates?name=%2Fservices%2Ftls_ca
   ```

   From the returned list, identify which on-demand service tiles use TLS in your deployment, such as MySQL for VMware Tanzu, VMware GemFire for TAS, Redis for VMware Tanzu, and RabbitMQ for VMware Tanzu [VMs].
5. For each on-demand service tile that uses TLS:
   1. Expand the errands.
   2. Enable the errand to upgrade all service instances.

   Note: The name of the upgrade all service instances errand may differ slightly between services.
6. Click **Apply Changes**.

Set New Services TLS CA Certificate

Set the new Services TLS CA certificate in CredHub. Do this on foundation A with foundation A's new TLS CA certificate, and do this on foundation B with foundation B's new TLS CA certificate.

Do one of the following:

- **If you have an existing certificate:** Obtain the CA certificate and private key file corresponding to the certificate that you got in the Get New TLS CA Certificates part of the procedure. Then, run:

  ```
  credhub set \
  --name="/services/tls_ca" \
  --type="certificate" \
  --certificate=PEM-PATH/root.pem \
  --private=CERT-KEY
  ```

  Where: - `certificate` is the location of the `root.pem` file for the certificate. Replace `PEM-PATH` with the path of your certificate's `root.pem` file. - `private` is the private key for the new certificate. Replace `CERT-KEY` with the value of the private key for your certificate.

- **If you created a new self-signed or intermediary certificate:** Set the `/services/new_ca` that you generated in the Get New TLS CA Certificates part of the procedure as the Services TLS CA:

  ```
  credhub get -n /services/new_ca -k ca > new_ca.ca
  credhub get -n /services/new_ca -k certificate > new_ca.certificate
  credhub get -n /services/new_ca -k private_key > new_ca.private_key
  credhub set -n /services/tls_ca \
  --type=certificate \
  --root=new_ca.ca \
  --certificate=new_ca.certificate \
  --private=new_ca.private_key
  ```

Apply Changes for the Second Time

> **Warning**: This procedure involves redeploying all of the VMs in your Ops Manager deployment to apply a CA certificate. The operation can take a long time to complete.
>
> You may apply the changes to both foundation A and foundation B at the same time.

In this step, you apply changes to ensure that all on-demand service instances generate new leaf certificates from the new Services TLS CA.

Follow these steps twice, once to apply the changes to foundation A, and once to apply the changes to foundation B.

1. Navigate back to the **Installation Dashboard**.
2. Click **Review Pending Changes**.
3. Ensure that all product tiles, including TAS for VMs, TAS for VMs [Windows], Isolation Segment, and partner tiles, are deselected in order to reduce deployment time.

4. Select only the on-demand services tiles that use TLS, such as MySQL for VMware Tanzu, VMware GemFire for TAS, Redis for VMware Tanzu, and RabbitMQ for VMware Tanzu [VMs].
5. For each on-demand service tile that uses TLS:
    1. Expand the errands.
    2. Enable the errand to upgrade all service instances.

    > **Note:** The name of the upgrade all service instances errand may differ slightly between services.
6. Click **Apply Changes**.

Remove the Old Services TLS CA Certificates

After your apps have reconnected to service instances with the certificates generated by the new CA, both of the old CA certificates may be removed. There will be one old CA certificate for foundation A and one old certificate for foundation B.

Follow these steps twice, once for foundation A, and once for foundation B.

1. Delete the two old CA certificates from the BOSH Director tile **Security > Trusted Certificates** field. You will remove one old CA certificate for foundation A and one old certificate for foundation B. Click **Save**.
2. Delete the two old CA certificates from the two fields of the VMware Tanzu Application Service tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. You will remove one old CA certificate for foundation A and one old certificate for foundation B. Click **Save**.
3. Delete the two old CA certificates from the two fields of the Isolation Segment tile, if the environment has this optional tile: **Networking > Certificate Authorities trusted by the Gorouter** and **Networking > Certificate Authorities trusted by the HAProxy**. You will remove one old CA certificate for foundation A and one old certificate for foundation B. Click **Save**.

Apply Changes for the Third Time

As a security best practice, you should remove outdated certificates as soon as possible from your deployment. You can schedule this step to a convenient time, because for most deployments you will not lose any deployment functionality if you do not perform this step immediately.

For some deployments, you may encounter an error if you create a service instance in a deployment that contains an expired Services TLS CA certificate. For more information, see Cloud Controller fails to create service instance when there is a expired cert installed on the system in the Tanzu Support Knowledge Base.

> **Warning**: This procedure involves redeploying all of the VMs in your Ops Manager deployment to apply a CA certificate. The operation can take a long time to complete.

You may apply the changes to both foundation A and foundation B at the same time.

Follow these steps twice, once to apply the changes to foundation A, and once to apply the changes to foundation B.

1. Navigate back to the **Installation Dashboard**.
2. Click **Review Pending Changes**.
3. Ensure that all product tiles, including TAS for VMs, TAS for VMs [Windows], Isolation Segment, and partner tiles, are selected.
4. Select only the on-demand services tiles that use TLS, such as MySQL for VMware Tanzu, VMware GemFire for TAS, Redis for VMware Tanzu, and RabbitMQ for VMware Tanzu [VMs].
5. For each on-demand service tile that uses TLS:
    1. Expand the errands.
    2. Enable the errand to upgrade all service instances.

    > **Note:** The name of the upgrade all service instances errand may differ slightly between services.

6. Click **Apply Changes**.

Restoring a WAN Connection

This topic describes the steps required to restore a WAN connection when using VMware GemFire for TAS.

This sequence of steps restores a bidirectional WAN connection to a service instance when the service instance has been restored from backup.

In this description, cluster A is an existing, running service instance that was formerly connected to cluster B over a WAN connection, and cluster B is the service that has been freshly restored from backup, and needs to be reconnected.

At this point in the process, cluster B should have the following characteristics:

- All data restored from backup
- The same `distributed_system_id` as its predecessor
- A freshly-generated service key

If the restored `distributed_system_id` matches that in the backup, then there is no need to recreate gateway senders and receivers; they are already defined. This procedure manually updates gateway senders with pointers to remote locators and credentials, re-enabling their ability to connect across the WAN without authentication errors.

1. Obtain the service key for cluster A. The service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster B in this example) to communicate with Cluster A:

    ```
    $ cf service-key A k1
    ```

The contents of the service key differ based upon the cluster configuration, such as whether an external authentication such as LDAP via User Account and Authentication (UAA) has been configured. Here is sample output from `cf service-key A k1`:

```
Getting key k1 for service instance A as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {}
}
```

2. Obtain the service key of cluster B:

   ```
   $ cf service-key B k2
   ```

   As above, the service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster A in this example) to communicate with Cluster B. Here is sample output from `cf service-key B k2`:

   ```
   Getting key k2 for service instance destination as admin...

   {
    "distributed_system_id": "2",
    "gfsh_login_string": "connect
    --url=https://cloudcache-url.com/gemfire/v1
    --user=cluster_operator_user --password=pass --skip-ssl-validation",
    "locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "remote_cluster_info": {
     "recursors": {
      "services-subnet-2.service-instance-id-2.bosh": [
       "10.1.16.7:1053",
       "10.1.16.6:1053",
       "10.1.16.8:1053"
      ]
     },
     "remote_locators": [
      "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
      "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
      "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
     ],
     "trusted_sender_credentials": [
      {
       "password": "gws-PQR-password",
       "username": "gateway_sender_PQR"
      }
     ]
    },
    "urls": {
     "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
     "pulse": "https://cloudcache-2.example.com/pulse"
    },
    "users": [
     {
      "password": "cl-op-JKL-password",
      "roles": [
       "cluster_operator"
      ],
      "username": "cluster_operator_JKL"
     },
     {
      "password": "dev-MNO-password",
      "roles": [
       "developer"
      ],
   ```

```
   "username": "developer_MNO"
  }
 ],
 "wan": {}
}
```

3. Update the Cluster A service instance, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster B service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster A to communicate with Cluster B, and to accept data from Cluster B:

```
$ cf update-service A -c '
{
  "remote_clusters":[
  {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-PQR-password",
      "username": "gateway_sender_PQR"
     }
    ]
   }
  }
 ]
}'
Updating service instance A as admin
```

4. To verify that a service instance has been correctly updated, delete and recreate the cluster service key. The recreated service key will have the same user identifiers and passwords as its predecessor, and will reflect the changes you specified in the recent `cf update-service` commands. In particular, the `wan{}` element at the end of a cluster's service key should be populated with the other cluster's remote connection information. For example, to verify that the Cluster A service key was updated correctly, log in as Cluster A administrator and issue these commands to delete and recreate the Cluster A service key:

```
$ cf delete-service-key A k1
  ...
$ cf create-service-key A k1
```

Verify that the `wan{}` field of the Cluster A service key contains a `remote_clusters` element which specifies contact information for Cluster B, including Cluster B's `recursors` array, `remote_locators` array, and `trusted_sender_credentials`:

```
Getting key k1 for service instance A as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-PQR-password",
      "username": "gateway_sender_PQR"
     }
    ]
   }
  ]
 }
}
```

5. Update the Cluster B service instance, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster A service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster B to communicate with Cluster A, and to accept data from Cluster A:

```
$ cf update-service B -c '
{
  "remote_clusters":[
  {
    "recursors": {
      "services-subnet.service-instance-id.bosh": [
        "10.0.8.5:1053",
        "10.0.8.7:1053",
        "10.0.8.6:1053"
      ]
    }
    "remote_locators":[
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
      "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    "trusted_sender_credentials":[
    {
      "username": "gateway_sender_GHI",
      "password":"gws-GHI-password"
    }]
  }]
}'
Updating service instance B as admin
```

6. To verify that the Cluster B service key was updated correctly, log in as Cluster B administrator and issue these commands to delete and recreate the Cluster B service key:

```
$ cf delete-service-key B k2
  ...
$ cf create-service-key B k2
```

Verify that the `wan{}` field of the Cluster B service key contains a `remote_clusters` element which specifies contact information for Cluster A, including Cluster A's `recursors` array, `remote_locators` array, and `trusted_sender_credentials`:

```
Getting key k1 for service instance B as admin...

{
 "distributed_system_id": "2",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-2.service-instance-id-2.bosh": [
    "10.1.16.7:1053",
    "10.1.16.6:1053",
    "10.1.16.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-PQR-password",
    "username": "gateway_sender_PQR"
   }
  ]
 },
 "urls": {
  "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
  "pulse": "https://cloudcache-2.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-JKL-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_JKL"
  },
  {
   "password": "dev-MNO-password",
   "roles": [
    "developer"
   ],
   "username": "developer_MNO"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "recursors": {
     "services-subnet.service-instance-id.bosh": [
      "10.0.8.6:1053",
      "10.0.8.7:1053",
      "10.0.8.5:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet.service-instance-id.bosh[55221]",
     "id2.locator.services-subnet.service-instance-id.bosh[55221]",
     "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-GHI-password",
      "username": "gateway_sender_GHI"
     }
    ]
   }
  ]
 }
}
```

7. To verify that the connection has been successfully restored, see Verify Bidirectional WAN Setup.

Working with Service Instances

This document describes how to choose a service plan, create and delete VMware GemFire for Tanzu Application Service service instances, and bind an app.

You must install the Cloud Foundry Command Line Interface (cf CLI) to run the commands in this topic. For more information, see Installing the cf CL in the VMware Tanzu Application Service for VMs product documentation.

In this topic:

View Available Plans

This topic explains how to view the plans available for VMware GemFire for TAS.

Run `cf marketplace` to view the installed VMware GemFire for Tanzu Application Service tile version and its associated VMware GemFire version. In this example, only a single plan is available:

```
% cf marketplace
Getting services from marketplace in org system / space system
 as admin...

service       plans     description
                  broker
p-cloudcache  dev-plan  VMware GemFire(version "1.13.0-build.24", Apache Geode version
 "1.13.0")  cloudcache-broker
```

Run `cf marketplace -e p-cloudcache` to view all plans available for GemFire for TAS. The plan names displayed are configured by the operator during tile installation. In this example, a subset of the available plans are enabled.

```
% cf marketplace -e p-cloudcache
Getting service plan information for service p-cloudcache
 as admin...

service plan       description      free or paid
extra-small        Caching Plan 1   free
small              Caching Plan 2   free
medium             Caching Plan 3   free
dev-plan1          Single-VM Plan   free
dev-plan2          Single-VM Plan   free
small-footprint1   Multi-VM Plan    free
small-footprint2   Multi-VM Plan    free
```

Create or Delete a Service Instance

This topic explains how to create and delete VMware GemFire for TAS services instances.

Create a Service Instance

Run `cf create-service p-cloudcache PLAN-NAME SERVICE-INSTANCE-NAME` to create a service instance. Replace `PLAN-NAME` with the name from the list of available plans. Replace `SERVICE-INSTANCE-NAME` with a name of your choice. Use this name to refer to your service instance with other commands. Service instance names can include alpha-numeric characters, hyphens, and underscores.

```
$ cf create-service p-cloudcache extra-large my-cloudcache
```

Service instances are created asynchronously. Run the `cf services` command to view the current status of the service creation, and of other service instances in the current org and space:

```
$ cf services
Getting services in org my-org / space my-space as user...
OK

name            service        plan    bound apps    last operation
my-cloudcache   p-cloudcache   small                 create in progress
```

When completed, the status changes from `create in progress` to `create succeeded`.

Provide Optional Parameters

You can create a customized service instance by passing optional parameters to `cf create-service` using the `-c` flag. The `-c` flag accepts a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

The VMware GemFire for Tanzu Application Service service broker supports the following parameters:

- `tls`: A boolean, that when true, enables TLS for all communication within the cluster.
- `service_gateway`: A boolean, that when true, causes the creation of a service gateway. The service instance must also use TLS for communication within the cluster.
- `num_servers`: An integer that specifies the number of server instances in the cluster. The minimum value is `4`. The maximum and default values are configured by the operator.
- `new_size_percentage`: An integer that specifies the percentage of the heap to allocate to young generation. This value must be between `5` and `83`. By default, the new size is 2 GB or 10% of heap, whichever is smaller.
- `distributed_system_id`: An integer that provides a unique identifier for a cluster that participates in a WAN.
- `shared_write_access`: A boolean, that when true, allows an app within a different space write access to the service instance when service-instance sharing is enabled.
- `pr_load_probe_class`: A property that specifies the algorithm to be used by the server when rebalancing data. Supported values are:
    - `org.apache.geode.internal.cache.partitioned.BucketCountLoadProbe`
      Rebalance based on the `total-num-buckets` configuration and the number of members hosting the region.
    - `org.apache.geode.internal.cache.partitioned.SizedBasedLoadProbe`
      Rebalance based on the actual size of data in each member hosting the region. This

is the default rebalancing algorithm.

- advanced_configuration: A property that allows users to configure custom JVM options, GemFire properties, credentials, and Resource Manager properties. **This property is not recommended for general use.** It is intended only for advanced users and support staff. See Advanced Service Instance Configuration for details.

This example enables TLS within the cluster:

```
$ cf create-service p-cloudcache small TLS-cluster -c '{"tls": true}'
```

This example creates a service gateway that is connected to the cluster:

```
$ cf create-service p-cloudcache small TLS-cluster -c '{"tls": true, “service_gateway”
: true}'
```

This example creates the service with five service instances in the cluster:

```
$ cf create-service p-cloudcache small my-cloudcache -c '{"num_servers": 5}'
```

This example assigns the service a specific distributed system ID, which is important when restoring a WAN participant from backups:

```
$ cf create-service p-cloudcache small my-cloudcache -c '{"distributed_system_id": 2}'
```

This example creates a cluster that allows write access by an app in a different space when service-instance sharing is enabled.

```
$ cf create-service p-cloudcache small my-cloudcache -c '{"shared_write_access":true}'
```

Single-VM Plans

The Colocated Single-VM Plan is a type of service plan that is useful for development and testing. This example creates a Colocated Single-VM Plan service instance:

```
$ cf create-service p-cloudcache dev-plan my-dev-cloudcache
```

The plan provides a single locator and a single server colocated within a single VM. Because the VM is recycled when the service instance is updated or upgraded, all data within the region is lost upon update or upgrade.

When post-deploy scripts are enabled for Ops Manager, the service instance is created with a single sample region called example_partition_region. The region is of type PARTITION_REDUNDANT_HEAP_LRU, as described in Partitioned Region Types for Creating Regions on the Server.

If example_partition_region has **not** been created, it is probably because post-deploy scripts are not enabled for Ops Manager, as described in Configure a Co-located Single VM Plan.

Delete a Service Instance

You can delete service instances using the cf CLI. Before doing so, you must remove any existing service keys and app bindings.

1. Run `cf delete-service-key SERVICE-INSTANCE-NAME KEY-NAME` to delete the service key.
2. Run `cf unbind-service APP-NAME SERVICE-INSTANCE-NAME` to unbind your app from the service instance.
3. Run `cf delete-service SERVICE-INSTANCE-NAME` to delete the service instance.

```
$ cf delete-service-key my-cloudcache my-service-key
$ cf unbind-service my-app my-cloudcache
$ cf delete-service my-cloudcache
```

Deletions are asynchronous. Run `cf services` to view the current status of the service instance deletion.

Upgrade a Single Service Instance

This topic describes how developers can upgrade a VMware GemFire for TAS service instance.

If individual service instance upgrades are permitted by the operator, and a newer tile has been made available, then developers may upgrade their own service instances by following the procedure here. Enable Individual Service Instance Upgrades describes how an operator enables developer upgrades of their own service instances.

The cf CLI, v6.46.0 or a more recent version, is a prerequisite for upgrading a VMware GemFire for Tanzu Application Service service instance.

To upgrade a single GemFire for TAS service instance:

1. Confirm that an upgrade is available for the service instance; the upgrade is available when the `upgrade available` column of the output says `yes`, as in the sample output:

```
$ cf services
Getting services in org system / space system as admin...

name   service     plan            last operation   broker            upgrade availab
le
testSI p-cloudcache small-footprint create succeeded cloudcache-broker yes
```

1. Upgrade the service instance with a command of the form:

```
cf update-service SERVICE-INSTANCE-NAME --upgrade
```

Replace `SERVICE-INSTANCE-NAME` with the name of the instance to be upgraded. Confirm that you want to update when prompted.

Updating a VMware GemFire for TAS Service Instance

This topic explains how to update a VMware GemFire for TAS services instance.

You can apply all optional parameters to an existing service instance using the `cf update-service` command. You can, for example, scale up a cluster by increasing the number of servers.

Previously specified optional parameters are persisted through subsequent updates. To return the service instance to default values, you must explicitly specify the defaults as optional parameters.

For example, if you create a service instance with five servers using a plan that has a default value of four servers:

```
$ cf create-service p-cloudcache small my-cloudcache -c '{"num_servers": 5}'
```

And you set the `new_size_percentage` to 50%:

```
$ cf update-service my-cloudcache -c '{"new_size_percentage": 50}'
```

Then the resulting service instance has `5` servers and `new_size_percentage` of 50% of heap.

Rebalancing a Cluster

When updating a cluster to increase the number of servers, the available heap size is increased. When this happens, VMware GemFire for Tanzu Application Service automatically rebalances data in the cache to distribute data across the cluster.

This automatic rebalancing does not occur when a server leaves the cluster and later rejoins, for example when a VM is re-created, or network connectivity is lost and restored. In this case, you must manually rebalance the cluster using the gfsh rebalance command while connected to the cluster under a role that can manage a cluster’s data.

Note: You must first connect with gfsh over HTTPS before you can use the `rebalance` command. For more information, see Connect with gfsh over HTTPS in *Accessing a Service Instance*.

By default, the server rebalances based on the size of data in each member hosting the region. You can change the algorithm to specify rebalancing based, instead, on bucket count, by specifying the `pr_load_probe_class` parameter in the `cf create-service` command. Be aware that changing the rebalancing algorithm is a data-intensive operation and can affect performance while the change is taking place.

Restarting a Cluster

Restarting a cluster stops and restarts each cluster member in turn, issuing a rebalance as each restarted server joins the cluster.

⚠ **WARNING:** Restart of a cluster may cause data loss.

There is a potential for data loss when restarting a cluster; the region type and number of servers in the cluster determine whether or not data is lost.

- **All data is lost when restarting a cluster with these region types and number of servers:**
  - Partitioned regions without redundancy or persistence. As each server is stopped, the region entries hosted in buckets on that stopped server are permanently lost.
  - Replicated regions without persistence on a cluster that has a single server.
  - A Colocated Single-VM Plan cluster loses all data, as there is a single server and no region persistence.

- **No data is lost when restarting the cluster with these region types and number of servers:**
    - Replicated regions for clusters with more than one server.
    - Persistent regions will not lose data, as all data is on the disk and available upon restart of a server.
    - Partitioned regions with redundancy. When the server with the primary copy of an entry is stopped, the redundant copy still exists on a running server.

To restart a cluster, use the VMware Tanzu Application Service for VMs (TAS for VMs) operator credentials to run the command:

```
cf update-service SERVICE-INSTANCE-NAME -c '{"restart": true}'
```

For example:

```
$ cf update-service my-cluster -c '{"restart": true}'
```

Adding a Service Gateway

A service gateway may be added to a service instance that has TLS connections enabled. Use the command

```
cf update-service SERVICE-INSTANCE-NAME -c '{"service_gateway":true, "tls":true}
```

where `SERVICE-INSTANCE-NAME` is the name assigned to the service instance upon creation.

Changes to the Service Plan

Your TAS for VMs operator can change details of the service plan available on the Marketplace. If your operator changes the default value of one of the optional parameters, this does not affect existing service instances.

However, if your operator changes the allowed values of one of the optional parameters, existing instances that exceed the new limits are not affected, but any subsequent service updates that change the optional parameter must adhere to the new limits.

For example, if the TAS for VMs operator changes the plan by decreasing the maximum value for `num_servers`, any future service updates must adhere to the new `num_servers` value limit.

You might see the following error message when attempting to update a service instance:

```
$ cf update-service  my-cloudcache -c '{"num_servers": 5}'
Updating service instance my-cloudcache as admin...
FAILED
Server error, status code: 502, error code: 10001, message:
Service broker error: Service cannot be updated at this time,
please try again later or contact your operator for more information
```

This error message indicates that the operator has made an update to the plan used by this service instance. You must wait for the operator to apply plan changes to all service instances before you can make further service instance updates.

Changes to Advanced Configuration

Advanced configuration allows users to configure custom JVM options, GemFire properties, credentials, and Resource Manager properties. **This property is not recommended for general use.** Do not attempt to modify a service instance's advanced configuration unless you are support staff or an experienced user with knowledge of the existing configuration. See Advanced Service Instance Configuration for details.

Monitoring VMware Tanzu GemFire for Tanzu Application Service Service Instances

This topic describes options to use when running VMware GemFire for TAS.

GemFire for TAS clusters and brokers emit service metrics. You can use any tool that has a corresponding Cloud Foundry nozzle to read and monitor these metrics in real time, including Tanzu Observability (formerly known as VMware Aria Operations for Applications).

As an app developer, when you opt to use a data service, you should be prepared to:

- monitor the state of that service
- triage issues that occur with that service
- be notified of any concerns

If you believe an issue relates to the underlying infrastructure (network, CPU, memory, or disk), you will need to capture evidence and notify your platform team. The metrics described in this section can help in characterizing the performance and resource consumption of your service instance.

GemFire for TAS does not publish locator metrics for service instances created as Co-located Single VM plans or Co-located Multi VM plans.

Service Instance Metrics

In the descriptions of the metrics, KPI stands for Key Performance Indicator.

Member Count

`serviceinstance_MemberCount`

Description	Returns the number of members in the distributed system.
Metric Type	number
Suggested measurement	Every second
Measurement Type	count
Warning Threshold	less than the manifest member count

Suggested Actions	This depends on the expected member count, which is available in the BOSH manifest. If the number expected is different from the number emitted, this is a critical situation that may lead to data loss, and the reasons for node failure should be investigated by examining the service logs.
Why a KPI?	Member loss due to any reason can potentially cause data loss.

Total Available Heap Size

`serviceinstance_TotalHeapSize`

Description	Returns the total available heap, in megabytes, across all instance members.
Metric Type	number
Suggested measurement	Every second
Measurement Type	pulse
Why a KPI?	If the total heap size and used heap size are too close, the system might see thrashing due to GC activity. This increases latency.

Total Used Heap Size

`serviceinstance_UsedHeapSize`

Description	Returns the total heap used across all instance members, in megabytes.
Metric Type	number
Suggested measurement	Every second
Measurement Type	pulse
Why a KPI?	If the total heap size and used heap size are too close, the system might see thrashing due to GC activity. This increases latency.

Total Available Heap Size as a Percentage

`serviceinstance_UnusedHeapSizePercentage`

Description	Returns the proportion of total available heap across all instance members, expressed as a percentage.
Metric Type	percent
Suggested measurement	Every second
Measurement Type	compound metric
Warning Threshold	40%
Critical Threshold	10%

Suggested Actions	If this is a spike due to eviction catching up with insert frequency, then customers need to keep a close watch that it should not hit the RED marker. If there is no eviction, then horizontal scaling is suggested.
Why a KPI?	If the total heap size and used heap size are too close, the system might see thrashing due to GC activity. This increases latency.

Per Member Metrics

Memory Used as a Percentage

`member_UsedMemoryPercentage`

Description	RAM being consumed.
Metric Type	percent
Suggested measurement	Average over last 10 minutes
Measurement Type	average
Warning Threshold	75%
Critical Threshold	85%

Count of Java Garbage Collections

`member_GarbageCollectionCount`

Description	The number of times that garbage has been collected over a rolling window of time. The rolling window is updated and the count is recalculated every second.
Metric Type	number
Suggested measurement	Sum over last 10 minutes
Measurement Type	count
Warning Threshold	Dependent on the IaaS and app use case.
Critical Threshold	Dependent on the IaaS and app use case.
Suggested Actions	Check the number of queries run against the system, which increases the deserialization of objects and increases garbage.
Why a KPI?	If the frequency of garbage collection is high, the system might see high CPU usage, which causes delays in the cluster.

CPU Utilization Percentage

`member_HostCpuUsage`

Description	This member's process CPU utilization, expressed as a percentage.
Metric Type	percent
Suggested measurement	Average over last 10 minutes
Measurement Type	average
Warning Threshold	85%
Critical Threshold	95%
Suggested Actions	If this is not happening with high GC activity, the system is reaching its limits. Horizontal scaling might help.
Why a KPI?	High CPU usage causes delayed responses and can also make the member non-responsive. This can cause the member to be kicked out of the cluster, potentially leading to data loss.

Average Latency of Get Operations

`member_GetsAvgLatency`

Description	The average latency of cache get operations, in nanoseconds.
Metric Type	number
Suggested measurement	Average over last 10 minutes
Measurement Type	average
Warning Threshold	Dependent on the IaaS and app use case.
Critical Threshold	Dependent on the IaaS and app use case.
Suggested Actions	If this is not happening with high GC activity, the system is reaching its limit. Horizontal scaling might help.
Why a KPI?	It is a good indicator of the overall responsiveness of the system. If this number is high, the service administrator should diagnose the root cause.

Average Latency of Put Operations

`member_PutsAvgLatency`

Description	The average latency of cache put operations, in nanoseconds.
Metric Type	number
Suggested measurement	Average over last 10 minutes
Measurement Type	average
Warning Threshold	Dependent on the IaaS and app use case.
Critical Threshold	Dependent on the IaaS and app use case.

Suggested Actions	If this is not happening with high GC activity, the system is reaching its limit. Horizontal scaling might help.
Why a KPI?	It is a good indicator of the overall responsiveness of the system. If this number is high, the service administrator should diagnose the root cause.

JVM pauses

`member_JVMPauses`

Description	The quantity of JVM pauses.
Metric Type	number
Suggested measurement	Sum over 2 seconds
Measurement Type	count
Warning Threshold	Dependent on the IaaS and app use case.
Critical Threshold	Dependent on the IaaS and app use case.
Suggested Actions	Check the cached object size; if it is greater than 1 MB, you may be hitting the limitation on JVM to garbage collect this object. Otherwise, you may be hitting the utilization limit on the cluster, and will need to scale up to add more memory to the cluster.
Why a KPI?	Due to a JVM pause, the member stops responding to "are-you-alive" messages, which may cause this member to be kicked out of the cluster.

File Descriptor Limit

`member_FileDescriptorLimit`

Description	The maximum number of open file descriptors allowed for the member's host operating system.
Metric Type	number
Suggested measurement	Every second
Measurement Type	pulse
Why a KPI?	If the number of open file descriptors exceeds number available, it causes the member to stop responding and crash.

Open File Descriptors

`member_TotalFileDescriptorOpen`

Description	The current number of open file descriptors.
Metric Type	number

Suggested measurement	Every second
Measurement Type	pulse
Why a KPI?	If the number of open file descriptors exceeds number available, it causes the member to stop responding and crash.

Quantity of Remaining File Descriptors

`member_FileDescriptorRemaining`

Description	The number of available file descriptors.
Metric Type	number
Suggested measurement	Every second
Measurement Type	compound metric
Warning Threshold	1000
Critical Threshold	100
Suggested Actions	Scale horizontally to increase capacity.
Why a KPI?	If the number of open file descriptors exceeds number available, it causes the member to stop responding and crash.

Threads Waiting for a Reply

`member_ReplyWaitsInProgress`

Description	The quantity of threads currently waiting for a reply.
Metric Type	number
Suggested measurement	Average over the past 10 seconds
Measurement Type	pulse
Warning Threshold	1
Critical Threshold	10
Suggested Actions	If the value does not average to zero over the sample interval, then the member is waiting for responses from other members. There are two possible explanations: either another member is unhealthy, or the network is dropping packets. Check other members' health, and check for network issues.
Why a KPI?	Unhealthy members are excluded from the cluster, possibly leading to data loss.

Gateway Sender and Gateway Receiver Metrics

These are metrics emitted through the CF Nozzle for gateway senders and gateway receivers.

Queue Size for the Gateway Sender

	`gatewaySender_<sender-id>_EventQueueSize`
Description	The current size of the gateway sender queue.
Metric Type	number
Measurement Type	count

Events Received at the Gateway Sender

	`gatewaySender_<sender-id>_EventsReceivedRate`
Description	A count of the events coming from the region to which the gateway sender is attached. It is the count since the last time the metric was checked. The first time it is checked, the count is of the number of events since the gateway sender was created.
Metric Type	number
Measurement Type	count

Events Queued by the Gateway Sender

	`gatewaySender_<sender-id>_EventsQueuedRate`
Description	A count of the events queued on the gateway sender from the region. This quantity of events might be lower than the quantity of events received, as not all received events are queued. It is a count since the last time the metric was checked. The first time it is checked, the count is of the number of events since the gateway sender was created.
Metric Type	number
Measurement Type	count

Events Received by the Gateway Receiver

	`gatewayReceiver_EventsReceivedRate`
Description	A count of the events received from the gateway sender which will be applied to the region on the gateway receiver's site. It is the count since the last time the metric was checked. The first time it is checked, the count is of the number of events since the gateway receiver was created.
Metric Type	number
Measurement Type	count

Disk Metrics

These are metrics emitted through the CF Nozzle for disks.

Average Latency of Disk Writes

`diskstore_DiskWritesAvgLatency`

Description	The average latency of disk writes in nanoseconds.
Metric Type	number
Measurement Type	time in nanoseconds

Quantity of Bytes on Disk

	`diskstore_TotalSpace`
Description	The total number of bytes on the attached disk.
Metric Type	number
Measurement Type	count

Quantity of Available Bytes on Disk

	`diskstore_UseableSpace`
Description	The total number of bytes of available space on the attached disk.
Metric Type	number
Measurement Type	count

Experimental Metrics

These metrics are experimental. Any of these metrics may be removed without advance notice, and the current name of any of these metrics may change. Expect changes to these metrics.

These experimental metrics are specific to a region (`REGION-NAME`):

- `region.REGION-NAME.BucketCount`
- `region.REGION-NAME.CreatesRate`
- `region.REGION-NAME.DestroyRate`
- `region.REGION-NAME.EntrySize`
- `region.REGION-NAME.FullPath`
- `region.REGION-NAME.GetsRate`
- `region.REGION-NAME.NumRunningFunctions`
- `region.REGION-NAME.PrimaryBucketCount`
- `region.REGION-NAME.PutAllRate`
- `region.REGION-NAME.PutLocalRate`
- `region.REGION-NAME.PutRemoteRate`
- `region.REGION-NAME.PutsRate`
- `region.REGION-NAME.RegionType`
- `region.REGION-NAME.SystemRegionEntryCount`

- `region.REGION-NAME.TotalBucketSize`
- `region.REGION-NAME.TotalRegionCount`
- `region.REGION-NAME.TotalRegionEntryCount`

These experimental metrics are specific to a member:

- `member.AverageReads`
- `member.AverageWrites`
- `member.BytesReceivedRate`
- `member.BytesSentRate`
- `member.CacheServer`
- `member.ClientConnectionCount`
- `member.CreatesRate`
- `member.CurrentHeapSize`
- `member.DeserializationRate`
- `member.DestroysRate`
- `member.FunctionExecutionRate`
- `member.GetRequestRate`
- `member.GetsRate`
- `member.MaxMemory`
- `member.MemberUpTime`
- `member.NumThreads`
- `member.PDXDeserializationRate`
- `member.PutAllRate`
- `member.PutRequestRate`
- `member.PutsRate`
- `member.TotalBucketCount`
- `member.TotalHitCount`
- `member.TotalMissCount`
- `member.TotalPrimaryBucketCount`

Total Memory Consumption

The BOSH `mem-check` errand calculates and outputs the quantity of memory used across all GemFire for TAS service instances. This errand helps GemFire for TAS (GemFire for TAS) operators monitor resource costs, which are based on memory usage.

The output from `mem-check` includes both locators and servers.

Consumption Calculation

`mem-check` does the following to calculate and output the quantity of memory used across all GemFire for TAS service instances:

1. Uses `gfsh` to `list members` and find all strings that match `locator|cacheserver` from the output.
2. For each member, uses `gfsh` to describe `member --name=<member_name>` and match on `(Used|Max) Heap`.
3. Adds all `used` and `max` and print them for the specific deployment.
4. After checking all deployments, outputs the global total `used` and `max` for all deployments

mem-check Usage

To use `mem-check`:

1. From the director, run the BOSH command:

   ```
   bosh -d <SERVICE_BROKER_NAME> run-errand mem-check
   ```

   Where `<SERVICE_BROKER_NAME>` is the name of your service broker.

   For example:

   ```
   bosh -d cloudcache-service-broker run-errand mem-check
   ```

The following is an anonymized portion of example output from the `mem-check` errand from a two cluster deployment:

```
    Analyzing deployment xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx1...
    JVM heap usage for service instance xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx1
    Used Total = 1204 MB
    Max Total = 3201 MB

    Analyzing deployment xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx2...
    JVM heap usage for service instance xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx2
    Used Total = 986 MB
    Max Total = 3201 MB

    JVM heap usage for all clusters everywhere:
    Used Global Total = 2390 MB
    Max Global Total = 6402 MB
```

Using the VMware GemFire for Redis Apps Add-on

VMware GemFire for Tanzu Application Service supports use of the VMware GemFire for Redis Apps add-on. The GemFire for Redis add-on allows applications that currently use a Redis client to connect to and send Redis commands to GemFire. This allows those applications to use GemFire as a datastore for Redis clients, with little to no code changes.

For more information about the add-on, see the VMware GemFire for Redis Apps documentation.

Usage Notes

- VMware GemFire for Redis Apps currently implements a subset of the full set of Redis commands. For more information, see Compatible Redis Commands in the *VMware GemFire for Redis Apps* documentation.
- If your application uses Spring Session, you must add the following code to deactivate Spring Session from calling the `CONFIG` command. Calling the `CONFIG` command is not supported.

  ```
  @Bean
  public static ConfigureRedisAction configureRedisAction() {
        return ConfigureRedisAction.NO_OP;
  }
  ```

  This is a known solution for many Managed Redis products (ElastiCache, Azure Cache for Redis, etc.) that deactivate the `CONFIG` command for security reasons. For more information, see ERR unknown command 'CONFIG' when using Secured Redis #124 in GitHub.
- If your application uses redis-py-cluster, you must specify the option `skip_full_coverage_check=True` when creating the connection to the cluster. This is a known solution for many Managed Redis products (ElastiCache, Azure Cache for Redis, etc) that deactivate the `CONFIG` command for security reasons. For more information, see AWS Elasticache Restrictions #189 in GitHub.

Service Instance Configuration Parameters

To use the APIs for Redis, you must start a VMware GemFire for Tanzu Application Service service instance with GemFire for Redis Apps enabled and redundant copies created.

You can pass other paramenters to the `cf create-service` command to further configure the VMware GemFire for Tanzu Application Service service instance.

The following parameters can be passed to configure the service instance:

`gemfire_for_redis_port: N`

Where `N` is a port number (default: `6379`).
Specifies the port number where the VMware GemFire server listens for Redis commands.

`gemfire_for_redis_use_default_region_config: B`

Where `B` is a Boolean (default: `true`).
Specifies whether to use the default configuration for the GemFire for Redis Apps region. If set to false, the VMware GemFire for Redis Apps service will not be enabled and the server will not accept Redis commands until a region with the configured region name is manually created.

`gemfire_for_redis_region_name: S`

Where `S` is a string (default: `GEMFIRE_FOR_REDIS`).
Specifies the name of the GemFire for Redis Apps region.

`gemfire_for_redis_maxmemory_policy: S`

Where `S` is a string. Valid values are `noeviction` and `allkeys-lru` (default: `noeviction`).
Specifies the eviction policy of the GemFire for Redis Apps region.

`gemfire_for_redis_maxmemory_eviction_percent: N`

Where `N` is a number (default: `0`).
Specifies the maximum amount of memory, as a percentage of the total heap size, to be used on an individual server before entries will be evicted. Must be less than the value set for `gemfire_for_redis_maxmemory_deny_commands_percent.`

`gemfire_for_redis_maxmemory_deny_commands_percent: N`

Where `N`is a number (default: `0`, no maximum).
Specifies the maximum amount of memory, as a percentage of the total heap size, to be used on an individual server before commands that write data will be denied. Must be greater than the value set for `gemfire_for_redis_maxmemory_eviction_percent`.

`gemfire_for_redis_notify_keyspace_events: S`

Where `S` is a string (default: an empty string).
Specifies the types of keyspace events that the VMware GemFire for Redis Apps server will notify clients of. Each event type is represented by a single letter.
For more information about the event types, see *--J=-Dgemfire-for-redis-notify-keyspace-events* in Configuring VMware GemFire for Redis Apps in the VMware GemFire for Redis Apps product documentation.

`gemfire_for_redis_hll_sparse_max_bytes: N`

Where `N` is a number (default: `3000`).
Specifies the maximum number of bytes to use in a HyperLogLog sparse representation. When a HyperLogLog using the sparse representation crosses this limit, it is converted into the dense representation. Decreasing this value, on average, results in lower CPU usage but higher memory usage, because the dense representation is less memory efficient than the sparse representation but performs PFADD in O(1) compared to O(N) for the sparse representation.

`gemfire_for_redis_expiration_interval_seconds: N`

Where `N` is a number (default: `180`).
Specifies the interval (in seconds) at which the active expiration check is performed. Every key present in the data store is iterated over and removed if its expiration time has passed. This can be resource intensive if many keys exist. Decreasing this value results in keys being expired and removed closer to their configured expiration time but also increases overall CPU usage.

`gemfire_for_redis_write_timeout_seconds: N`

Where `N` is a number (default: `10`).
Specifies the timeout (in seconds) for write operations between the VMware GemFire for Redis Apps server and connected clients. If a write operation cannot finish before this timeout is exceeded, an error is returned to the client. While you may be able to prevent timeout exceptions by increasing this value from the default, we recommended that you address the cause of the slow writes instead.

Note: To prevent confusion with the JVM arguments of the redis extension app being passed as advanced configuration, they have been added to the disallow list. If you need to modify these arguments, you must include them when starting a VMware GemFire for Tanzu Application Service service instance.

Start a Service Instance with GemFire for Redis Apps

To use the APIs for Redis, you must start a VMware GemFire for Tanzu Application Service service instance with GemFire for Redis Apps enabled and redundant copies created as follows:

1. To access the API, open port `6379` in VMware GemFire for Tanzu Application Service. For more information, see Networking for On-Demand Services.

2. Enable APIs for Redis and add the “redis” tag to your VMware GemFire for Tanzu Application Service service.

   - When you create a new service, you must add `-c '{"gemfire_for_redis_enabled":true}', "gemfire_for_redis_redundant_copies":N`, and `-t "redis"` to the `cf create-service` command. `N` must be `1`, `2`, or `3`, and sets the number of redundant copies created. VMware GemFire for Tanzu Application Service supports a maximum of three redundant copies.

     You can pass other paramenters to the `cf create-service` command to further configure the VMware GemFire for Tanzu Application Service service instance. For a list of other parameters, see Service Instance Configuration Parameters.

     For example:

     ```
     $ cf create-service p-cloudcache PLAN-NAME YOUR-SERVICE-NAME -c '{"gemfir
     e_for_redis_enabled":true,"gemfire_for_redis_redundant_copies":2,"gemfire
     _for_redis_region_name": S}' -t redis
     ```

   - When updating an existing service, you must add `-c '{"gemfire_for_redis_enabled":true}', "gemfire_for_redis_redundant_copies":N`, and `-t "redis"` to the `cf update-service` command. `N` must be `1`, `2`, or `3`, and sets the number of redundant copies created. VMware GemFire for Tanzu Application Service supports a maximum of three redundant copies.

     You can pass other paramenters to the `cf update-service` command to further configure the VMware GemFire for Tanzu Application Service service instance. For a list of other parameters, see Service Instance Configuration Parameters.

     For example:

     ```
     $ cf update-service YOUR-SERVICE-NAME -c '{"gemfire_for_redis_enabled":tr
     ue,"gemfire_for_redis_redundant_copies":2,"gemfire_for_redis_region_name"
     : S}' -t "redis"
     ```

     Note: Updating the service causes all members of the cluster to restart.

If you migrate from an existing application that uses the Redis tile, you typically must unbind and rebind the app to the Redis service.

3. If your app is still bound to the Redis service, unbind it:

   ```
   cf unbind-service APP-NAME SERVICE-INSTANCE-NAME
   ```

4. Bind your app to the new VMware GemFire for Tanzu Application Service service:

   ```
   cf bind-service APP-NAME SERVICE-INSTANCE-NAME
   ```

(Optional) Transport Layer Security (TLS)

VMware GemFire for Tanzu Application Service allows service instances to be created with Transport Layer Security (TLS) enabled. This differs from the Redis tile which allows TLS and non-TLS ports to be active at the same time on a single service instance. TLS can only be enabled during service instance creation.

If you enable TLS, all connections must be made using a TLS-capable client. The non-TLS port will be unavailable

To enable APIs for Redis and TLS on a VMware GemFire for Tanzu Application Service service instance, add `"tls":true` to the parameters passed to the `cf create-service` command:

For example:

```
$ cf create-service p-cloudcache PLAN-NAME YOUR-SERVICE-NAME -c '{"gemfire_for_redis_e
nabled":true,"tls":true}' -t redis
```

Note: This enables TLS for the entire cluster.

App Development

To connect a Redis application to the VMware GemFire for Tanzu Application Service service, you must create a service key. For a Redis-enabled service, the following properties are added to the service key as well as to the binding environment for the bound app:

- `"uri": "redis://REDIS-USER:REDIS-PASSWORD@REDIS-HOST-NAME:6379"`
- `"hostname": "REDIS-HOST-NAME"`
- `"port": 6379`
- `"username": "REDIS-USER"`
- `"password": "REDIS-PASSWORD"`

Before you begin, note that: - Applications must be using a Redis client that supports **Redis Cluster mode**. - VMware GemFire for Redis Apps currently implements a subset of the full set of Redis commands. For more information, see Compatible Redis Commands in the *VMware GemFire for Redis Apps* documentation.

Follow the steps below to connect a Redis application to the VMware GemFire for Tanzu Application Service service.

1. Create a service key by running the following command:

   ```
   cf create-service-key MY-INSTANCE MY-KEY
   ```

2. Inspect the service key:

   ```
   cf service-key MY-INSTANCE MY-KEY
   ```

   This return a JSON response (`VCAP_SERVICES`) containing the fields shown in this example:

   ```
   {
    "uri": "redis://developer_GUID:EXAMPLE-PASSWORD@redis-endpoint.service-instance-INSTANCE-GUID.bosh:6379",
    "hostname": "redis-endpoint.service-instance-INSTANCE-GUID.bosh",
    "username": "developer_GUID",
    "password": "EXAMPLE-PASSWORD",
    "port": 6379,
    "users": [
     {
      "password": "EXAMPLE-PASSWORD",
      "roles": [
       "developer"
      ],
      "username": "developer_GUID"
     }
    ]
   }
   ```

   Note: The `username` field is the same as the username configured for the `developer` role. This role has the necessary permissions to access all GemFire for Redis Apps functionality.

   If TLS is enabled for the service, the `uri` scheme will be `rediss:`, and the `port` field will be replaced by a `tls_port` field which must be used for TLS enabled connections. Only one of `port` or `tls_port` will be present. Redis clients cannot establish both TLS and non-TLS connections to the same GemFire for Redis Apps service.

3. Configure the application.

   You can use the values from the service key to configure the application, as appropriate, to connect to the service. For example, a Spring Boot application would need the following set in its `application.properties` file:

   ```
   spring.redis.cluster.nodes=redis-endpoint.service-instance-INSTANCE-GUID.bosh:6379
   spring.redis.username=developer_GUID
   spring.redis.password=PASSWORD
   ```

Security

If the VMware GemFire for Tanzu Application Service service is configured to use external authentication (UAA), Redis applications must be configured with a user and associated password that has the `PCC_DATA-ACCESS` role as described in Configuring User Account and Authentication (UAA) Roles.

When UAA is configured, the `VCAP_SERVICES` data will not include a `username` or `password` field.

Back Up and Restore

The APIs for Redis do not persist any data to disk, so it is not possible to back up and restore.

Additional Information

For additional information, including supported commands, see the VMware GemFire for Redis Apps documentation.

Using a Service Gateway

The service gateway allows apps that run in a different location than the Services Foundation where the GemFire for TAS service instance runs to communicate with the service instance. At its core, the service gateway is a TCP router.

There are three locations where an app may run. The locations are defined in The App's Location.

Apps that run within an App Foundation and apps that run completely outside of any foundation must have a service gateway enabled and running to communicate with the GemFire for TAS service instance. To enable and run the service instance:

- The operator must Configuring a Service Gateway.
- The developer must Provide Optional Parameters when creating the GemFire for TAS service instance.
- The app must be given and use properties that authenticate and authorize Communicating with a Service Instance through the service gateway.

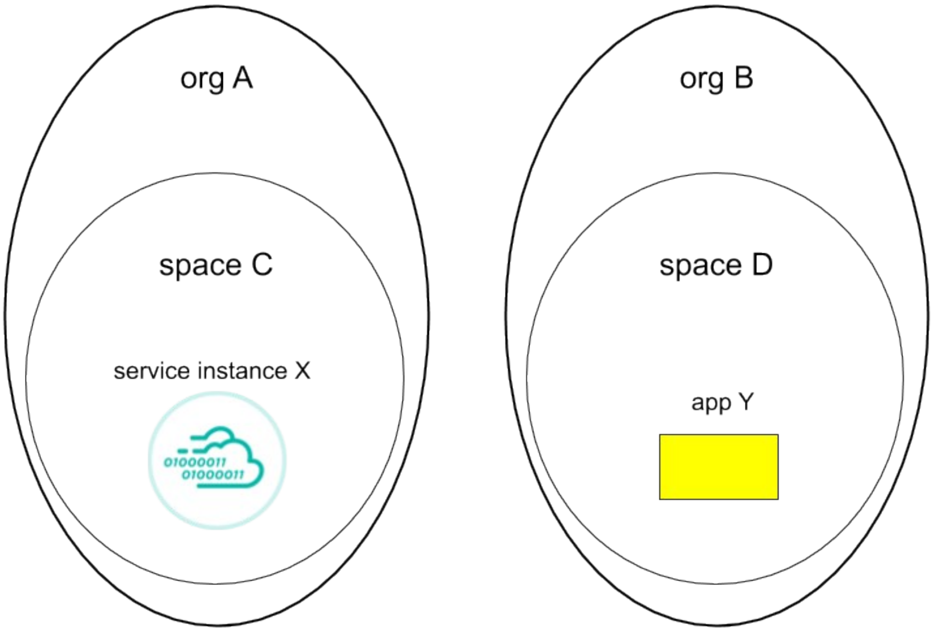
Service-Instance Sharing

Service-instance sharing for VMware GemFire for Tanzu Application Service permits access to a GemFire for TAS service instance by an app within a different space. Read-only access of the data by the app is configured by default.

Follow these steps to set up sharing:

1. A Cloud Foundry operator enables instance sharing as detailed in Enabling Service-Instance Sharing
2. (Optional) To give the app write access to the data, a developer creates the service instance with service instance sharing enabled as defined in Provide Optional Parameters.
3. A developer shares a service instance
4. A developer binds the app to the shared service instance

These instructions require identification of the org and the space of both the service instance and the app. The following diagram names the components for use in the configuration instructions. Service instance X resides within space C, which is part of org A. App Y resides within space D, which is part of org B.



Share a Service Instance

The GemFire for TAS service instance must be up and running prior to sharing.

To share the service instance:

1. An org A developer does a `cf login` with a space developer role. Target the space that contains the service to be shared: org A, space C.
2. The org A developer shares the space with a command of the form

   ```
   cf share-service SERVICE-X -s SPACE-D -o ORG-B
   ```

   Replace `SERVICE-X` with the GemFire for TAS service instance name. Replace `SPACE-D` with the space name where the app resides. Replace `ORG-B` with the org name where the app resides.

Bind an App to a Shared Service Instance

The app must be bound to the shared service instance prior to starting the app.

To bind the app to the shared service instance:

1. An org B developer does a `cf login` with a space developer role. Target the org and space that contains the app: org B, space D.
2. Verify that the GemFire for TAS service instance is available and shared across the spaces in the output of the command:

   ```
   $ cf services
   ```

3. The org B developer binds the app with a command of the form

```
cf bind APP-Y SERVICE-X
```

Replace SERVICE-X with the GemFire for TAS service instance name. Replace APP-Y with the name of the app.

App Authentication

Apps that interact with a shared GemFire for TAS service instance which resides in a different space will be given a set of credentials. The app must acquire and use this set of credentials for authentication. Apps built with Spring Boot Data GemFire version 1.1.1 or a more recent version will automatically pick up the credentials, so these apps do not need to acquire the credentials.

By default, the role of these credentials is authorized only for read access of region data. If the cluster is created with the shared_write_access parameter, the cluster operator role will be used, authorizing the operations for that role, as defined in Security.

The set of credentials are in the VCAP_SERVICES environment variable, with a role of readonly or cluster_operator_XXX. The app must parse the VCAP_SERVICES environment variable to extract the credentials. The app uses the credentials to set a property that then gets passed to the ClientCacheFactory for the purpose of authentication prior to creating the cache.

Set Up Service Instances Across a Wide Area Network (WAN)

This topic explains how to set up service instances of VMware GemFire for TAS across a Wide Area Network (WAN).

Two service instances may form a single distributed system across a WAN. The interaction of the two service instances may follow one of the patterns described in the section on Design Patterns.

Call the two service instances A and B. The cluster within each service instance uses an identifier called a distributed_system_id. This example assigns distributed_system_id = 1 to cluster A and distributed_system_id = 2 to cluster B. Cluster gateway senders provide the communication path and construct that propagates region operations from one cluster to another. On the receiving end are cluster gateway receivers. Creating a service instance also creates gateway receivers.

The procedures in this section assume that service instances communicate using TLS encryption.

Note: To set up more than two service instances across a WAN, set up the interaction between the first two service instances A and B following the directions in either Set Up a Bidirectional System or Set Up a Unidirectional System, as appropriate. After that, set up the interaction between service instance A and another service instance (called C) following the directions in either Set Up an Additional Bidirectional Interaction or Set Up an Additional Unidirectional Interaction, as appropriate.

Establishing Mutually Trusted TLS Credentials

This topic describes establishing mutually trusted TLS certificates for use with VMware GemFire for TAS.

In order for services instances in two foundations to communicate using TLS encryption, the CredHub “/services/tls_ca” certificate must be trusted in both foundations; otherwise, WAN connections with TLS will fail.

Assumptions

- You have two VMware Tanzu Application Service for VMs (TAS for VMs) foundations, A and B, with a network connection between them.
- You wish to establish a TLS-encrypted WAN connection between a service instance on Foundation A and a service instance on Foundation B.
- The Preparing for TLS procedure has been followed for each foundation, establishing a CredHub “/services/tls_ca” certificate for each.

Establish Mutual Trust

In order for services instances in two foundations to communicate with TLS, the CredHub “/services/tls_ca” certificate must be trusted in both foundations; otherwise, WAN connections requiring TLS will fail.

To be trusted in both foundations, their certificates must be signed by:

- a single CA that is identical in both foundations, or
- two CAs, one in each foundation, which is trusted by the other foundation.

If one of these conditions is satisfied, mutually trusted credentials are already in place; there is no need to implement the following procedure.

If the two foundations have different “/services/tls_ca” certificates that are not already mutually trusted, follow these steps to establish mutual trust.

Assuming you have two different TAS for VMs foundations, A and B, connected by a WAN:

1. Access the CredHub of Foundation A using instructions from Access BOSH CredHub
2. Fetch the certificate from Foundation A using CredHub:

   ```
   credhub get -n /services/tls_ca -k certificate
   ```

3. Record the output.
4. Navigate to the **Ops Manager Installation Dashboard** of **Foundation B** and click the **BOSH Director** tile.
5. Click **Security**.
6. Append the contents of the new CA certificate to the old CA certificate under **Trusted Certificates**. Do not remove the old CA certificate.
7. Click **Save**.
8. Distribute the new CA certificate to your VMware GemFire for Tanzu Application Service VMs and regenerate each server certificate using the new CA:

   - Navigate back to the **Installation Dashboard**.
   - Click **Review Pending Changes**.
   - Click **ERRANDS**.
   - Select **Upgrade All Service Instances**.

9. Return to the **Installation Dashboard** in Ops Manager and click **Apply Changes**.

Repeat Steps 2 - 9 for Foundation B to put its services CA into Foundation A.

Set Up a Bidirectional System

This section describes how to set up a bidirectional WAN connection, using TLS encryption, between two VMware GemFire for Tanzu Application Service service instances in two different foundations. A bidirectional connection like this is needed if you wish to implement an active-active pattern, as described in Bidirectional Replication Across a WAN.

Assumptions

- You have two VMware Tanzu Application Service for VMs (TAS for VMs) foundations, A and B, with a network connection between them.
- You wish to establish a TLS-encrypted WAN connection between a service instance on Foundation A and a service instance on Foundation B.
- The connection will be bidirectional, such that operations in each cluster can be replicated in the other.
- The Preparing for TLS procedure has been followed for each foundation, establishing a CredHub "/services/tls_ca" certificate for each.
- The Establishing Mutually Trusted TLS Credentials procedure has been followed, establishing mutually trusted TLS credentials between Foundations A and B.

Create Clusters

1. Log into Foundation A using Foundation A Cloud Foundry credentials.
2. Use the `cf create-service` command to create a service instance, which we will call Cluster A in this example. In the `cf create-service` command, use the `-c` option with the argument `"tls" : true` to enable TLS. This example also uses the `-c` option to set the `distributed_system_id` of Cluster A to "1":

   ```
   $ cf create-service p-cloudcache wan-cluster wan1 -c '{
   "tls" : true,
   "distributed_system_id" : 1 }'
   ```

   Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.
3. Log into Foundation B using Foundation B Cloud Foundry credentials.

4. Use the `cf create-service` command to create a service instance, which we will call Cluster B. In the `cf create-service` command, use the `-c` option with the argument `"tls" : true` to enable TLS. This example also uses the `-c` option to set the `distributed_system_id` of Cluster B to “2”:

   ```
   $ cf create-service p-cloudcache wan-cluster wan2 -c '{
   "tls" : true,
   "distributed_system_id" : 2 }'
   ```

   Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.

Create Service Keys

1. While logged in to Foundation A, create a service key for Cluster A. The contents of the service key differ based upon the cluster configuration, such as whether an external authentication such as LDAP via User Account and Authentication (UAA) has been configured.

   ```
   $ cf create-service-key wan1 k1
   Creating service key wan1 for service instance k1 ...
   OK
   ```

   The service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster B in this example) to communicate with Cluster A:

   ```
   $ cf service-key wan1 k1
   Getting key k1 for service instance wan1 ...

   {
    "distributed_system_id": "1",
    "gfsh_login_string": "connect
    --url=https://cloudcache-url.com/gemfire/v1
    --user=cluster_operator_user --password=pass --skip-ssl-validation",
    "locators": [
     "id1.locator.services-subnet.service-instance-id.bosh[55221]",
     "id2.locator.services-subnet.service-instance-id.bosh[55221]",
     "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "remote_cluster_info": {
     "recursors": {
      "services-subnet.service-instance-id.bosh": [
       "10.0.8.6:1053",
       "10.0.8.7:1053",
       "10.0.8.5:1053"
      ]
     },
     "remote_locators": [
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
      "id3.locator.services-subnet.service-instance-id.bosh[55221]"
     ],
   ```

```
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {}
}
```

2. While logged in to Foundation B, create a service key for Cluster B:

```
$ cf create-service-key wan2 k2
Creating service key wan2 for service instance k2 ...
OK
```

As above, the service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster A in this example) to communicate with Cluster B:

```
$ cf service-key wan2 k2
Getting key k2 for service instance wan2 ...

{
 "distributed_system_id": "2",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-2.service-instance-id-2.bosh": [
    "10.1.16.7:1053",
    "10.1.16.6:1053",
    "10.1.16.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-PQR-password",
    "username": "gateway_sender_PQR"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
  "pulse": "https://cloudcache-2.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-JKL-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_JKL"
  },
  {
   "password": "dev-MNO-password",
   "roles": [
    "developer"
   ],
   "username": "developer_MNO"
  }
 ],
 "wan": {}
}
```

Establish a Bidirectional Connection

To establish bidirectional communication between Clusters A and B, The `remote_clusters` element in each must be updated with the contents of the other's `remote_cluster_info`, which contains three elements:

- The `recursors` array and the `remote_locators` array enable communication between clusters.
- The `trusted_sender_credentials` element identifies a cluster from which data can be accepted for replication. For example, when Cluster B has been updated with Cluster A's `trusted_sender_credentials`, then Cluster B can accept and store data sent from Cluster A.

In order to implement a bidirectional WAN connection between two clusters, each must have the `trusted_sender_credentials` of the other.

1. While logged in to Foundation A, update the Cluster A service instance, using the `-c` option

to specify a `remote_clusters` element that includes the contents of the Cluster B service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster A to communicate with Cluster B, and to accept data from Cluster B:

```
$ cf update-service wan1 -c '
{
  "remote_clusters":[
  {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-PQR-password",
      "username": "gateway_sender_PQR"
     }
    ]
   }
 ]
}'
Updating service instance wan1 as admin
```

Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

2. While logged in to Foundation B, update the Cluster B service instance, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster A service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster B to communicate with Cluster A, and to accept data from Cluster A:

```
$ cf update-service wan2 -c '
{
  "remote_clusters": [
    {
      "recursors": {
        "services-subnet.service-instance-id.bosh": [
          "10.0.8.5:1053",
          "10.0.8.7:1053",
          "10.0.8.6:1053"
        ]
      },
      "remote_locators": [
        "id1.locator.services-subnet.service-instance-id.bosh[55221]",
        "id2.locator.services-subnet.service-instance-id.bosh[55221]",
        "id3.locator.services-subnet.service-instance-id.bosh[55221]"
      ],
      "trusted_sender_credentials": [
        {
          "username": "gateway_sender_GHI",
          "password":"gws-GHI-password"
        }
      ]
    }
  ]
}'
Updating service instance wan2 as admin
```

Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

3. To verify that a service instance has been correctly updated, delete and recreate the cluster service key. The recreated service key will have the same user identifiers and passwords as its predecessor, and will reflect the changes you specified in the recent `cf update-service` commands. In particular, the `wan{}` element at the end of a cluster's service key should be populated with the other cluster's remote connection information. For example, to verify that the Cluster A service key was updated correctly, log in as Cluster A administrator and issue these commands to delete and recreate the Cluster A service key:

```
$ cf delete-service-key wan1 k1
  ...
$ cf create-service-key wan1 k1
```

Verify that the `wan{}` field of the Cluster A service key contains a `remote_clusters` element which specifies contact information for Cluster B, including Cluster B's `remote_locators` array and `trusted_sender_credentials`:

```
$ cf service-key wan1 k1

Getting key k1 for service instance wan1 ...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_PQR"
    ]
   }
  ]
 }
}
```

Similarly, to verify that the Cluster B service key was updated correctly, log in as Cluster B administrator and issue these commands to delete and recreate the Cluster B service key:

```
$ cf delete-service-key wan2 k2
  ...
$ cf create-service-key wan2 k2
```

Verify that the `wan{}` field of the Cluster B service key contains a `remote_clusters` element which specifies contact information for Cluster A, including Cluster A's `remote_locators` array, and `trusted_sender_credentials`:

```
$ cf service-key wan2 k2

Getting key k2 for service instance wan2 ...

{
 "distributed_system_id": "2",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-2.service-instance-id-2.bosh": [
    "10.1.16.7:1053",
    "10.1.16.6:1053",
    "10.1.16.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-PQR-password",
    "username": "gateway_sender_PQR"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
  "pulse": "https://cloudcache-2.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-JKL-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_JKL"
  },
  {
   "password": "dev-MNO-password",
   "roles": [
    "developer"
   ],
   "username": "developer_MNO"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
      "remote_locators": [
       "id1.locator.services-subnet.service-instance-id.bosh[55221]",
       "id2.locator.services-subnet.service-instance-id.bosh[55221]",
       "id3.locator.services-subnet.service-instance-id.bosh[55221]"
      ],
      "trusted_sender_credentials": [
       "gateway_sender_GHI"
      ]
    }
  ]
 }
}
```

Create Gateway Senders and Regions

On each cluster, create a gateway sender that sends to the other cluster, then create regions on each cluster that have the same name and type. Note: the gateway sender must be created before the region that uses it, so there is a window of time in which region operations that occur after the region is created on Cluster A, but before the corresponding region is created on Cluster B, will be lost.

1. While logged in to Foundation A, use gfsh to create the Cluster A gateway sender and the region.

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the Cluster A gateway sender. The required `remote-distributed-system-id` option identifies the `distributed-system-id` of the destination cluster. It is 2 for this example:

      ```
      gfsh>create gateway-sender --id=send_to_2 --remote-distributed-system-id=
      2 --enable-persistence=true
      ```

    - Create the Cluster A region. The `gateway-sender-id` associates region operations with a specific gateway sender. The region must have an associated gateway sender in order to propagate region events across the WAN.

      ```
      gfsh>create region --name=regionX --gateway-sender-id=send_to_2 --type=PA
      RTITION_REDUNDANT
      ```

2. While logged in to Foundation B, use gfsh to create the Cluster B gateway sender and region.

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the Cluster B gateway sender:

      ```
      gfsh>create gateway-sender --id=send_to_1 --remote-distributed-system-id=
      1 --enable-persistence=true
      ```

    - Create the Cluster B region:

```
gfsh>create region --name=regionX --gateway-sender-id=send_to_1 --type=PA
RTITION_REDUNDANT
```

Verify Bidirectional WAN Setup

This verification uses gfsh to put a region entry into each of Cluster A and Cluster B, in order to observe that the entry appears in the other cluster.

1. While logged in to Foundation A, run gfsh and connect following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.
2. Verify that Cluster A has a complete set of gateway senders and gateway receivers:

   ```
   gfsh>list gateways
   ```

   Also verify that there are no queued events in the gateway senders.
3. Log in to Foundation B in a separate terminal window, then run gfsh and connect following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.
4. Verify that Cluster B has a complete set of gateway senders and gateway receivers:

   ```
   gfsh>list gateways
   ```

   Also verify that there are no queued events in the gateway senders.
5. From the Cluster A gfsh connection, use gfsh to perform a `put` operation. This example assumes that both the key and the value for the entry are strings. The gfsh `help put` command shows the put command’s options. Here is an example:

   ```
   gfsh>put --region=regionX --key=mykey1 --value=stringvalue1
   Result      : true
   Key Class   : java.lang.String
   Key         : mykey1
   Value Class : java.lang.String
   Old Value   : null
   ```
6. Wait approximately 30 seconds, then use the Cluster B gfsh connection to perform a `get` of the same key that was put into the region on Cluster A.

   ```
   gfsh>get --region=regionX --key=mykey1
   Result      : true
   Key Class   : java.lang.String
   Key         : mykey1
   Value Class : java.lang.String
   Value       : "stringvalue1"
   ```

   You should see that Cluster B has the value.
7. Repeat steps 5 and 6 to perform a `put` operation on Cluster B and verify that the entry is sent to Cluster A.
8. Remove both test entries from both clusters with two gfsh commands, issuing the commands on one of the clusters. WAN replication will cause the removal of the test entries on the

other cluster.

```
gfsh>remove --region=regionX --key=mykey1
Result    : true
Key Class : java.lang.String
Key       : mykey1

gfsh>remove --region=regionX --key=mykey2
Result    : true
Key Class : java.lang.String
Key       : mykey2
```

Set Up a Unidirectional System with TLS

This topic describes how to set up a unidirectional WAN connection, using TLS encryption, between two VMware GemFire for Tanzu Application Service instances in two different foundations, such that all operations in Cluster A are replicated in Cluster B. Two design patterns that use unidirectional replication are described in Blue-Green Disaster Recovery and CQRS Pattern Across a WAN.

Assumptions

- You have two VMware Tanzu Application Service for VMs (TAS for VMs) foundations, A and B, with a network connection between them.
- You wish to establish a TLS-encrypted WAN connection between a service instance on Foundation A and a service instance on Foundation B.
- The connection will be unidirectional, such that all operations in Cluster A are replicated in Cluster B, but Cluster B does not send operations to Cluster A.
- The Preparing for TLS procedure has been followed for each foundation, establishing a CredHub “/services/tls_ca” certificate for each.
- The Establishing Mutually Trusted TLS Credentials procedure has been followed, establishing mutually trusted TLS credentials between Foundations A and B.

Create Clusters

1. Log into Foundation A using Foundation A Cloud Foundry credentials.
2. Use the `cf create-service` command to create a service instance, which we will call Cluster A in this example. In the `cf create-service` command, use the `-c` option with the argument `"tls" : true` to enable TLS. This example also uses the `-c` option to set the `distributed_system_id` of Cluster A to “1”:

   ```
   $ cf create-service p-cloudcache wan-cluster wan1 -c '{
   "tls" : true,
   "distributed_system_id" : 1 }'
   ```

   Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.

3. Log into Foundation B using Foundation B Cloud Foundry credentials.

4. Use the `cf create-service` command to create a service instance, which we will call Cluster B. In the `cf create-service` command, use the `-c` option with the argument `"tls" : true` to enable TLS. This example also uses the `-c` option to set the `distributed_system_id` of Cluster B to “2”:

   ```
   $ cf create-service p-cloudcache wan-cluster wan2 -c '{
   "tls" : true,
   "distributed_system_id" : 2 }'
   ```

   Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.

Create Service Keys

1. While logged in to Foundation A, create a service key for Cluster A. The contents of the service key differ based upon the cluster configuration, such as whether an external authentication such as LDAP via User Account and Authentication (UAA) has been configured.

   ```
   $ cf create-service-key wan1 k1
   Creating service key wan1 for service instance k1 ...
   OK
   ```

   The service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster B in this example) to communicate with Cluster A:

   ```
   $ cf service-key wan1 k1
   Getting key k1 for service instance wan1 as admin...

   {
    "distributed_system_id": "1",
    "gfsh_login_string": "connect
    --url=https://cloudcache-url.com/gemfire/v1
    --user=cluster_operator_user --password=pass --skip-ssl-validation",
    "locators": [
     "id1.locator.services-subnet.service-instance-id.bosh[55221]",
     "id2.locator.services-subnet.service-instance-id.bosh[55221]",
     "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "remote_cluster_info": {
     "recursors": {
      "services-subnet.service-instance-id.bosh": [
       "10.0.8.6:1053",
       "10.0.8.7:1053",
       "10.0.8.5:1053"
      ]
     },
     "remote_locators": [
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   ```

```
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {}
}
```

2. While logged in to Foundation B, create a service key for Cluster B:

```
$ cf create-service-key wan2 k2
Creating service key wan2 for service instance k2 ...
OK
```

As above, the service key contains generated credentials, in a JSON element called `remote_cluster_info`, that enable other clusters (Cluster A in this example) to communicate with Cluster B:

```
$ cf service-key wan2 k2
Getting key k2 for service instance destination as admin...

{
 "distributed_system_id": "2",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-2.service-instance-id-2.bosh": [
    "10.1.16.7:1053",
    "10.1.16.6:1053",
    "10.1.16.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-PQR-password",
    "username": "gateway_sender_PQR"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
  "pulse": "https://cloudcache-2.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-JKL-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_JKL"
  },
  {
   "password": "dev-MNO-password",
   "roles": [
    "developer"
   ],
   "username": "developer_MNO"
  }
 ],
 "wan": {}
}
```

Establish a Unidirectional TLS Connection

To establish unidirectional communication between Clusters A and B, The `remote_clusters` element in each must be updated with selected contents of the other's `remote_cluster_info`, which contains three elements:

- The `recursors` array and the `remote_locators` array enable communication between clusters.
- The `trusted_sender_credentials` element identifies a cluster from which data can be accepted for replication. For example, when Cluster B has been updated with Cluster A's `trusted_sender_credentials`, then Cluster B can accept and store data sent from Cluster A.

In order to implement a unidirectional WAN connection in which Cluster A sends data to Cluster B:

- Each cluster must be updated with the `recursors` array and the `remote_locators` array of the other.
- Cluster B must be updated with the `trusted_sender_credentials` of Cluster A.
- Cluster A must *NOT* be updated with the `trusted_sender_credentials` of Cluster B.

Follow these steps to configure a unidirectional connection in which Cluster A is the active member, and Cluster B accepts data for replication, but does not send operations to Cluster A:

1. While logged into Foundation A, update the Cluster A service instance, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster B service key `remote_cluster_info` element, including Cluster B’s `recursors` array and the `remote_locators` array, but **NOT** Cluster B’s `trusted_sender_credentials`. This allows Cluster A to send messages to Cluster B, but blocks data flow from Cluster B to Cluster A.

```
$ cf update-service wan1 -c '
{
  "remote_clusters": [
    {
      "recursors": {
        "services-subnet-2.service-instance-id-2.bosh": [
          "10.1.16.7:1053",
          "10.1.16.6:1053",
          "10.1.16.8:1053"
        ]
      },
      "remote_locators": [
        "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
        "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
        "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
      ]
    }
  ]
}'
Updating service instance wan1 as admin
```

   Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

2. While logged in to Foundation B, update the Cluster B service instance, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster A service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster B to communicate with Cluster A, and to accept data from Cluster A:

```
$ cf update-service wan2 -c '
{
  "remote_clusters": [
    {
      "recursors": {
        "services-subnet.service-instance-id.bosh": [
          "10.0.8.5:1053",
          "10.0.8.7:1053",
          "10.0.8.6:1053"
        ]
      },
      "remote_locators":[
        "id1.locator.services-subnet.service-instance-id.bosh[55221]",
        "id2.locator.services-subnet.service-instance-id.bosh[55221]",
        "id3.locator.services-subnet.service-instance-id.bosh[55221]"
      ],
      "trusted_sender_credentials":[
        {
          "username": "gateway_sender_GHI",
          "password":"gws-GHI-password"
        }
      ]
    }
  ]
}'
Updating service instance wan2 as admin
```

Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

3. To verify that each service instance has been correctly updated, delete and recreate the cluster service key. The recreated service key will have the same user identifiers and passwords as its predecessor, and will reflect the changes you specified in the recent `cf update-service` commands. In particular the `wan{}` elements at the end of each cluster's service key should be populated with the other cluster's remote connection information. For example, to verify that the Cluster A service key was updated correctly, log in as Cluster A administrator and issue these commands to delete and recreate the Cluster A service key:

```
$ cf delete-service-key wan1 k1
  ...
$ cf create-service-key wan1 k1
```

Verify that the `wan{}` field of the Cluster A service key contains a `remote_clusters` element which specifies contact information for Cluster B, including Cluster B's `remote_locators` array, but **NOT** Cluster B's `trusted_sender_credentials`:

```
$ cf service-key wan1 k1
Getting key k1 for service instance wan1 ...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ]
   }
  ]
 }
}
```

Similarly, to verify that the Cluster B service key was updated correctly, log in as Cluster B administrator and issue these commands to delete and recreate the Cluster B service key:

```
$ cf delete-service-key wan2 k2
  ...
$ cf create-service-key wan2 k2
```

Verify that the `wan{}` field of the Cluster B service key contains a `remote_clusters` element which specifies contact information for Cluster A, including Cluster A’s `remote_locators` array and `trusted_sender_credentials`:

```
$ cf service-key wan2 k2
Getting key k2 for service instance wan2 ...

{
 "distributed_system_id": "2",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
  "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-2.service-instance-id-2.bosh": [
    "10.1.16.7:1053",
    "10.1.16.6:1053",
    "10.1.16.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
   "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-PQR-password",
    "username": "gateway_sender_PQR"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-2.example.com/gemfire/v1",
  "pulse": "https://cloudcache-2.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-JKL-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_JKL"
  },
  {
   "password": "dev-MNO-password",
   "roles": [
    "developer"
   ],
   "username": "developer_MNO"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet.service-instance-id.bosh[55221]",
     "id2.locator.services-subnet.service-instance-id.bosh[55221]",
     "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_GHI"
    ]
   }
  ]
 }
}
```

Create a Gateway Sender and Regions

Create a gateway sender on the active cluster that sends to the passive cluster, then create regions on each cluster that have the same name and type. Note: the gateway sender must be created before the region that uses it, so there is a window in which region operations that occur after the region is created on Cluster A, but before the corresponding region is created on Cluster B, will be lost.

1. While logged in to Foundation A, use gfsh to create the Cluster A gateway sender and the region.
   - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the Cluster and the data.
   - Create the Cluster A gateway sender. The required `remote-distributed-system-id` option identifies the `distributed-system-id` of the destination cluster. It is 2 for this example:

     ```
     gfsh>create gateway-sender --id=send_to_2 --remote-distributed-system-id=
     2 --enable-persistence=true
     ```

   - Create the Cluster A region. The `gateway-sender-id` associates region operations with a specific gateway sender. The region must have an associated gateway sender in order to propagate region events across the WAN.

     ```
     gfsh>create region --name=regionX --gateway-sender-id=send_to_2 --type=PA
     RTITION_REDUNDANT
     ```

2. While logged in to Foundation B, use gfsh to create the Cluster B region. (Because Cluster B is the passive member in this unidirectional connection, it does not need a gateway sender.)
   - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.
   - Create the Cluster B region:

     ```
     gfsh>create region --name=regionX --type=PARTITION_REDUNDANT
     ```

Verify Unidirectional WAN Setup

This verification uses gfsh to put a region entry into Cluster A, in order to observe that the entry appears in Cluster B.

1. While logged in to Foundation A, run gfsh and connect following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.
2. Verify that Cluster A has a complete set of gateway senders and gateway receivers:

   ```
   gfsh>list gateways
   ```

   Also verify that there are no queued events in the gateway senders.
3. Log in to Foundation B in a separate terminal window, then run gfsh and connect following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.
4. Verify that Cluster B has a complete set of gateway receivers (Cluster B should have no gateway senders):

   ```
   gfsh>list gateways
   ```

5. From the Cluster A gfsh connection, use gfsh to perform a `put` operation. This example assumes that both the key and the value for the entry are strings. The gfsh `help put` command shows the put command's options. Here is an example:

   ```
   gfsh>put --region=regionX --key=mykey1 --value=stringvalue1
   Result      : true
   Key Class   : java.lang.String
   Key         : mykey1
   Value Class : java.lang.String
   Old Value   : null
   ```

6. Wait approximately 30 seconds, then use the Cluster B gfsh connection to perform a `get` of the same key that was put into the region on Cluster A.

   ```
   gfsh>get --region=regionX --key=mykey1
   Result      : true
   Key Class   : java.lang.String
   Key         : mykey1
   Value Class : java.lang.String
   Value       : "stringvalue1"
   ```

   You should see that Cluster B has the value.
7. From the Cluster A gfsh connection, remove the test entry from Cluster A. WAN replication will cause the removal of the test entry from Cluster B.

   ```
   gfsh>remove --region=regionX --key=mykey1
   Result    : true
   Key Class : java.lang.String
   Key       : mykey1
   ```

Set Up an Additional Bidirectional Interaction

This topic describes how to add a bidirectional TLS-encrypted WAN connection to an additional VMware GemFire for Tanzu Application Service service instance, once an initial setup is in place for a first pair of GemFire for TAS service instances.

Call the first pair of GemFire for TAS service instances Cluster A and Cluster B, which reside on Foundation A and Foundation B, respectively. This set of directions sets up an interaction between Cluster A and and a third service instance, Cluster C, which resides on third foundation, Foundation C.

Assumptions

- You have already established a TLS-encrypted WAN connection between a service instance on Foundation A, referred to here as Cluster A, and a service instance on Foundation B, which we'll call Cluster B, following the Set Up a Bidirectional System procedure.
- You wish to add an additional TLS-encrypted WAN connection between Cluster A and a third service instance on Foundation C, which we will call Cluster C.
- The connection will be bidirectional, such that an operation on either cluster can be replicated in the other.
- The Preparing for TLS procedure has been followed for Foundation C, establishing a CredHub "/services/tls_ca" certificate.
- The Establishing Mutually Trusted TLS Credentials procedure has been followed, establishing mutually trusted TLS credentials between Foundations A and C.

Get Cluster A's Remote Credentials

1. Log into Foundation A using Foundation A Cloud Foundry credentials.
2. View the Cluster A service key and save a copy of the `remote_cluster_info` element, which includes the `remote_locators` array and `trusted_sender_credentials`. These are the credentials that enable other clusters to communicate with Cluster A.

```
$ cf service-key wan1 k1
Getting key k1 for service instance wan1 as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_PQR"
    ]
   }
  ]
 }
}
```

Create the New Cluster

The cluster within each service instance uses an identifier called a `distributed_system_id`. This example assumes the assignment of `distributed_system_id = 1` for cluster A, `distributed_system_id = 2` for cluster B, and `distributed_system_id = 3` for cluster C.

1. Log into Foundation C using Foundation C Cloud Foundry credentials.

2. Use the `cf create-service` command to create a service instance, which we will call Cluster C in this example. In the `cf create-service` command, use the `-c` option to specify the following arguments:

   - `"tls" : true` to enable TLS
   - `"distributed_system_id" : 3` to set the distributed system id of Cluster C to “3”
   - `remote_clusters" : [CLUSTER-A-CREDENTIALS]` to enable communication with Cluster A, where CLUSTER-A-CREDENTIALS are the contents of Cluster A’s `remote_cluster_info` element.

   The following command specifies the Cluster A credentials you copied in an earlier step, enabling Cluster C to communicate with and receive data from Cluster A:

   ```
   $ cf create-service p-cloudcache wan-cluster wan3 -c '{
   "tls" : true,
   "distributed_system_id" : 3,
     "remote_clusters":[
      {
       "recursors": {
         "services-subnet.service-instance-id.bosh": [
           "10.0.8.5:1053",
           "10.0.8.7:1053",
           "10.0.8.6:1053"
         ]
       },
       "remote_locators":[
         "id1.locator.services-subnet.service-instance-id.bosh[55221]",
         "id2.locator.services-subnet.service-instance-id.bosh[55221]",
         "id3.locator.services-subnet.service-instance-id.bosh[55221]"
       ],
       "trusted_sender_credentials":[
       {
         "username": "gateway_sender_GHI",
         "password":"gws-GHI-password"
       }]
     }]
   }'
   ```

   Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.

3. While logged in to Foundation C, create a service key for Cluster C.

   ```
   $ cf create-service-key wan3 k3
   Creating service key wan3 for service instance k3 ...
   OK
   ```

   Display the service key and save a copy of the `remote_cluster_info` element, which includes the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. These are the credentials that enable other clusters to communicate with Cluster C.

   ```
   $ cf service-key wan3 k3
   ```

```
Getting key k3 for service instance destination as admin...

{
 "distributed_system_id": "3",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
  "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
  "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-3.service-instance-id-3.bosh": [
    "10.2.32.7:1053",
    "10.2.32.6:1053",
    "10.2.32.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
   "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
   "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "username": "gateway_sender_XYZ",
    "password": "gws-XYZ-password"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-3.example.com/gemfire/v1",
  "pulse": "https://cloudcache-3.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-STU-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_STU"
  },
  {
   "password": "dev-VWX-password",
   "roles": [
    "developer"
   ],
   "username": "developer_VWX"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
      "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "trusted_sender_credentials": [
      "gateway_sender_GHI"
    ]
   }
  ]
 }
}
```

4. While logged in to Foundation A, update the Cluster A service instance to add the Cluster C credentials, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster C service key `remote_cluster_info` element, including the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. This allows Cluster A to communicate with Cluster C, and to accept data from Cluster C.

   IMPORTANT: You must also retain the credentials for all clusters with which Cluster A interacts, in this example those of Cluster B. The existing credentials can be copied from the `remote_cluster_info` entry in Cluster B's service key, before updating the Cluster A service instance to append credentials for the additional cluster.

   The following example adds Cluster C's credentials to Cluster A's `remote_clusters` element:

```
$ cf update-service wan1 -c '
{
  "remote_clusters":[
   {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "username": "gateway_sender_PQR",
      "password": "gws-PQR-password"
     }
    ]
   },
   {
    "recursors": {
     "services-subnet-3.service-instance-id-3.bosh": [
      "10.2.32.7:1053",
      "10.2.32.6:1053",
      "10.2.32.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "username": "gateway_sender_XYZ",
      "password": "gws-XYZ-password"
     }
    ]
   }
 ]
}'
Updating service instance wan1 as admin
```

Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

5. To verify that a service instance has been correctly updated, delete and recreate the cluster service key. The recreated service key will have the same user identifiers and passwords as its predecessor, and will reflect the changes you specified in the recent `cf update-service` commands. In particular, the `wan{}` element at the end of a cluster's service key should be populated with the other cluster's remote connection information. For example, to verify that the Cluster A service key was updated correctly, log in as Cluster A administrator and issue these commands to delete and recreate the Cluster A service key:

```
$ cf delete-service-key wan1 k1
  ...
$ cf create-service-key wan1 k1
```

Verify that the `wan{}` field of the Cluster A service key contains a `remote_clusters` element which specifies contact information for Cluster B, including Cluster B's `remote_locators` array and `trusted_sender_credentials`:

```
$ cf service-key wan1 k1

Getting key k1 for service instance wan1 as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters":[
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_PQR"
    ]
   },
   {
    "remote_locators": [
     "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_XYZ"
    ]
   }
  ]
 }
}
```

Create Gateway Senders and Regions

1. While logged in to Foundation A, use gfsh to create the cluster A gateway sender and alter the existing region.

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the cluster A gateway sender. The required `remote-distributed-system-id` option identifies the `distributed-system-id` of the destination cluster. It is 3 for this example:

      ```
      gfsh>create gateway-sender --id=send_to_3 --remote-distributed-system-id=
      3 --enable-persistence=true
      ```

    - Alter the existing cluster A region so that it specifies all gateway senders associated with the region. There are two gateway senders in this example, one that goes to cluster B and a second that goes to cluster C.

      ```
      gfsh>alter region --name=regionX --gateway-sender-id=send_to_2,send_to_3
      ```

2. While logged in to Foundation C, use gfsh to create the cluster C gateway sender and region.

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the cluster C gateway sender:

      ```
      gfsh>create gateway-sender --id=send_to_1 --remote-distributed-system-id=
      1 --enable-persistence=true
      ```

    - Create the cluster C region:

      ```
      gfsh>create region --name=regionX --gateway-sender-id=send_to_1 --type=PA
      RTITION_REDUNDANT
      ```

Set Up an Additional Unidirectional Interaction

This topic describes how to add a unidirectional TLS-encrypted WAN connection to an additional VMware GemFire for Tanzu Application Service service instance, once an initial setup is in place for a first pair of GemFire for TAS service instances.

Call the first pair of GemFire for TAS service instances Cluster A and Cluster B, which reside on Foundation A and Foundation B, respectively. This set of directions sets up an interaction between Cluster A and and a third service instance, Cluster C, which resides on a third foundation, Foundation C.

Assumptions

- You have already established a TLS-encrypted WAN connection between a service instance on Foundation A, referred to here as Cluster A, and a service instance on Foundation B, which we'll call Cluster B, following the Set Up a Unidirectional System procedure.

- You wish to add an additional TLS-encrypted WAN connection between Cluster A and a third service instance on Foundation C, which we will call Cluster C.
- The connection will be unidirectional, such that all operations in Cluster A are replicated in Cluster C, but Cluster C does not send operations to Cluster A.
- The Preparing for TLS procedure has been followed for Foundation C, establishing a CredHub "/services/tls_ca" certificate.
- The Establishing Mutually Trusted TLS Credentials procedure has been followed, establishing mutually trusted TLS credentials between Foundations A and C.

Get Cluster A's Remote Credentials

1. Log into Foundation A using Foundation A Cloud Foundry credentials.
2. View the Cluster A service key and save a copy of the `remote_cluster_info` element, which includes the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. These are the credentials that enable other clusters to communicate with Cluster A.

```
$ cf service-key wan1 k1
Getting key k1 for service instance wan1 as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ]
   }
  ]
 }
}
```

Create the New Cluster

The cluster within each service instance uses an identifier called a `distributed_system_id`. This example assumes the assignment of `distributed_system_id = 1` for cluster A, `distributed_system_id = 2` for cluster B, and `distributed_system_id = 3` for cluster C.

1. Log into Foundation C using Foundation C Cloud Foundry credentials.

2. Use the `cf create-service` command to create a service instance, which we will call Cluster C in this example. In the `cf create-service` command, use the `-c` option to specify the following arguments:

    - `"tls" : true` to enable TLS
    - `"distributed_system_id" : 3` to set the distributed system id of Cluster C to “3”
    - `remote_clusters" : [CLUSTER-A-CREDENTIALS]` to enable communication with Cluster A, where CLUSTER-A-CREDENTIALS are the contents of Cluster A’s `remote_cluster_info` element.

    The following command specifies the Cluster A credentials you copied in an earlier step, enabling Cluster C to communicate with and receive data from Cluster A:

    ```
    $ cf create-service p-cloudcache wan-cluster wan3 -c '{
    "tls" : true,
    "distributed_system_id" : 3,
    ```

```
  "remote_clusters":[
   {
    "recursors": {
      "services-subnet.service-instance-id.bosh": [
        "10.0.8.5:1053",
        "10.0.8.7:1053",
        "10.0.8.6:1053"
      ]
    },
    "remote_locators":[
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
      "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "trusted_sender_credentials":[
    {
      "username": "gateway_sender_GHI",
      "password":"gws-GHI-password"
    }]
  }]
}'
```

Verify the completion of service creation prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `create succeeded` when service creation is completed.

3. While logged in to Foundation C, create a service key for Cluster C.

```
$ cf create-service-key wan3 k3
Creating service key wan3 for service instance k3 ...
OK
```

Display the service key and save a copy of the `remote_cluster_info` element, which includes the `recursors` array, `remote_locators` array, and `trusted_sender_credentials`. These are the credentials that enable other clusters to communicate with Cluster C.

```
$ cf service-key wan3 k3

Getting key k3 for service instance destination as admin...

{
 "distributed_system_id": "3",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
  "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
  "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet-3.service-instance-id-3.bosh": [
    "10.2.32.7:1053",
    "10.2.32.6:1053",
    "10.2.32.8:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
   "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
   "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "username": "gateway_sender_XYZ",
    "password": "gws-XYZ-password"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-3.example.com/gemfire/v1",
  "pulse": "https://cloudcache-3.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-STU-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_STU"
  },
  {
   "password": "dev-VWX-password",
   "roles": [
    "developer"
   ],
   "username": "developer_VWX"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
      "id1.locator.services-subnet.service-instance-id.bosh[55221]",
      "id2.locator.services-subnet.service-instance-id.bosh[55221]",
      "id3.locator.services-subnet.service-instance-id.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     "gateway_sender_GHI"
    ]
   }
  ]
 }
}
```

4. While logged in to Foundation A, update the Cluster A service instance to add the Cluster C credentials, using the `-c` option to specify a `remote_clusters` element that includes the contents of the Cluster C service key `remote_cluster_info` element, including the `recursors` array and the `remote_locators` array, but **NOT** the `trusted_sender_credentials`. This allows Cluster A to send messages to Cluster C, but blocks data flow from Cluster C to Cluster A.

   IMPORTANT: You must also retain the credentials for all clusters with which Cluster A interacts, in this example those of Cluster B. The existing `remote_locators` and `recursors` credentials can be copied from the `remote_cluster_info` entry in Cluster B's service key, before updating the Cluster A service instance to append credentials for the additional cluster. Do not include Cluster B's `trusted_sender_credentials`.

The following example adds Cluster C's credentials to Cluster A's `remote_clusters` element:

```
$ cf update-service wan1 -c '
{
  "remote_clusters":[
   {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ]
   },
   {
    "recursors": {
     "services-subnet-3.service-instance-id-3.bosh": [
      "10.2.32.7:1053",
      "10.2.32.6:1053",
      "10.2.32.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
    ]
   }
 ]
}'
Updating service instance wan1 as admin
```

Verify the completion of the service update prior to continuing to the next step. Output from the `cf services` command will show the `last operation` as `update succeeded` when service update is completed.

5. To verify that a service instance has been correctly updated, delete and recreate the cluster service key. The recreated service key will have the same user identifiers and passwords as its predecessor, and will reflect the changes you specified in the recent `cf update-service` commands. In particular, the `wan{}` element at the end of a cluster's service key should be populated with the other cluster's remote connection information. For example, to verify that the Cluster A service key was updated correctly, log in as Cluster A administrator and issue these commands to delete and recreate the Cluster A service key:

```
$ cf delete-service-key wan1 k1
```

```
  ...
$ cf create-service-key wan1 k1
```

Verify that the `wan{}` field of the Cluster A service key contains a `remote_clusters` element which specifies contact information for Cluster B, including Cluster B's `remote_locators` array, but **NOT** `trusted_sender_credentials`:

```
$ cf service-key wan1 k1

Getting key k1 for service instance wan1 as admin...

{
 "distributed_system_id": "1",
 "gfsh_login_string": "connect
 --url=https://cloudcache-url.com/gemfire/v1
 --user=cluster_operator_user --password=pass --skip-ssl-validation",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "tls-enabled": "true",
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "cl-op-ABC-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_ABC"
  },
  {
   "password": "dev-DEF-password",
   "roles": [
    "developer"
   ],
   "username": "developer_DEF"
  }
 ],
 "wan": {
  "remote_clusters": [
   {
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ]
   },
   {
    "remote_locators": [
     "id1.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id2.locator.services-subnet-3.service-instance-id-3.bosh[55221]",
     "id3.locator.services-subnet-3.service-instance-id-3.bosh[55221]"
    ]
   }
  ]
 }
}
```

Create Gateway Senders and Regions

1. While logged in to Foundation A, use gfsh to create the cluster A gateway sender and alter the existing region.

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the cluster A gateway sender. The required `remote-distributed-system-id` option identifies the `distributed-system-id` of the destination cluster. It is 3 for this example:

      ```
      gfsh>create gateway-sender --id=send_to_3 --remote-distributed-system-id=
      3 --enable-persistence=true
      ```

    - Alter the existing cluster A region so that it specifies all gateway senders associated with the region. There are two gateway senders in this example, one that goes to cluster B and a second that goes to cluster C.

      ```
      gfsh>alter region --name=regionX --gateway-sender-id=send_to_2,send_to_3
      ```

2. While logged in to Foundation C, use gfsh to create the cluster C region. (Because Cluster C is a passive member in this unidirectional connection, it does not need a gateway sender.)

    - Connect using gfsh following the instructions in Connect with gfsh over HTTPS with a role that is able to manage both the cluster and the data.

    - Create the cluster C region:

      ```
      gfsh>create region --name=regionX --gateway-sender-id=send_to_1 --type=PA
      RTITION_REDUNDANT
      ```

Setting Up Servers for an Inline Cache

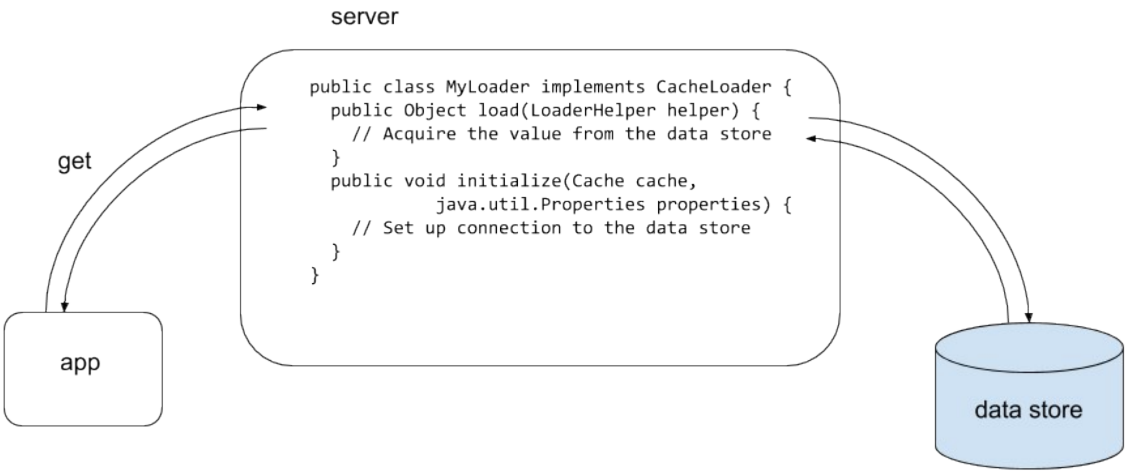
This topic explains how to set up servies for an inline cache in VMware GemFire for TAS.

See The Inline Cache for an introductory description of an inline cache. The implementation of an inline cache requires custom code deployed on the cluster servers to interact with the backend data store for read misses and for writes.

The custom code always implements a cache loader for read misses. The custom code and configuration setup differs for writes. A write-behind implementation uses an asynchronous event queue (AEQ) and an AEQ listener. A write-through implementation uses a cache writer.

Implement a Cache Loader for Read Misses

An app's get operation is a cache read. If the desired entry is in the region, it is a cache hit, and the value is quickly returned to the app. If the desired entry is not in the region, it is a cache miss. For an inline cache, that value is acquired from the backend data store. You implement the `CacheLoader` interface to handle cache misses. Each cache miss invokes the `CacheLoader.load` method. The `CacheLoader.load` method must acquire and return the value for the specified key. See VMware GemFire Java API Reference.



```
public class MyLoader implements CacheLoader {
  public Object load(LoaderHelper helper) {
    // Acquire the value from the data store
  }
  public void initialize(Cache cache,
              java.util.Properties properties) {
    // Set up connection to the data store
  }
}
```

The value returned from the `CacheLoader.load` method will be put into the region and then returned to the waiting app, completing the app's get operation. Since the app blocks while waiting for the result of the get operation, design the `CacheLoader.load` method to acquire the value as quickly as possible.

The `CacheLoader` implementation must be thread-safe. You will deploy the implementation to the servers during configuration.

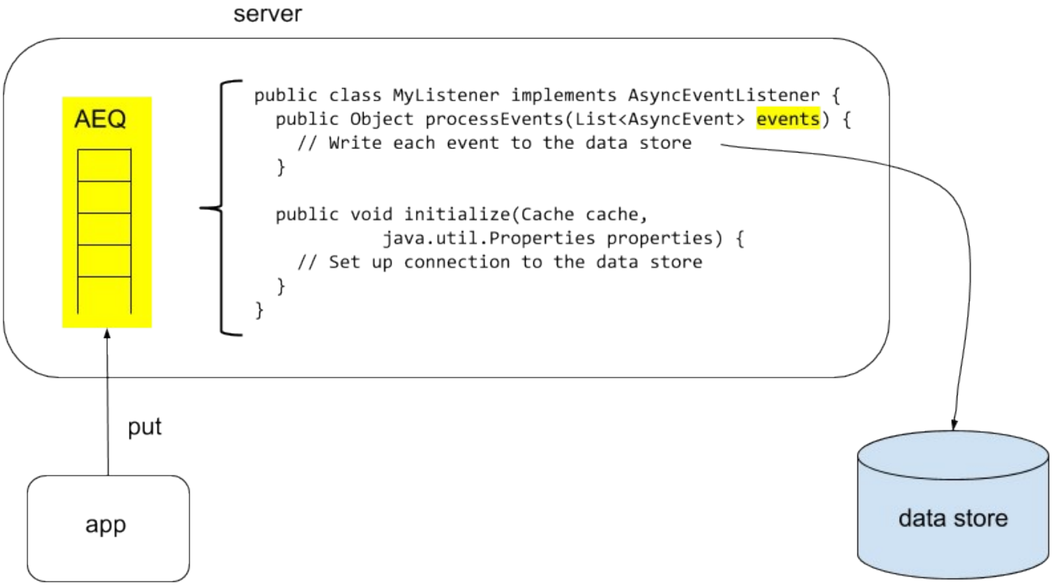
The `CacheLoader.load` method queries the backend data store for the desired entry. That communication between the server process and the backend data store requires a connection, and establishing a connection is likely to use a set of credentials. You provide a custom implementation of the `CacheLoader.initialize` method to establish the connection.

You specify the credentials during configuration with the gfsh `create region` command by adding the JSON description to the `--cache-loader` option. The credentials will be passed as parameters to the invoked `CacheLoader.initialize` method as part of the `CacheLoader` instance construction.

Implement an Asynchronous Event Queue and Cache Listener for Write Behind

An app's put operation is a cache write. For a write-behind implementation, the value is placed into the region, and it will also be asynchronously written to the backend data store, allowing the app's write operation to complete without waiting for the backend-data-store write to complete.

An asynchronous event queue (AEQ) to queue the write events together with an implementation of the `AsyncEventListener` interface provides the desired behavior. See VMware GemFire Java API Reference.



```
public class MyListener implements AsyncEventListener {
  public Object processEvents(List<AsyncEvent> events) {
    // Write each event to the data store
  }

  public void initialize(Cache cache,
            java.util.Properties properties) {
    // Set up connection to the data store
  }
}
```

With a configured AEQ, all put operations first create or update the entry in the hosted region on the server and then add the event to the AEQ.

You provide a custom implementation of the `AsyncEventListener` interface. Your `AsyncEventListener.processEvents` method's task is to iterate through the events in the AEQ, writing each newly created or updated entry in the AEQ to the backend data store. The `AsyncEventListener.processEvents` method is invoked when either the AEQ holds a configured quantity of events, or a configured quantity of time has elapsed since the earliest entry entered the AEQ.

The communication between the server process and the backend data store to do the writes requires a connection, and establishing a connection is likely to use a set of credentials. You provide a custom implementation of the `AsyncEventListener.initialize` method to establish the connection.

You specify the credentials during configuration in the gfsh `create async-event-queue` command with the `--listener-param` option as described in Configure Using gfsh for Write Behind. The credentials will be passed as parameters to the invoked `AsyncEventListener.initialize` method as part of `AsyncEventListener` instance construction.

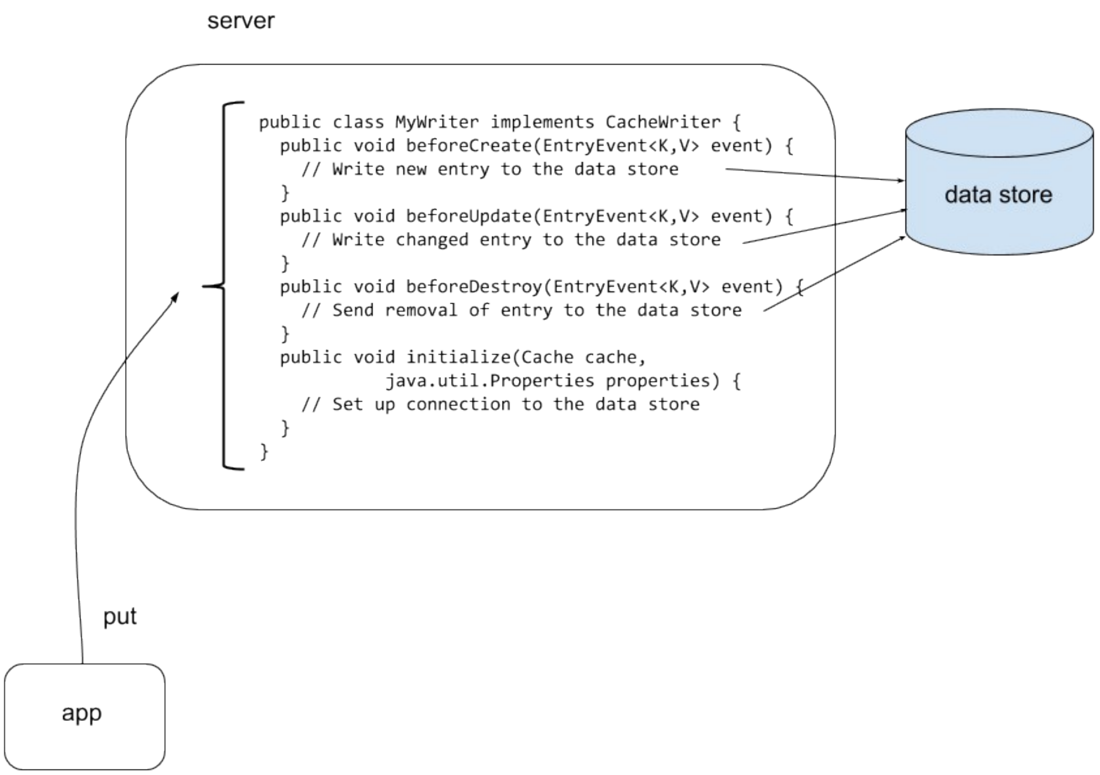
Your configuration will specify the AEQ as persistent, such that it does not lose queued backend-data-store writes across unexpected server restarts.

Implement a Cache Writer for Write Through

An app's put operation is a cache write. For a write-through implementation, the value will be written to the backend data store prior to being placed into the region. After both writes, the app's put operation completes.

An implementation of the `CacheWriter` interface implementation provides the correct behavior for write through. See VMware GemFire Java API Reference. You provide a custom implementation of the `CacheWriter.beforeCreate` method to handle backend-data-store writes for put operations that add a new entry to the region. You provide a custom implementation of the `CacheWriter.beforeUpdate` method to handle backend-data-store writes for put operations that modify an existing entry in the region. You provide a custom implementation of `CacheWriter.beforeDestroy`, as appropriate, to handle an update of the backend data store for a region operation that removes an entry.

The `CacheWriter` implementation must be thread-safe. You will deploy the implementation to the servers during configuration.



```
public class MyWriter implements CacheWriter {
  public void beforeCreate(EntryEvent<K,V> event) {
    // Write new entry to the data store
  }
  public void beforeUpdate(EntryEvent<K,V> event) {
    // Write changed entry to the data store
  }
  public void beforeDestroy(EntryEvent<K,V> event) {
    // Send removal of entry to the data store
  }
  public void initialize(Cache cache,
            java.util.Properties properties) {
    // Set up connection to the data store
  }
}
```

Communication between the server process and the backend data store to do the writes requires a connection, and establishing a connection is likely to use a set of credentials. You provide a custom implementation of the `CacheWriter.initialize` method to establish the connection.

Specify the credentials in the gfsh `create region` command during configuration as described in Configure Using gfsh for Write Through. Add the JSON description to the `--cache-writer` option. The credentials will be passed as parameters to the invoked `CacheWriter.initialize` method as part of the `CacheWriter` instance construction.

Configure Using gfsh for Write Behind

Follow this procedure to deploy your custom implementations of the interfaces to the servers, create the AEQ, and configure the region to use the AEQ and the deployed interface implementations.

1. Follow the directions in Connect with gfsh over HTTPS to connect to the cluster with a role that is able to manage both the cluster and the data.
2. Deploy the cache loader and the AEQ listener code to the servers within the VMware GemFire for Tanzu Application Service service instance:

```
gfsh>deploy --jars=/path/to/MyLoader.jar,/path/to/MyListener.jar
```

3. Create the AEQ, assigning a name for the AEQ (called `WB-AEQ` in this example), specifying the AEQ listener, and specifying the AEQ listener's parameters:

```
gfsh>create async-event-queue --id=WB-AEQ \
  --parallel=true --persistent \
  --listener=com.myCompany.MyListener \
  --listener-param=url#jdbc:db2:SAMPLE,username#admin,password#gobbledeegook
```

The persistence of the AEQ uses the default disk store, since no disk store is specified in this command.

4. Create the region, specifying the cache loader and the assigned AEQ name.

```
gfsh>create region --name=myRegion --type=PARTITION_REDUNDANT \
  --cache-loader=com.myCompany.MyLoader{'url':'jdbc:db2:SAMPLE','username':'adm
in',password:'gobbledeegook'}
  --async-event-queue-id=WB-AEQ
```

Configure Using gfsh for Write Through

Follow this procedure to deploy your custom implementations of the interfaces to the servers, and configure the region to use the deployed interface implementations.

1. Follow the directions in Connect with gfsh over HTTPS to connect to the cluster with a role that is able to manage both the cluster and the data.
2. Deploy the cache loader and the cache writer code to the servers within the GemFire for TAS service instance:

```
gfsh>deploy --jars=/path/to/MyLoader.jar,/path/to/MyWriter.jar
```

3. Create the region, specifying the cache loader and the cache writer:

   ```
   gfsh>create region --name=myRegion --type=PARTITION_REDUNDANT \
     --cache-loader=com.myCompany.MyLoader{'url':'jdbc:db2:SAMPLE','username':'adm
   in',password:'gobbledeegook'}
     --cache-writer=com.myCompany.MyWriter{'url':'jdbc:db2:SAMPLE','username':'adm
   in',password:'gobbledeegook'}
   ```

Accessing a Service Instance

This topic explains how to access a service instance of VMware GemFire for TAS.

After you have created a service instance, you can access it. The gfsh command line tool provides cluster maintenance and data access functionality. Many gfsh commands require an authenticated connection that can be set up with the `gfsh connect` command.

Connecting requires these one-time, setup steps:

1. Create a service key.
2. Create a truststore.
3. Acquire the correct version of `gfsh`.
4. Set a `JAVA_ARGS` environment variable.

Create a Service Key

A service key provides a way to access your service instance outside the scope of a deployed app. Run `cf create-service-key SERVICE-INSTANCE-NAME KEY-NAME` to create a service key. Replace `SERVICE-INSTANCE-NAME` with the name you chose for your service instance. Replace `KEY-NAME` with a name of your choice. You can use this name to refer to your service key with other commands.

```
$ cf create-service-key my-cloudcache my-service-key
```

View a Service Key

To view a service key, run `cf service-key SERVICE-INSTANCE-NAME KEY-NAME`.

```
$ cf service-key my-cloudcache my-service-key
```

The service key reveals the configuration of your service instance. It shows addresses and ports of its locators, and contains an element called `remote_cluster-info` that provides fields by which the service instance can be reached from other service instances. The service key specifies two URLs, one URL used to connect the gfsh client to the service instance, and another URL used to view the Pulse dashboard in a web browser, which allows monitoring of the service instance status.

View a Service Key with External Authentication

If external authentication such as LDAP through User Account and Authentication (UAA) is in place, the external authentication system handles user credentials, and they will not appear in the service key.

If an external authentication system has been configured, then the `cf service-key` command returns output in the following format, without role-based login credentials:

```
{
 "distributed_system_id": "0",
 "gfsh_login_string": "connect
  --url=https://cloudcache-1.example.com/gemfire/v1
  --user=\u003cusername\u003e --password=\u003cpassword\u003e",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "management": "https://cloudcache-1.example.com/management/docs",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "wan": {
  "remote_clusters": [
   {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-PQR-password",
      "username": "gateway_sender_PQR"
     }
    ]
   }
  ]
 }
}
```

View a Service Key without External Authentication

If external authentication such as LDAP through User Account and Authentication (UAA) has not been configured, the service key shows role-based login credentials as username/password pairs.

The `cf service-key` command returns output in the following format and includes role-based login credentials:

```
{
 "distributed_system_id": "0",
 "gfsh_login_string": "connect
  --url=https://cloudcache-1.example.com/gemfire/v1
  --user=cluster_operator_XXX --password=cluster_operator-password --skip-ssl-validati
on",
 "locators": [
  "id1.locator.services-subnet.service-instance-id.bosh[55221]",
  "id2.locator.services-subnet.service-instance-id.bosh[55221]",
  "id3.locator.services-subnet.service-instance-id.bosh[55221]"
 ],
 "remote_cluster_info": {
  "recursors": {
   "services-subnet.service-instance-id.bosh": [
    "10.0.8.6:1053",
    "10.0.8.7:1053",
    "10.0.8.5:1053"
   ]
  },
  "remote_locators": [
   "id1.locator.services-subnet.service-instance-id.bosh[55221]",
   "id2.locator.services-subnet.service-instance-id.bosh[55221]",
   "id3.locator.services-subnet.service-instance-id.bosh[55221]"
  ],
  "trusted_sender_credentials": [
   {
    "password": "gws-GHI-password",
    "username": "gateway_sender_GHI"
   }
  ]
 },
 "urls": {
  "gfsh": "https://cloudcache-1.example.com/gemfire/v1",
  "management": "https://cloudcache-1.example.com/management/docs",
  "pulse": "https://cloudcache-1.example.com/pulse"
 },
 "users": [
  {
   "password": "developer-password",
   "roles": [
    "developer"
   ],
   "username": "developer_XXX"
  },
  {
   "password": "cluster_operator-password",
   "roles": [
    "cluster_operator"
   ],
   "username": "cluster_operator_XXX"
   }
 ],
 "wan": {
  "remote_clusters": [
   {
    "recursors": {
     "services-subnet-2.service-instance-id-2.bosh": [
      "10.1.16.7:1053",
      "10.1.16.6:1053",
      "10.1.16.8:1053"
     ]
    },
    "remote_locators": [
     "id1.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id2.locator.services-subnet-2.service-instance-id-2.bosh[55221]",
     "id3.locator.services-subnet-2.service-instance-id-2.bosh[55221]"
    ],
    "trusted_sender_credentials": [
     {
      "password": "gws-PQR-password",
      "username": "gateway_sender_PQR"
     }
    ]
   }
  ]
 }
}
```

This service key for a VMware GemFire for Tanzu Application Service installation in which an external authentication such as LDAP via UAA has not been configured specifies these user roles that are predefined, for interacting with and within the cluster:

- The cluster operator administers the pool, performing operations such as creating and destroying regions, and creating gateway senders. The identifier assigned for this role is of the form `cluster_operator_XXX`, where `XXX` is a unique string generated upon service instance creation and incorporated in this user role's name.
- The developer does limited cluster administration such as region creation, and the developer role is expected to be used by applications that are interacting with region entries. The developer does CRUD operations on regions. The identifier assigned for this role is of the form `developer_XXX`, where `XXX` is a unique string generated upon service instance creation and incorporated in this user role's name.

Connect with gfsh over HTTPS

When connecting over HTTPS, you must use the same certificate you use to secure traffic into VMware Tanzu Application Service for VMs (TAS for VMs); that is, the certificate you use where your TLS termination occurs. See Determine Your TLS Termination.

Before you can connect, you must create a truststore.

Create a Truststore

To create a truststore, use the same certificate you used to configure TLS termination. We suggest using the `keytool` command line utility to create a truststore file.

1. Locate the certificate you use to configure TLS termination. See Determine Your TLS Termination.
2. Using your certificate, run the `keytool` command:

   ```
   keytool -import -file CERTIFICATE.CER -keystore TRUSTSTORE-FILE-PATH -storetype
   JKS
   ```

   where + `CERTIFICATE.CER` is your certificate file + `TRUSTSTORE-FILE-PATH` is the path to the location where you want to create the truststore file, including the name you want to give the file
3. When you run this command, you are prompted to enter a keystore password. Create a password and remember it!
4. When prompted for the certificate details, enter **yes** to trust the certificate.

The following example shows how to run `keytool` and what the output looks like:

```
$ keytool -import -file /tmp/loadbalancer.cer -keystore /tmp/truststore/prod.myTrustSt
ore -storetype JKS
Enter keystore password:
Re-enter new password:
Owner: CN=*.url.example.com, OU=Cloud Foundry, O=Pivotal, L=New York, ST=New York, C=U
S
Issuer: CN=*.url.example.com, OU=Cloud Foundry, O=Pivotal, L=New York, ST=New York, C=
US
Serial number: bd84912917b5b665
Valid from: Sat Jul 29 09:18:43 EDT 2017 until: Mon Apr 07 09:18:43 EDT 2031
Certificate fingerprints:
   MD5:  A9:17:B1:C9:6C:0A:F7:A3:56:51:6D:67:F8:3E:94:35
   SHA1: BA:DA:23:09:17:C0:DF:37:D9:6F:47:05:05:00:44:6B:24:A1:3D:77
   SHA256: A6:F3:4E:B8:FF:8F:72:92:0A:6D:55:6E:59:54:83:30:76:49:80:92:52:3D:91:4D:61:
1C:A1:29:D3:BD:56:57
   Signature algorithm name: SHA256withRSA
   Version: 3

Extensions:

#1: ObjectId: 2.5.29.10 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:0
]

#2: ObjectId: 2.5.29.11 Criticality=false
SubjectAlternativeName [
  DNSName: *.sys.url.example.com
  DNSName: *.apps.url.example.com
  DNSName: *.uaa.sys.url.example.com
  DNSName: *.login.sys.url.example.com
  DNSName: *.url.example.com
```

```
  DNSName: *.ws.url.example.com
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Establish the Connection with HTTPS

After you have created the truststore, you can use the cluster's command line interface, `gfsh`, to connect to the cluster over HTTPS.

1. Acquire the compressed TAR file of Apache Geode from the VMware GemFire Product Page. Discover the correct version to download by referencing the table in Product Snapshot and Version Compatibility. Note that a JDK or JRE will also be required.
2. Expand the VMware GemFire compressed TAR file. `gfsh` is within the `bin` directory in the expanded VMware GemFire.
3. Run `gfsh` on Unix with a command of the form:

   ```
   /PATH-TO-VMWARE-GEMFIRE/bin/gfsh
   ```

   or, on Windows with a command of the form:

   ```
   \PATH-TO-VMWARE-GEMFIRE\bin\gfsh.bat
   ```

   and then at the gfsh prompt, issue a `connect` command of the form:

   ```
   connect --use-http=true --url=HTTPS-gfsh-URL
    --trust-store=TRUSTSTORE-FILE-PATH --trust-store-password=PASSWORD
    --user=USER-NAME --password=USER-PASSWORD
   ```

   The `HTTPS-gfsh-URL` is the gfsh URL from the service key. See Create Service Keys for instructions on how to view the service key. `TRUSTSTORE-FILE-PATH` is the path to the trustStore file you created in Create a Truststore, and `PASSWORD` is the associated password you created. If you omit the `--trust-store-password` option from the command line, you will be prompted to enter the password.

   For an installation configured *without* an external authentication such as LDAP via UAA, the `USER-NAME` and `USER-PASSWORD` are taken from the service key, and they will either be for the developer role or for the cluster operator role, depending on the gfsh operations to be performed.

   For an installation configured *with* an external authentication such as LDAP via UAA, the `USER-NAME` and `USER-PASSWORD` are your credentials as issued and set up by your organization within your SSO system.

Establish the Connection in a Development Environment

When working in a non-production, development environment, a developer may choose to work in a less secure manner by eliminating the truststore and SSL mutual authentication.

The steps to establish the `gfsh` connection become:

1. Acquire the `connect` command (the `gfsh_login_string`) from the service key. This command will connect using the cluster_operator role. This `connect` command assumes that the installation is configured *without* an external authentication such as LDAP via UAA.
2. Acquire the compressed TAR file of Apache Geode from the VMware GemFire Product Page. Discover the correct version to download by referencing the table in Product Snapshot and Version Compatibility. Note that a JDK or JRE will also be required.
3. Expand the VMware GemFire compressed TAR file. `gfsh` is within the `bin` directory in the expanded VMware GemFire.
4. Run `gfsh` on Unix with a command of the form:

   ```
   /PATH-TO-VMWARE-GEMFIRE/bin/gfsh
   ```

   or, on Windows with a command of the form:

   ```
   \PATH-TO-VMWARE-GEMFIRE\bin\gfsh.bat
   ```

   and then issue the acquired `connect` command:

   ```
   gfsh>connect --url=https://cloudcache-1.example.com/gemfire/v1
     --user=cluster_operator_XXX --password=cluster_operator-password
     --skip-ssl-validation
   ```

   At each of the nine `gfsh` prompts that ask for keystore, truststore, and SSL details, hit `Enter` to step through the prompts and connect.

Determine Your TLS Termination

To connect your VMware GemFire for Tanzu Application Service service instance using `gfsh`, you will need the certificate from where your TLS termination occurs. The TLS termination may be at the Gorouter, at the HAProxy, or at your load balancer. Request the needed certificate from your VMware Tanzu Application Service for VMs (TAS for VMs) operator.

The TAS for VMs operator determines the location of your TLS termination:

1. Bring up the Ops Manager dashboard.
2. Click on the TAS for VMs product tile.
3. Click on the Networking section under the Settings tab.

The choice of TLS termination is labeled with **TLS termination point**.

If the choice names the **Gorouter** or **HAProxy**, the certificate is in the same section, labeled with **Certificates and private keys for the Gorouter and HAProxy**.

If the choice names the **Infrastructure load balancer**, then the TAS for VMs operator can retrieve the certificate from the load balancer.

Using gfsh

This topic describes ways to use the `gfsh` command-line interface to manage VMware GemFire for TAS.

gfsh Command Restrictions

Developers may invoke all `gfsh` commands. Given credentials with sufficient permissions, those `gfsh` commands will be invoked. However, not all `gfsh` commands are supported. An invocation of an unsupported command may lead to incorrect results. Those results range from ineffective results to inconsistent region entries. **Do not use these listed `gfsh` commands; each has an explanation why it must not be used.**

The following `gfsh start` commands will bring up members contrary to the configured plan. Their configuration will be wrong, and their existence is likely to contribute to data loss. Since they are not part of the configured plan, any upgrade will not include them, and if they were to stop or crash, the BOSH Director will not restart them.

- `gfsh start locator`
- `gfsh start server`

The following `gfsh` cluster stop commands will temporarily stop the member or cluster. However, the BOSH Director will notice that members are not running and restart them. This renders the following commands ineffective:

- `gfsh stop locator`
- `gfsh stop server`
- `gfsh shutdown`

The following Lucene-related commands are not supported:

- `gfsh create lucene index`
- `gfsh describe lucene index`
- `gfsh destroy lucene index`
- `gfsh list lucene indexes`
- `gfsh search lucene`

The following JNDI binding-related commands are not supported:

- `gfsh create jndi-binding`
- `gfsh describe jndi-binding`
- `gfsh destroy jndi-binding`
- `gfsh list jndi-binding`

The following configure command will instill configuration contrary to the already-configured plan. Since it is not part of the configured plan, any upgrade will not include it. Therefore, do not use the following command:

- `gfsh configure pdx`

The creation of a gateway receiver will never be appropriate for any situation. The VMware GemFire for Tanzu Application Service cluster will already have gateway receivers, and there is no situation in which the cluster can benefit from creating more. Therefore, do not use the following command:

- `gfsh create gateway receiver`

Do Not Export from an VMware GemFire Cluster to a GemFire for TAS Cluster

While the expectation is that configuration and data can be exported from an VMware GemFire cluster and then imported into a GemFire for TAS cluster, this does **not** work. Using export and import commands will not have the desired effect of migration from one cluster to another. The import of cluster configuration requires a state that cannot be provided by a GemFire for TAS cluster. The GemFire for TAS cluster will already have its configuration, and upon restart or upgrade, that same configuration will be used. Given that the configuration cannot be imported, data import is problematic. Therefore, do not use the following command:

- `gfsh import cluster-configuration`
- `gfsh import data`

Create Regions

After connecting with `gfsh` with a role that is able to manage a cluster’s data, you can define a new cache region.

The following command creates a partitioned region with a single redundant copy:

```
gfsh>create region --name=my-cache-region --type=PARTITION_REDUNDANT_HEAP_LRU
     Member      | Status
---------------- | -------------------------------------------------------
cacheserver-z2-1 | Region "/my-cache-region" created on "cacheserver-z2-1"
cacheserver-z3-2 | Region "/my-cache-region" created on "cacheserver-z3-2"
cacheserver-z1-0 | Region "/my-cache-region" created on "cacheserver-z1-0"
cacheserver-z1-3 | Region "/my-cache-region" created on "cacheserver-z1-3"
```

See Region Design for guidelines about choosing a region type.

You can test the newly created region by writing and reading values with `gfsh`:

```
gfsh>put --region=/my-cache-region --key=test --value=thevalue
Result      : true
Key Class   : java.lang.String
Key         : test
Value Class : java.lang.String
Old Value   : NULL


gfsh>get --region=/my-cache-region --key=test
Result      : true
Key Class   : java.lang.String
Key         : test
Value Class : java.lang.String
Value       : thevalue
```

In practice, you should perform these get/put operations from a deployed app. To do this, you must bind the service instance to these apps.

Working with Disk Stores

Persistent regions and regions that overflow upon eviction use disk stores. Use `gfsh` to create or destroy a disk store.

Create a Disk Store

To create a disk store for use with a persistent or overflow type of region:

1. Use the directions in Connect with gfsh over HTTPS to connect to the GemFire for TAS service instance with a role that is able to manage a cluster's data.
2. Create the disk store with a `gfsh` command of the form:

   ```
   create disk-store --name=<name-of-disk-store>
    --dir=<relative/path/to/diskstore/directory>
   ```

   Specify a relative path for the disk store location. That relative path will be created within `/var/vcap/store/gemfire-server/`. For more details about further options, see create disk-store in the `gfsh` Command Reference in the VMware GemFire documentation.

Destroy a Disk Store

To destroy a disk store:

1. Use the directions in Connect with gfsh over HTTPS to connect to the GemFire for TAS service instance with a role that is able to manage a cluster's data.
2. Destroy the disk store with a `gfsh` command of the form:

   ```
   destroy disk-store --name=<name-of-disk-store>
   ```

   For more details about further options, see destroy disk-store in the `gfsh` Command Reference in the VMware GemFire documentation.

Use the Pulse Dashboard

You can access the Pulse dashboard for a service instance in a web browser using the Pulse URL specified in the service key you created following the directions in Create Service Keys.

Access Service Instance Metrics

To access service metrics, you must have **Enable Plan** selected under **Service Plan Access** on the page where you configure your tile properties. (For details, see the Configure Service Plans page.)

GemFire for TAS service instances output metrics to the Loggregator Firehose. You can use the Firehose plugin to view metrics output on the cf CLI directly. For more information about using the Firehose plugin, see Installing the Loggregator Firehose Plugin for cf CLI in the VMware Tanzu Application Service for VMs documentation.

You can also connect the output to a Firehose nozzle. Nozzles are programs that consume data from the Loggregator Firehose. For example, to connect the output to the nozzle for Datadog, see Datadog Firehose Nozzle on GitHub.

GemFire for TAS supports metrics for the whole cluster and metrics for each member. Each server and locator in the cluster outputs metrics.

Service Instance (Cluster-wide) Metrics

- serviceinstance.MemberCount: the number of VMs in the cluster
- serviceinstance.TotalHeapSize: the total MBs of heap available in the cluster
- serviceinstance.UsedHeapSize: the total MBs of heap in use in the cluster

Member (per-VM) Metrics

- member.GarbageCollectionCount: the number of JVM garbage collections that have occurred on this member since startup
- member.CpuUsage: the percentage of CPU time used by the cluster process
- member.GetsAvgLatency: the avg latency of GET requests to this cluster member
- member.PutsAvgLatency: the avg latency of PUT requests to this cluster member
- member.JVMPauses: the number of JVM pauses that have occurred on this member since startup
- member.FileDescriptorLimit: the number of files this member allows to be open at once
- member.TotalFileDescriptorOpen: the number of files this member has open now
- member.FileDescriptorRemaining: the number of files that this member could open before hitting its limit
- member.TotalHeapSize: the number of megabytes allocated for the heap
- member.UsedHeapSize: the number of megabytes currently in use for the heap
- member.UnusedHeapSizePercentage: the percentage of the total heap size that is not currently being used

Access Service Broker Metrics

Service broker metrics are on by default and can be accessed through the Loggregator Firehose CLI plugin. For more information, see Installing the Loggregator Firehose Plugin for cf CLI in the VMware Tanzu Application Service for VMs documentation. For more information about broker metrics, see (Optional) Configure Service Metrics in *Operating an On-Demand Broker* in the On-Demand Services SDK for VMware Tanzu documentation.

Export gfsh Logs

You can retrieve logs and `.gfs` stats files from your GemFire for TAS service instances using the `export logs` command in `gfsh`.

1. Use the Connect with gfsh over HTTPS procedure to connect to the service instance for

which you want to see logs. Use a role that can read cluster information.

2. Run `export logs`. If you see a message of the following form, your logs exceed an expanded size of 100 megabytes, and they were not exported:

   ```
   Estimated exported logs expanded file size = 289927115, file-size-limit = 10485
   7600. To deactivate exported logs file size check use option "--file-size-limit
   =0".
   ```

   To export in this case, use the `gfsh` command:

   ```
   export logs --file-size-limit=0
   ```

3. Find the ZIP file in the directory where you started `gfsh`. This file contains a folder for each member of the cluster. The member folder contains the associated log files and stats files for that member.

For more information about the `gfsh` export logs command, see the gfsh export logs documentation.

Deploy or Redeploy an App JAR File to the Servers

You can deploy or redeploy an app JAR file to the servers in the cluster.

Deploy an App JAR File to the Servers

To initially deploy an app JAR file after connecting within `gfsh` using credentials that can manage the cluster, run the following `gfsh` command:

```
deploy --jar=PATH-TO-JAR/FILENAME.jar
```

For example:

```
gfsh>deploy --jar=working-directory/myJar.jar
```

Redeploy an App JAR File to the Servers

To redeploy an app JAR file after connecting within `gfsh` using credentials that can manage the cluster:

1. Deploy the updated JAR file by running the following `gfsh` command:

   ```
   deploy --jar=PATH-TO-UPDATED-JAR/FILENAME.jar
   ```

   For example:

   ```
   gfsh>deploy --jar=newer-jars/myJar.jar
   ```

2. Restart the cluster and load the updated JAR file by running the following cf CLI command:

   ```
   cf update-service SERVICE-INSTANCE-NAME -c '{"restart": true}'
   ```

   For example:

```
$ cf update-service my-service-instance -c '{"restart": true}'
```

Scripting Common Operations

You can place `gfsh` commands in files to provide a way to script common maintenance operations. Scripting operations reduces errors that might occur when manually entering commands, and can be used for running test cases and in deployment automation.

Running a gfsh Script

The `gfsh run` command invokes the `gfsh` commands that are in a file. Start up the `gfsh` command line interface first, and then run the command:

```
gfsh>run --file=thecommands.gfsh
```

To see the `gfsh run` options:

```
gfsh>help run
```

File specification can be a relative or absolute path and file name. Scripted commands are not interactive, and any command that would have prompted for input will instead use default values.

To eliminate placing cleartext passwords within a script, run `gfsh` and connect to the cluster prior to running a script with the `gfsh run` command. The password will appear on the command line, but it will not appear in the file that logs and captures `gfsh` commands.

Example Scripts

Tightly focussed scripts will ease cluster maintenance. Most of these scripts will do cluster management operations, so connect to the cluster with a role that is able to manage both the cluster and the data.

Creating the regions hosted on the servers is a common operation. The following example `gfsh` script contents creates two regions:

```
create region --name=sessions --type=PARTITION_REDUNDANT
create region --name=customers --type=REPLICATE
```

The following example `gfsh` script contents deploys a app JAR file to the cluster servers:

```
deploy --jar=/path/to/app-classes.jar
```

The following example `gfsh` script contents creates the disk store needed for persistent regions:

```
create disk-store --name=all-regions-disk --dir=regions
```

Connecting a Spring Boot App to VMware GemFire with

Session State Caching

This section describes the two ways in which you can connect a Spring Boot app to VMware GemFire for Tanzu Application Service:

- Using a Tomcat app with a WAR file. This is the default method for Tomcat apps.
- Using the spring-session-data-gemfire library. This method requires that you use the correct version of these libraries.

Use the Tomcat App

To get a Spring Boot app running with session state caching (SSC) on GemFire for TAS, you must create a WAR file using the `spring-boot-starter-tomcat` plugin instead of the `spring-boot-maven` plugin to create a JAR file.

For example, if you want your app to use SSC, you cannot use `spring-boot-maven` to build a JAR file and push your app to VMware Tanzu Application Service for VMs (TAS for VMs), because the Java buildpack does not pull in the necessary JAR files for SSC when it detects a Spring JAR file.

To build your WAR file, add this dependency to your `pom.xml`:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-tomcat</artifactId>
  <scope>provided</scope>
</dependency>
```

For a full example of running a Spring Boot app that connects with SSC, run this app and use the following for your `pom.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>io.pivotal.gemfire.demo</groupId>
  <artifactId>HttpSessionCaching-Webapp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>HttpSessionCaching-Webapp</name>
  <description>Demo project for GemFire Http Session State caching</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.8.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

Use a Spring Session Data GemFire App

You can connect your Spring app to GemFire for TAS to do session state caching. Use the correct version of the `spring-session-data-gemfire` library; apps built for GemFire for TAS v1.6.0 and later versions are compatible with Spring Session Data GemFire v2.1.5.RELEASE and later versions.

Upgrade GemFire for TAS and Spring Session Data GemFire

1. Before your operator upgrades GemFire for TAS, stop your app. This avoids breaking the app in this upgrade process.

2. Upgrade GemFire for TAS. See Upgrading VMware GemFire for Tanzu Application Service for details.

3. Rebuild your app using a `build.gradle` file that depends on the correct version of VMware GemFire. Here is an example `build.gradle` file:

```
version = '0.0.1-SNAPSHOT'

buildscript {
  ext {
    springBootVersion = '2.1.8.RELEASE'
  }
  repositories {
    mavenCentral()
    maven { url "https://repo.spring.io/snapshot" }
    maven { url "https://repo.spring.io/milestone" }
  }
  dependencies {
    classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootV
ersion}")
  }
}

apply plugin: 'java'
apply plugin: 'org.springframework.boot'
apply plugin: 'idea'

idea{
  module{
    downloadSources = true
    downloadJavadoc = true
  }
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
  mavenCentral()
  maven { url "https://repo.spring.io/libs-milestone" }
  maven { url "https://repo.spring.io/milestone" }
  maven { url "http://repo.springsource.org/simple/ext-release-local" }
  maven { url "http://repo.spring.io/libs-release/" }
  maven { url "https://repository.apache.org/content/repositories/snapshots" }
}


dependencies {
  compile("org.springframework.boot:spring-boot-starter-web:2.1.8.RELEASE")
  compile("org.springframework.session:spring-session-data-gemfire:2.1.5.RELEAS
E")
  compile("org.springframework.geode:spring-gemfire-starter:1.1.0.RELEASE")
}
```

4. Clear the session state region.
5. Start the rebuilt app.

Creating Continuous Queries Using Spring Data GemFire

This topic discusses creating continuous queries using Spring Data GemFire for use with VMware GemFire for TAS.

To create continuous queries with the Spring Data GemFire library, you must have the following:

- Spring Data GemFire v2.1.0+
- Spring Boot v2.1.0+

To create continuous queries, do the following items:

- Specify attributes `subscriptionEnabled` and `readyForEvents` for the `ClientCacheApplication` annotation. Apply this annotation to the Spring Boot client application class:

```
@ClientCacheApplication(name = "GemFireSpringApplication", readyForEvents = tru
e,
  subscriptionEnabled = true)
```

The annotation for a durable event queue for continuous queries also sets the `durableClientId` and `keepAlive` attributes. For example:

~~~
```java
@ClientCacheApplication(name = "GemFireSpringApplication",
  durableClientId = "durable-client-id", keepAlive = true,
  readyForEvents = true, subscriptionEnabled = true)
```
~~~

- Annotate the method that handles the events to specify the query. To make the event queue durable across server failures and restarts, include the `durable = true` attribute in the annotation, as is done in the example:

  ```
  @Component
  public class ContinuousQuery {

      @ContinuousQuery(name = "yourQuery",
         query = "SELECT * FROM /yourRegion WHERE someAttribute == true",
         durable = true)
      public void handleChanges(CqEvent event) {
        //PERFORM SOME ACTION
      }
  }
  ```

  The class that contains the method with the @ContinuousQuery annotation must have the @Component annotation, such that the continuous query is wired up correctly for the server.

For more information, see the Spring Data GemFire documentation.

Advanced Service Instance Configuration

This topic explains the advanced configuration settings when creating a service instance of VMware GemFire for TAS.

Warning: This is an advanced feature and can have a significant impact on the stability of the GemFire cluster and the underlying JVM. Users should advance with caution unless they are an advanced user or are being instructed by support staff.

The advanced configuration feature allows users to configure custom JVM options, GemFire properties, Credentials, and Resource Manager properties. While the majority of parameters have the ability to be changed, there is a small group that cannot be changed due to their impact on the TAS platform.

Warning: Do not modify the security properties listed below using advanced configuration.

Some security properties set using advanced configuration are overwritten by the default settings, and modifying the security properties can lead to unintended consequences. For example, when SSL endpoint verification is enabled, the configuration for protocols and ciphers reverts to the SSLContext’s client mode defaults. If advanced configuration was used to modify the security properties, increased difficulty when upgrading the JDK can occur when the newer JDK uses different defaults for client and server mode SSL. Additionally, modifying the security properties can lead to duplicate properties being set.

Do not modify the following security properties:

- security-gateway-password

- security-gateway-username
- security-json
- security-password
- security-username
- ssl-enabled-components
- ssl-endpoint-identification-enabled
- ssl-keystore
- ssl-keystore-password
- ssl-require-authentication
- ssl-truststore
- ssl-truststore-password
- ssl-web-require-authentication

The following JVM options are unavailable for customization:

```
-Xms
-Xmx
-Xloggc
-XX:NumberOfGCLogFiles
-XX:GCLogFileSize
-XX:OnOutOfMemoryError
-XX:+DisableExplicitGC
```

The following GemFire property is unavailable for customization:

```
-Dgemfire.OSProcess.ENABLE_OUTPUT_REDIRECTION
```

Furthermore, if one of the following properties below is set, then VMware GemFire for TAS will NOT set default values for any of the other properties listed below. The responsibility lies on the user to determine and set all of the values.

```
-XX:NewSize
-XX:MaxNewSize
-XX:CMSInitiatingOccupancyFraction
-XX:+UnlockDiagnosticVMOptions
-XX:ParGCCardsPerStrideChunk=32768
-XX:+UseConcMarkSweepGC
-XX:+UseCMSInitiatingOccupancyOnly
-XX:+CMSClassUnloadingEnabled
```

Setting Advanced Configurations

Users can set an advanced_configuration when creating a service instance such as:

```
cf create-service p-cloudcache dev-plan qa_instance -c
'{
   "advanced_configuration":{
      "server":{

         "jvm_options":["-XX:+UnlockDiagnosticVMOptions",
             "-XX:ParGCCardsPerStrideChunk=1274"],

         "gemfire_properties":{"enable-cluster-configuration":"true"},

         "credentials": {
           "db_user_credhub_key": "credhub/ref/myuser",
           "db_password_credhub_key": "credhub/ref/mypassword"
         },

      "resource_manager": {
            "critical_heap_percentage": 80,
            "eviction_heap_percentage": 60
          }
      },

      "locator":{

         "jvm_options":["-XX:+UseNUMA"],

         "gemfire_properties":{"groups":"QA_DB_1"},

         "credentials": {
           "db_user_credhub_key": "credhub/ref/myuser",
           "db_password_credhub_key": "credhub/ref/mypassword"
         },

      "resource_manager": {
            "critical_heap_percentage": 80,
            "eviction_heap_percentage": 60,
    }
      }
   }
}'
```

The advanced_configuration allows users to set options for the servers and the locators. Within each (server and locator) users can set jvm_options, gemfire_properties, credentials, and resource managers.

Updating a Service Instance with Advanced Configuration

To update a service instance with advanced configurations, run the `cf update-service` command with the configuration that you would like to apply to the service instance.

Warning

This new configuration WILL OVERWRITE any previous advanced configuration that may have been set on the service instance. Any configuration option not set in the advanced_configuration update to the service instance will revert back to the default value.

```
cf update-service qa_instance -c
'{
   "advanced_configuration":{
      "server":{
         "jvm_options":["-XX:+UnlockDiagnosticVMOptions",
             "-XX:ParGCCardsPerStrideChunk=1274"],
         "gemfire_properties":{"enable-cluster-configuration":"true"},
         "credentials": {
           "db_user_credhub_key": "credhub/ref/myuser",
           "db_password_credhub_key": "credhub/ref/mypassword"
         },
      "resource_manager": {
            "critical_heap_percentage": 80,
            "eviction_heap_percentage": 60
          }

      },
      "locator":{
         "jvm_options":["-XX:+UseNUMA"],
         "gemfire_properties":{"groups":"QA_DB_1"},
         "credentials": {
           "db_user_credhub_key": "credhub/ref/myuser",
           "db_password_credhub_key": "credhub/ref/mypassword"
         },
         "resource_manager": {
            "critical_heap_percentage": 80,
            "eviction_heap_percentage": 60
          }
      }
   }
}'
```

Reset Advanced Configuration to Default Values

To reset all advanced settings to their default values, run the `cf update-service` command and set the advanced_configuration settings to empty brackets {}.

```
cf update-service qa_instance -c
'{
   "advanced_configuration":{}
}'
```

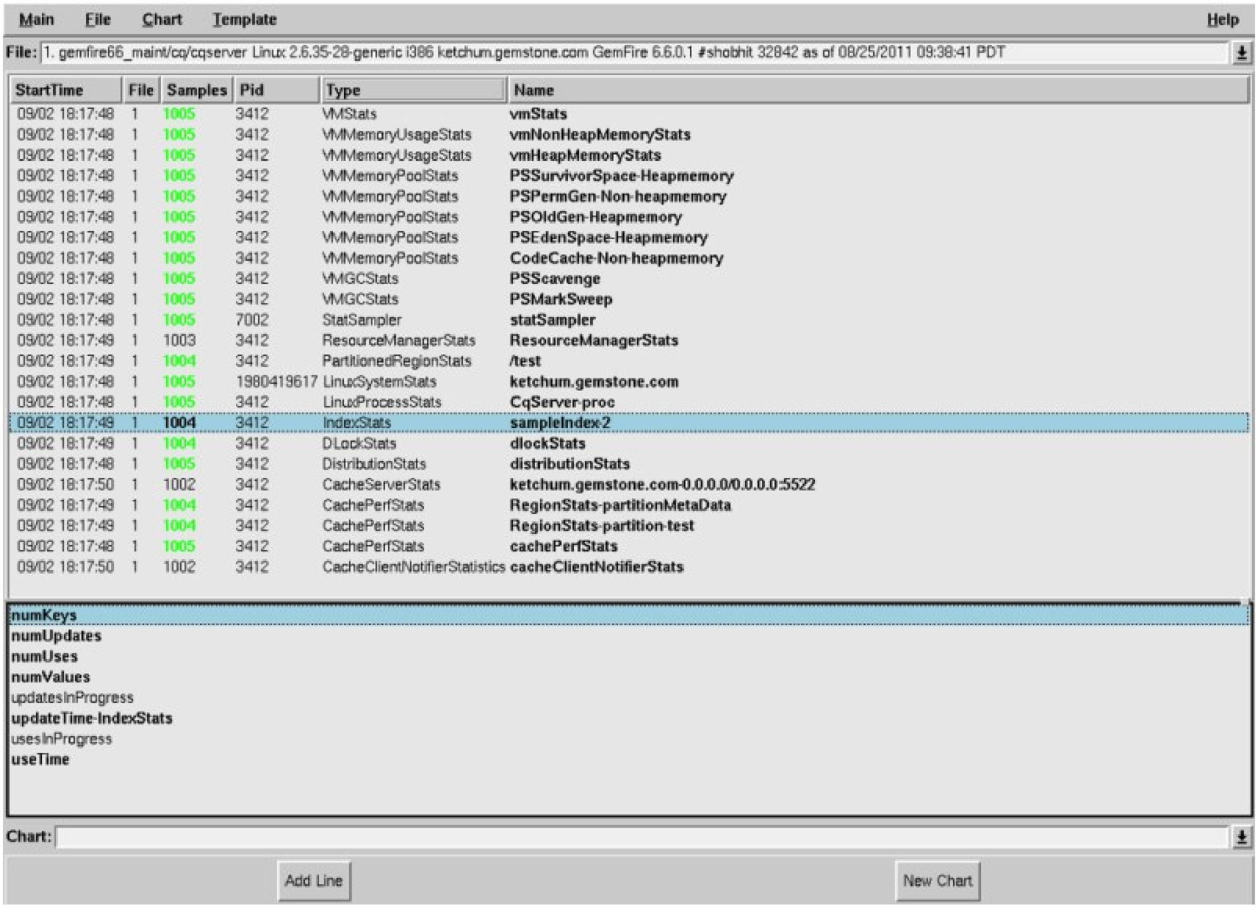
Visual Statistics Display (VSD)

The Visual Statistics Display (VSD) utility reads VMware GemFire for TAS statistics and produces graphical displays for analysis. VSD helps you monitor the performance of GemFire for TAS and diagnose performance problems.

Your service instance creates a statistical archive file named `filename.gfs`. The file logs useful statistics — counters and gauges that describe the state of the system at a particular moment in time. The file collects statistics at specific sampling intervals, which you can set at various levels to monitor different types of behavior.

The VSD tool reads the sampled statistics and produces graphical displays for analysis. Typically, the points on a line of a VSD graph represent the values for a particular statistic over time. VSD's online help offers more complete reference information about the tool.

The following screenshots of the VSD tool display statistics and a graph analysis of selected statistics.

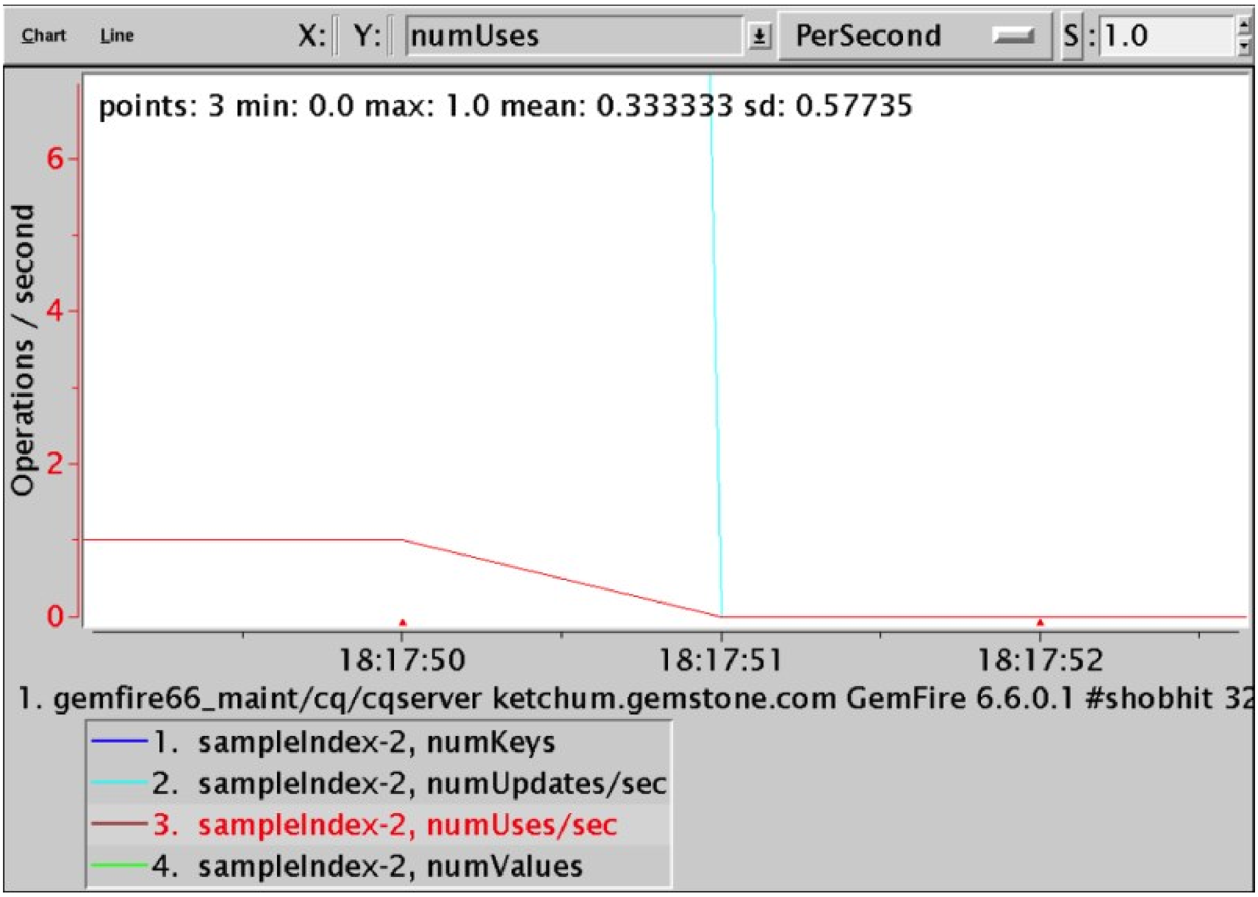


File: 1. gemfire66_maint/cq/cqserver Linux 2.6.35-28-generic i386 ketchum.gemstone.com GemFire 6.6.0.1 #shobhit 32842 as of 08/25/2011 09:38:41 PDT

StartTime	File	Samples	Pid	Type	Name
09/02 18:17:48	1	1005	3412	VMStats	vmStats
09/02 18:17:48	1	1005	3412	VMMemoryUsageStats	vmNonHeapMemoryStats
09/02 18:17:48	1	1005	3412	VMMemoryUsageStats	vmHeapMemoryStats
09/02 18:17:48	1	1005	3412	VMMemoryPoolStats	PSSurvivorSpace-Heapmemory
09/02 18:17:48	1	1005	3412	VMMemoryPoolStats	PSPermGen-Non-heapmemory
09/02 18:17:48	1	1005	3412	VMMemoryPoolStats	PSOldGen-Heapmemory
09/02 18:17:48	1	1005	3412	VMMemoryPoolStats	PSEdenSpace-Heapmemory
09/02 18:17:48	1	1005	3412	VMMemoryPoolStats	CodeCache-Non-heapmemory
09/02 18:17:48	1	1005	3412	VMGCStats	PSScavenge
09/02 18:17:48	1	1005	3412	VMGCStats	PSMarkSweep
09/02 18:17:48	1	1005	7002	StatSampler	statSampler
09/02 18:17:49	1	1003	3412	ResourceManagerStats	ResourceManagerStats
09/02 18:17:49	1	1004	3412	PartitionedRegionStats	/test
09/02 18:17:48	1	1005	1980419617	LinuxSystemStats	ketchum.gemstone.com
09/02 18:17:48	1	1005	3412	LinuxProcessStats	CqServer-proc
09/02 18:17:49	1	1004	3412	IndexStats	sampleIndex-2
09/02 18:17:49	1	1004	3412	DLockStats	dlockStats
09/02 18:17:48	1	1005	3412	DistributionStats	distributionStats
09/02 18:17:50	1	1002	3412	CacheServerStats	ketchum.gemstone.com-0.0.0.0/0.0.0.0:5522
09/02 18:17:49	1	1004	3412	CachePerfStats	RegionStats-partitionMetaData
09/02 18:17:49	1	1004	3412	CachePerfStats	RegionStats-partition-test
09/02 18:17:48	1	1005	3412	CachePerfStats	cachePerfStats
09/02 18:17:50	1	1002	3412	CacheClientNotifierStatistics	cacheClientNotifierStats

numKeys
numUpdates
numUses
numValues
updatesInProgress
updateTime-IndexStats
usesInProgress
useTime

Chart:



points: 3 min: 0.0 max: 1.0 mean: 0.333333 sd: 0.57735

1. gemfire66_maint/cq/cqserver ketchum.gemstone.com GemFire 6.6.0.1 #shobhit 32

1. sampleIndex-2, numKeys
2. sampleIndex-2, numUpdates/sec
3. sampleIndex-2, numUses/sec
4. sampleIndex-2, numValues

VSD System Requirements

This topic lists the sytem requirements necessary to run VMware GemFire for TAS.

VSD works on Linux, macOS, and Windows platforms.

64-Bit Platform Support

VSD is a 32-bit application. If you are running VSD on a 64-bit operating system, you may need to install 32-bit OS libraries to run the application if they are not already installed. On Linux, to find out which libraries are missing you can try running the following:

```
ldd PRODUCT-DIR/tools/vsd/bin/vsdwishLinux
```

For 64-bit Windows, you can modify the scripts and executables as described in the note below.

Windows 7 and Later Support

To use VSD on Windows 7, perform the following steps:

1. Start Windows Explorer and navigate to the `PRODUCT-DIR\tools\vsd\bin\` directory, where `PRODUCT-DIR` corresponds to the installation location.
2. Right click and select properties for `vsd.bat`.
3. Select the Compatibility tab.
4. Click “Run this program in compatibility mode for” and then select Windows XP SP3.
5. Repeat steps for all the other executables in the `bin` directory.

Installing and Running VSD

This topic describes installing and running VSD for use with VMware GemFire for TAS.

Install VSD

VSD is a free analysis tool and is provided as-is. See VSD System Requirements to view a list of platforms that are known to work with VSD.

VSD is available for download from the VMware GemFire for TAS page on VMware Tanzu Network. It can be installed anywhere, but if Apache Geode is installed in `PRODUCT-DIR`, you may wish to install VSD in `PRODUCT-DIR/tools/vsd` so it can be conveniently launched from gfsh using the `start vsd` command.

Download the VSD archive, usually named something like `pivotal-gemfire-vsd.zip`, and unpack it in the directory of your choosing. For this example, assume and VSD was downloaded, unzipped, and installed in `PRODUCT-DIR/tools/vsd`.

Note: VSD is a 32-bit application. If you are running VSD on a 64-bit machine, you may need to install 32-bit OS libraries to run the application if they are not already installed. On Linux, to find out which libraries are missing, run the following `ldd` command:

```
ldd PRODUCT-DIR/tools/vsd/bin/vsdwishLinux
```

The VSD tool installation has two subdirectories, `bin` and `lib`:

- `bin`. Contains scripts and binaries that can be used to run VSD on a variety of operating systems, including:

  - vsd - script for Solaris, Linux, and Mac
  - vsd.bat - script for Windows
  - vsdwishSunOS - binary for Solaris
  - vsdwishLinux - binary for Linux
  - vsdwishDarwin - binary for Mac
  - vsdwishWindows_NT.exe - binary for Windows

- `lib`. The jars and binary libraries needed to run VSD.

Start VSD

To start VSD, you can either execute the scripts directly or you can start VSD through the gfsh interface. To start VSD using the provided scripts, change directories to `PRODUCT-DIR/tools/vsd/bin` and the enter the following command at the prompt:

- **Windows:**

  ```
  prompt>vsd.bat
  ```

  Note: To run VSD on Windows 7 or later, go to the *`product-dir`*`/tools/vsd/bin` directory. Right-click on `vsd.bat` and select Properties. Click Compatibility and set it to **Windows XP**. Repeat this step for all other executables in the `bin` directory.

- **Linux/Unix, MacOS or Other OS:**

  ```
  $ vsd
  ```

To start VSD using `gfsh`, start a `gfsh` prompt and enter the following command:

```
gfsh>start vsd
```

Acquire the Statistics Files

VSD displays captured statistics. Those statistics must be copied from the TAS environment where the service instance is to the local environment where VSD is.

There are two ways to acquire the files. Choose one:

- After connecting to the service instance with `gfsh export logs` command with the `--stats-only` as directed in Export gfsh Logs.
- Use the BOSH command as specified in View Statistics Files and Logs. The statistics are in `/var/vcap/sys/log/gemfire-server/gemfire/statistics.gfs` for servers, and `/var/vcap/sys/log/gemfire-locator/gemfire/statistics.gfs` for locators.

Load a Statistics File into VSD

You have several options for loading a statistics file into VSD:

- Include the name of one or more statistics files on the VSD command line. Example:

  ```
  vsd STAT-FILE-NAME.gfs ...
  ```

- Browse for an existing statistics file through Main > Load Data File.
- Type the full path in the File entry box, then press Enter.
- Switch to a statistics file that you've already loaded by clicking the down-arrow next to the File entry.

After you load the data file, the VSD main window displays a list of entities for which statistics are available. VSD uses color to distinguish between entities that are still running (shown in green) and those that have stopped (shown in black).

The File > Auto Update is not supported, since the statistics file is static when downloaded.

.gfs Time Zone Information for Matching Statistics to Log Files

When opening a .gfs file, statistics are shown in the time zone used on the *local computer* where VSD is launched. This can made it harder to relate log files to statistics if the logs are from another time zone.

To open a VSD file with the time zone used when generating it, first you need to know in which time zone the .gfs file is created. To obtain this information, use the following command:

```
strings file.gfs | head
```

For example:

```
$ strings ObjLoader?-31-03.gfs | head
Hongkong
hklp162p.oocl.com
:GemFire? x.x.x
14:46:33 PST
Linux x.x.x
```

After you obtain the time zone, modify your local computer to use the time zone used when obtaining statistics in the .gfs file. For example, on a Mac computer, you can first list available time zones:

```
sudo systemsetup -listtimezones
```

And then export the specific timezone to your environment:

```
export TZ=<timezone>
```

For example, for Hong Kong:

```
export TZ=Asia/Hong_Kong
```

Then use VSD to open the .gfs file that will now display timestamps from the original time zone.

Viewing Statistics in VSD

This topic explains how to select statistics and view them using chart templates and customized charts in VMware GemFire for TAS.

Statistic Levels

Each statistic has a characteristic called a level that reflects the amount of background knowledge that you would need to use the statistic with understanding. You can set up VSD to list (in its main window and in associated charts) only those statistics that are at, or below, a certain level of complexity — common, advanced, or wizard.

To establish the levels of statistics that you want to display in VSD, choose the menu item Main > Statistic Level in the main VSD window.

Select Statistics for Viewing

1. In the VSD list, click the left mouse button to select the entity or entities you want to view.
   - Search for a specific session name or process ID. To find a specific entity, click the mouse in the process list, then press Ctrl-S. When the dialog box appears, enter the PID or name of the entity that you're looking for. VSD highlights the first entity with that PID or name. To find the next match, press Ctrl-S again. To select the highlighted item, click on it. When you're done, press Return.
   - Select all entities or by statistic or by type. Statistics are available for various entities, which are listed under the heading Type in the process list.

   Note: You can use the right mouse button to perform these functions:

   - Combine multiple entities into a single line. This can be quite helpful. For example, if you want to measure page reads per second for several hundred entities, you could select all the entities, then combine them into a single line in the chart, thus rendering the data much more readable.
   - Combine multiple entities from different files into a single line.
   - Eliminate flatlines — entities whose values are always zero.
   - Select Single File mode so only one loaded file can be enabled at a time.
   - Select for created lines to have absolute timestamps, which is useful when merging files.
2. Select a statistic for viewing from the statistics list just below the process list.
3. With the selecting statistic, do one of the following:
   - To display the statistic in a new chart, type the name of the chart in the Chart entry box, then click New Chart. (Note: If you don't explicitly specify a chart name, VSD will assign one for you.)
   - To display the statistic in an existing chart, select the chart name in the Chart entry box. Then click on Add Line.

4. To add another statistic to the chart, repeat steps 1 through 3.

Using VSD Chart Templates

VSD templates let you quickly add a set of lines to a chart. Templates are helpful if you find yourself performing the same task frequently in VSD — for example, monitoring the same five or six statistics. By creating a template for the statistics that you want to monitor most frequently, you can automate the task of building charts.

In your template, you can assign a filter for each statistic, to determine how much information is displayed for that statistic. You can also restrict the template to look for extreme conditions (for example, processes that are consuming 90% or more of the CPU).

VSD is shipped with a set of predefined templates, which are maintained in the .vsdtemplates file in your home directory.

Task	Procedure
Create a new chart from a template	In the VSD main window, choose the menu item Template > New Template Chart. This is a good way to display some of the more useful system statistics.
Apply a template to the chart you are viewing	In the Chart window, choose the menu item Chart > Add From Template. (Note: If you have zoomed in on a chart, the template filter is only applied to values within the zoomed range.)
Reread the `.vsdtemplates` template file into VSD after you've edited it	In the VSD main window, choose the menu item Template > Reload Template File.
Save the current chart as a template	In the Chart window, configure the chart as you desire, then choose the menu item Chart > Save Template. The template is saved to the `.vsdtemplates` file. If you save the current chart as a template, you may still need to edit the `.vsdtemplates` file so that you can give it a more useful name and make the information and patterns it captures more general.

Chart Menu (Chart Window)

To customize the way VSD displays statistics in your chart, you can choose items from the Chart window's Chart menu.

Chart Menu Item	Effect
Add from Template	Expand template and add resulting lines to chart.
Save Template	Save all lines on chart as a template.
Paste	Paste last item on clipboard.
Print	Print chart.
Snapshot	Write this chart as a graphic to `snapshot.gif`.
Help …	Open Help window.

Chart Menu Item	Effect
Zoom In	Zoom in to improve your view of the chart. After you choose this menu item, click to select one corner of the area that you want to zoom. Move the mouse pointer to the opposite corner of the zoom area, then click again. If you have a middle mouse button, you can quickly zoom in on an area by clicking the middle mouse button over it.
Zoom Out	Zoom out by using the menu button or by right-clicking in the chart window.
Compare Two Points	Log information by comparing two points.
Compute Scale All, Unscale All	Adjust the scale of the chart. This helps you view multiple statistics on the same axis.
Show Legend	Display the legend for this chart.
Time Format	Change the format of the time displayed along the X axis.
Show Time Axis Title, Show Left Axis Title, Show Right Axis Title	Display the title alongside the respective axes.
Show Current Values	Display the current X and Y values for the selected line at the top of the chart
Show Min and Max	Display the minimum and maximum values for the selected line at the top of the chart.
Show Line Stats	Display these statistics for the selected line: the number of data samples, the min, max, mean, and standard deviation. The statistics are calculated from all of the data points on the selected line in the region defined by the graph’s current X axis. (To change the region, select Zoom In or Zoom Out from the Chart menu.)
Show CrossHairs	Draw cross hairs on graph of item.
Show Grid Lines	Draw grid line on graph of item.
Close	Close chart window.

For additional information about the Chart window, choose Help from the Chart menu.

Line Menu (Chart Window)

Customize your VSD chart display using items from the Line menu (in the Chart window). Line menu commands operate on the currently selected line. To select a line, click on it or on its entry in the chart legend. VSD highlights the selected line.

Line Menu Item	Effect
Log Info	Display a log file showing the line statistics for all data samples in the region defined by the graph’s current X axis.
Log Delta	Measure the difference between two values on the selected line. Select the line before choosing this menu item then click on the two points whose difference you want to compute. VSD responds by displaying a log file showing the difference in time and value between the two points; the number of data samples in the selected line segment; and the min, max, mean, and standard deviation of those samples.

Line Menu Item	Effect
Compute Scale	Compute a scale value for the selected line that will make it visible on the current chart. You can also use the Scale entry box to manually change the scale. The default scale value is 1.
Unscale	Reverse the effect of Compute Scale.
Graph on Left Axis	Display the Y axis for the selected line to the left of the chart. Otherwise, the Y axis is displayed to the right. You can use this to view multiple lines on the same chart, by graphing large values on one axis and small values on the opposite axis.
Symbol	Select a new symbol.
Style	Select a line style for connecting points: linear (default), step, natural, quadratic.
Update	Update files used by current line.
Add Lines	Add a line to the current line.
Diff Lines	Remove a line from the current line.
Divide Lines	Divide current line by another line.
Normalize	Normalize current line.
Trim Left, Right	Trim line to left or right of a data point.
Untrim Left, Right	Undo any trim line operations.
Copy	Copy current line to clipboard.
Cut	Cut current line to clipboard.
Delete	Remove the selected line from this chart.

Customizing Your VSD Chart

You can customize and manipulate a VSD chart in many ways:

- To select a line in the chart, click on it or on its entry in the chart legend.
- To delete a line from the chart, click the middle mouse button (if available) on its entry in the chart legend. Alternatively, select the line's entry in the chart legend and choose Line >

Delete.

- To find out about a specific point in a chart, hold the mouse pointer over it.

View Statistic Information

To view a description of the most recently selected statistic, along with information about its type, level, and default filter, go to the VSD main window, then choose the menu item Main > Show Statistic Info.

In the Statistic Information window, you can redefine the level and default filter for any VSD statistic.

- The statistic's level — common, advanced, or wizard — allows you to determine whether the statistic is displayed in the VSD statistic list.
- Whenever you add a line to a chart, the filter determines how information is displayed for the selected statistic.

Default Filter	Effect
Default	No Filter if the statistic represents a snapshot of a value. PerSecond if the statistic represents a value that always increases.
No Filter	Display the raw values for the statistic with no filtering.
PerSample	Display the difference between two consecutive samples of the statistic.
PerSecond	Display the difference between two consecutive samples of the statistic, divided by the number of elapsed seconds between the two samples.
Aggregate	Display a running total of per-sample deltas for the statistic. Reset to zero when the delta is zero or changes direction.

Once you have added the line to a chart, you can override its default filter by specifying a new filter from the drop-down menu at the top of the Chart window.

If you leave the Statistic Information window up as you work, it changes to reflect the current statistic. In this way, you can get a quick explanation of any statistic that you're currently examining.

Quick Guide to Useful Statistics

This topic explains how to select statistics and view them using chart templates and customized charts in VMware GemFire for TAS.

A large number of statistics about VMware GemFire for TAS are intended only for product support and engineering. This topic describes the most important categories and the useful statistics they contain.

For a reference on Geode statistics, see Geode Statistics List.

Runtime Configuration

As the name implies, these statistics can help with verifying the runtime configuration of a service instance:

- The **number of peer nodes** (i.e. servers or peer accessors) in the system: *DistributionStats:nodes*. This value should be the same for every node in the system.
- The **number of clients and client connections for each server**: *CacheServerStats: currentClients*, and *currentClientConnections*
- The **number of data entries**:
  - *CachePerfStats:entries*. Each region has its own *CachePerfStats* instance per JVM named *RegionStats-REGION-NAME*, or *RegionStats-partition-REGION-NAME* for partitioned regions. Its entries statistic is the number of entries for that region in the JVM.
  - *DiskRegionStatistics* (a per region disk statistic category about the region's disk use): *entriesInVM*, and *entriesOnlyOnDisk* show the number of entries in the JVM (which can also be on disk too), and the number of entries that are only on disk, respectively.
- **Partitioned Region Configuration**: One of the main parameters of Partitioned Region (PR) configuration is the primary bucket distribution. To make sure that primary buckets for a PR are evenly distributed, check the *PartitionedRegionStats.primaryBucketCount* statistic for each partition. This statistic shows the number of primary buckets in a partition.

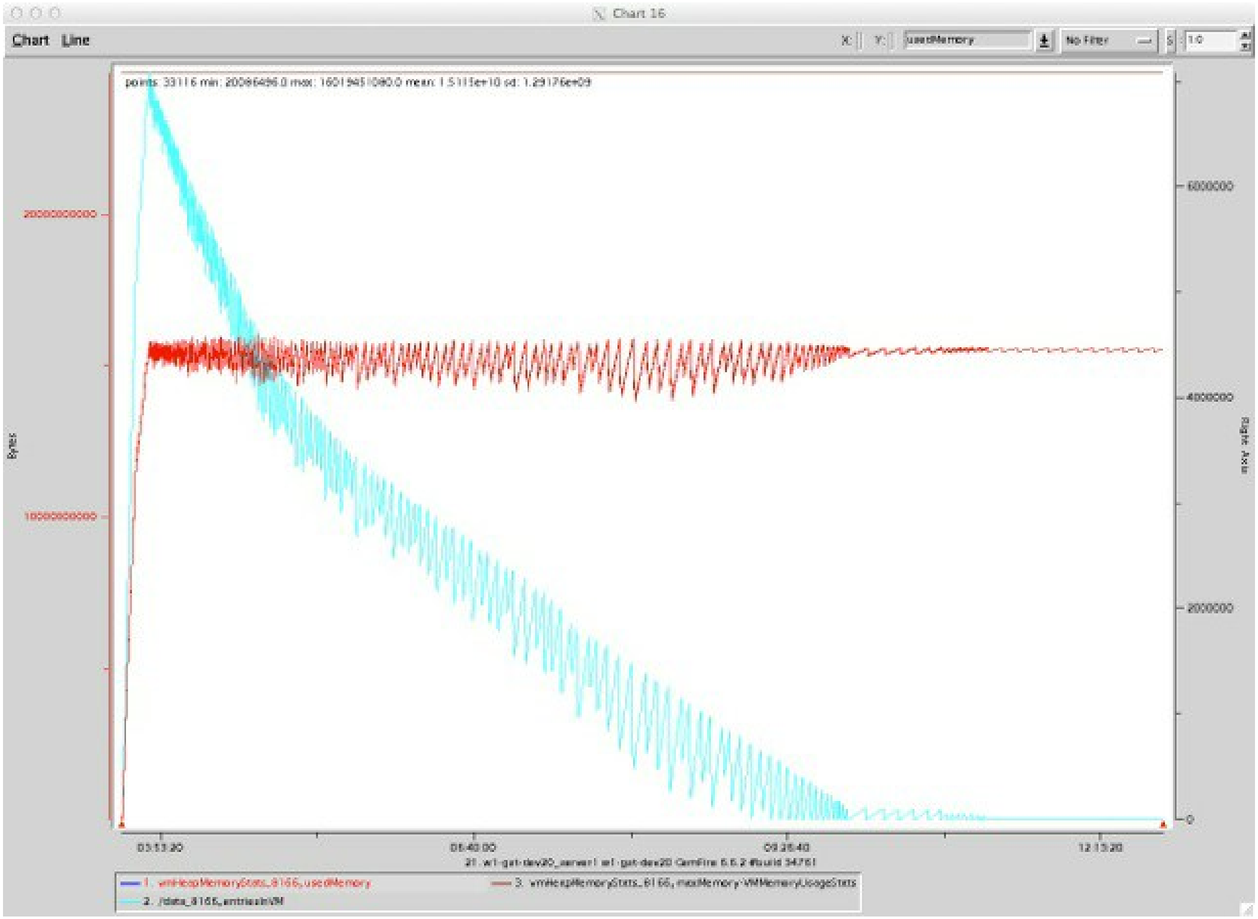
Resources

The resources that are vital for normal operation and performance are memory, file descriptors (most importantly sockets, then files), CPU, network, and disk (when disk operations, such as overflow and persistence, are involved). The following stats cover all those:

- **Memory**: There are several stats categories that show memory usage, for different types and granularity of memory.
  - **Heap**: *VMMemoryUsageStats:vmHeapMemoryStats* are all about heap usage, as are the memory stats under *VMStats:vmStats*: *freeMemory*, *totalMemory*, and *maxMemory*.
  - **Non-heap memory**: *VMMemoryUsageStats:vmNonHeapMemoryStats*.
  - **System-wide memory stats as reported by the OS**: The OS statistic category (e.g. *LinuxSystemStats* on Linux) includes various system level memory statistics, such as *freeMemory*, which shows the free memory on the host (as opposed to related to the JVM process), *physicalMemory* (total physical memory on the host), paging related statistics (*pagesSwappedIn*, *pagesSwappedOut*, *unallocatedSwap*).
  - **Client and gateway queue sizes**: while not actual resources, these queues may be responsible for increased memory usage, so it's good to keep them in mind when investigating memory issues. The client queue stats are in *ClientSubscriptionStats* category: *eventsQueued*, and *eventsRemoved*. The difference between the two is the current queue size. The gateway queue stats are in *GatewaySenderStatistics* category: *eventQueueSize* is the size of the queue.
- **File Descriptors**: file descriptor related statistics are captured in the category *VMStats: fdsOpen* and *fdLimit* show the number of open file descriptors, and the limit on file descriptors for the host, respectively.

- **CPU**: The CPU usage is captured in OS statistic category, e.g. *LinuxSystemStats*. The statistic *cpuActive* shows the percentage of the total available CPU time that has been used in a non-idle state.
- **System load**: OS statistic category (e.g. *LinuxSystemStats*) includes the *loadAverage1*, *loadAverage5*, *loadAverage15* statistics, which show the average system load for 1, 5, and 15 minutes.
- **Network**: OS stats also include network related stats for received (*recv*) and transmitted traffic (*recvBytes*, *xmitBytes*, *recvErrors*, *xmitErrors*).
- **Disk: DiskDirStatistics:diskSpace** shows the amount of disk space used for disk storage on a given disk. Above mentioned *entriesOnlyOnDisk*, and *entriesInVM* from *DiskRegionStatistics* are useful for determining the distribution of data between memory and disk, for regions that use disk overflow/persistence.

The following chart is an example of examining the *vmHeapMemoryStats* in relation to the *entriesInVM* statistic.



Throughput for Different Operations

There are several stat categories that capture the throughput for gemfire operations: *CachePerfStats* (non-PR, and PR specific), and*CacheServerStats*, which capture throughput statistics with respect to clients. Note that the PR specific instances of *CachePerfStats* cover only the specific partitioned regions, while the *CachePerfStats* instance includes aggregate stats for all non-PR regions.

- **CachePerfStats** category includes the following stats (all measured in the number of operations per second):

  - *gets*: the number of successful `gets`
  - *puts*: the number of times an entry has been added or replaced as a result of a local operation (put, create, or get which results in a load, netsearch, or netload of a value)
  - *updates*: the number of updates originating remotely
  - *putalls*: the number of `putAll` operations
  - *destroys*: the number of destroys
  - *Function execution: FunctionService*
  - *Queries: queryExecutions*: the number of query executions
  - *Transactions: txCommits*, *txFailures*, *txRollbacks*: the number of successful, failed, and rolled back transactions, respectively

- **CacheServerStats** category includes the following throughput stats for client operations on the cache server:
  - *getRequests, getResponses*
  - *getAllRequests, getAllResponses*
  - *putRequests, putResponses*
  - *putAllRequests, putAllResponses*
  - *queryRequests, queryResponses*
- **Disk operations**: If any disk related statistic categories are present in VSD, it means that there is disk activity (some entries are on disk). Presence of disk operations may explain a drop in throughput, as disk use slows things down.
  - *DiskRegionStatistics* (statistics about a region disk use): *writes, writeTime, writtenBytes, reads, readTime, readBytes*
  - *DiskStoreStatistics* are statistics about a specific disk store’s use of disk. In addition to write/read as those in *DiskRegionStatistics*, this category includes *queueSize* statistic, which shows the current number of entries in the asynchronous queue waiting to be flushed to disk.

Application Development

This section introduces design patterns for structuring app design for VMware GemFire for Tanzu Application Service. It presents a minimal view of cluster data organization to help with data architecture design. A complete presentation of a cluster's capabilities is in the VMware GemFire Product Documentation.

An app that interacts with a VMware GemFire for Tanzu Application Service service instance will use the VMware GemFire cluster within that service instance. Architecting the data storage for the app requires some familiarity with how clusters work.

In this section:

- Design Patterns
    - The Inline Cache
    - The Cache-Aside Cache
    - Bidirectional Replication Across a WAN
    - Blue-Green Disaster Recovery
    - CQRS Pattern Across a WAN
    - Hub-and-Spoke Topology with WAN Replication
    - Follow-the-Sun Pattern
- Region Design
    - Keys
    - Partitioned Regions
    - Replicated Regions
    - Persistence
    - Overflow
    - Regions as Used by the App
    - An Example to Demonstrate Region Design
- Handling Events
- Example Applications
    - A Simple Java App
    - A Spring Boot App
- Running an App
- Developing an App Under TLS

- Tomcat Session State Caching
  - Enable Session State Caching
  - Deactivate Near Caching Within the App
  - Create a Custom Buildpack
  - Use an External Repository for Configuration
- Spring Session Caching

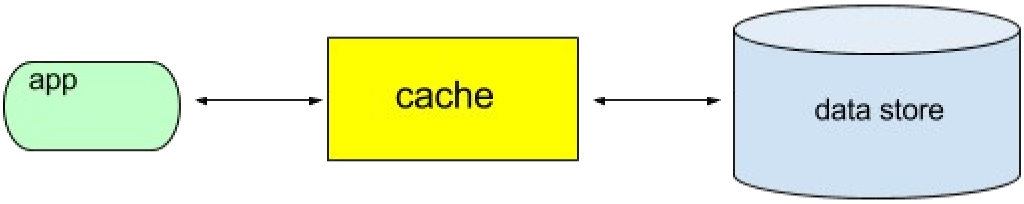
Design Patterns

This topic discusses design patterns for use with VMware GemFire for TAS.

The design patterns in this section are intended as overviews of common and useful configurations. Effective implementations will require planning—consult with a system architect to fill in the design details for your app.

The Inline Cache

An inline cache places the caching layer between the app and the backend data store.



The app will want to accomplish CRUD (create, read, update, delete) operations on its data. The app's implementation of the CRUD operations result in cache operations that break down into cache lookups (reads) and/or cache writes.

The algorithm for a cache lookup quickly returns the cache entry when the entry is in the cache. This is a cache hit. If the entry is not in the cache, it is a cache miss, and code on the cache server retrieves the entry from the backend data store. In the typical implementation, the entry returned from the backend data store on a cache miss is written to the cache, such that subsequent cache lookups of that same entry result in cache hits.

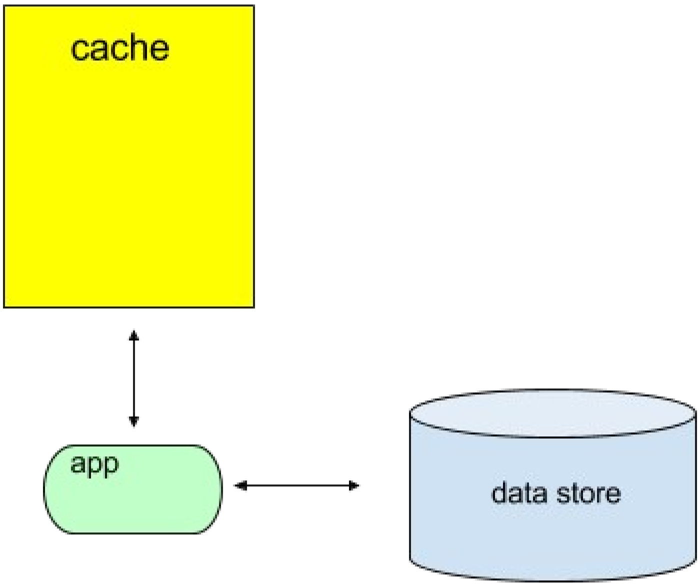
The implementation for a cache write typically creates or updates the entry within the cache. It also creates or updates the data store in one of the following ways:

- Synchronously, in a write-through manner. Each write operation from the app is sent on to be written to the backend data store. After the backend data store write finishes, the value is also written to the cache. The app blocks until the writes to both the backend data store and the cache complete.
- Asynchronously, in a write-behind manner. The cache gets updated, and the value to be written to the backend data store gets queued. Control then returns to the app, which continues independent of the write to the backend data store.

Developers design the server code to implement this inline-caching pattern. See Setting Up Servers for an Inline Cache for details about the custom server code and how to configure an inline cache.

The Cache-Aside Cache

The cache-aside pattern of caching places the app in charge of communication with both the cache and the backend data store.



The app will want to accomplish CRUD (CREATE, READ, UPDATE, DELETE) operations on its data. That data may be

- in both the data store and the cache
- in the data store, but not in the cache
- not in either the data store or the cache

The app's implementation of the CRUD operations result in cache operations that break down into cache lookups (reads) and/or cache writes.

The algorithm for a cache lookup returns the cache entry when the entry is in the cache. This is a cache hit. If the entry is not in the cache, it is a cache miss, and the app attempts to retrieve the entry from the data store. In the typical implementation, the entry returned from the backend data store is written to the cache, such that subsequent cache lookups of that same entry result in cache hits.

The cache-aside pattern of caching leaves the app free to implement whatever it chooses if the data store does not have the entry.

The algorithm for a cache write implements one of these:

- The entry is either updated or created within the data store, and the entry is updated within or written to the cache.

- The entry is either updated or created within the backend data store, and the copy currently within the cache is invalidated.

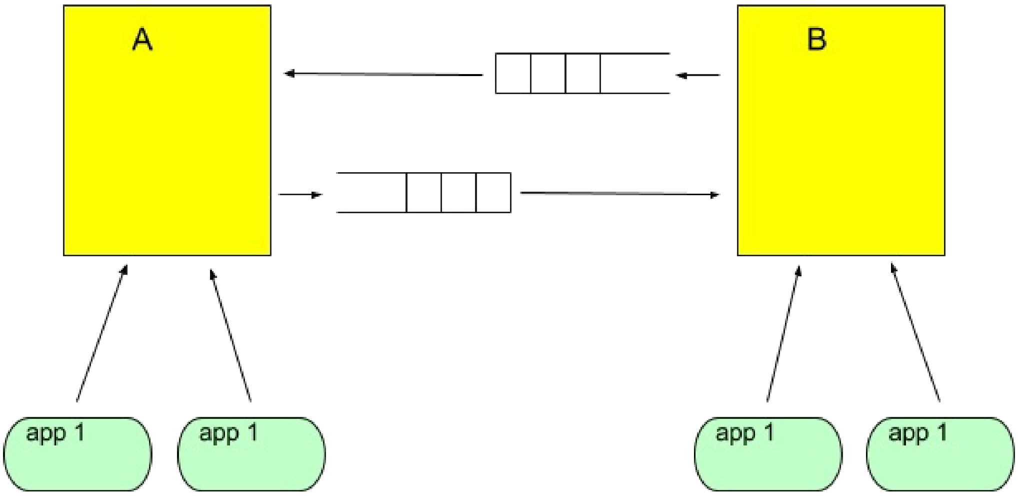
Note: SDG (Spring Data Geode) supports the cache-aside pattern, as detailed at Configuring Spring's Cache Abstraction.

Bidirectional Replication Across a WAN

Two VMware GemFire for Tanzu Application Service service instances may be connected across a WAN to form a single distributed system with asynchronous communication. The cluster within each of the GemFire for TAS service instances will host the same region. Updates to either GemFire for TAS service instance are propagated across the WAN to the other GemFire for TAS service instance. The distributed system implements an eventual consistency of the region that also handles write conflicts which occur when a single region entry is modified in both GemFire for TAS service instances at the same time.

In this active-active system, an external entity implements load-balancing by directing app connections to one of the two service instances. If one of the GemFire for TAS service instances fails, apps may be redirected to the remaining service instance.

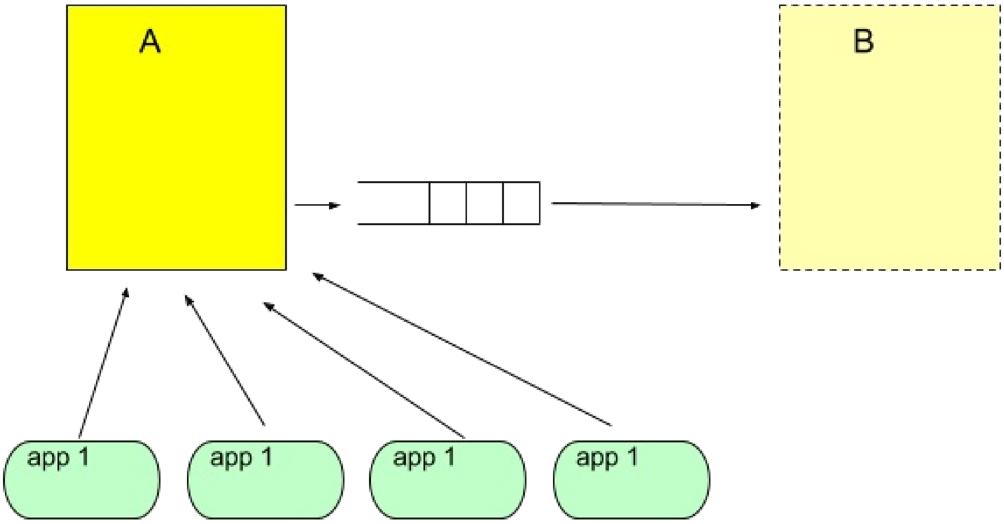
This diagram shows multiple instances of an app interacting with one of the two GemFire for TAS service instances, cluster A and cluster B. Any change made in cluster A is sent to cluster B, and any change made in cluster B is sent to cluster A.



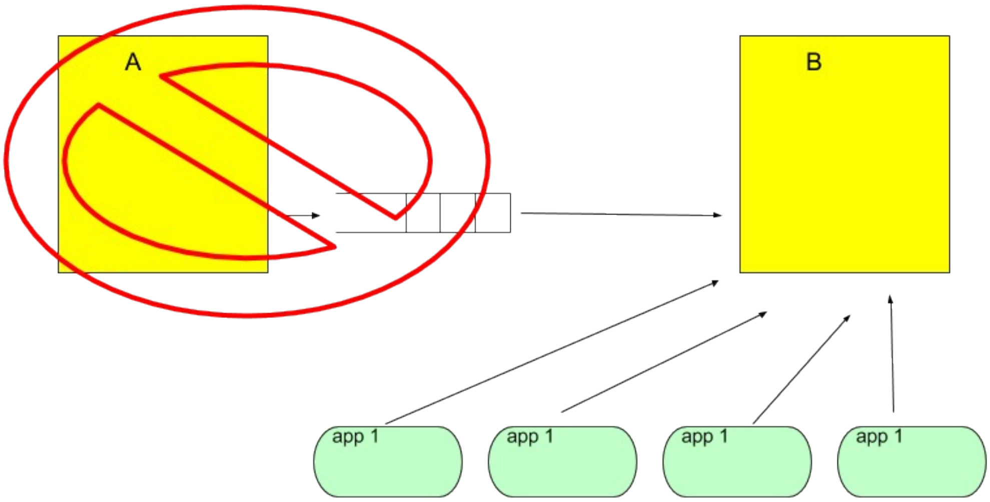
Blue-Green Disaster Recovery

Two GemFire for TAS service instances may be connected across a WAN to form a single distributed system with asynchronous communication. An expected use case propagates all changes to a region’s data from the cluster within one service instance (the primary) to the other, where both service instances reside in the same foundation. The replicate increases the fault tolerance of the system by acting as a “hot” spare. In the scenario of the failure of an entire data center or an availability zone, you can rebind apps to the replicate and restage them. The replicate then takes over as the primary.

In this diagram, cluster A is primary, and it replicates all data across a WAN to cluster B.



If cluster A fails, you can manually rebind and restage the apps so that cluster B takes over.

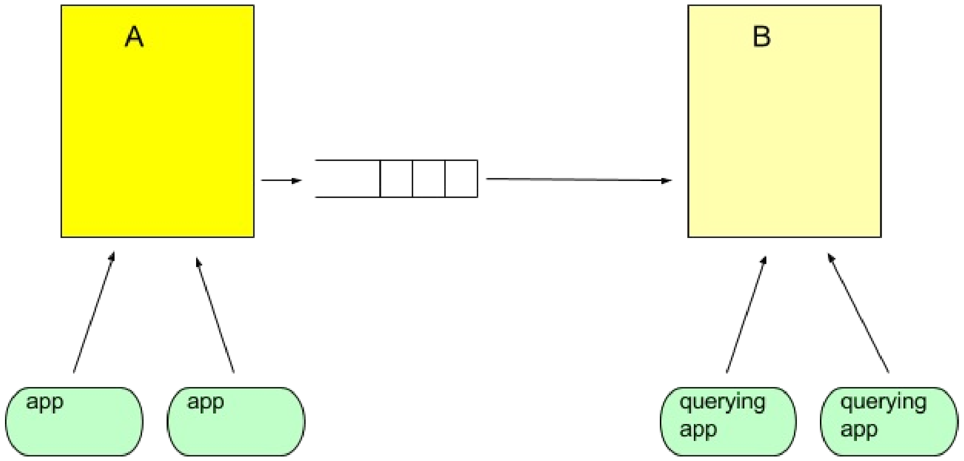


CQRS Pattern Across a WAN

Two GemFire for TAS service instances may be connected across a WAN to form a single distributed system that implements a CQRS (Command Query Responsibility Segregation) pattern. Within this pattern, commands are those that change the state, where state is represented by region contents. All region operations that change state are directed to the cluster within one GemFire for TAS service instance. The changes are propagated asynchronously to the cluster within the other

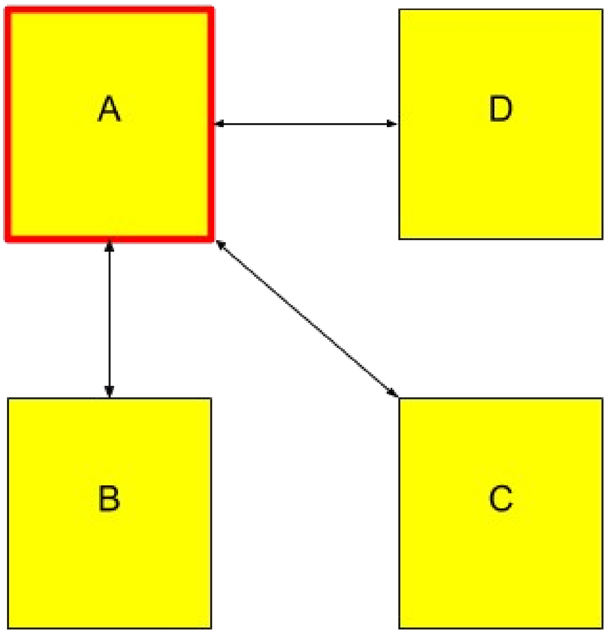
GemFire for TAS service instance via WAN replication, and that other cluster provides only query access to the region data.

This diagram shows an app that may update the region within the GemFire for TAS service instance of cluster A. Changes are propagated across the WAN to cluster B. The app bound to cluster B may only query the region data; it will not create entries or update the region.



Hub-and-Spoke Topology with WAN Replication

Multiple GemFire for TAS service instances connected across a WAN form a single hub and a set of spokes. This diagram shows GemFire for TAS service instance A is the hub, and GemFire for TAS service instances B, C, and D are spokes.



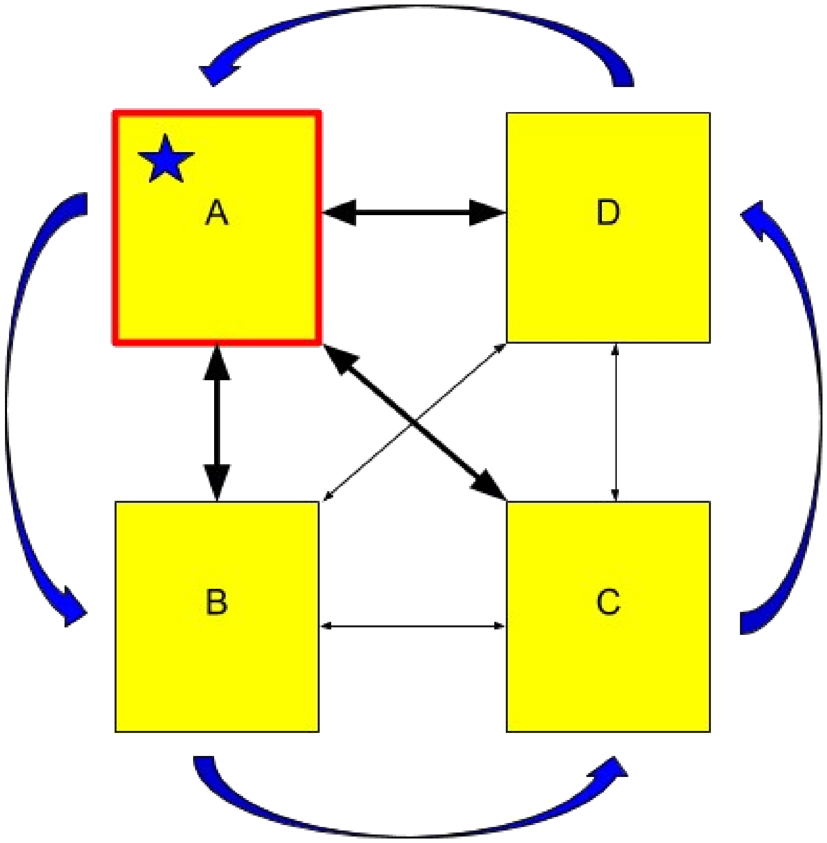
A common implementation that uses this topology directs all app operations that write or update region contents to the hub. Writes and updates are then propagated asynchronously across the WAN from the hub to the spokes.

Follow-the-Sun Pattern

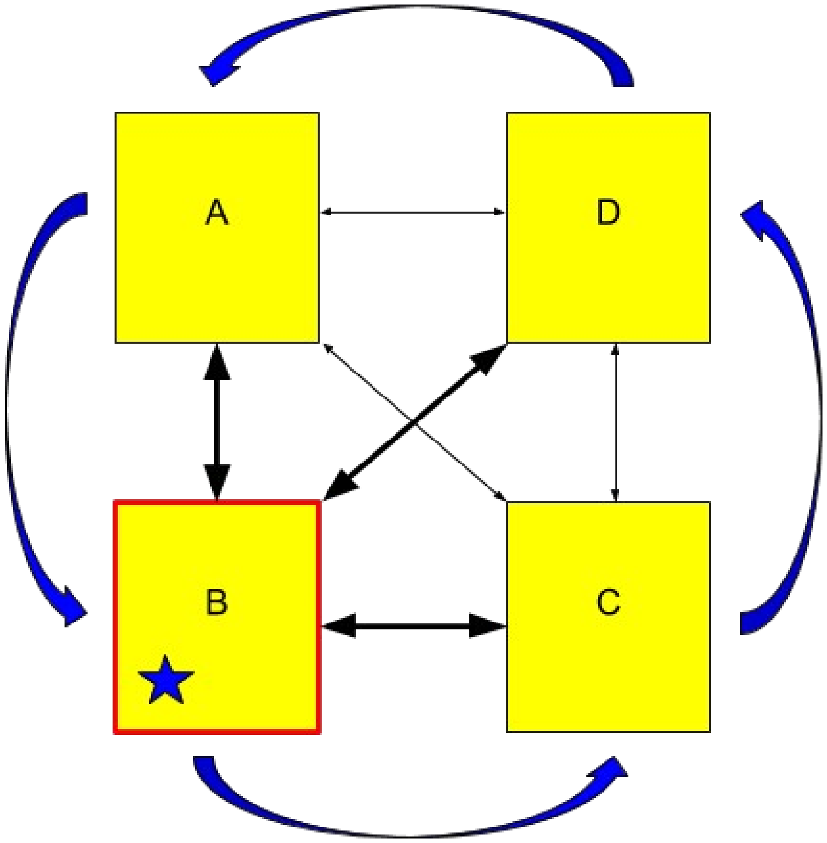
Performance improves when operation requests originate in close proximity to the service instance that handles those requests. Yet many data sets are relevant and used all over the world. If the most active location for write and update operations moves over the course of a day, then a performant design pattern is a variation on the hub-and-spoke implementation that changes which GemFire for TAS service instance is the hub to the most active location.

Form a ring that contains each GemFire for TAS service instance that will act as the hub. Define a token to identify the hub. Over time, pass the token from one GemFire for TAS service instance to the next, around the ring.

This diagram shows GemFire for TAS service instance A is the hub, as it has the token, represented in this diagram as a star. GemFire for TAS service instances B, C, and D are spokes. Write and update operations are directed to the hub.



This diagram shows that the token has passed from A to B, and B has become the hub.



Region Design

This topic describes region design considerations and requirements to use when running VMware GemFire for TAS.

Cached data are held in regions. Each entry within a region is a key/value pair. The choice of key and region type affect the performance of the design. There are two basic types of regions: partitioned and replicated. The distinction between the two types is based on how entries are distributed among servers that host the region.

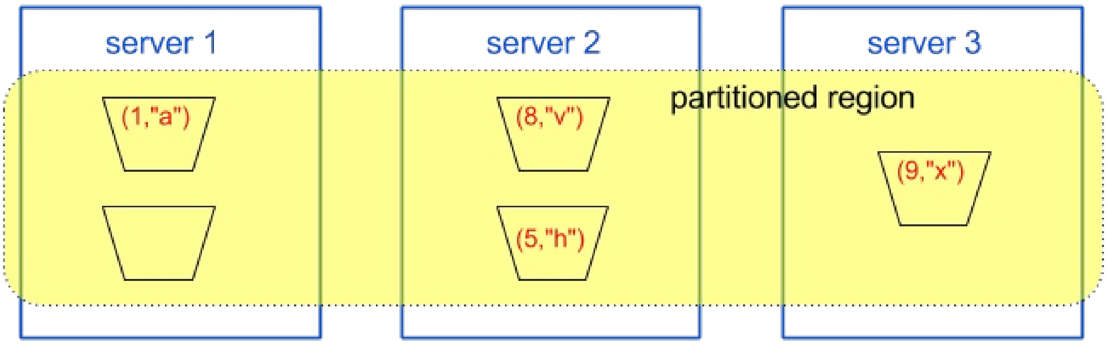
Keys

Each region entry must have a unique key. Use a wrapped primitive type of `String`, `Integer`, or `Long`. Experienced designers have a slight preference of `String` over `Integer` or `Long`. Using a `String` key enhances the development and debugging environment by permitting the use of a REST API (Swagger UI), as it only works with `String` types.

Partitioned Regions

A partitioned region distributes region entries across servers by using hashing. The hash of a key maps an entry to a bucket. A fixed number of buckets are distributed across the servers that host the region.

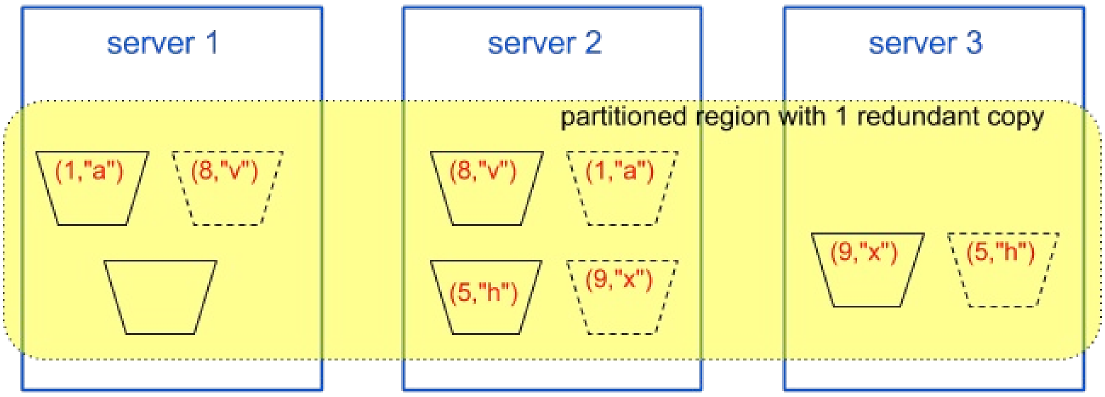
Here is a diagram that shows a single partitioned region (highlighted) with very few entries to illustrate partitioning.



A partitioned region is the proper type of region to use when one or both of these situations exist:

- The region holds vast quantities of data. There may be so much data that you need to add more servers to scale the system up. VMware GemFire for Tanzu Application Service can be scaled up without downtime; to learn more, see Updating a VMware GemFire for TAS Service Instance.
- Operations on the region are write-heavy, meaning that there are a lot of entry updates.

Redundancy adds fault tolerance to a partitioned region. Here is that same region, but with the addition of a single redundant copy of each each entry. The buckets drawn with dashed lines are redundant copies. Within the diagram, the partitioned region is highlighted.



With one redundant copy, the cluster can tolerate a single server failure or a service upgrade without losing any region data. With one less server, the cluster revises which server holds the primary copy of an entry.

A partitioned region without redundancy permanently loses data during a service upgrade or if a server goes down. All entries hosted in the buckets on the failed server are lost.

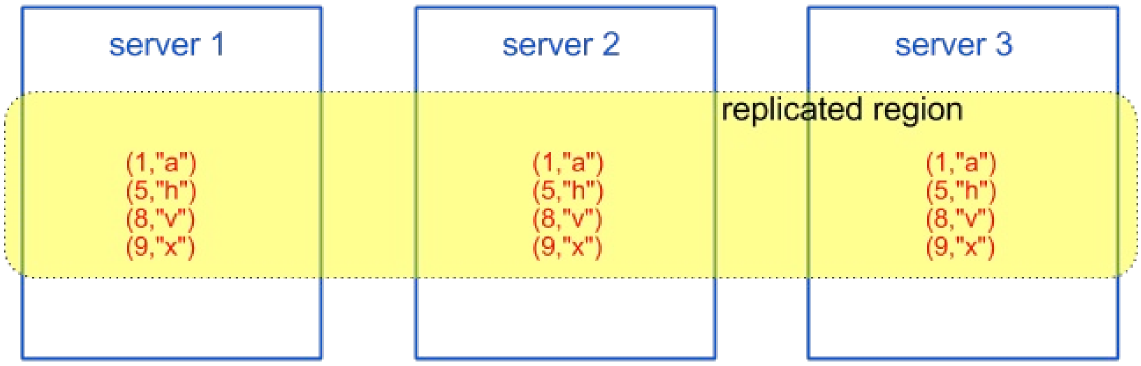
Partitioned Region Types for Creating Regions on the Server

Region types associate a name with a particular region configuration. The type is used when creating a region. Although more region types than these exist, use one of these types to ensure that no region data is lost during service upgrades or if a server fails. These partitioned region types are supported:

- `PARTITION_REDUNDANT` Region entries are placed into the buckets that are distributed across all servers hosting the region. In addition, the cluster keeps and maintains a declared number of redundant copies of all entries. The default number of redundant copies is 1.
- `PARTITION_REDUNDANT_HEAP_LRU` Region entries are placed into the buckets that are distributed across all servers hosting the region. The cluster keeps and maintains a declared number of redundant copies. The default number of redundant copies is 1. As a server (JVM) reaches a heap usage of 75% of available heap, the server destroys entries as space is needed for updates. The oldest entry in the bucket where a new entry lives is the one chosen for destruction.
- `PARTITION_PERSISTENT` Region entries are placed into the buckets that are distributed across all servers hosting the region, and all servers persist all entries to disk.
- `PARTITION_REDUNDANT_PERSISTENT` Region entries are placed into the buckets that are distributed across all servers hosting the region, and all servers persist all entries to disk. In addition, the cluster keeps and maintains a declared number of redundant copies of all entries. The default number of redundant copies is 1.
- `PARTITION_REDUNDANT_PERSISTENT_OVERFLOW` Region entries are placed into the buckets that are distributed across all servers hosting the region, and all servers persist all entries to disk. In addition, the cluster keeps and maintains a declared number of redundant copies of all entries. The default number of redundant copies is 1. As a server (JVM) reaches a heap usage of 75% of available heap, the server overflows entries to disk when it needs to make space for updates.
- `PARTITION_PERSISTENT_OVERFLOW` Region entries are placed into the buckets that are distributed across all servers hosting the region, and all servers persist all entries to disk. As a server (JVM) reaches a heap usage of 75% of available heap, the server overflows entries to disk when it needs to make space for updates.

Replicated Regions

Here is a replicated region with very few entries (four) to illustrate the distribution of entries across servers. For a replicated region, all servers that host the region have a copy of every entry.



The cluster maintains copies of all region entries on all servers. The cluster takes care of distribution and keeps the entries consistent across the servers.

A replicated region is the proper type of region to use when one or more of these situations exist:

- The region entries do not change often. Each write of an entry must be propagated to all servers that host the region. As a consequence, where there are many concurrent write accesses, the resulting subsequent writes to all other servers hosting the region cause a loss of performance.
- The overall quantity of entries is not so large as to push the limits of memory space for a single server. The TAS for VMs service plan sets the server memory size.
- The entries of a region are frequently accessed together with entries from other regions. The entries in the replicated region are always available on the server that receives the access request, leading to better performance.

Replicated Region Types for Creating Regions on the Server

Region types associate a name a particular region configuration. These replicated region types are supported:

- `REPLICATE` All servers hosting the region have a copy of all entries.
- `REPLICATE_HEAP_LRU` All servers hosting the region have a copy of all entries. As a server (JVM) reaches a heap usage of 75% of available heap, the server destroys entries as it needs to make space for updates.
- `REPLICATE_PERSISTENT` All servers hosting the region have a copy of all entries, and all servers persist all entries to disk.
- `REPLICATE_PERSISTENT_OVERFLOW` All servers hosting the region have a copy of all entries. As a server (JVM) reaches a heap usage of 75% of available heap, the server overflows entries to disk as it need to make space for updates.

Persistence

Persistence adds a level of fault tolerance to a GemFire for TAS service instance by writing all region updates to local disk. Disk data, hence region data, is not lost upon cluster failures that exceed redundancy failure tolerances. Upon cluster restart, regions are reloaded from the disk, avoiding the slower method of restart that reacquires data using a database of record.

Creating a region with one of the region types that includes `PERSISTENT` in its name causes the instantiation of local disk resources with these default properties:

- Synchronous writes. All updates to the region generate operating system disk writes to update the disk store.
- The disk size is part of the instance configuration. See Configure Service Plans for details on setting the persistent disk types for the server. Choose a size that is at least twice as large as the expected maximum quantity of region data, with an absolute minimum size of 2 GB. Region data includes both the keys and their values.
- Warning messages are issued when a 90% disk usage threshold is crossed.

Overflow

Region overflow is an eviction action that keeps heap memory space usage below a fixed threshold of 75% of available heap memory space. For a region that pushes at the limits of memory usage, overflow reduces the number of or eliminates pauses for stop-the-world garbage collection.

The action of overflow writes one or more least recently used region entries to disk to make room in memory for another entry. The least recently used entry within the bucket to which new entry maps is the chosen overflow victim. The key of the victim remains in memory, but the value of the entry is written to disk. An operation on an entry that has overflowed to disk causes the entry to be swapped back into memory.

If using a region type with overflow, be sure to configure a plan with sufficient disk space for the Server VM, allocating at least the minimums given for the *Persistent disk type for the Server VMs*, as described in Configure Tile Properties.

If no disk store is created, region creation with a region type that uses a disk store will cause the creation of one called `DEFAULT` with a default size (2 Gbyte). Alternatively, create the disk store using `gfsh`, as described in Working with Disk Stores. Then, create the region using the `--disk-store` option to specify the created disk store. If the disk store has been created, but the gfsh region creation command neglects to specify a disk store, a new `DEFAULT` disk store will be created and used. For more details on region creation options, see the VMware GemFire create region Command Reference Page.

Regions as Used by the App

The client accesses regions hosted on the servers by creating a cache and the regions. The type of the client region determines if data is only on the servers or if it is also cached locally by the client in addition to being on the servers. Locally cached data can introduce consistency issues, because region entries updated on a server are not automatically propagated to the client’s local cache.

Client region types associate a name with a particular client region configuration.

- `PROXY` forwards all region operations to the servers. No entries are locally cached. Use this client region type unless there is a compelling reason to use one of the other types. Use this type for all Twelve-Factor apps in order to assure stateless processes are implemented. Not caching any entries locally prevents the app from accidentally caching state.

- `CACHING_PROXY` forwards all region operations to the servers, and entries are locally cached.
- `CACHING_PROXY_HEAP_LRU` forwards all region operations to the servers, and entries are locally cached. Locally cached entries are destroyed when the app's usage of cache space causes its JVM to hit the threshold of being low on memory.

An Example to Demonstrate Region Design

Assume that on servers, a region holds entries representing customer data. Each entry represents a single customer. With an ever-increasing number of customers, this region data is a good candidate for a partitioned region.

Perhaps another region holds customer orders. This data also naturally maps to a partitioned region. The same could apply to a region that holds order shipment data or customer payments. In each case, the number of region entries continues to grow over time, and updates are often made to those entries, making the data somewhat write heavy.

A good candidate for a replicated region would be the data associated with the company's products. Each region entry represents a single product. There are a limited number of products, and those products do not often change.

Consider that as the client app goes beyond the most simplistic of cases for data representation, the GemFire for TAS instance hosts all of these regions such that the app can access all of these regions. Operations on customer orders, shipments, and payments all require product information. The product region benefits from access to all its entries available on all the cluster's servers, again pointing to a region type choice of a replicated region.

Handling Events

This topic explains discusses events in VMware GemFire for TAS.

The cluster within a VMware GemFire for Tanzu Application Service service instance can handle events. An app registers interest in a particular event, and when the server detects the event, an app-defined callback (also called a handler or a listener) is invoked to handle the event.

There are three aspects to configuring and implementing an event:

- the app defines or specifies the event
- the app registers the event with or identifies the event to the system component that will detect the event
- the app defines the callback, which handles the event

Continuous Queries

Continuous queries use an OQL query on region data to define an event. A change in query results is the event. The app registers both the query and the callback to set up a continuous query. Each region operation invokes the query, leading to the naming of this type of event handling as continuous.

See Querying with OQL for details on the Object Query Language (OQL) and generating queries.

Example Applications

This topic provides links to example applications intended to provide insight into aspects of app design for VMware GemFire for TAS Service.

- A Simple Java App
- A Spring Boot App

An Example Java App

This topic presents an example Java app to use with VMware GemFire for TAS.

The sample Java client app at https://github.com/cf-gemfire-org/cloudcache-sample-app.git demonstrates how to connect an app to a service instance.

These instructions assume:

- A GemFire for TAS service instance is running.
- You have Cloud Foundry credentials for accessing the GemFire for TAS service instance.
- You have a service key for the GemFire for TAS service instance.
- You have a login on the Pivotal Commercial Maven Repository at https://commercial-repo.pivotal.io.
- You have a `gfsh` client of the same version as is used within your GemFire for TAS service instance.

Follow these instructions to run the app.

1. Clone the sample Java app from https://github.com/cf-gemfire-org/cloudcache-sample-app.git.
2. Update your clone of the sample Java app to work with your GemFire for TAS service instance:
    - Modify the manifest in `manifest.yml` by replacing `service0` with the name of your GemFire for TAS service instance.
    - Replace the username and password in the `gradle.properties` file with your username and password for the Pivotal Commercial Maven Repository.
    - Update the VMware GemFire version in the dependencies section of the `build.gradle` file to be the same as the version within your GemFire for TAS service instance.
3. Build the app with

    ```
    $ ./gradlew clean build
    ```

4. In a second shell, run `gfsh`.
5. Use `gfsh` to connect to the GemFire for TAS service instance as described in Connect with gfsh over HTTPS.
6. Use `gfsh` to create a region named `test` as described in Create Regions. This sample app places a single entry into the region, so the region type is not important.

PARTITION_REDUNDANT is a good choice.

7. In the shell where the app was built, deploy and run the app with

   ```
   cf push -f manifest.yml
   ```

8. After the app starts, there will be an entry of (“1”, “one”) in the test region. you can see that there is one entry in the region with the gfsh command:

   ```
   gfsh>describe region --name=test
   ```

   For this very small region, you can print the contents of the entire region with a gfsh query:

   ```
   gfsh>query --query='SELECT * FROM /test'
   ```

An Example Spring Boot App

This topic presents an example Spring Boot app to use with VMware GemFire for TAS.

The versioned example Spring Boot Data GemFire app at PCC-Sample-App-PizzaStore uses the VMware GemFire for Tanzu Application Service service instance as a system of record. Directions for running the app are in the GitHub repository’s README.md file. The app is versioned, and branches of the repository represent GemFire for TAS versions. For GemFire for TAS v1.15, use the branch named release/1.15.

The app uses Spring Boot Data GemFire. The documentation for this opinionated extension of Spring Data GemFire is at Spring Boot for Apache Geode & Pivotal GemFire Reference Guide.

The app saves pizza orders within the VMware GemFire servers of a GemFire for TAS service instance. A REST interface provides a way for the app to take orders for pizza toppings and sauces.

The app demonstrates how to set up and run the app based on The App’s Location.

Top Down Explanation

In this opinionated version of Spring Boot Data GemFire, at the topmost level of the app, the @SpringBootApplication annotation causes the app to have a ClientCache instance. Within the example app’s CloudcachePizzaStoreApplication class, place the annotation at the class definition level:

```
@SpringBootApplication
public class CloudcachePizzaStoreApplication
```

This app is the client within a standard client/server architecture. The GemFire for TAS service instance contains the servers. The client cache is a driver for interactions with the servers.

The Spring repository construct is the preferred choice to use for the data storage, which will be an VMware GemFire region. To implement it, annotate this example’s PizzaRepository implementation with @Repository:

```
@Repository
public interface PizzaRepository extends CrudRepository<Pizza, String>
```

An VMware GemFire region underlies the Spring repository, storing the ordered pizzas. Annotate the `Pizza` class model with `@Region`:

```
@Region("Pizza")
public class Pizza {
```

Within the `Pizza` class, identify the key of the VMware GemFire region entries with the `@Id` annotation. It is the `name` field in this example:

```
@Getter @Id @NonNull
private final String name;
```

The `@SpringBootApplication` annotation results in a chain of opinionated defaults, all of which are appropriate for this app. It identifies the app as a client. The client receives an VMware GemFire client cache. Any regions will default to type `PROXY`. A proxy type of region forwards all region operations to the GemFire servers; no data is stored within the app's client cache.

See Configuring Regions for Spring details. See Region Design for GemFire for TAS details on regions.

The App Controller

The `AppController` class implements the REST interface, by annotating the class with `@RestController`:

```
@RestController
public class AppController
```

As pizzas are ordered, a `CrudRepository.save()` operation causes an VMware GemFire `put` operation that updates the region on the VMware GemFire server.

Running an App

This topic describes the requirements and options to use when running an app with VMware GemFire for TAS.

Java Build Pack Requirements

To ensure that your app can use all the features from VMware GemFire for Tanzu Application Service, use the latest buildpack. The buildpack is available on GitHub at cloudfoundry/java-buildpack.

Bind an App to a Service Instance

Binding your apps to a service instance enables the apps to connect to the service instance and read or write data to the region. Run `cf bind-service APP-NAME SERVICE-INSTANCE-NAME` to bind an app to your service instance. Replace `APP-NAME` with the name of the app. Replace `SERVICE-INSTANCE-NAME` with the name you chose for your service instance.

```
$ cf bind-service my-app my-cloudcache
```

Binding an app to the service instance provides connection information through the VCAP_SERVICES environment variable. Your app can use this information to configure components, such as the GemFire client cache, to use the service instance.

Communicating with a Service Instance

An app may be running in one of three locations:

- The app is in the same foundation as the GemFire for TAS service instance. For this discussion, the app is a services foundation app.
- The app is in a different foundation than the GemFire for TAS service instance. For this discussion, the app is an app foundation app.
- The app is not running within any foundation. For this discussion, the app is an off-platform app, where a platform is composed of all foundations.

To communicate with the GemFire for TAS service instance, app foundation apps and off-platform apps require a service gateway.

Run an App Foundation Spring Boot Data Geode App

Follow these steps to run a Spring Boot Data Geode app that is located within an app foundation.

1. An operator takes the steps needed for Configuring a Service Gateway.
2. Create a GemFire for TAS service instance with service-gateway access enabled as described in Create a Service Instance.
3. Follow the directions to Create a Truststore for the App.
4. Copy the truststore to the resources directory within the app source code.
5. If the app foundation does not include a user-defined service instance, populate a resources/application.properties file within the app source code, as described in Specifying Application Properties.
6. Build the app.
7. If the @EnableClusterConfiguration annotation is used by the app, update the app's manifest.yml specification to include Java options that specify the truststore and the password created for the truststore. Deactivate the foundation's additional security, since this specification provides what is necessary. The env portion of the manifest for an app that uses @EnableClusterConfiguration will be of the form:

```
env:
    JAVA_OPTS: '-Djavax.net.ssl.trustStore=/home/vcap/app/BOOT-INF/classes/mytr
uststore.jks -Djavax.net.ssl.trustStorePassword=TRUST-STORE-PASSWD-HERE'
    JBP_CONFIG_CONTAINER_SECURITY_PROVIDER: '{ key_manager_enabled: false }'
```

8. Push the app to the app foundation.

Run an Off-Platform Spring Boot Data Geode App

Follow these steps to run a Spring Boot Data Geode app that is not located within any foundation.

1. An operator takes the steps needed for Configuring a Service Gateway.
2. Create a GemFire for TAS service instance with service-gateway access enabled as described in Create a Service Instance.
3. Follow the directions to Create a Truststore for the App.
4. Copy the truststore to the `resources` directory within the app source code.
5. Populate a `resources/application.properties` file within the app source code as described in Specifying Application Properties.
6. Build the app.
7. Run the app. If the `@EnableClusterConfiguration` annotation is used by the Spring Boot Data Geode app, the app must specify the truststore and its password in both the `application.properties` file and in the command that invokes the app. An example maven command might look similar to:

   ```
   mvn spring-boot:run -Dspring-boot.run.jvmArguments="-Djavax.net.ssl.trustStore=
   /PATH/TO/truststore.jks -Djavax.net.ssl.trustStorePassword=TRUST-STORE-PASSWD"
   ```

   where `TRUST-STORE-PASSWD` is the invented password specified when creating the truststore.

   If the app does not use the `@EnableClusterConfiguration` annotation, the example maven command becomes

   ```
   mvn spring-boot:run
   ```

Create a Truststore for the App

The app needs a truststore so that it can establish a TLS connection with the GemFire for TAS service instance. This truststore needs two CA certificates within it.

Follow this procedure to create the truststore.

1. Acquire the `services/tls_ca` from CredHub on the services foundation where the GemFire for TAS service instance runs.

   ```
   credhub get --name="/services/tls_ca" -k certificate > services_ca
   ```

2. Acquire the CA certificate from the location of your TLS termination within the services foundation where the GemFire for TAS service instance runs. For the example provided here, the file is named `root_ca_certificate`. If your TLS termination is at the GoRouter, then the certificate can be acquired from the Ops Manager tile under **Settings**, **Advanced Options**, **Download Root CA Cert**.
3. Use these `keytool` commands to form the truststore using the two acquired certificates:

   ```
   keytool -importcert -file services_ca -keystore apptruststore.jks -storetype JK
   S
   keytool -importcert -alias root_ca -file root_ca_certificate -keystore apptrust
   store.jks -storetype JKS
   ```

Specifying Application Properties

An `application.properties` file will contain these properties, with property values filled in to the right of the equals sign:

```
spring.data.gemfire.pool.locators=
service-gateway.hostname=
service-gateway.port=
spring.data.gemfire.pool.default.socket-factory-bean-name=
spring.data.gemfire.security.username=
spring.data.gemfire.security.password=
spring.data.gemfire.security.ssl.components=
gemfire.ssl-truststore=
gemfire.ssl-truststore-password=
gemfire.ssl-keystore=
gemfire.ssl-keystore-password=
```

Here are descriptions of the value needed for each property:

- Define `spring.data.gemfire.pool.locators` with a comma-separated list of the locators given within the `locators` element of the GemFire for TAS service instance service key, near the beginning of the service key. Include the entire specification for each locator with its port number, but do not include the quotation marks.
- Define `service-gateway.hostname` with the value of the `services_gateway` element of the GemFire for TAS service instance service key. Do not include the colon or the port number.
- Define `service-gateway.port` with only the value of the port number from the `services_gateway` element of the GemFire for TAS service instance service key.
- Define `spring.data.gemfire.pool.default.socket-factory-bean-name` with the bean which points to the SNI proxy socket factory.
- Define `spring.data.gemfire.security.username` with the `username` element of the `cluster_operator` role, from the `users` element of the GemFire for TAS service instance service key.
- Define `spring.data.gemfire.security.password` with the `password` element of the `cluster_operator` role, from the `users` element of the GemFire for TAS service instance service key.
- Define `spring.data.gemfire.security.ssl.components` with the value `all`.
- Define `gemfire.ssl-truststore` with the path to and file name of the truststore within the app’s environment.
- Define `gemfire.ssl-truststore-password` with the password specified when creating the truststore.
- Include `gemfire.ssl-keystore=` in the application properties file, but leave the value blank.
- Include `gemfire.ssl-keystore-password=` in the application properties file, but leave the value blank.

Developing an App Under TLS

This topic discusses developing an app under TLS for use with VMware GemFire for TAS.

Apps that connect to a TLS-enabled VMware GemFire for Tanzu Application Service service instance must set properties to configure the communication with the cluster components within the GemFire for TAS service instance.

Ensure that the cluster-level prerequisite step of Preparing for TLS has been completed.

For a Spring Data GemFire app with a Spring Data GemFire library dependency of 2.2.0.BUILD-SNAPSHOT or a more recent version, attach the `@EnableSsl` annotation to your configuration class to enable the TLS encryption for all cluster components. Also set these properties:

```
ssl-use-default-context=true
ssl-endpoint-identification-enabled=false
```

For other apps, the properties should be

```
ssl-enabled-components=all
ssl-use-default-context=true
ssl-endpoint-identification-enabled=false
```

An app may set these properties with the `ClientCacheFactory.set()` method, prior to creating a `ClientCache` instance.

The build and `cf push` of the app does not require any changes to work with a TLS-enabled GemFire for TAS service instance.

Tomcat Session State Caching

This topic explains Tomcat session state caching for apps with VMware GemFire for TAS.

Enable Session State Caching

By default, the Tomcat instance is configured to store all sessions and their data in memory. Under certain circumstances it may be appropriate to persist the sessions and their data to a repository. When this is the case (small amounts of data that should survive the failure of any individual instance), the buildpack can automatically configure Tomcat to do so by binding an appropriate service. In this case, the service is VMware GemFire for Tanzu Application Service.

When the `session-replication` tag is specified, the sessions will be stored in the GemFire for TAS service instance, thus moving the data out of Tomcat. An application bound to this service will start running in a Tomcat instance. A region for storing sessions will be created. The region will be named `gemfire_modules_sessions`. A 30-minute expiration for sessions will be automatically configured.

For session state caching, use a version 4.30 or more recent Java buildpack.

To enable Tomcat session state caching, do *one* of these options:

- Option 1: When creating your service instance name, specify the `session-replication` tag. For example:

  ```
   $ cf create-service p-cloudcache small-plan my-service-instance -t session-rep
  lication
  ```

- Option 2: Update your service instance, specifying the `session-replication` tag. For example:

  ```
  $ cf update-service new-service-instance -t session-replication
  ```

- Option 3: When creating the service, include the substring `session-replication` anywhere within the service instance name. An example with the substring is `my-service-instance-session-replication`.

Java Buildpack Version Compatibility with Tomcat Session State Caching

If your version of the Cloud Foundry Java buildpack specifies a `geode_store` version that is more recent than your tile's version of GemFire for TAS, you must use a different version of VMware GemFire for Tanzu Application Service than is provided by the Java buildpack.

The complete list of `geode_store` versions is given in https://java-buildpack-tomcat-gemfire-store.s3-us-west-2.amazonaws.com/index.yml.

You can specify the `geode_store` version environment variable in your application's `manifest.yml` file or set it with the cf set-env command. For details about specifying configuration values through environment variable in the Java Buildpack, see the Configuration and Extension section of the *Cloud Foundry Java Buildpack README* in GitHub.

You can find the buildpack's `geode_store` version in file `/config/tomcat.yml`.

If you use the java_offline_buildpack, you cannot change the `geode_store` value by using environment variables. You must either use an online version of the Java Buildpack alongside the environment variable or create a custom buildpack which specifies the version as described in Create a Custom Buildpack.

When using session state caching you may see a warning during app deployment that looks like the following:

```
WARNING: Tomcat version X does not match Geode Tomcat Y module. If you
encounter compatibility issues, please make sure these versions match.
```

If you have issues with Tomcat version compatibility when deploying applications using session state caching you can resolve this error by changing the version of Tomcat used by the Java Buildpack. To change the version of Tomcat used by the Java Buildpack, use an environment variable as described in the Configuration and Extension section of the *Cloud Foundry Java Buildpack README* or modify it using the instructions in Create a Custom Buildpack.

For a full list of environment variables related to configuring Tomcat in the Java Buildpack, see Configuration in *Tomcat Container* in GitHub. If you do not want to set an environment variable, you can also configure these values using the process described in Create a Custom Buildpack

Do not mix Tomcat session state caching as described here with Spring Session caching as described in Spring Session Caching. Mixing Tomcat session state caching with Spring Session caching can cause issues.

Java Buildpack Version Incompatibility

The Cloud Foundry Java buildpack specifies a Geode Tomcat Session Store that corresponds to your GemFire for TAS version. If your version of the Cloud Foundry Java buildpack specifies a `geode_store` version that is more recent than your tile's version of GemFire for TAS, you will need to specify a different version of VMware GemFire for Tanzu Application Service than is provided by the Java buildpack. The complete list of `geode_store` versions is given in https://java-buildpack-tomcat-gemfire-store.s3-us-west-2.amazonaws.com/index.yml.

Three ways to specify and use a different buildpack exist. Select and implement one of these three ways.

- Option 1: Replace the Tanzu Application Service (TAS) environment's Java buildpack:
    1. Log in using `cf login`, targeting the correct org and space.
    2. Delete the current TAS environment's offline Java buildpack:

        ```
        cf delete-buildpack java_buildpack_offline
        ```

    3. Download the replacement Java buildpack (offline) from Tanzu Network.
    4. Upload the replacement Java buildpack (offline) to your TAS environment. This following example specifies a version 4.37 buildpack:

        ```
        cf create-buildpack java_buildpack_offline /PATH/TO/java-buildpack-offlin
        e-v4.37.zip POSITION
        ```

        Where `POSITION` is the order in which this buildpack will be selected.
- Option 2: Specify both a `geode_store` version environment variable and an online buildpack for your application.
    - Set the `geode_store` version environment variable in one of the following ways:
        - Set the `geode_store` version environment variable in your application's `manifest.yml` file. For details about specifying configuration values with an environment variable in the Java Buildpack, see Configuration and Extension in the Cloud Foundry Java Buildpack repository in GitHub. You can find the buildpack's `geode_store` version in the `/config/tomcat.yml` file. For example:

            ```
            env:
              JBP_CONFIG_TOMCAT: '{ geode_store: { version: 1.13.0 } }'
            ```

        - Set the `geode_store` version environment variable with the `cf set-env` command. For more information about this command, see set-env in the *Cloud Foundry CLI Reference Guide*.
    - Specify an online buildpack in one of two ways:
        - Specify an online buildpack in your application's `manifest.yml` file. For example: as in the example:

            ```
            buildpacks:
              - https://github.com/cloudfoundry/java-buildpack.git#v4.37
            ```

    - Specify an online buildpack in the `cf push` command of your application.

- Option 3: Create a custom buildpack that specifies the version as described in Create a Custom Buildpack.

Tomcat Versions Incompatibility

This Tomcat session state caching warning issued during app deployment may indicate a version incompatibility:

```
WARNING: Tomcat version X does not match Geode Tomcat Y module. If you
encounter compatibility issues, please make sure these versions match.
```

This warning indicates an issue that needs to be addressed if your version of Tomcat is significantly older (such as a Tomcat version prior to version 8) than the current default Tomcat version specifications.

You can fix this warning by changing the version of Tomcat being used by the Java buildpack in one of two ways:

- Create a Custom Buildpack and modify the version of Tomcat in your custom buildpack.
- Specify the online buildpack in the app's `manifest.yml` file, and set an environment variable in the app's `manifest.yml` file as described in Cloud Foundry Java Buildpack Configuration and Extension. For example:

  ```
  buildpacks:
    - https://github.com/cloudfoundry/java-buildpack.git#v4.37
  env:
    JBP_CONFIG_TOMCAT: '{ tomcat: { version: 9.0.+ } }'
  ```

  For a full list of environment variables related to configuring Tomcat in the Java buildpack, see Tomcat Container Configuration.

Deactivate Near Caching Within the App

Near caching is when an app locally caches data. Near caching uses an embedded cache within the app. Web apps that deploy Tomcat with VMware GemFire session state caching have near caching by default. To keep an app stateless, deactivate near caching.

If you are not using an external repository for configuration, Create a Custom Buildpack to deactivate near caching.

Create a Custom Buildpack

If your VMware GemFire for Tanzu Application Service version is not compatible with the dependencies included in the Java buildpack, you can create a custom buildpack.

1. Clone the Cloud Foundry Java buildpack repository:

   ```
   $ git clone git@github.com:cloudfoundry/java-buildpack.git
   ```

2. Change directories to the newly-created repository clone:

```
$ cd java-buildpack
```

3. Check out the desired version or release of the java-buildpack repository on a local branch. Use v4.36 or the most recent version available.

4. To change the version of VMware GemFire for Tanzu Application Service used by the buildpack, edit the `/config/tomcat.yml` configuration file to specify the `geode_store`. Here is the portion of the file to be changed, with the string to change in bold:

```
geode_store:
  version: 1.13.+
  repository_root: https://java-buildpack-tomcat-gemfire-store.s3-us-west-2.ama
zonaws.com
```

   For VMware GemFire for Tanzu Application Service version 1.11 and more recent versions, change the `geode_store` version to match your VMware GemFire for Tanzu Application Service' minor version.

   Change the `geode_store` version to `0.0.2` for all previous VMware GemFire for Tanzu Application Service versions. The complete list of `geode_store` versions is given in https://java-buildpack-tomcat-gemfire-store.s3-us-west-2.amazonaws.com/index.yml.

5. If you changed the version of VMware GemFire for Tanzu Application Service used by the buildpack, you may also need to change the version of the Geode Session State Module used by the buildpack. Edit the following line in the java-buildpack repository file `lib/java_buildpack/container/tomcat/tomcat_geode_store.rb` to specify the appropriate version of Tomcat for your current `geode_store` version:

```
SESSION_MANAGER_CLASS_NAME = 'org.apache.geode.modules.session.catalina.Tomcat9
DeltaSessionManager'
```

   For versions of `geode_store` that are 1.11 or greater, use `Tomcat9DeltaSessionManager`. For versions of `geode_store` previous to 1.11 and those with `geode_store` version 0.0.2, use `Tomcat8DeltaSessionManager`.

6. To deactivate near caching within the custom buildpack, edit the java-buildpack repository file `lib/java_buildpack/container/tomcat/tomcat_geode_store.rb` such that it deactivates caching within the app. Change the string `true` to instead be `false`. Before the change, here is the portion of the file to be changed, with the string to change highlighted:

```
def add_manager(context)
  context.add_element 'Manager',
                      'className' => SESSION_MANAGER_CLASS_NAME,
                      'enableLocalCache' => 'true',
                      'regionAttributesId' => REGION_ATTRIBUTES_ID
end
```

   After changing the highlighted string from `true` to `false`, here is the portion of the file with the change highlighted:

```
def add_manager(context)
  context.add_element 'Manager',
                      'className' => SESSION_MANAGER_CLASS_NAME,
                      'enableLocalCache' => 'false',
                      'regionAttributesId' => REGION_ATTRIBUTES_ID
end
```

7. Commit the changes to your local branch.
8. Create your custom Java buildpack on the platform with a command of the form:

   ```
   cf create-buildpack BUILDPACK-NAME PATH POSITION
   ```

   Where: * `BUILDPACK-NAME` is the name you choose for your buildpack. * `PATH` is the location of your local Java buildpack directory. * `POSITION` is the order in which your buildpack will be selected.
9. After building your application, push it such that it uses your buildpack:

   ```
   cf push -f ./manifest.yml -b BUILDPACK-NAME
   ```

   Where `BUILDPACK-NAME` is the name you chose for your buildpack.
10. Bind your app as described in Bind an App to a Service Instance.
11. Restage the app to ensure proper configuration:

   ```
   cf restage APP-NAME
   ```

   where `APP-NAME` is the name you chose for app.

Use an External Repository for Configuration

The Tomcat container can be customized if you place the custom configuration in a repository. Using an external repository for configuration also facilitates having a single configuration that may be used by a variety of apps.

There are two parts to using an external repository:

- Part 1: Build a repository to hold the configuration, and then push the repository to the space, such that the app will have access.
- Part 2: Change the app such that it uses the external repository.

Part 1 builds the repository:

1. Make a directory to hold the configuration:

   ```
   $ mkdir tomcat-config
   ```

2. Make other needed files and directories within the newly created directory:

   ```
   $ cd tomcat-config
   $ mkdir public
   $ mkdir -p tomcat-1.0.0/conf
   $ touch Staticfile
   ```

3. Edit `Staticfile` to contain:

```
root: public
directory: visible
```

4. Create a `tomcat-1.0.0/conf/context.xml` file. This example content specifies Tomcat version 9 and deactivates near caching:

   ```
   <?xml version='1.0' encoding='UTF-8'?>
   <!--
   ~ Copyright 2013-2019 the original author or authors.
   ~
   ~ Licensed under the Apache License, Version 2.0 (the "License");
   ~ you may not use this file except in compliance with the License.
   ~ You may obtain a copy of the License at
   ~
   ~      http://www.apache.org/licenses/LICENSE-2.0
   ~
   ~ Unless required by applicable law or agreed to in writing, software
   ~ distributed under the License is distributed on an "AS IS" BASIS,
   ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   ~ See the License for the specific language governing permissions and
   ~ limitations under the License.
   -->
   <Context>
       <Resources allowLinking='true'/>
       <Manager className='org.apache.geode.modules.session.catalina.Tomcat9DeltaSessionManager' enableLocalCache='false' regionAttributesId='PARTITION_REDUNDANT_HEAP_LRU'/>
   </Context>
   ```

   You can customize Tomcat further by placing custom configuration files within the `tomcat-1.0.0/conf` directory.

5. Place a compressed TAR format version of the `tomcat-1.0.0` directory into the `public` directory:

   ```
   $ tar -czf tomcat-1.0.0.tar.gz tomcat-1.0.0/
   $ cp tomcat-1.0.0.tar.gz public/
   ```

6. With a current working directory of `tomcat-config`, use `cf push` to place the configuration into the VMware Tanzu Application Service for VMs space that will host the app:

   ```
   $ cf push tomcat-config
   ```

7. Issue the command

   ```
   $ cf apps
   ```

   to acquire the URL of the `tomcat-config` app. Prepend the listed URL with `http://` to have the URL that will identify the location of the configuration needed by the app.

   For example, your complete URL will look something like `http://tomcat-config.apps.yellow-green.cf-app.com`.

8. Within the `tomcat-config` directory, edit `public/index.yml` to have URL-specific contents. Using the example URL as a guide, the contents of this `public/index.yml` file will contain:

```
---
1.0.0: http://tomcat-config.apps.yellow-green.cf-app.com/tomcat-1.0.0.tar.gz
```

9. Push the configuration a second time with its completed configuration:

   ```
   cf push tomcat-config
   ```

Part 2 modifies the app to configure the use of the external repository:

1. Edit the `manifest.yml` file by appending this URL-specific configuration to the `applications` section of the `manifest.yml` file:

   ```
     env:
       JBP_CONFIG_TOMCAT: "{ tomcat: { external_configuration_enabled: true }, ext
   ernal_configuration: { repository_root: "http://tomcat-config.apps.yellow-green
   .cf-app.com" } }"
   ```

   Substitute your complete URL for the URL in this example.

   A complete `manifest.yml` file will appear similar to:

   ```
   ---
   applications:
   - name: http-session-caching
     path: build/libs/http-session-caching-0.0.1.war
     buildpack: java_buildpack_offline
     env:
       JBP_CONFIG_TOMCAT: "{ tomcat: { external_configuration_enabled: true }, ext
   ernal_configuration: { repository_root: "http://tomcat-config.apps.yellow-green
   .cf-app.com" } }"
   ```

2. Push, but do not start, your app:

   ```
   cf push -f ./manifest.yml --no-start -b BUILDPACK-NAME
   ```

   where `BUILDPACK-NAME` is the name you chose for your custom buildpack, or it is the URL of the most recent Java buildpack version, if you did not create a custom buildpack.

   The specification of the buildpack on the `cf push` command line takes precedence over specification within the manifest.

3. Bind and start your app:

   ```
   cf bind-service APP-NAME SERVICE-INSTANCE-NAME

   cf start APP-NAME
   ```

4. To verify that near caching is deactivated for the app, use `cf ssh` to access the app and visually verify that `enableLocalCache='false'` appears within the `context.xml` file. Use this sequence of commands:

   ```
   cf ssh APP-NAME
   ```

```
cat app/.java-buildpack/tomcat/conf/context.xml
```

Spring Session Caching

This topic explains how to use Spring Session for Apache Geode for session state caching for apps with VMware GemFire for TAS.

To use Spring Session for Apache Geode for session state caching for apps with GemFire for TAS, follow these steps:

1. In the app, replace the existing Spring Session `@EnableXXXHttpSession` annotation with:

   ```
   @EnableGemFireHttpSession(maxInactiveIntervalInSeconds = N)
   ```

   Where `N` is an integer representing seconds.

2. In the app, add the `spring-session-data-geode` and `spring-data-geode` dependencies to the build.

3. In the app, add beans to the Spring app configuration.

4. On a command line, connected with a security role that can manage cluster data, use `gfsh` to create a region on the cluster servers named `ClusteredSpringSessions`:

   ```
   gfsh>create region --name=ClusteredSpringSessions --type=PARTITION_HEAP_LRU
   ```

Do not enable Tomcat session state caching as described in Enable Session State Caching in *Tomcat Session State Caching*. Mixing Tomcat session state caching with Spring Session caching can cause issues.

The Session State Code Example demonstrates session state caching.

For reference documentation, see Spring Session in the Spring documentation.

Troubleshooting

This topic describes issues and solutions related to using VMware GemFire for Tanzu Application Service.

Acquire Artifacts for Troubleshooting

Gather Cluster Logs

Cluster statistics and log files may be obtained by using the `gfsh export logs` command. See Export gfsh Logs for details.

View Statistics Files and Logs

You can visualize the performance of your cluster by downloading the statistics files from your servers. These files are located in the persistent store on each VM. To copy these files to your workstation, run the command:

```
bosh2 -e BOSH-ENVIRONMENT -d DEPLOYMENT-NAME scp server/0:/var/vcap/store/gemfire-serv
er/statistics.gfs /tmp
```

Alternatively, use `bosh ssh` to access GemFire for TAS service instance server VMs and directly obtain the VMware GemFire logs that reside within the directory `/var/vcap/sys/log/gemfire-server`.

See Visual Statistics Display (VSD) for information about loading the statistics files into VSD.

Acquire Thread Dumps

Thread dumps may be useful for debugging. Take at least three thread dumps on each VM, separating them by about one second.

1. To list your VMs, run:

   ```
   bosh -e ENV -d DEPLOYMENT vms
   ```

   Acquire the Deployment Name instructs how to acquire the string to substitute for `DEPLOYMENT`.

2. Use the VM in a `bosh ssh` command to ssh in to the GemFire for TAS VM where you want to produce the thread dumps. GemFire for TAS VMs can be referenced using a 0-based index, for example server/0, or locator/2:

   ```
   bosh -e ENV -d DEPLOYMENT ssh server/0
   ```

3. Get into the Bosh Process Manager (bpm) shell by running

   ```
   sudo /var/vcap/packages/bpm/bin/bpm shell JOB-NAME
   ```

   where JOB-NAME is either `gemfire-server` or `gemfire-locator`, depending on which GemFire for TAS VM you are on.

4. Find the process ID (PID) that is running the VMware GemFire Java process by running

   ```
   ps -aux | grep java
   ```

   Typically, the PID is 1.

5. As you take multiple thread dumps, redirect the output of each to a uniquely named file. This example uses the file name `threaddump1.txt`:

   ```
   /var/vcap/packages/jdk8/bin/jcmd 1 Thread.print > /tmp/threaddump1.txt
   ```

   Files in `/tmp` will be accessible on the VM in directory `/var/vcap/data/gemfire-server/tmp` or `/var/vcap/data/gemfire-locator/tmp`.

6. Move the files to the `/tmp` directory on the VM by running

   ```
   mv /var/vcap/data/gemfire-server/tmp/threaddump1.txt /tmp/
   ```

   or

   ```
   mv /var/vcap/data/gemfire-locator/tmp/threaddump1.txt /tmp/
   ```

7. Files can be copied to your local machine using `bosh scp` command. From your local machine, run:

   ```
   bosh -d DEPLOYMENT scp VM:/tmp/threaddump1.txt .
   ```

   For example:

   ```
   $ bosh -d service-instance_1fd2850e-b754-4c5e-aa5c-ddb54ee301e6 scp server/0:/tmp/threaddump1.txt .
   ```

Acquire the Deployment Name

The `DEPLOYMENT` name is needed in several troubleshooting procedures. To acquire the `DEPLOYMENT` name:

1. Use the Cloud Foundry CLI. Target the space where the service instance runs.

2. Discover the globally unique identifier (GUID) for the service instance:

   ```
   cf service INSTANCE-NAME --guid
   ```

   The output is the GUID. For example:

   ```
   $ cf service dev-instance --guid
   1fd2850e-b754-4c5e-aa5c-ddb54ee301e6
   ```

3. Prefix the GUID with the string `service-instance_` to obtain the `DEPLOYMENT` name. For the example GUID, the `DEPLOYMENT` name is `service-instance_1fd2850e-b754-4c5e-aa5c-ddb54ee301e6`.

Troubleshooting for Operators

Smoke Test Failures

- **Error message:** “Creating p-cloudcache SERVICE-NAME failed”

  Cause of the Problem: The smoke tests could not create an instance of GemFire.

  Action: To troubleshoot why the deployment failed, use the CF CLI to create a new service instance using the same plan and download the logs of the service deployment from BOSH.

- **Error message:** “Deleting SERVICE-NAME failed”

  Cause of the Problem: The smoke test attempted to clean up a service instance it created and failed to delete the service using the `cf delete-service` command.

  Action: Run BOSH `logs` to view the logs on the broker or the service instance to see why the deletion may have failed.

- **Error message:** “Cannot connect to the cluster SERVICE-NAME”

  Cause of the Problem: The smoke test was unable to connect to the cluster.

  Action: Review the logs of your load balancer, and review the logs of your CF Router to ensure the route to your GemFire for TAS cluster is properly registered.

  You also can create a service instance and try to connect to it using the gfsh CLI. This requires creating a service key.

- **Error message:** “Could not perform create/put on locator LOCATOR-ADDRESS”

  Cause of the Problem: `LOCATOR-ADDRESS` is the BOSH DNS name of the locator. The smoke test was unable to write data to the cluster. The user may not have permissions to create a region or write data.

General Connectivity

- **Problem:** Client-to-server communication

  Cause of the Problem: GemFire for TAS clients communicate to GemFire for TAS servers on port 40404 and with locators on port 55221. Both of these ports must be reachable from the VMware Tanzu Application Service for VMs network to service the network.

- **Problem:** Membership port range

  Cause of the Problem: GemFire for TAS servers and locators communicate with each other using UDP and TCP. The current port range for this communication is `49152-65535`.

  Solution: If you have a firewall between VMs, ensure this port range is open.

- **Problem:** Port range usage across a WAN

  Cause of the Problem: Gateway receivers and gateway senders communicate across WAN-separated service instances. Each GemFire for TAS service instance uses VMware GemFire defaults for the gateway receiver ports. The default is the inclusive range of port numbers 5000 to 5499.

Solution: Ensure this port range is open when WAN-separated service instances will communicate.

Troubleshooting for Developers

- **Problem:** An error occurs when creating a service instance or when running a smoke test. The service creation issues an error message that starts with

  ```
  Instance provisioning failed: There was a problem completing your request.
  ```

  VMware GemFire server logs at `/var/vcap/sys/log/gemfire-server/gemfire/server-<N>.log` will contain a disk-access error with the string

  ```
  A DiskAccessException has occurred
  ```

  and a stack trace similar to this one that begins with

  ```
  org.apache.geode.cache.persistence.ConflictingPersistentDataException
      at org.apache.geode.internal.cache.persistence.PersistenceAdvisorImpl.check
  MyStateOnMembers(PersistenceAdvisorImpl.java:743)
      at org.apache.geode.internal.cache.persistence.PersistenceAdvisorImpl.getIn
  itialImageAdvice(PersistenceAdvisorImpl.java:819)
      at org.apache.geode.internal.cache.persistence.CreatePersistentRegionProces
  sor.getInitialImageAdvice(CreatePersistentRegionProcessor.java:52)
      at org.apache.geode.internal.cache.DistributedRegion.getInitialImageAndReco
  very(DistributedRegion.java:1178)
      at org.apache.geode.internal.cache.DistributedRegion.initialize(Distributed
  Region.java:1059)
      at org.apache.geode.internal.cache.GemFireCacheImpl.createVMRegion(GemFireC
  acheImpl.java:3089)
  ```

  Cause of the Problem: The GemFire for TAS VMs are underprovisioned; the quantity of disk space is too small.

  Solution: Use Ops Manager to provision VMs of at least the minimum size. See Configure Service Plans for minimum-size details.