Redis for Tanzu Application Service 3.2

Redis for VMware Tanzu Application Service 3.2

vmware
by Broadcom

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

Redis for VMware Tanzu Application Service

This is documentation for Redis for VMware Tanzu® Application Service™. You can download the Redis for Tanzu Application Service tile from Broadcom's Customer Support Portal.

This documentation:

- Describes features and architecture of Redis for Tanzu Application Service.
- Instructs operators on how to install, configure, maintain, and backup Redis for Tanzu Application Service.
- Instructs app developers on how to choose a service plan, create and delete Redis service instances, and bind an app.

Product snapshot

Element	Details
Version	3.2.5
Release date	November 30, 2023
Software component version	Redis OSS 6.2.13
Compatible Tanzu Operations Manager version(s)	2.9, 2.10, and 3.0
Compatible VMware Tanzu Application Service for VMs version(s)	2.11, 2.12, 2.13, 3.0, 4.0, and 5.0
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	Yes

About Redis

Redis®* is an easy-to-use, high-speed key-value store that can be used as a database, cache, and message broker. It supports a range of data structures including strings, lists, hashes, sets, bitmaps, HyperLogLogs, and geospatial indexes. It's easy to install and configure and is popular with engineers as a straightforward NoSQL datastore. It's used for everything from a quick way to store data for development and testing through to enterprise-scale apps like Twitter.

About Redis for Tanzu Application Service

Redis for Tanzu Application Service packages Redis for deployment and operability.

There are two service offerings:

- **On-Demand Service**—Provides a dedicated VM running a Redis instance. The operator can configure up to three plans with different configurations, memory sizes, and quotas App developers can provision an instance for any of the On-Demand plans offered and configure certain Redis settings.
- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development purposes only, **do not use the Shared-VM service in production environments**. The Shared-VM instances are pre-provisioned by the operator with a fixed number of instances and memory size. App developers can then use one of these pre-provisioned instances.

For more information about the plans, see:

- On-Demand service offering
- Shared-VM service offering

Is Redis for Tanzu Application Service right for your enterprise?

For information about recommended use cases, and the enterprise-readiness of Redis for Tanzu Application Service, see Is Redis for Tanzu Application Service right for your enterprise?.

Upgrading to the latest version

For information about how to upgrade and the supported upgrade paths, see Upgrading Redis for VMware Tanzu Application Service.

Addtional information

The following table lists where you can find topics related to the information about this page:

For more information about...	See...
Product compatibility	Upgrading your Tanzu Operations Manager deployment
How to upgrade Redis for Tanzu Application Service	Upgrading Redis for VMware Tanzu Application Service
How to use Redis	Redis Documentation

Redis for Tanzu Application Service and other services

As well as Redis for Tanzu Application Service, other services offer *on-demand* service plans. These plans allow developers to 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
VMware RabbitMQ for Tanzu Application Service	Pivotal RabbitMQ	Yes	Yes. Only recommended for test environments.
Redis for Tanzu Application Service	Redis	Yes	Yes (shared-VM plan). Recommended only for test environments.
VMware Tanzu SQL with MySQL for VMs	MySQL	Yes	No
VMware Tanzu GemFire for VMs	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.

---

*Redis is a registered trademark of Redis Ltd. Any rights therein are reserved to Redis Ltd. Any use by VMware by Broadcom is for referential purposes only and does not indicate any sponsorship, endorsement, or affiliation between Redis and VMware by Broadcom.

Feedback

Please provide any issues, feature requests, or questions to the Feedback list.

Redis for VMware Tanzu Application Service release notes

This topic describes the changes in this minor release of Redis for VMware Tanzu Application Service.

For product versions and upgrade paths, see Upgrade Planner.

v3.2.5

Release date: 30 November 2023

Features

There are no new features for this release.

Resolved issues

This release has the following fix:

- **Resolved large backups failing to upload for Azure storage:** Now successfully upload backups greater than 4 MB to Azure storage.

Security fixes

This release includes the following security fixes:

- CVE-2009-5155
- CVE-2020-10029
- CVE-2020-1751
- CVE-2020-1752
- CVE-2020-27618
- CVE-2020-29562
- CVE-2020-29573
- CVE-2020-6096
- CVE-2021-27645
- CVE-2021-3326
- CVE-2021-33574
- CVE-2021-35942

- CVE-2021-38604
- CVE-2021-3998
- CVE-2021-3999
- CVE-2021-43396
- CVE-2022-39046
- CVE-2023-0687
- CVE-2023-25139
- CVE-2023-4527
- CVE-2023-4806
- CVE-2023-4813
- CVE-2023-4911
- CVE-2023-5156
- CVE-2023-24532
- CVE-2023-24534
- CVE-2023-24536
- CVE-2023-24537
- CVE-2023-24539
- CVE-2023-24540
- CVE-2023-29400
- CVE-2023-29402
- CVE-2023-29403
- CVE-2023-29404
- CVE-2023-29405
- CVE-2023-29406
- CVE-2023-29409
- CVE-2023-39318
- CVE-2023-39319
- CVE-2023-39323
- CVE-2023-24534
- CVE-2023-24536
- CVE-2023-24537
- CVE-2023-24538
- CVE-2023-24539
- CVE-2023-24540

- CVE-2023-29400
- CVE-2023-29402
- CVE-2023-29403
- CVE-2023-29404
- CVE-2023-29405
- CVE-2023-29406
- CVE-2023-29409
- CVE-2023-39318
- CVE-2023-39319
- CVE-2023-39323
- CVE-2023-39318
- CVE-2023-39319
- CVE-2023-39323
- CVE-2023-39323
- CVE-2023-39323
- CVE-2015-8863
- CVE-2016-4074
- CVE-2019-6706
- CVE-2020-15888
- CVE-2020-15945
- CVE-2020-24342
- CVE-2020-24369
- CVE-2020-24370
- CVE-2020-24371
- CVE-2021-44647
- CVE-2022-28805
- CVE-2022-33099
- CVE-2023-0464
- CVE-2023-0465
- CVE-2023-0466
- CVE-2023-2650
- CVE-2023-3446
- CVE-2023-3817
- CVE-2023-4807

- CVE-2023-27043
- CVE-2023-33595
- CVE-2023-36632
- CVE-2023-40217
- CVE-2023-41105
- CVE-2018-18074
- CVE-2023-32681
- CVE-2023-32681
- CVE-2022-40897
- CVE-2018-20060
- CVE-2019-11236
- CVE-2019-11324
- CVE-2020-26137
- CVE-2021-33503
- CVE-2023-43804
- CVE-2023-43804

Known issues

There are no known issues for this release.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	1.301
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0, 4.0, 5.0
shared-redis-release	437.0.42
on-demand-service-broker	0.45.1
routing	0.284.0
service-metrics	2.0.33
service-backup	18.4.9
loggregator-agent	7.7.1
bpm	1.2.11
cf-cli	1.53.0-lite
Redis OSS	6.2.13

v3.2.4

Release date: 18 October 2023

Features

New features and changes in this release:

- **VMware Tanzu Application Service for VMs compatibility:** This release is compatible with TAS for VMs v5.0.
- **Stemcell compatibility:** This release is compatible with Ubuntu Jammy Stemcell.

Security fixes

This release includes the following security fixes:

- CVE-2023-39325
- CVE-2023-29406
- CVE-2018-25091
- CVE-2023-44487
- CVE-2023-39320
- CVE-2023-39321
- CVE-2023-39322

Known issues

There are no known issues for this release.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	1.250
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0, 4.0, 5.0
shared-redis-release	437.0.40
on-demand-service-broker	0.44.0
routing	0.281.0
service-metrics	2.0.32
service-backup	18.4.7
loggregator-agent	7.6.3
bpm	1.2.8
cf-cli	1.49.0-lite

Component	Version
Redis OSS	6.2.13

v3.2.3

Release date: 06 September 2023

Features

There are no new features for this release.

Resolved issues

This release has the following fix:

- **Resolved large backups failing to upload:** Backups greater than 5 GB now successfully upload to AWS S3 buckets.

Known issues

There are no known issues for this release.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	621.655
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0, 4.0
shared-redis-release	437.0.38
on-demand-service-broker	0.43.2
routing	0.278.0
service-metrics	2.0.31
service-backup	18.4.6
loggregator-agent	7.6.1
bpm	1.2.6
cf-cli	1.45.0-lite
Redis OSS	6.2.13

v3.2.2

Release date: 10 August 2023

Security fixes

This release includes the following security fixes:

- CVE-2007-4559
- CVE-2015-20107
- CVE-2017-17522
- CVE-2017-20052
- CVE-2018-1000117
- CVE-2018-25032
- CVE-2019-16056
- CVE-2019-16935
- CVE-2019-20477
- CVE-2019-20478
- CVE-2019-20916
- CVE-2020-14343
- CVE-2020-15523
- CVE-2020-1747
- CVE-2020-27619
- CVE-2020-28366
- CVE-2020-28367
- CVE-2020-29509
- CVE-2020-29510
- CVE-2020-29511
- CVE-2020-36242
- CVE-2021-29923
- CVE-2021-3115
- CVE-2021-3177
- CVE-2021-33194
- CVE-2021-33195
- CVE-2021-4189
- CVE-2021-44717
- CVE-2021-45960
- CVE-2021-46143
- CVE-2022-0391
- CVE-2022-22822
- CVE-2022-22823

- CVE-2022-22824
- CVE-2022-22825
- CVE-2022-22826
- CVE-2022-22827
- CVE-2022-23806
- CVE-2022-23990
- CVE-2022-25235
- CVE-2022-25236
- CVE-2022-25314
- CVE-2022-25315
- CVE-2022-28805
- CVE-2022-29526
- CVE-2023-29402
- CVE-2023-29403
- CVE-2023-29404
- CVE-2023-29405
- CVE-2023-29406
- CVE-2023-32643

Resolved issues

This release has the following fixes:

- **Smoke tests errand error:** When running the Shared-VM and On-Demand smoke tests errand, tests no longer enable public access to Redis private plans when running the command `cf service-access`.

Known issues

This release has the following known issues:

- **Large backups fail to upload:** Backups fail to upload to AWS S3 buckets if their size is greater than 5 GB.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	621.615
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0, 4.0
shared-redis-release	437.0.38

Component	Version
on-demand-service-broker	0.43.2
routing	0.265.0
service-metrics	2.0.30
service-backup	18.4.5
loggregator-agent	7.4.0
bpm	1.2.5
Redis OSS	6.2.13

v3.2.1

Release date: 08 June 2023

Resolved issues

This release has the following fixes:

- This release is compatible with Tanzu Application Service for VMs v4.0.
- **Smoke tests errand error:** When running the Shared-VM and On-Demand smoke tests errand, the service access to the Redis plans no longer appears as limited when running the command `cf service-access`.

Known issues

This release has the following known issues:

- **Smoke tests errand error:** When running the Shared-VM and On-Demand smoke tests errand, private Redis plans change to public Redis plans.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	621.463
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0, 4.0
shared-redis-release	437.0.36
on-demand-service-broker	0.42.7
routing	0.265.0
service-metrics	2.0.28
service-backup	18.4.0
loggregator-agent	7.2.1

Component	Version
bpm	1.2.2
Redis OSS	6.2.7

v3.2.0

Release date: 19 April 2023

Features

New features and changes in this release:

- **Service-gateway access** Service-gateway access enables a Redis for Tanzu Application Service on-demand service instance to connect to external components that are not on the same foundation as the service instance. These components might be on another foundation or hosted outside of the foundation. For more information, see Service-Gateway Access.
- **Backups for AWS S3 with CA Certificate** When selecting AWS S3 option for backups, now the connection can be verified with a CA Certificate that you enter.

Resolved Issues

This release has the following fixes:

- **Single AZ only:** Made the necessary adjustments to the `cf-redis-broker` configuration to ensure that it runs as a singleton job.

  This fix might cause the Availability Zones (AZs) for the VM to be changed. This only affects the Shared-VM. As a result, an orphaned disk might occur. For instructions on how to reattach the disk, see the BOSH documentation.
- **loggr-syslog-agent drain certificate:** Added `drain_ca.crt` to the broker so that `loggr-syslog-agent` can pull the certificate for trust purposes.
- **Ruby Buildpack in smoke test errand:** The smoke tests errand no longer fails with Ruby Buildpack v1.9.x and v1.10.x.

Known Issues

This release has the following known issues:

- **Smoke tests errand error:** When running the Shared-VM and On-Demand smoke tests errand, Redis plans change to `limited` access.
- The fix **Single AZ only** might cause the Availability Zones (AZs) for the VM to be changed. This only affects the Shared-VM. As a result, an orphaned disk might occur. For instructions on how to reattach the disk, see the BOSH documentation.

Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	621.463
VMware Tanzu Application Service for VMs	2.11, 2.12, 2.13, 3.0
shared-redis-release	437.0.35
on-demand-service-broker	0.42.7
routing	0.261.0
service-metrics	2.0.26
service-backup	18.4.0
loggregator-agent	7.1.3
bpm	1.1.21
Redis OSS	6.2.7

View Release Notes for Another Version

To view the release notes for another product version, select the version from the drop-down menu at the top of this page.

Is Redis for VMware Tanzu Application Service right for your enterprise

This topic gives you recommended use cases for Redis for VMware Tanzu Application Service and information for determining the product's fit for your enterprise's use case.

Recommended use cases

On-demand plans are configured by default for cache use cases but can also be used as a datastore.

Shared-VM plans are designed for datastore use cases in testing or development environments.

⚠ **Caution**

The shared-VM service should only be used for development and testing. Do not use for production.

Redis can be used in many different ways, including:

- Key/value store: For strings and more complex data structures including Hashes, Lists, Sets, and Sorted Sets
- Session cache: Persistence enabled preservation of state
- Full page cache: Persistence enabled preservation of state
- Database cache: Middle-tier database caching to speed up common queries
- Data ingestion: Because Redis is in memory, it can ingest data very quickly
- Message queues: List and set operations. `PUSH`, `POP`, and blocking queue commands.
- Leaderboards and counting: Increments and decrements sets and sorted sets using `ZRANGE`, `ZADD`, `ZREVRANGE`, `ZRANK`, `INCRBY`, and `GETSET`
- Pub/Sub: Built in publish and subscribe operations: `PUBLISH`, `SUBSCRIBE`, and `UNSUBSCRIBE`

Service offerings

For descriptions of the service offerings for Redis for Tanzu Application Service, see:

- On-Demand Service offering
- Shared-VM Service offering

Enterprise readiness checklist

Review the following table to determine if Redis for Tanzu Application Service has the features needed to support your enterprise.

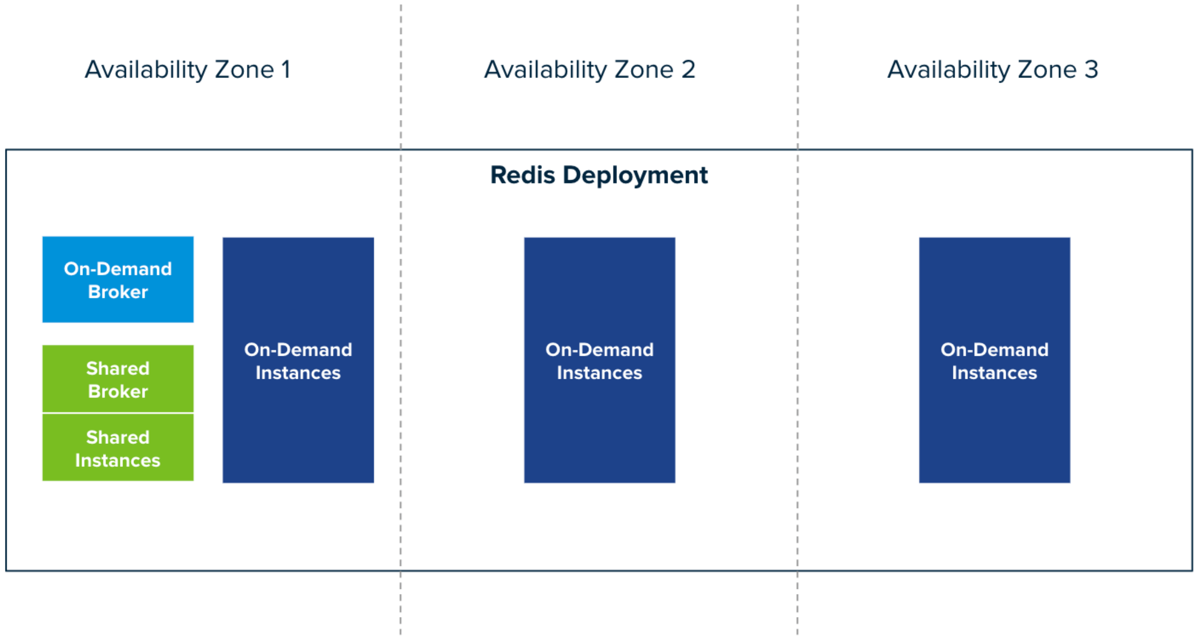
Resilience		More information
Availability	All service offerings of Redis for Tanzu Application Service are single VMs without clustering capabilities. This means that planned maintenance jobs (e.g., upgrades) can result in 2–10 minutes of downtime, depending on the nature of the upgrade. Unplanned downtime (e.g., VM failure) also affects the Redis service. Redis for Tanzu Application Service has been used successfully in enterprise-ready apps that can tolerate downtime. Pre-existing data is not lost during downtime with the default persistence configuration. Successful apps include those where the downtime is passively handled or where the app handles failover logic.	Recommended use cases Support for multiple AZs
Failure recovery	Recovery from VM failures and process failures are provided for by: • Automated service backups (both the on-demand and shared-VM Redis services) • BBR backup and recovery (only on-demand Redis services) • Manual backup and restore (both the on-demand and shared-VM Redis services)	Configuring automated service backups BOSH backup and restore (BBR) for on-demand Redis for VMware Tanzu Application Service Manually backing up and restoring Redis for Pivotal Cloud Foundry
Isolation	Isolation is provided when using the on-demand service. Individual apps and workflows should have their own Redis for Tanzu Application Service instance to maximize isolation.	
Day 2 Operations		**More information**
Resource planning	Operators can configure the number of VMs and the size of those VMs. For the on-demand service, the operator does this by creating plans with specific VM sizes and quotas for each plan. For the shared-VM service, the number and size of VMs are pre-provisioned by the operator. BOSH errands used for registration, upgrade and cleanup use short-lived VMs that cannot be configured but can be turned on or off.	On-demand resource planning Shared-VM plan
Health monitoring	Both the on-demand and shared service instances emit metrics. These include Redis-specific metrics and Redis for Tanzu Application Service metrics. Guidance on critical metrics and alerting levels is captured with the Redis for Tanzu Application Service Key Performance Indicators (KPIs).	Key performance indicators
Scalability	For the on-demand service, the operator can configure three plans with different resource sizes. The operator can also scale up the VM size associated with the plan. Additionally, the operator can increase the quota, which caps the number of instances allowed for each on-demand plan. To prevent data loss, only scaling up is supported. For the shared-VM service, the operators can change the Redis instance memory limit as well as change the instance limit. To prevent data loss, only scaling up is supported.	Scaling the on-demand service

Resilience		More information
Logging	All Redis services emit logs. Operators can configure syslog forwarding to a remote destination. This enables viewing logs from every VM in the Redis for Tanzu Application Service deployment in one place, effective troubleshooting when logs are lost on the source VM, and setting up alerts for important error logs to monitor the deployment.	Configuring syslog forwarding
Customization	The on-demand service can be configured to best fit the needs of a specific app. The shared-VM service cannot be customized.	Configuring the on-demand service
Upgrades	For information about preparing an upgrade and about understanding the effects on your Redis for Tanzu Application Service and other services, see Upgrading Redis for Tanzu Application Service. Redis for Tanzu Application Service upgrades run a post deployment BOSH errand called smoke tests to validate the success of the upgrade.	Upgrades Smoke tests
Encryption		**More information**
Encrypted communication in transit	You can enable TLS encryption between apps and service instances. Additionally, Redis for Tanzu Application Service has been tested with the IPsec Add-on for PCF.	OS Redis security TLS in Redis for Tanzu Application Service Securing data in transit with the IPsec add-on

Availability zones

On-demand Redis for Tanzu Application Service supports configuring multiple availability zones (AZs) to improve resiliency. However, assigning multiple AZs to Redis service instances does not provide high availability. This is because each individual Redis service instance is a single VM without clustering capabilities.

The following diagram shows a Redis deployment configured with three availability zones.



Click here to view a larger version of this image

Service instance VMs are placed in availability zones as follows:

- **For on-demand plans**: Service instances can be configured to deploy to any AZ. If you select multiple AZs, service instances are distributed randomly between them. This improves resiliency.
- **For the shared-VM plan**: Service instances run on a single VM in the AZ in which the tile is deployed.

Is Redis for VMware Tanzu Application Service right for your enterprise

This topic gives you recommended use cases for Redis for VMware Tanzu Application Service and information for determining the product's fit for your enterprise's use case.

Recommended use cases

On-demand plans are configured by default for cache use cases but can also be used as a datastore.

Shared-VM plans are designed for datastore use cases in testing or development environments.

> ⚠ **Caution**
>
> The shared-VM service should only be used for development and testing. Do not use for production.

Redis can be used in many different ways, including:

- Key/value store: For strings and more complex data structures including Hashes, Lists, Sets, and Sorted Sets

- Session cache: Persistence enabled preservation of state
- Full page cache: Persistence enabled preservation of state
- Database cache: Middle-tier database caching to speed up common queries
- Data ingestion: Because Redis is in memory, it can ingest data very quickly
- Message queues: List and set operations. `PUSH`, `POP`, and blocking queue commands.
- Leaderboards and counting: Increments and decrements sets and sorted sets using `ZRANGE`, `ZADD`, `ZREVRANGE`, `ZRANK`, `INCRBY`, and `GETSET`
- Pub/Sub: Built in publish and subscribe operations: `PUBLISH`, `SUBSCRIBE`, and `UNSUBSCRIBE`

Service offerings

For descriptions of the service offerings for Redis for Tanzu Application Service, see:

- On-Demand Service offering
- Shared-VM Service offering

Enterprise readiness checklist

Review the following table to determine if Redis for Tanzu Application Service has the features needed to support your enterprise.

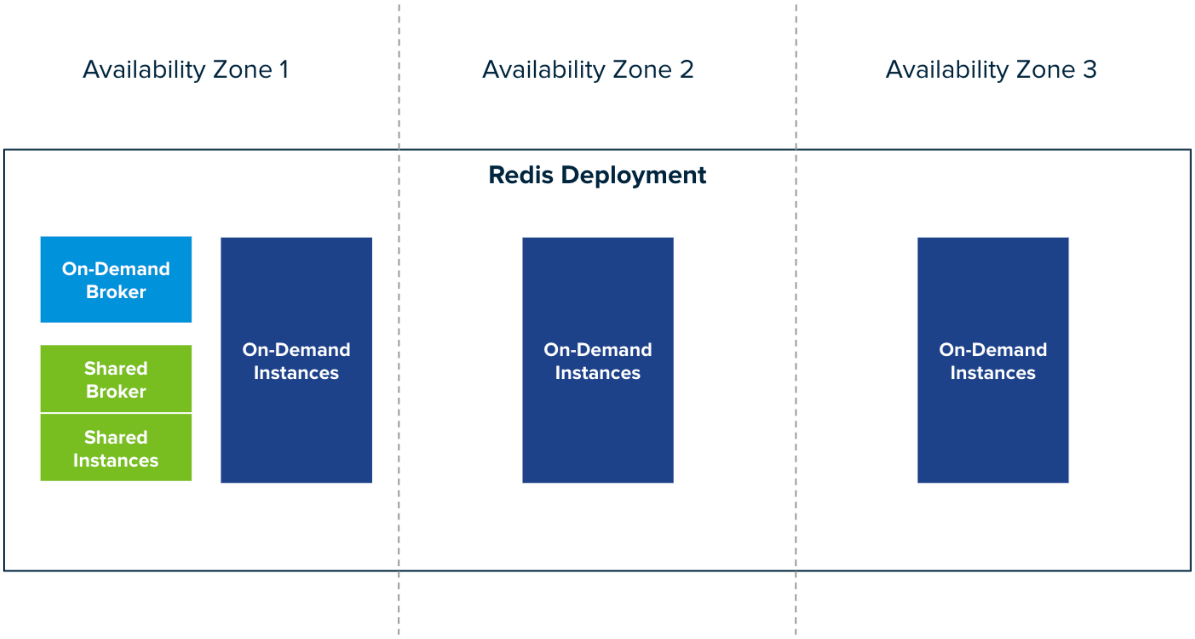
Resilience		More information
Availability	All service offerings of Redis for Tanzu Application Service are single VMs without clustering capabilities. This means that planned maintenance jobs (e.g., upgrades) can result in 2–10 minutes of downtime, depending on the nature of the upgrade. Unplanned downtime (e.g., VM failure) also affects the Redis service. Redis for Tanzu Application Service has been used successfully in enterprise-ready apps that can tolerate downtime. Pre-existing data is not lost during downtime with the default persistence configuration. Successful apps include those where the downtime is passively handled or where the app handles failover logic.	Recommended use cases Support for multiple AZs
Failure recovery	Recovery from VM failures and process failures are provided for by: • Automated service backups (both the on-demand and shared-VM Redis services) • BBR backup and recovery (only on-demand Redis services) • Manual backup and restore (both the on-demand and shared-VM Redis services)	Configuring automated service backups BOSH backup and restore (BBR) for on-demand Redis for VMware Tanzu Application Service Manually backing up and restoring Redis for Pivotal Cloud Foundry
Isolation	Isolation is provided when using the on-demand service. Individual apps and workflows should have their own Redis for Tanzu Application Service instance to maximize isolation.	
Day 2 Operations		**More information**

Resilience		More information
Resource planning	Operators can configure the number of VMs and the size of those VMs. For the on-demand service, the operator does this by creating plans with specific VM sizes and quotas for each plan. For the shared-VM service, the number and size of VMs are pre-provisioned by the operator. BOSH errands used for registration, upgrade and cleanup use short-lived VMs that cannot be configured but can be turned on or off.	On-demand resource planning Shared-VM plan
Health monitoring	Both the on-demand and shared service instances emit metrics. These include Redis-specific metrics and Redis for Tanzu Application Service metrics. Guidance on critical metrics and alerting levels is captured with the Redis for Tanzu Application Service Key Performance Indicators (KPIs).	Key performance indicators
Scalability	For the on-demand service, the operator can configure three plans with different resource sizes. The operator can also scale up the VM size associated with the plan. Additionally, the operator can increase the quota, which caps the number of instances allowed for each on-demand plan. To prevent data loss, only scaling up is supported. For the shared-VM service, the operators can change the Redis instance memory limit as well as change the instance limit. To prevent data loss, only scaling up is supported.	Scaling the on-demand service
Logging	All Redis services emit logs. Operators can configure syslog forwarding to a remote destination. This enables viewing logs from every VM in the Redis for Tanzu Application Service deployment in one place, effective troubleshooting when logs are lost on the source VM, and setting up alerts for important error logs to monitor the deployment.	Configuring syslog forwarding
Customization	The on-demand service can be configured to best fit the needs of a specific app. The shared-VM service cannot be customized.	Configuring the on-demand service
Upgrades	For information about preparing an upgrade and about understanding the effects on your Redis for Tanzu Application Service and other services, see Upgrading Redis for Tanzu Application Service. Redis for Tanzu Application Service upgrades run a post deployment BOSH errand called smoke tests to validate the success of the upgrade.	Upgrades Smoke tests
Encryption		**More information**
Encrypted communication in transit	You can enable TLS encryption between apps and service instances. Additionally, Redis for Tanzu Application Service has been tested with the IPsec Add-on for PCF.	OS Redis security TLS in Redis for Tanzu Application Service Securing data in transit with the IPsec add-on

Availability zones

On-demand Redis for Tanzu Application Service supports configuring multiple availability zones (AZs) to improve resiliency. However, assigning multiple AZs to Redis service instances does not provide high availability. This is because each individual Redis service instance is a single VM without clustering capabilities.

The following diagram shows a Redis deployment configured with three availability zones.



Click here to view a larger version of this image

Service instance VMs are placed in availability zones as follows:

- **For on-demand plans**: Service instances can be configured to deploy to any AZ. If you select multiple AZs, service instances are distributed randomly between them. This improves resiliency.
- **For the shared-VM plan**: Service instances run on a single VM in the AZ in which the tile is deployed.

On-demand service offering for Redis for VMware Application Service

Redis for VMware Tanzu Application Service offers on-demand and shared-VM service plans.

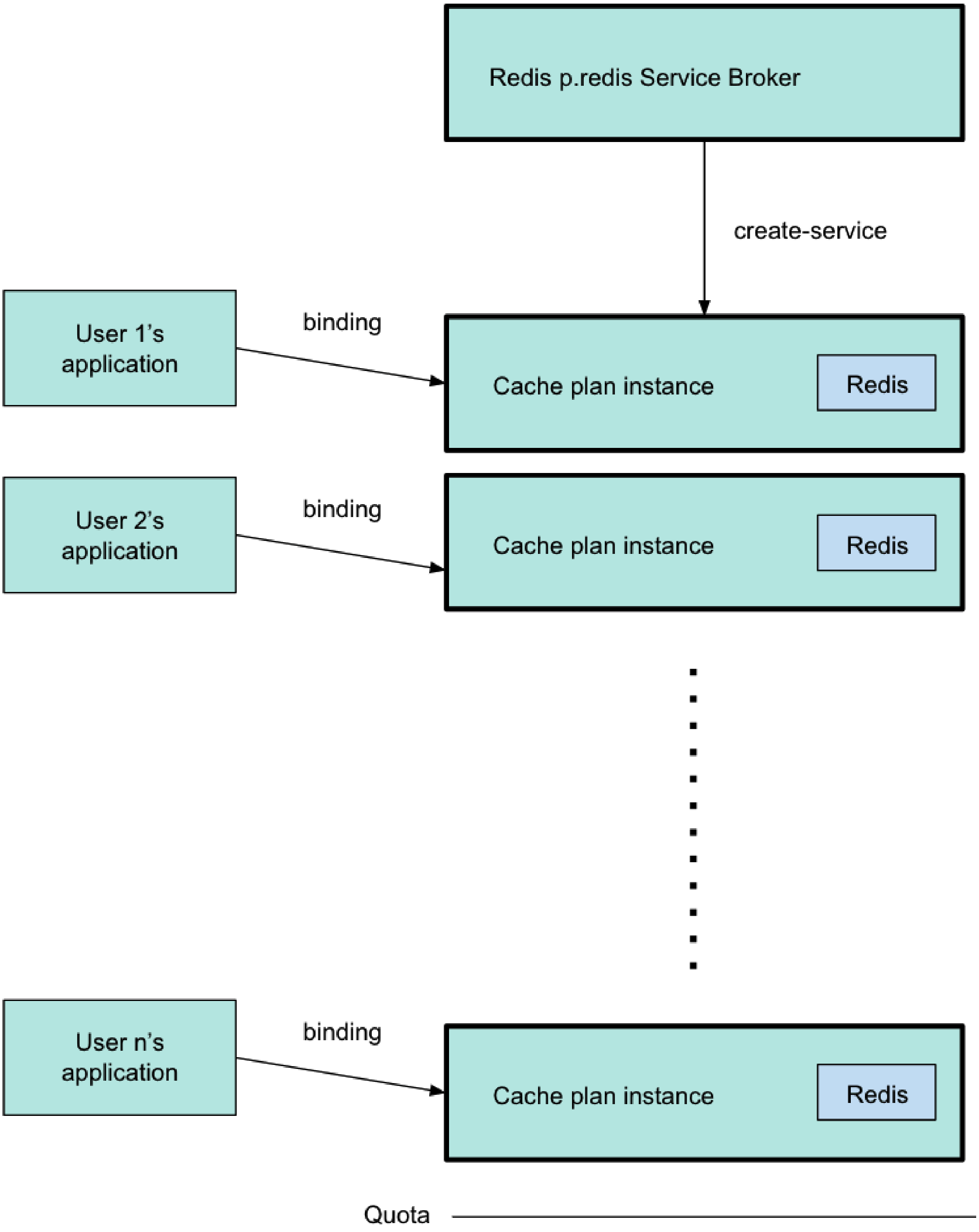
Learn about the architecture, lifecycle, and configurations of the on-demand plan, as well as networking information for the on-demand service.

For similar information for the Shared-VM plans, see Shared-VM Service offering.

Architecture of the on-demand plan

The p.redis service broker manages the on-demand service plan instances.

The following diagram shows the architecture of the service broker and on-demand plans and how the user's app binds to a Redis instance.



You can configure plans in Tanzu Operations Manager, and set global and per-plan quotas for the maximum number of instances.

Developers can create instances of each plan when needed, until a quota is reached, and bind their apps to the instances. The previous diagram shows the p.redis service broker pointing to a cache plan instance, which was created by running `cf create-service`. For more information about this command, see Create a Service Instance in *Using Redis for VMware Tanzu Application Service*.

The diagram shows three different users' apps, each one bound to a separate cache plan instance. Each instance has its own VM. The line below the final instance shows that the quota has been reached, and developers cannot create more instances.

TLS in Redis for Tanzu Application Service

You can enable TLS to secure traffic between apps and service instances. In Redis for Tanzu Application Service, the available options are **Optional** and **Not Configured**.

TLS set to optional

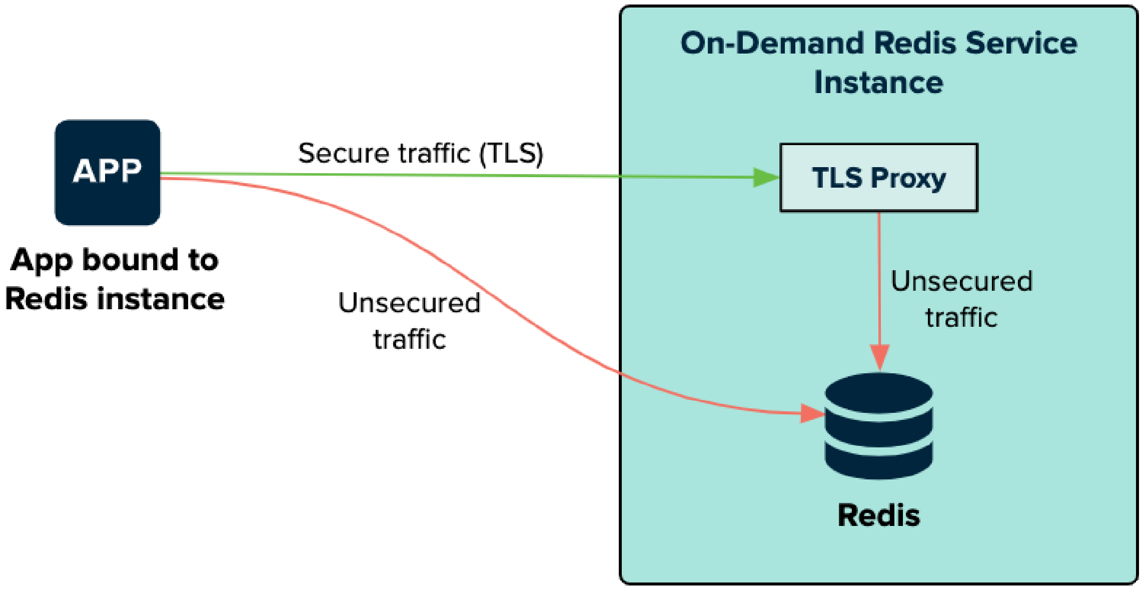
When setting TLS to **Optional** within On-Demand Service Settings, both TLS and non-TLS connections are accepted. TLS traffic goes through a proxy as shown in the diagram below. Enabling TLS is not expected to noticeably reduce performance. This depends, however, on network infrastructure, application architecture, and other such resources being in good shape.

VMware recommends setting TLS as **Optional**, because it allows app developers to migrate to TLS connections regardless of whether traffic is restricted to just TLS connections.

> **Note**
>
> The option to enforce TLS only is not supported in Redis for Tanzu Application Service.

Steeltoe and Spring apps use the TLS port by default, if it is available. Other apps might require further configuration to make use of the correct port.

The following diagram shows how apps communicate with on-demand Redis instances when you set TLS to **Optional**.

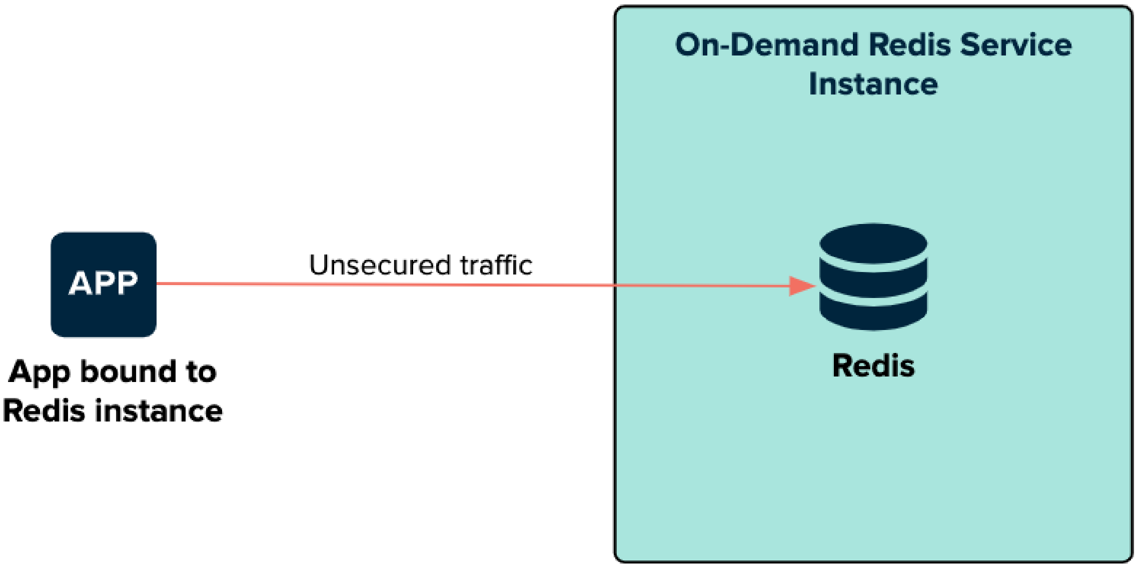


The bound app can connect to the Redis service on the on-demand Redis service instance VM through a TLS proxy or connect directly. The TLS proxy and Redis are both on the Redis service instance. The traffic is secure from the app to the TLS proxy. When on the service instance, the traffic from the TLS proxy to Redis is unsecured.

TLS set to not configured

When setting TLS to **Not Configured** within the On-Demand Service Settings, the communication with service instances remains unchanged from Redis for Pivotal Cloud Foundry v2.1 and earlier.

The following diagram shows how apps communicate with on-demand Redis instances when you set TLS to **Not Configured**.



The bound app connects directly to the Redis service on the on-demand Redis service instance VM. The traffic on this connection is unsecured.

On-demand service plans

Redis for Tanzu Application Service offers on-demand plans as the `p.redis` service within the tile. On-demand plans are best suited to caching. Redis for Tanzu Application Service has tailored the default configuration to this use case.

The default on-demand plan is the **On-Demand Cache Plan**. Service instances of this plan are deployed to a dedicated VM. VMware recommends that you configure these VMs to have 2.5 times more persistent disk than memory.

You can customize service plans by configuring the **Plan name**, **Plan description**, **Server VM type**, and **Server Disk type**. You can add and configure as many service plans as required.

Features of on-demand service plans

- Each on-demand service instance is deployed to its own VM and is suitable for production workloads.
- The service plans are operator-configured and enabled. When enabled, app developers can view the available plans in the Marketplace and provision a Redis instance from that plan.

- Operators can update the cache plan settings, including the VM size and disk size, after the plans have been created.
- Operators and app developers can change certain Redis configurations from the default.
- The default `maxmemory-policy` is `allkeys-lru` and can be updated for other cache policies.
- On-Demand Redis supports Redis Database Backup (RDB) snapshots, but not Append-Only File (AOF) persistence. For more information, see Redis Persistence in the Redis documentation.
- The maximum number of instances is managed by a per-plan and global quota.

For information about setting quotas, see Setting Limits for On-Demand Service Instances.

Configuration of on-demand service plans

For on-demand plans, certain Redis configurations can be set by the operator during plan configuration, and by the app developer during instance provisioning. Other Redis configurations cannot be changed from the default.

Operator configurable Redis settings

The Redis settings that an operator can configure in the tile UI include:

- Redis Client Timeout
- Redis TCP Keepalive
- Max Clients
- Lua Scripting
- Plan Quota

For more information, see Configure On-Demand Plan settings.

App developer configurable Redis settings

The Redis settings that an app developer can configure include:

- `maxmemory-policy`
- `notify-keyspace-events`
- `slowlog-log-slower-than`
- `slowlog-max-len`

For more information, see Customize an On-Demand Service Instance.

Operator notes for on-demand service plans

- Instances of the on-demand plan can be deployed until their number reaches either an operator-set per-plan quota or a global quota. For information about setting quotas, see Setting Limits for On-Demand Service Instances.
- Instances are provisioned based on the On-Demand Services SDK and service broker adapter associated with this plan.

- `maxmemory` in `redis.conf` is set to 45% of the system memory.
- Any on-demand plan can be deactivated from the plan page in Tanzu Operations Manager.

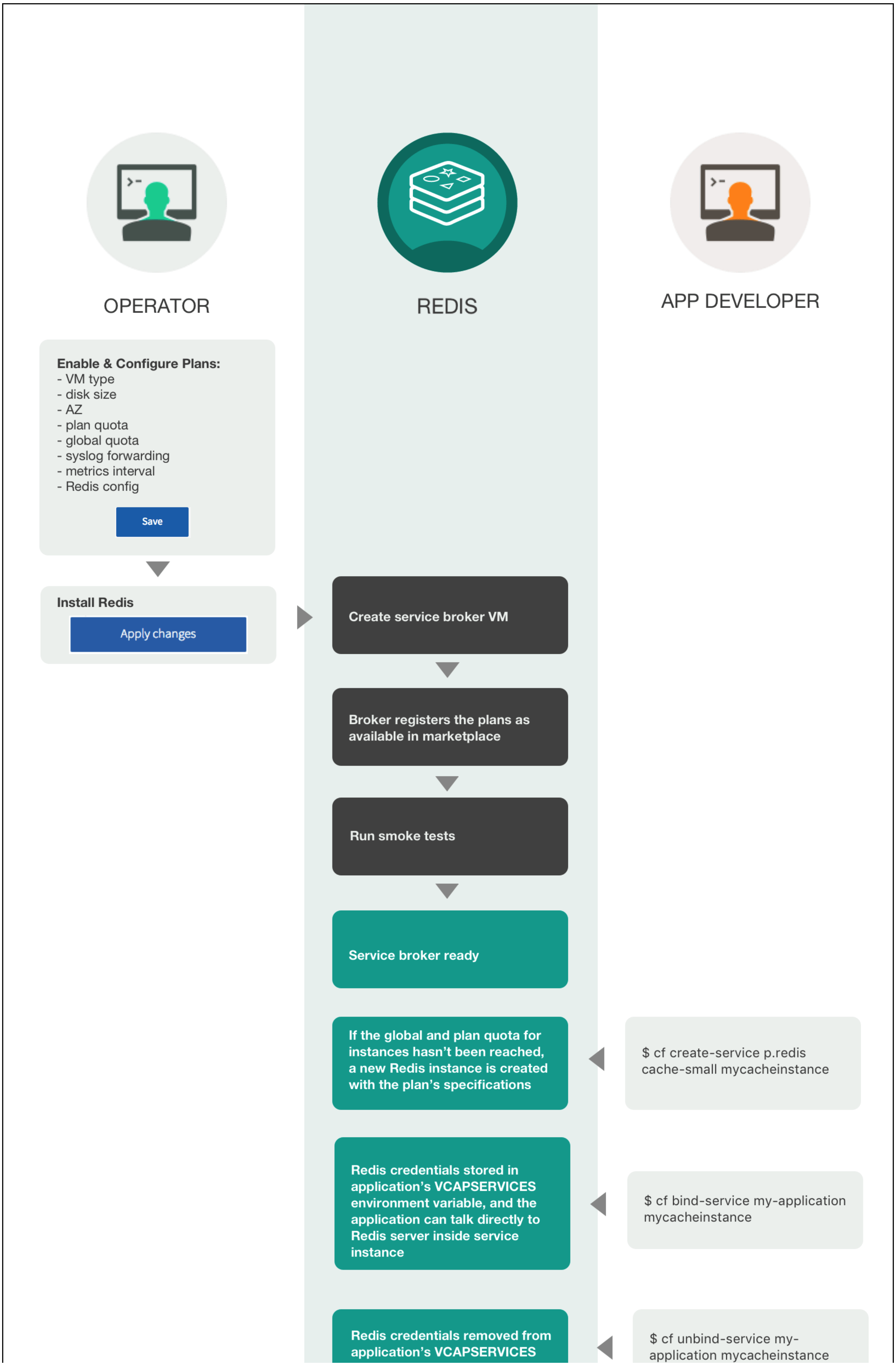
Known limitations for on-demand service plans

Limitations for the on-demand service include:

- Operators must not downsize the VMs or disk size as this can cause data loss in pre-existing instances.
- Operators can update certain plan settings after the plans have been created. To ensure upgrades happen across all instances, set the **upgrade instances** errand to **On**.
- If you update the VM size, disk size, or the Redis configuration settings, thereby enabling Lua Scripting, max-clients, timeout, and TCP keepalive, the settings are implemented in all existing instances.

Lifecycle for on-demand service plan

Here is the lifecycle of Redis for Tanzu Application Service, from an operator installing the tile through an app developer using the service then an operator deleting the tile.



Enable & Configure Plans:
- VM type
- disk size
- AZ
- plan quota
- global quota
- syslog forwarding
- metrics interval
- Redis config

Broker registers the plans as available in marketplace

If the global and plan quota for instances hasn't been reached, a new Redis instance is created with the plan's specifications

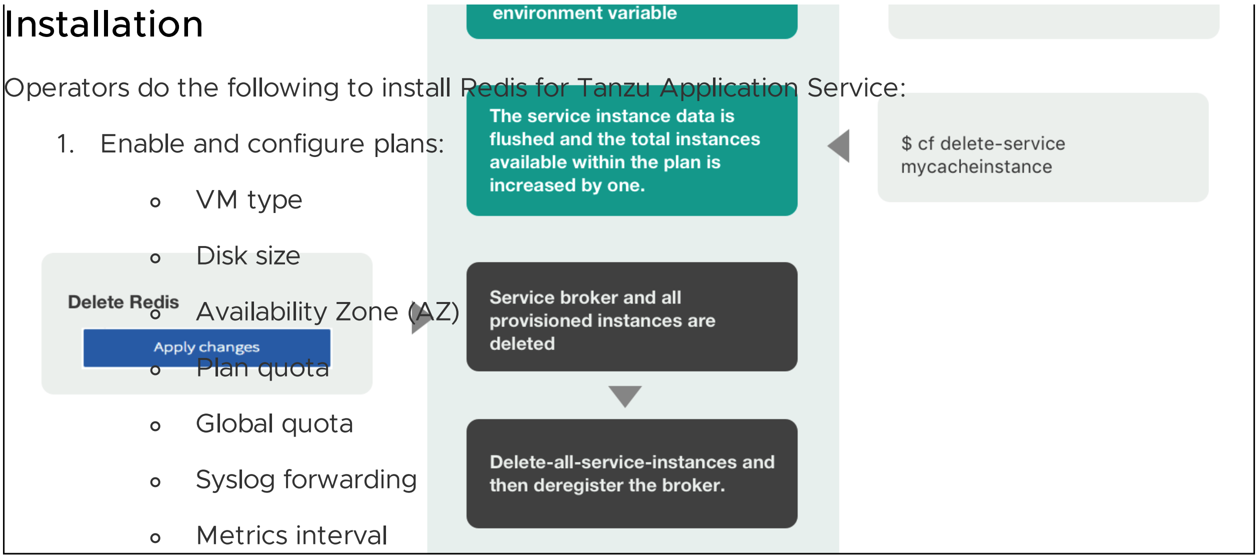
```
$ cf create-service p.redis cache-small mycacheinstance
```

Redis credentials stored in application's VCAPSERVICES environment variable, and the application can talk directly to Redis server inside service instance

```
$ cf bind-service my-application mycacheinstance
```

Redis credentials removed from application's VCAPSERVICES

```
$ cf unbind-service my-application mycacheinstance
```


environment variable
The service instance data is flushed and the total instances available within the plan is increased by one.
$ cf delete-service mycacheinstance
Delete Redis
Apply changes
Service broker and all provisioned instances are deleted
Delete-all-service-instances and then deregister the broker.

Installation

Operators do the following to install Redis for Tanzu Application Service:

1. Enable and configure plans:
    - VM type
    - Disk size
    - Availability Zone (AZ)
    - Plan quota
    - Global quota
    - Syslog forwarding
    - Metrics interval
    - Backup destination
    - Metrics interval
    - Redis config
    - Click **Save**
2. Install Redis
    - Click **Apply changes**

After the operator click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Creates a service broker VM
2. Broker registers the plans as available in marketplace
3. Run smoke tests
4. Service broker ready

Using Redis for Tanzu Application Service

After you have installed Redis for Tanzu Application Service, developers can create service instances, bind and unbind the service instances to apps, and delete service instances.

Create service

When a developer runs the `cf create-service` command, for example:

```
$ cf create-service p-redis cache-small mycacheinstance
```

Redis for Tanzu Application Service does the following:

- If the global and plan quota for instances has not been reached, a new Redis instance is created with the plan’s specifications.

Bind service

When a developer runs the `cf bind-service` command, for example:

```
$ cf bind-service my-application mycacheinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are stored in the app's `VCAP_SERVICES` environment variable and the app can communicate directly with the Redis server inside the service instance

Unbind service

When a developer runs the `cf unbind-service` command, for example:

```
$ cf unbind-service my-application mycacheinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are removed from the app's `VCAP_SERVICES` environment variable

Delete service

When a developer runs the `cf delete-service` command, for example:

```
$ cf delete-service mycacheinstance
```

Redis for Tanzu Application Service does the following:

- The service instance data is flushed and the total instances available within the plan is increased by one

Deletion

You can do the following to delete Redis for Tanzu Application Service:

1. Delete Redis:
    - Click **Apply Changes**

After you click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Service broker and all provisioned instances are deleted.
2. Delete-all-service-instances and then deregister the broker.

On-demand service offering for Redis for VMware Application Service

Redis for VMware Tanzu Application Service offers on-demand and shared-VM service plans.

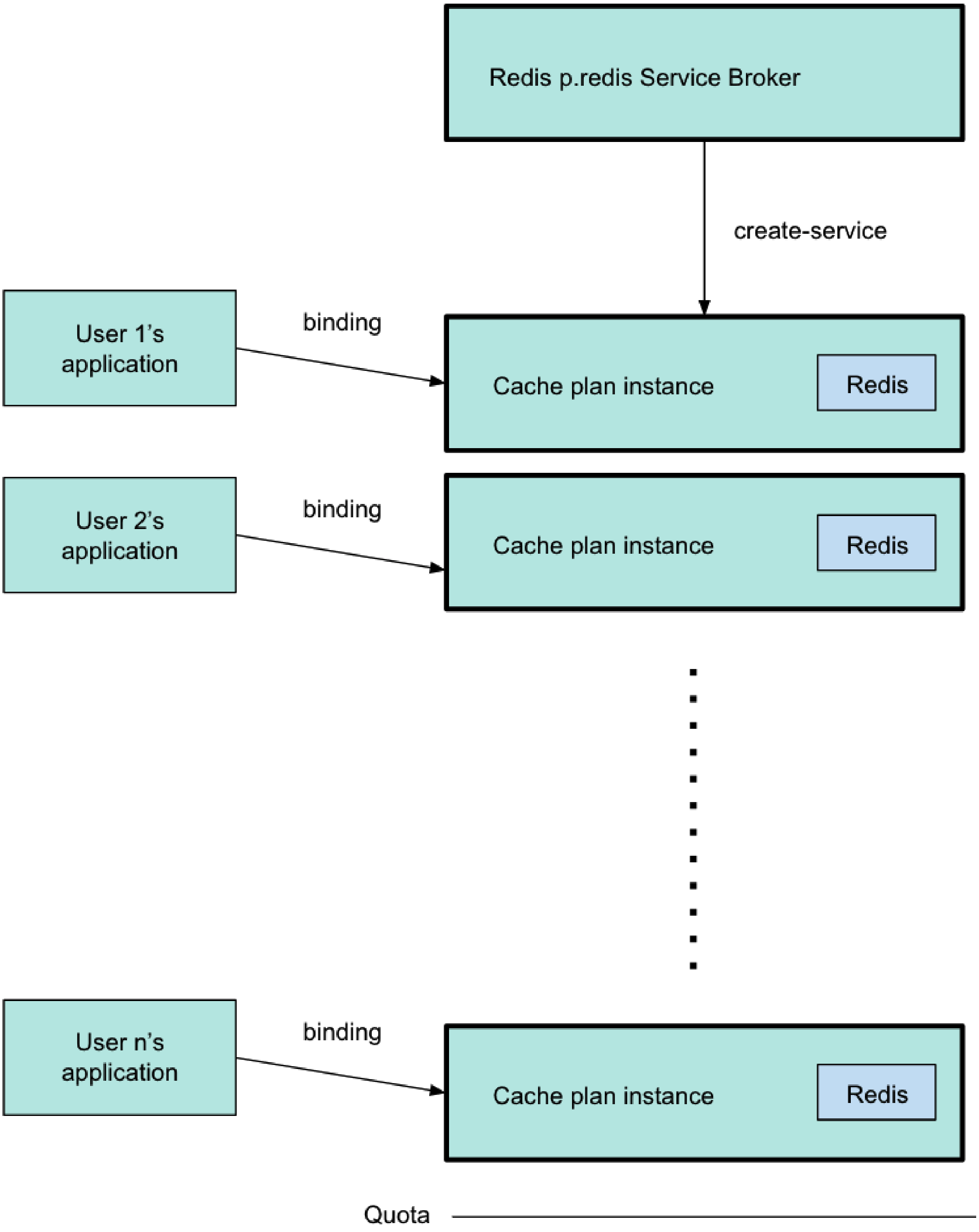
Learn about the architecture, lifecycle, and configurations of the on-demand plan, as well as networking information for the on-demand service.

For similar information for the Shared-VM plans, see Shared-VM Service offering.

Architecture of the on-demand plan

The p.redis service broker manages the on-demand service plan instances.

The following diagram shows the architecture of the service broker and on-demand plans and how the user's app binds to a Redis instance.



You can configure plans in Tanzu Operations Manager, and set global and per-plan quotas for the maximum number of instances.

Developers can create instances of each plan when needed, until a quota is reached, and bind their apps to the instances. The previous diagram shows the p.redis service broker pointing to a cache plan instance, which was created by running `cf create-service`. For more information about this command, see Create a Service Instance in *Using Redis for VMware Tanzu Application Service*.

The diagram shows three different users' apps, each one bound to a separate cache plan instance. Each instance has its own VM. The line below the final instance shows that the quota has been reached, and developers cannot create more instances.

TLS in Redis for Tanzu Application Service

You can enable TLS to secure traffic between apps and service instances. In Redis for Tanzu Application Service, the available options are **Optional** and **Not Configured**.

TLS set to optional

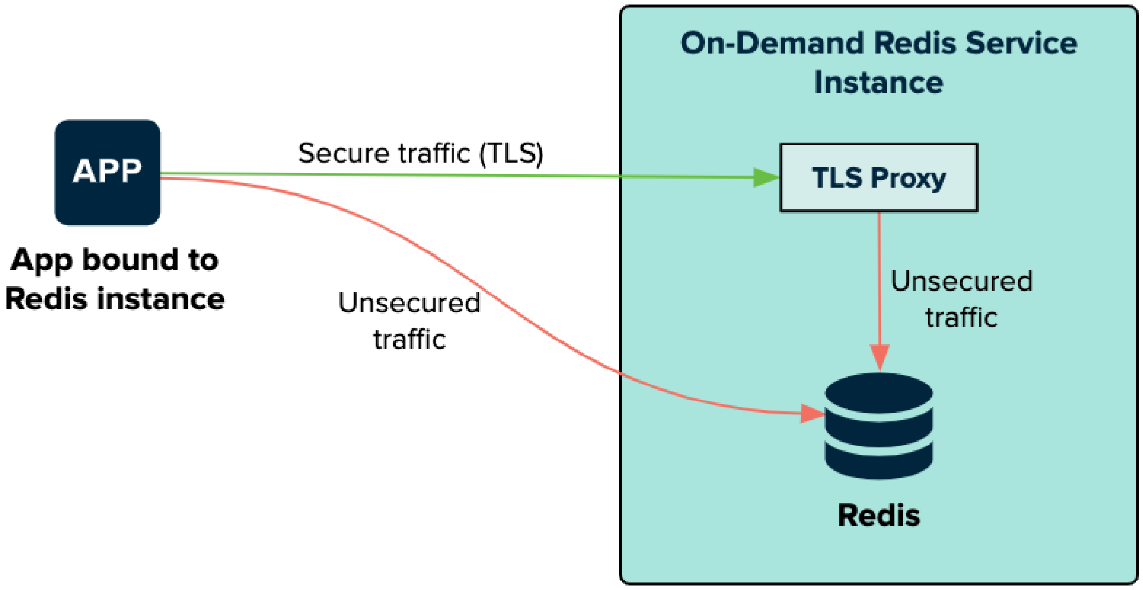
When setting TLS to **Optional** within On-Demand Service Settings, both TLS and non-TLS connections are accepted. TLS traffic goes through a proxy as shown in the diagram below. Enabling TLS is not expected to noticeably reduce performance. This depends, however, on network infrastructure, application architecture, and other such resources being in good shape.

VMware recommends setting TLS as **Optional**, because it allows app developers to migrate to TLS connections regardless of whether traffic is restricted to just TLS connections.

> **Note**
>
> The option to enforce TLS only is not supported in Redis for Tanzu Application Service.

Steeltoe and Spring apps use the TLS port by default, if it is available. Other apps might require further configuration to make use of the correct port.

The following diagram shows how apps communicate with on-demand Redis instances when you set TLS to **Optional**.

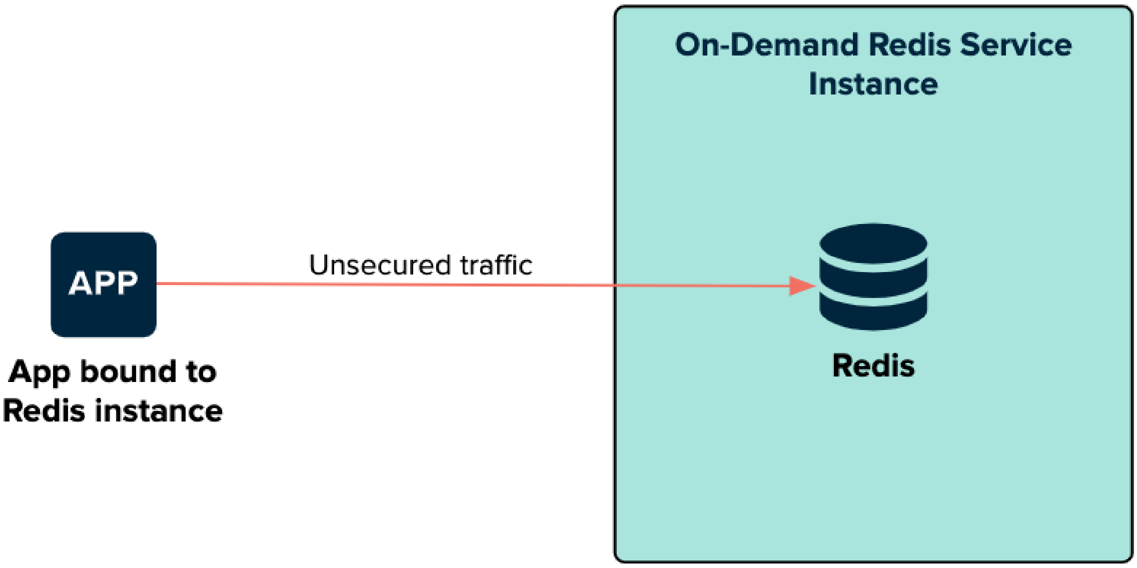


The bound app can connect to the Redis service on the on-demand Redis service instance VM through a TLS proxy or connect directly. The TLS proxy and Redis are both on the Redis service instance. The traffic is secure from the app to the TLS proxy. When on the service instance, the traffic from the TLS proxy to Redis is unsecured.

TLS set to not configured

When setting TLS to **Not Configured** within the On-Demand Service Settings, the communication with service instances remains unchanged from Redis for Pivotal Cloud Foundry v2.1 and earlier.

The following diagram shows how apps communicate with on-demand Redis instances when you set TLS to **Not Configured**.



The bound app connects directly to the Redis service on the on-demand Redis service instance VM. The traffic on this connection is unsecured.

On-demand service plans

Redis for Tanzu Application Service offers on-demand plans as the `p.redis` service within the tile. On-demand plans are best suited to caching. Redis for Tanzu Application Service has tailored the default configuration to this use case.

The default on-demand plan is the **On-Demand Cache Plan**. Service instances of this plan are deployed to a dedicated VM. VMware recommends that you configure these VMs to have 2.5 times more persistent disk than memory.

You can customize service plans by configuring the **Plan name**, **Plan description**, **Server VM type**, and **Server Disk type**. You can add and configure as many service plans as required.

Features of on-demand service plans

- Each on-demand service instance is deployed to its own VM and is suitable for production workloads.

- The service plans are operator-configured and enabled. When enabled, app developers can view the available plans in the Marketplace and provision a Redis instance from that plan.
- Operators can update the cache plan settings, including the VM size and disk size, after the plans have been created.
- Operators and app developers can change certain Redis configurations from the default.
- The default `maxmemory-policy` is `allkeys-lru` and can be updated for other cache policies.
- On-Demand Redis supports Redis Database Backup (RDB) snapshots, but not Append-Only File (AOF) persistence. For more information, see Redis Persistence in the Redis documentation.
- The maximum number of instances is managed by a per-plan and global quota.

For information about setting quotas, see Setting Limits for On-Demand Service Instances.

Configuration of on-demand service plans

For on-demand plans, certain Redis configurations can be set by the operator during plan configuration, and by the app developer during instance provisioning. Other Redis configurations cannot be changed from the default.

Operator configurable Redis settings

The Redis settings that an operator can configure in the tile UI include:

- Redis Client Timeout
- Redis TCP Keepalive
- Max Clients
- Lua Scripting
- Plan Quota

For more information, see Configure On-Demand Plan settings.

App developer configurable Redis settings

The Redis settings that an app developer can configure include:

- `maxmemory-policy`
- `notify-keyspace-events`
- `slowlog-log-slower-than`
- `slowlog-max-len`

For more information, see Customize an On-Demand Service Instance.

Operator notes for on-demand service plans

- Instances of the on-demand plan can be deployed until their number reaches either an operator-set per-plan quota or a global quota. For information about setting quotas, see Setting Limits for On-Demand Service Instances.

- Instances are provisioned based on the On-Demand Services SDK and service broker adapter associated with this plan.
- `maxmemory` in `redis.conf` is set to 45% of the system memory.
- Any on-demand plan can be deactivated from the plan page in Tanzu Operations Manager.

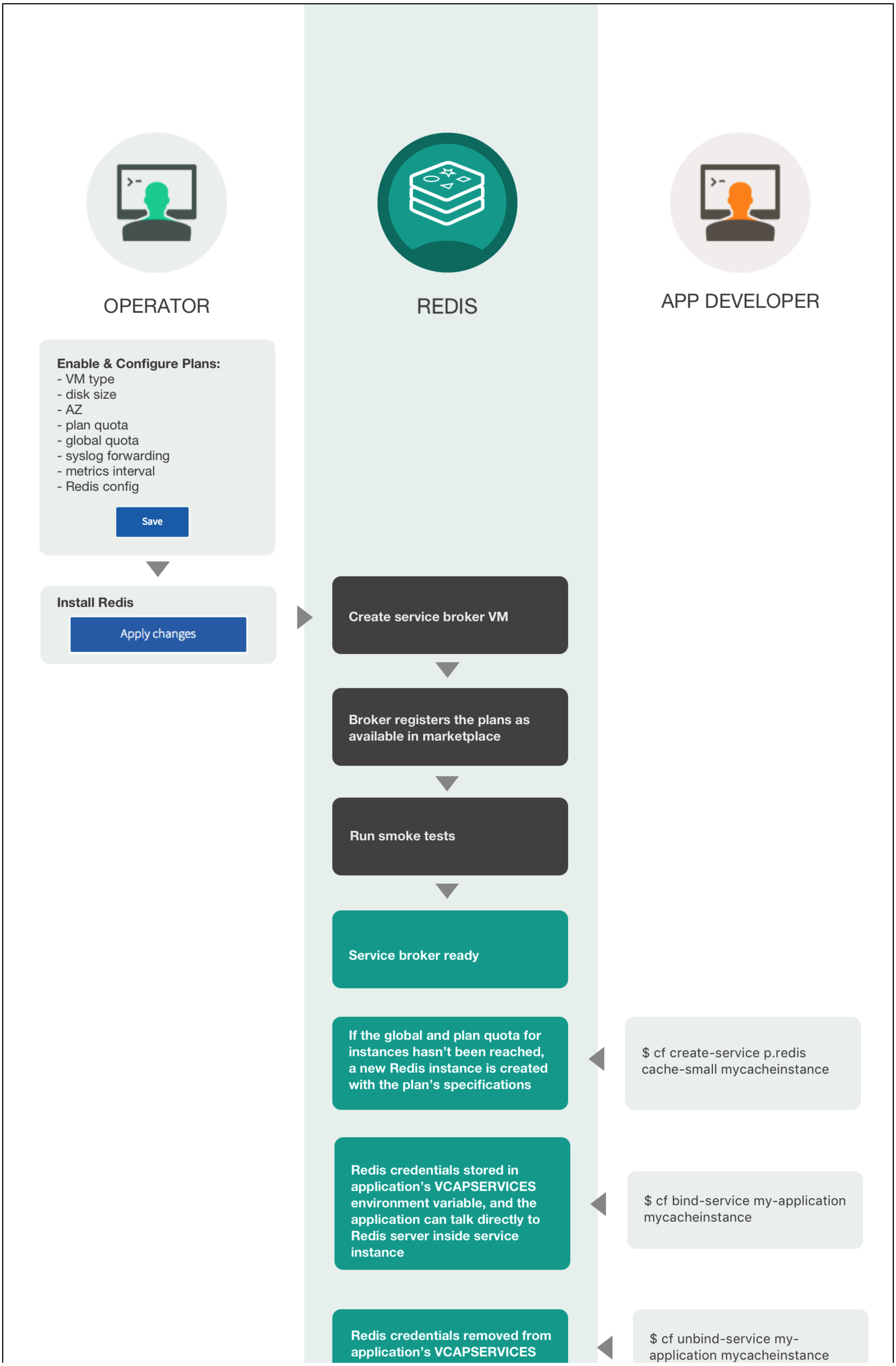
Known limitations for on-demand service plans

Limitations for the on-demand service include:

- Operators must not downsize the VMs or disk size as this can cause data loss in pre-existing instances.
- Operators can update certain plan settings after the plans have been created. To ensure upgrades happen across all instances, set the **upgrade instances** errand to **On**.
- If you update the VM size, disk size, or the Redis configuration settings, thereby enabling Lua Scripting, max-clients, timeout, and TCP keepalive, the settings are implemented in all existing instances.

Lifecycle for on-demand service plan

Here is the lifecycle of Redis for Tanzu Application Service, from an operator installing the tile through an app developer using the service then an operator deleting the tile.



OPERATOR

REDIS

APP DEVELOPER

Enable & Configure Plans:
- VM type
- disk size
- AZ
- plan quota
- global quota
- syslog forwarding
- metrics interval
- Redis config

Install Redis

Create service broker VM

Broker registers the plans as available in marketplace

Run smoke tests

Service broker ready

If the global and plan quota for instances hasn't been reached, a new Redis instance is created with the plan's specifications

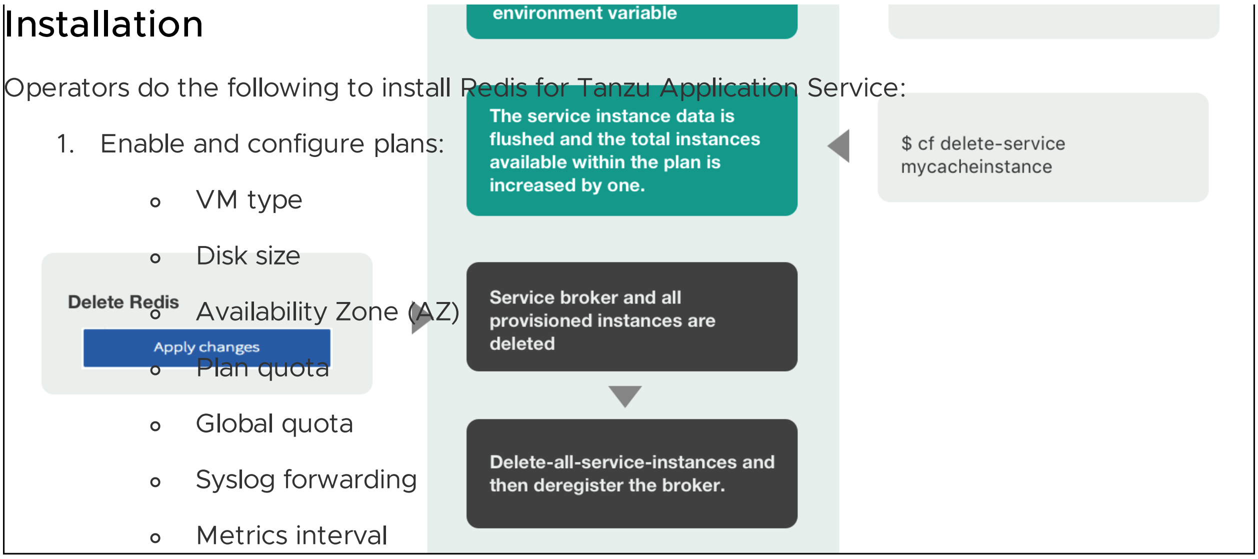
```
$ cf create-service p.redis
cache-small mycacheinstance
```

Redis credentials stored in application's VCAPSERVICES environment variable, and the application can talk directly to Redis server inside service instance

```
$ cf bind-service my-application
mycacheinstance
```

Redis credentials removed from application's VCAPSERVICES

```
$ cf unbind-service my-
application mycacheinstance
```


environment variable
The service instance data is flushed and the total instances available within the plan is increased by one.
$ cf delete-service mycacheinstance
Delete Redis
Apply changes
Service broker and all provisioned instances are deleted
Delete-all-service-instances and then deregister the broker.

Installation

Operators do the following to install Redis for Tanzu Application Service:

1. Enable and configure plans:
   - VM type
   - Disk size
   - Availability Zone (AZ)
   - Plan quota
   - Global quota
   - Syslog forwarding
   - Metrics interval
   - Backup destination
   - Metrics interval
   - Redis config
   - Click **Save**
2. Install Redis
   - Click **Apply changes**

After the operator click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Creates a service broker VM
2. Broker registers the plans as available in marketplace
3. Run smoke tests
4. Service broker ready

Using Redis for Tanzu Application Service

After you have installed Redis for Tanzu Application Service, developers can create service instances, bind and unbind the service instances to apps, and delete service instances.

Create service

When a developer runs the `cf create-service` command, for example:

```
$ cf create-service p-redis cache-small mycacheinstance
```

Redis for Tanzu Application Service does the following:

- If the global and plan quota for instances has not been reached, a new Redis instance is created with the plan's specifications.

Bind service

When a developer runs the `cf bind-service` command, for example:

```
$ cf bind-service my-application mycacheinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are stored in the app's `VCAP_SERVICES` environment variable and the app can communicate directly with the Redis server inside the service instance

Unbind service

When a developer runs the `cf unbind-service` command, for example:

```
$ cf unbind-service my-application mycacheinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are removed from the app's `VCAP_SERVICES` environment variable

Delete service

When a developer runs the `cf delete-service` command, for example:

```
$ cf delete-service mycacheinstance
```

Redis for Tanzu Application Service does the following:

- The service instance data is flushed and the total instances available within the plan is increased by one

Deletion

You can do the following to delete Redis for Tanzu Application Service:

1. Delete Redis:
    - Click **Apply Changes**

After you click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Service broker and all provisioned instances are deleted.
2. Delete-all-service-instances and then deregister the broker.

Shared-VM service offering for Redis for VMware Tanzu Application Service

Redis for VMware Tanzu Application Service offers on-demand and shared-VM service plans. This topic tells you about the architecture, lifecycle, and configurations of the shared-VM plan.

For similar information for the on-demand service plan, see On-Demand Service Offering.

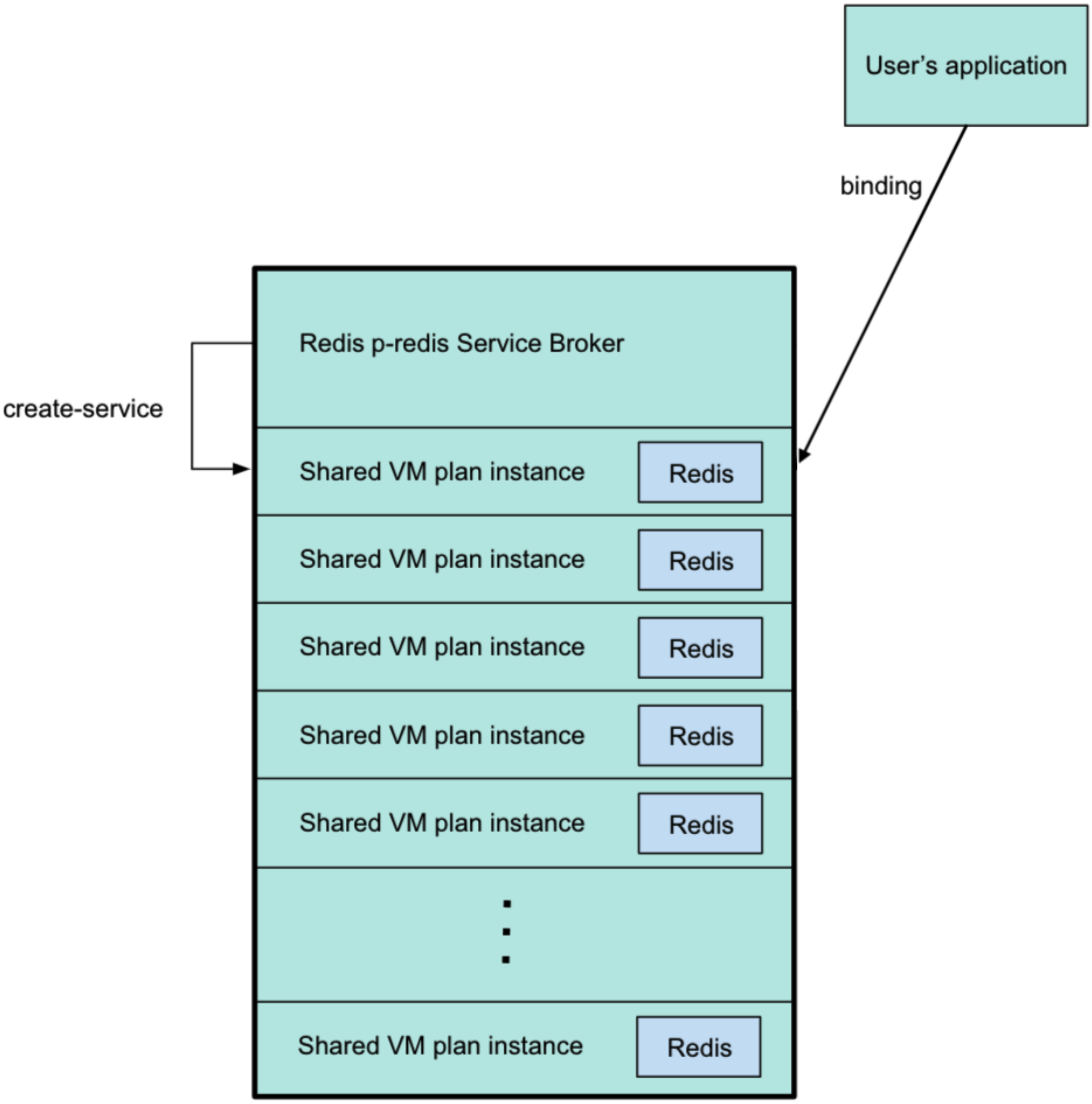
About the shared-VM plan

The shared-VM plan is a pre-provisioned service plan for development and testing purposes only. An instance of this plan provisions a single Redis process on a single shared VM. This plan is suitable for workloads that do not require dedicated hardware. This plan is **not** suitable for production purposes.

Architecture diagram for shared plans

The p-redis service broker manages the shared-vm plan service instances.

The following diagram shows the architecture of the service broker and shared-VM plans and how the user's app binds to a Redis instance.



The previous diagram shows the p-redis service broker pointing to a shared-VM plan instance, which was created by running `cf create-service`. For more information about this command, see Create a Service Instance in *Using Redis for VMware Tanzu Application Service*.

The service broker VM contains the shared-VM plan instances. Six shared-VM plan instances are shown. These are provisioned when created by the app developer. The maximum number is specified by the operator.

Each shared instance has its own Redis server, with credentials stored in the `VCAP_SERVICES` environment variable.

The user’s app is bound to a shared-VM plan instance, shown above by an arrow labeled *binding* pointing from the app to a shared-VM plan instance. For information about binding, see Bind a Service Instance to your App in *Using Redis for VMware Tanzu Application Service*.

Settings for shared-VM service plans

You cannot change the default Redis settings for shared-VM plans. Because of this, you cannot run `cf update-service` with the `-c` flag to set config parameters, as described in the Cloud Foundry documentation.

The default Redis settings are as follows:

Memory policy

Redis is configured with a `maxmemory-policy` of `no-eviction`. This policy means that when the memory is full, the service does not evict any keys or perform any write operations until memory becomes available.

Persistence

Shared-VM Redis supports both Redis Database Backup (RDB) and Append-Only File (AOF) persistence options. Redis writes to the AOF log every second. For more information, see Redis Persistence in the Redis documentation.

Maximum number of connections

The maximum number of connections, `maxclients`, is set at 10,000 by default. Redis might reduce this number when run on a system with a low maximum number of file descriptors. You can retrieve the actual setting on your Redis service instances with the Redis command `CONFIG GET maxclients`.

You can run the Redis command `CONFIG SET maxclients NUMBER` in your service instance to reduce `maxclients` until the next BOSH action occurs. For example:

```
$ CONFIG SET maxclients 9000
```

You cannot set `maxclients` above 10,000 and you cannot configure shared plans to permanently use a custom limit.

Replication and event notification

The replication and event notifications are not configured.

Change the service instances limit

This plan deploys a Redis instance on a shared VM and a single service broker VM. To prevent this, set the **Max instances limit** on the **Shared-VM Plan** tab in Tanzu Operations Manager to `0`.

You can increase the maximum number of service instances that can run on a shared VM from the default five to 250. There is a hard maximum of 250 shared instances.

If you increase the number of instances that can be run on a VM, consider increasing the resources allocated to the VM, especially RAM and CPU. Failure to do so might lead to a degradation of performance.

You can also increase the maximum amount of RAM allocated to each service instance that is running on this VM.

If you decrease the service instance limit, any instances that are now running beyond the limit are not automatically terminated. You cannot create any new instances until the total falls below the new limit.
For example, if you use 10 service instances, and you then reduce the limit to 8, the two instances outside the limit continue to run until you terminate them.

The number of shared-VM instances available to developers is set by the operator.
The maximum number of shared-VM instances is relative to the memory allocated to each shared-VM instance and the total memory of the Redis service broker.
For more information, see Configure Shared-VM Plan settings.

Lua scripting

You can activate or deactivate Lua scripting. Changes to this configuration apply to all existing shared-VM instances. Lua scripting can adversely affect the performance of other service instances on the VM, so VMware recommends deactivating Lua scripting unless developers need it enabled. For more information, see Configure Shared-VM Plan settings.

> ⚠ **Caution**
>
> The Steeltoe connector for Redis requires Redis for Tanzu Application Service to support Lua scripting. Check if any of your apps require Lua scripting. By default, Lua scripting is deactivated for Redis for Tanzu Application Service, but an operator can change the setting to enable it by selecting the **Lua Scripting** checkbox in the Shared-VM Plan configuration pane.

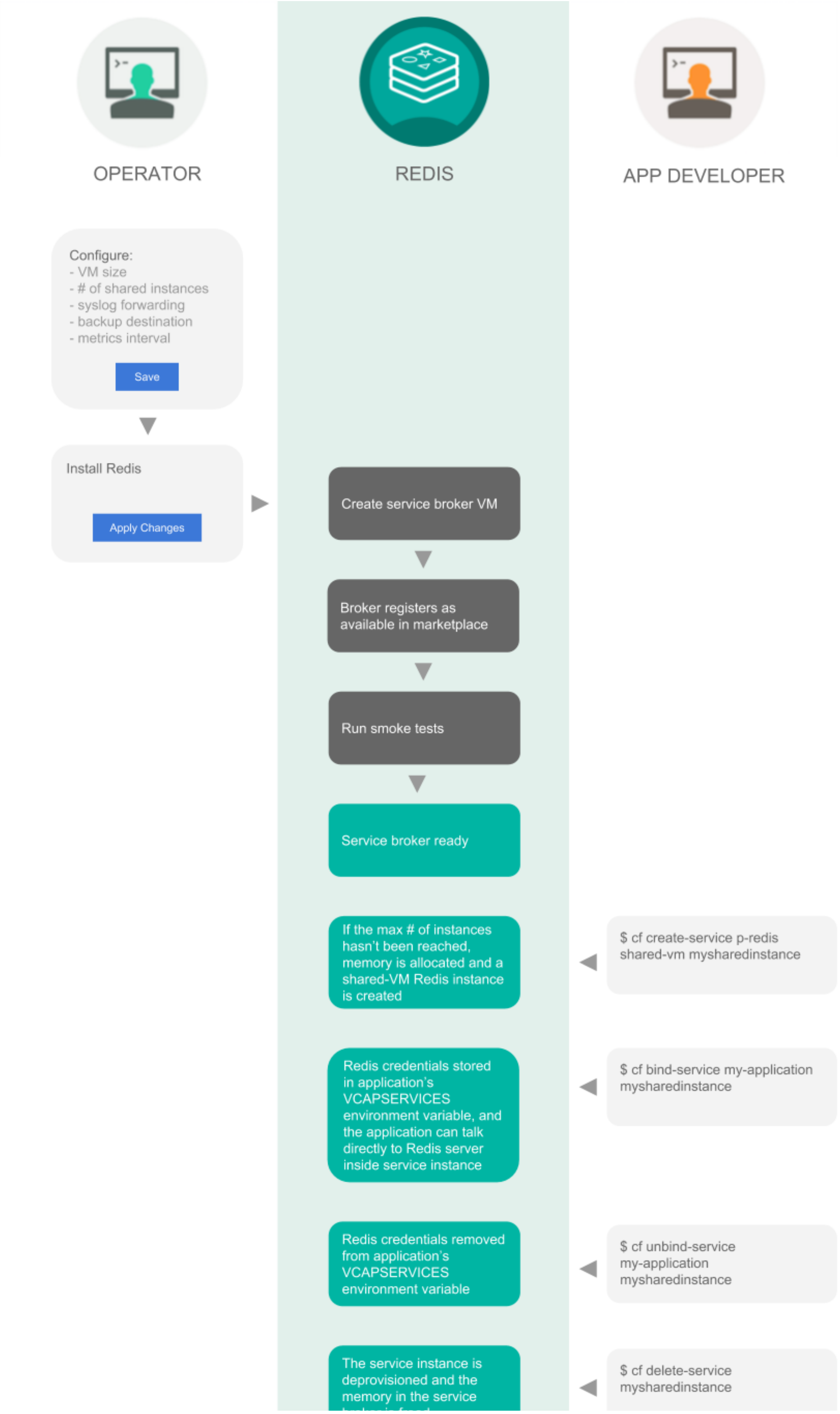
Known limitations of the shared-VM plan

The shared-vm plan cannot:

- Scale beyond a single VM
- Run the commands `CONFIG`, `MONITOR`, `SAVE`, `BGSAVE`, `SHUTDOWN`, `BGREWRITEAOF`, `REPLICAOF`, `SLAVEOF`, `DEBUG`, or `SYNC`
- Constrain CPU or disk usage
- Manage “noisy neighbor” problems, which makes it unsuitable for production apps

Lifecycle for shared-VM service plan

This is the lifecycle of Redis for Tanzu Application Service, from an operator installing the tile, to an app developer using the service, to an operator deleting the tile.



Configure:
- VM size
- # of shared instances
- syslog forwarding
- backup destination
- metrics interval

Install Redis

Create service broker VM

Broker registers as available in marketplace

Run smoke tests

Service broker ready

If the max # of instances hasn't been reached, memory is allocated and a shared-VM Redis instance is created

$ cf create-service p-redis shared-vm mysharedinstance

Redis credentials stored in application's VCAPSERVICES environment variable, and the application can talk directly to Redis server inside service instance

$ cf bind-service my-application mysharedinstance

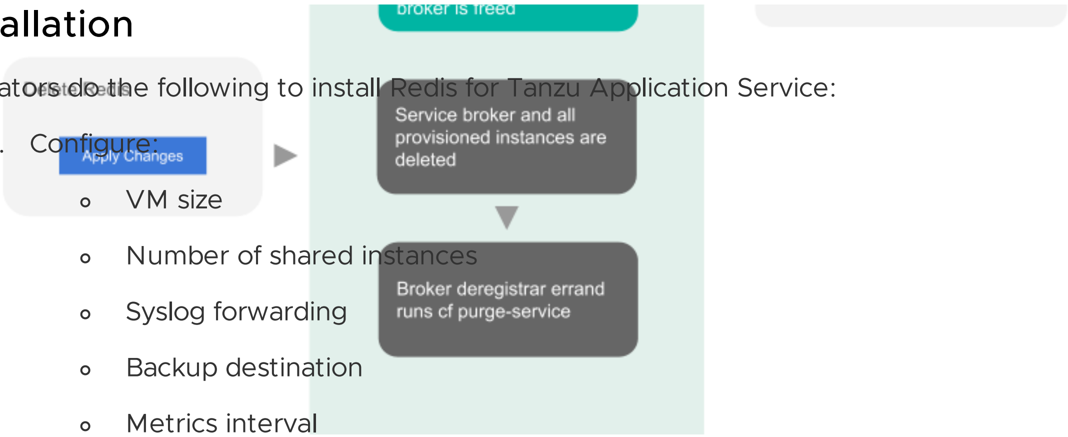
Redis credentials removed from application's VCAPSERVICES environment variable

$ cf unbind-service my-application mysharedinstance

The service instance is deprovisioned and the memory in the service

$ cf delete-service mysharedinstance

Installation


broker is freed
Apply Changes
Service broker and all provisioned instances are deleted
Broker deregistrar errand runs cf purge-service

Operators do the following to install Redis for Tanzu Application Service:

1. Configure:
   - VM size
   - Number of shared instances
   - Syslog forwarding
   - Backup destination
   - Metrics interval
   - Click **Save**
2. Install Redis
   - Click **Apply changes**

After you click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Create service broker VM
2. Broker registers as available in marketplace
3. Run smoke tests
4. Service broker ready

Using Redis for Tanzu Application Service

After you have installed Redis for Tanzu Application Service, developers can create service instances, bind and unbind the service instances to apps, and delete service instances.

Create service

When a developer runs the `cf create-service` command, for example:

```
$ cf create-service p-redis shared-vm mysharedinstance
```

Redis for Tanzu Application Service does the following:

- If the maximum number of instances has not been reached, memory is allocated and a shared-VM Redis instance is created

Bind service

When a developer runs the `cf bind-service` command, for example:

```
$ cf bind-service my-application mysharedinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are stored in the app's `VCAP_SERVICES` environment variable and the app can talk directly to the Redis server inside the service instance

Unbind service

When a developer runs the `cf unbind-service` command, for example:

```
$ cf unbind-service my-application mysharedinstance
```

Redis for Tanzu Application Service does the following:

- Redis credentials are removed from the app's `VCAP_SERVICES` environment variable

Delete service

When a developer runs the `cf delete-service` command, for example:

```
$ cf delete-service mysharedinstance
```

Redis for Tanzu Application Service does the following:

- The service instance is deprovisioned and the memory in the service broker is freed

Deletion

You do the following to delete Redis for Tanzu Application Service:

1. Delete Redis
    - Click **Apply Changes**

After you click **Apply Changes**, Redis for Tanzu Application Service does the following:

1. Service broker and all provisioned instances are deleted
2. Broker deregistrar errand runs `cf purge-service`

Networking for on-demand Redis services

This topic tells you about the networking considerations for the Redis for VMware Tanzu Application Service on-demand service.

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. You can use dynamic networking with asynchronous service provisioning to define dynamically-provisioned service networks. For more information, see Default network and service network.

On-Demand services are enabled by default on all networks. You can create separate networks to host services in BOSH Director, if required. You can select which network hosts on-demand service instances when you configure the tile for that service.

Default network and service network

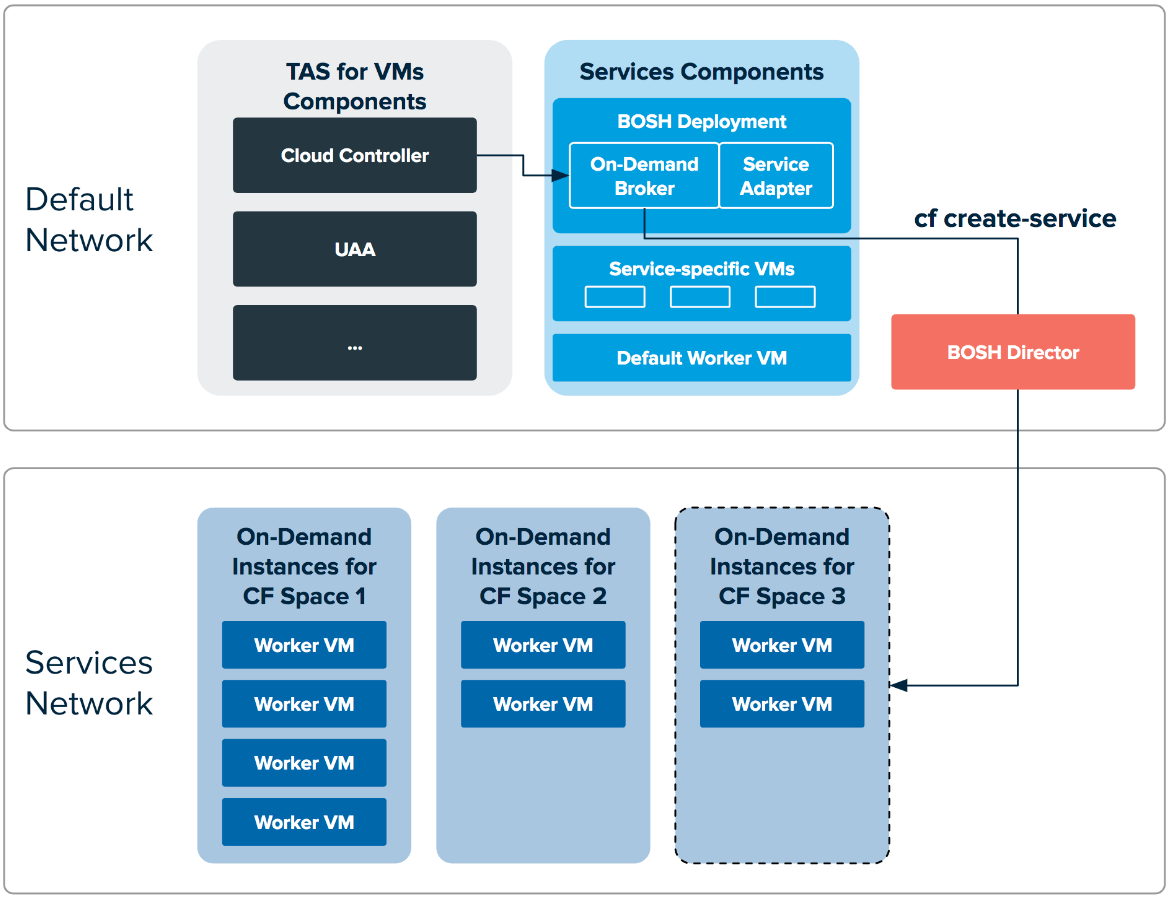
On-Demand Redis for Tanzu Application Service services use BOSH to dynamically deploy VMs and create single-tenant service instances in a dedicated network. On-Demand services use the dynamically-provisioned service network to host single-tenant worker VMs. These worker VMs run as service instances within 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 shows worker VMs in an on-demand service instance running on a separate services network, while other components run on the default network.



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 Redis for Tanzu Application Service service instance runs the Redis for Tanzu Application Service 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.

Source component	Destination component	Default TCP port	Notes
ODB	**BOSH Director** **BOSH UAA**	25555 8443 8844	The default ports are not configurable.
ODB	**TAS for VMs**	8443	The default port is not configurable.
Errand VMs	**TAS for VMs** **ODB** **Deployed service instances**	8443 8080 6379 16379	The default ports are not configurable.
BOSH Agent	**BOSH Director**	4222	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. The communication between these components is two-way.
Deployed apps on TAS for VMs	**Deployed service instances**	6379 16379	This is the default port where Redis is deployed and is the default for using Redis with TLS.
TAS for VMs	**ODB**	8080	The default port is not configurable.

For a complete list of ports and ranges used in Redis for Tanzu Application Service, see Network configuration.

Security for Redis for Tanzu Application Service

This topic gives you security recommendations for Redis for VMware Tanzu Application Service.

To allow Redis for VMware Tanzu Application Service to have network access you must create app security groups (ASGs). For more information, see Networks, security, and assigning AZs.

VMware recommends the following best practices for security:

- Run Redis for Tanzu Application Service in its own network. For more information, see Creating networks in Tanzu Operations Manager.
- Use Redis for Tanzu Application Service with the IPsec Add-on. For information about the IPsec Add-on, see Securing data in transit with the IPsec add-on.
- Do not use a single Redis for Tanzu Application Service instance for multi-tenancy. A single Redis instance of the On-Demand service should only support a single workload.
- Do not use the Shared-VM service for production use cases. It is not considered adequately secure for that purpose, even though it is designed for multi-tenancy.
- Set TLS to **Optional** and encourage app developers to make use of the TLS port. For more information, see Using TLS.

Service-gateway access for Redis for Tanzu Application Service

Service-gateway access enables a Redis for VMware Tanzu Application Service on-demand service instance to connect to external components that are not on the same foundation as the service instance. These components might be on another foundation or hosted outside of the foundation.

For related procedures, see:

- Enabling Service-Gateway access
- Create a Service Instance with Service-Gateway access

There are multiple use cases for service-gateway access.

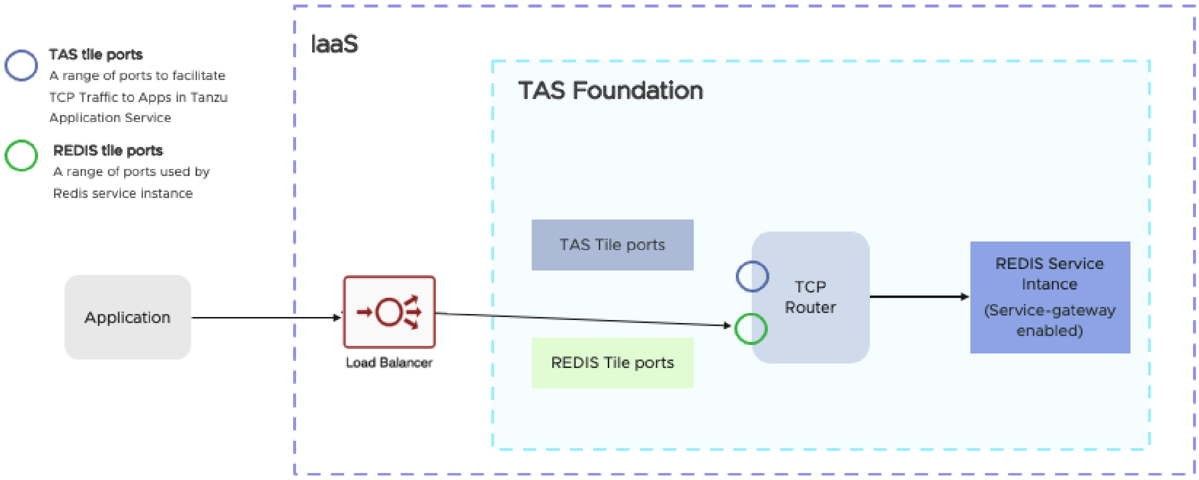
For example:

- Accessing Redis from apps deployed to VMware Tanzu Application Service for VMs (TAS for VMs) in a different foundation.
- Using Redis as a service for apps that are not deployed to TAS for VMs.

Architecture

Service Gateway access to Redis for Tanzu Application Service instances leverages the TCP Router in TAS for VMs.

Any Redis requests that an app makes are forwarded through DNS to a load balancer that can route traffic from outside to inside the foundation. This load balancer (the TCP Router) opens a range of ports that are reserved for any TAS application traffic. When an app developer creates a service instance on a plan with service-gateway access enabled, a port from such a range is provisioned for that service instance. The load balancer then forwards the requests for this Redis for Tanzu Application Service service instance to the TCP router.


TAS tile ports
A range of ports to facilitate TCP Traffic to Apps in Tanzu Application Service
REDIS tile ports
A range of ports used by Redis service instance
IaaS
TAS Foundation
Application
Load Balancer
TAS Tile ports
REDIS Tile ports
TCP Router
REDIS Service Intance (Service-gateway enabled)

Introduction for Redis operators

This topic for operators introduces you to some best practices for Redis for VMware Tanzu Application Service. It does not provide details about operation.

Best practices

VMware recommends that operators follow these guidelines:

- **Resource allocation**—Work with app developers to anticipate memory requirements and to configure VM sizes. Instances of the Shared-VM service have identical VM sizes. However, with the On-Demand service, app developers can choose from three different plans, each with its own VM size and quota. See the service offering for the On-Demand Service Offering and Resource Usage Planning for On-Demand plans.
- **Logs**—Configure a syslog output. Storing logs in an external service helps operators debug issues both current and historical. See Configure Syslog Output. In particular, set up alerts on critical logs, such as service backups so that you are alerted if a backup fails.
- **Monitoring**—Set up a monitoring dashboard for metrics to track the health of the installation.
- **Backing up data**—When using Redis for persistence, configure automatic backups so that data can be restored in an emergency. Validate the backed-up data with a test restore. See Configuring Automated Backups and also Manually backing up and restoring Redis for Pivotal Cloud Foundry in the VMware Tanzu Support knowledge base.
- **Using**—Instances of the On-Demand service run on dedicated VMs. Apps in production should have an on-demand instance to prevent performance issues caused by sharing an instance. The Shared-VM service shares a VM across many instances. VMware recommends that you only use the Shared-VM service for development and testing, but not in production environments. For more information about the plans, see the On-Demand Service Offering and the Shared-VM Service Offering.

Redis key count and memory size

Redis can handle up to 2^{32} keys, and was tested in practice to handle at least 250 million keys per instance. Every hash, list, set, and sorted set, can hold 2^{32} elements. VM memory is more likely to be a limiting factor than number of keys that can be handled.

Errands

Redis for VMware Tanzu Application Service includes the following errands.

Post-deploy errands

The following post-deploy errands are run by default when **Apply Changes** is triggered. These errands run whether or not there has been a configuration change in the Redis for Tanzu Application Service tile.

Tanzu Operations Manager UI name	BOSH errand name	Description
Broker registrar	`broker-registrar`	Registers the cf-redis-broker with TAS for VMs to offer the `p-redis` service, that is, the shared-VM plan.
Smoke tests	`smoke-tests`	Runs lifecycle tests for shared-VM plans if these have been enabled and there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding, and deprovisioning of service instances.
Register on-demand broker	`register-broker`	Registers the on-demand Redis broker with TAS for VMs to offer the `p.redis` service (on-demand plans).
On-demand smoke tests	`on-demand-broker-smoke-tests`	Runs lifecycle tests for enabled plans of the `p.redis` service if there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding and deprovisioning of service instances.
Upgrade all on-demand service instances	`upgrade-all-service-instances`	Upgrades on-demand service instances to use the latest plan configuration, service releases, and stemcell. This causes downtime to any service instances with available upgrades.

The following post-deploy errands do not run by default when **Apply Changes** is triggered. These errands help operators to troubleshoot and maintain their service fleet.

Tanzu Operations Manager UI name	BOSH errand name	Description
Recreate all on-demand service instances	`recreate-all-service-instances`	Re-creates on-demand service instances one-by-one. This causes downtime for all service instances.
Find orphan on-demand service instances	`orphan-deployments`	Finds all orphan on-demand service instances. The cleanup of orphan on-demands service instances can be carried out manually.

Pre-delete errands

The following pre-delete errands are run by default when the Redis for Tanzu Application Service tile is deleted:

Tanzu Operations Manager UI name	BOSH errand name	Description
Broker deregistrar	`broker-deregistrar`	Deregisters the `cf-redis-broker`.

Tanzu Operations Manager UI name	BOSH errand name	Description
Delete all on-demand service instances and deregister broker	`delete-all-service-instances-and-deregister-broker`	Deletes all on-demand instances and deregisters the on-demand Redis broker.

Turning off post-deploy errands

VMware recommends that you run the post-deploy errands at any trigger of **Apply Changes**. However, this practice can extend the duration of applying changes by several minutes every time. This section helps you decide when it is safe to skip some post-deploy errands.

Changes to Redis for Tanzu Application Service tile configuration

If the changes include configuration changes on the Redis for Tanzu Application Service tile or a new stemcell version, the operator must run all post-deploy errands.

Installing another tile

When installing another tile that does not make any changes to the BOSH Director or the VMware Tanzu Application Service for VMs (TAS for VMs), it is not necessary to run any of the Redis for Tanzu Application Service tile’s post-deploy errands.

Changes to other tiles

Sometimes the change does not include changes to the Redis for Tanzu Application Service tile’s configuration. Then it might not be necessary to run all of the Redis for Tanzu Application Service tile’s post-deploy errands.

Broker registrar errand

- Required to run if the CF system domain is changed in the TAS for VMstile.
- Not necessary to run if the change only involves other tiles except TAS for VMstile.

Register on-demand broker errand

- Required to run if the network range that the Redis on-demand broker is deployed in is changed in the BOSH Director tile.
- Not necessary to run if the change only involves other tiles except BOSH Director.

> **Note**
>
> VMware recommends against changing the BOSH Director's network configuration in a way that changes the ranges where the Redis for Tanzu Application Service tile deploys VMs.

Smoke tests and on-demand smoke tests errands

- Required to run if their respective register broker errand is required.
- Required to run both if a newer stemcell minor version is uploaded. The Redis for Tanzu Application Service tile floats to the newest minor version. For more information, see Benefits of floating stemcells.
- Good practice to run both for any change in the BOSH Director or TAS for VMstile.
- Not necessary to run either if the change only involves other tiles except TAS for VMsand BOSH Director.

Upgrade all on-demand service instances errand

- Required to run if a newer stemcell minor version is uploaded. The Redis for Tanzu Application Service tile floats to the newest minor version. For more information, see Benefits floating stemcells.
- Not necessary to run if there are no on-demand instances provisioned.

Recreate all on-demand service instances

- Necessary when an instance must be re-created with different resources, such as when rotating CA certificates.
- Might increase the time that `Apply Changes` takes because it follows the typical instance lifecycle.
- Not necessary to run if there are no on-demand instances provisioned. Recommended to be turned off unless needed.

Find orphan on-demand service instances

- Queries BOSH for any orphaned Redis on-demand instances and then displays them during `Apply Changes`.
- Does not remove any instances. Informs the operator of the details of orphaned instances so the operator can decide when and how to remove them.

Smoke tests

Tanzu Operations Manager runs Redis for Tanzu Application Service smoke tests as a post-install errand. To run the smoke tests errand manually:

1. Retrieve the deployment name of the installed product. To find the deployment name:
    1. From the Tanzu Operations Manager UI, click the Redis for Tanzu Application Service tile.
    2. Copy the part of the URL that starts with “p-redis-”.
2. Run the smoke tests errand:

```
bosh -d REDIS-DEPLOYMENT-NAME run-errand smoke-tests
```

For more information, see Redis for VMware Tanzu Application Service Smoke Tests.

Note

Smoke tests fail unless you enable global default app 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`.

Introduction for Redis operators

This topic for operators introduces you to some best practices for Redis for VMware Tanzu Application Service. It does not provide details about operation.

Best practices

VMware recommends that operators follow these guidelines:

- **Resource allocation**—Work with app developers to anticipate memory requirements and to configure VM sizes. Instances of the Shared-VM service have identical VM sizes. However, with the On-Demand service, app developers can choose from three different plans, each with its own VM size and quota. See the service offering for the On-Demand Service Offering and Resource Usage Planning for On-Demand plans.
- **Logs**—Configure a syslog output. Storing logs in an external service helps operators debug issues both current and historical. See Configure Syslog Output. In particular, set up alerts on critical logs, such as service backups so that you are alerted if a backup fails.
- **Monitoring**—Set up a monitoring dashboard for metrics to track the health of the installation.
- **Backing up data**—When using Redis for persistence, configure automatic backups so that data can be restored in an emergency. Validate the backed-up data with a test restore. See Configuring Automated Backups and also Manually backing up and restoring Redis for Pivotal Cloud Foundry in the VMware Tanzu Support knowledge base.
- **Using**—Instances of the On-Demand service run on dedicated VMs. Apps in production should have an on-demand instance to prevent performance issues caused by sharing an instance. The Shared-VM service shares a VM across many instances. VMware recommends that you only use the Shared-VM service for development and testing, but not in production environments. For more information about the plans, see the On-Demand Service Offering and the Shared-VM Service Offering.

Redis key count and memory size

Redis can handle up to 2^{32} keys, and was tested in practice to handle at least 250 million keys per instance. Every hash, list, set, and sorted set, can hold 2^{32} elements. VM memory is more likely to be a limiting factor than number of keys that can be handled.

Errands

Redis for VMware Tanzu Application Service includes the following errands.

Post-deploy errands

The following post-deploy errands are run by default when **Apply Changes** is triggered. These errands run whether or not there has been a configuration change in the Redis for Tanzu Application Service tile.

Tanzu Operations Manager UI name	BOSH errand name	Description
Broker registrar	`broker-registrar`	Registers the cf-redis-broker with TAS for VMs to offer the `p-redis` service, that is, the shared-VM plan.
Smoke tests	`smoke-tests`	Runs lifecycle tests for shared-VM plans if these have been enabled and there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding, and deprovisioning of service instances.
Register on-demand broker	`register-broker`	Registers the on-demand Redis broker with TAS for VMs to offer the `p.redis` service (on-demand plans).
On-demand smoke tests	`on-demand-broker-smoke-tests`	Runs lifecycle tests for enabled plans of the `p.redis` service if there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding and deprovisioning of service instances.
Upgrade all on-demand service instances	`upgrade-all-service-instances`	Upgrades on-demand service instances to use the latest plan configuration, service releases, and stemcell. This causes downtime to any service instances with available upgrades.

The following post-deploy errands do not run by default when **Apply Changes** is triggered. These errands help operators to troubleshoot and maintain their service fleet.

Tanzu Operations Manager UI name	BOSH errand name	Description
Recreate all on-demand service instances	`recreate-all-service-instances`	Re-creates on-demand service instances one-by-one. This causes downtime for all service instances.
Find orphan on-demand service instances	`orphan-deployments`	Finds all orphan on-demand service instances. The cleanup of orphan on-demands service instances can be carried out manually.

Pre-delete errands

The following pre-delete errands are run by default when the Redis for Tanzu Application Service tile is deleted:

Tanzu Operations Manager UI name	BOSH errand name	Description
Broker deregistrar	`broker-deregistrar`	Deregisters the `cf-redis-broker`.
Delete all on-demand service instances and deregister broker	`delete-all-service-instances-and-deregister-broker`	Deletes all on-demand instances and deregisters the on-demand Redis broker.

Turning off post-deploy errands

VMware recommends that you run the post-deploy errands at any trigger of **Apply Changes**. However, this practice can extend the duration of applying changes by several minutes every time. This section helps you decide when it is safe to skip some post-deploy errands.

Changes to Redis for Tanzu Application Service tile configuration

If the changes include configuration changes on the Redis for Tanzu Application Service tile or a new stemcell version, the operator must run all post-deploy errands.

Installing another tile

When installing another tile that does not make any changes to the BOSH Director or the VMware Tanzu Application Service for VMs (TAS for VMs), it is not necessary to run any of the Redis for Tanzu Application Service tile's post-deploy errands.

Changes to other tiles

Sometimes the change does not include changes to the Redis for Tanzu Application Service tile's configuration. Then it might not be necessary to run all of the Redis for Tanzu Application Service tile's post-deploy errands.

Broker registrar errand

- Required to run if the CF system domain is changed in the TAS for VMstile.
- Not necessary to run if the change only involves other tiles except TAS for VMstile.

Register on-demand broker errand

- Required to run if the network range that the Redis on-demand broker is deployed in is changed in the BOSH Director tile.
- Not necessary to run if the change only involves other tiles except BOSH Director.

> **Note**

> VMware recommends against changing the BOSH Director's network configuration in a way that changes the ranges where the Redis for Tanzu Application Service tile deploys VMs.

Smoke tests and on-demand smoke tests errands

- Required to run if their respective register broker errand is required.
- Required to run both if a newer stemcell minor version is uploaded. The Redis for Tanzu Application Service tile floats to the newest minor version. For more information, see Benefits of floating stemcells.
- Good practice to run both for any change in the BOSH Director or TAS for VMstile.
- Not necessary to run either if the change only involves other tiles except TAS for VMsand BOSH Director.

Upgrade all on-demand service instances errand

- Required to run if a newer stemcell minor version is uploaded. The Redis for Tanzu Application Service tile floats to the newest minor version. For more information, see Benefits floating stemcells.
- Not necessary to run if there are no on-demand instances provisioned.

Recreate all on-demand service instances

- Necessary when an instance must be re-created with different resources, such as when rotating CA certificates.
- Might increase the time that `Apply Changes` takes because it follows the typical instance lifecycle.
- Not necessary to run if there are no on-demand instances provisioned. Recommended to be turned off unless needed.

Find orphan on-demand service instances

- Queries BOSH for any orphaned Redis on-demand instances and then displays them during `Apply Changes`.
- Does not remove any instances. Informs the operator of the details of orphaned instances so the operator can decide when and how to remove them.

Smoke tests

Tanzu Operations Manager runs Redis for Tanzu Application Service smoke tests as a post-install errand. To run the smoke tests errand manually:

1. Retrieve the deployment name of the installed product. To find the deployment name:
    1. From the Tanzu Operations Manager UI, click the Redis for Tanzu Application Service tile.

  2. Copy the part of the URL that starts with “p-redis-”.

2. Run the smoke tests errand:

```
bosh -d REDIS-DEPLOYMENT-NAME run-errand smoke-tests
```

For more information, see Redis for VMware Tanzu Application Service Smoke Tests.

Note

Smoke tests fail unless you enable global default app 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`.

Preparing for TLS with Redis for Tanzu Application Service

This topic gives you an overview of how to prepare for using Transport Layer Security (TLS) with Redis for VMware Tanzu Application Service to secure communication between apps and service instances.

Caution

This procedure involves restarting all of the VMs in your deployment to apply a CA certificate. The operation can take a long time to complete.

When you use TLS, a new port is co-located with Redis for Tanzu Application Service service instances. Apps and clients can use this secure port to establish encrypted connections with the service.

Using BOSH CredHub, Tanzu Operations Manager generates a server certificate using a Certificate Authority (CA) certificate.

If you do not want to use the CA certificate generated, you can provide your own CA certificate and add it through the CredHub CLI. For an overview of the purpose and capabilities of the CredHub component, see CredHub.

Apps and clients use this CA certificate to verify that the server certificate is trustworthy. A trustworthy server certificate allows apps and clients to securely communicate with the Redis for Tanzu Application Service server.

VMware Tanzu Application Service for VMs (TAS for VMs) shares the CA certificate public component:

- TAS for VMs provisions a copy of the CA certificate in the trusted store of each container’s operating system. Apps written in Java and Spring, or C# and Steeltoe, automatically discover the CA certificate in the trusted store. Apps not written in Java and Spring, or C# and Steeltoe, can retrieve the public component of the CA certificate from `VCAP_SERVICES` and use it to establish an encrypted connection with the data service.

Generated or Provided CA Certificate

Tanzu Operations Manager can generate a CA certificate for TLS to use.

Alternatively, you can choose to provide your own CA certificate for TLS to use.

Workflow

The workflow you follow to prepare for TLS depends on whether you use the CA certificate generated by Tanzu Operations Manager or if you bring your own CA certificate.

Using the Generated CA Certificate

To use the CA certificate that Tanzu Operations Manager generates through CredHub, follow this workflow to enable TLS for Redis for VMware Tanzu Application Service:

1. An operator adds the CredHub-generated certificate to Tanzu Operations Manager by performing the procedures:
    1. Find the CredHub Credentials in Tanzu Operations Manager
    2. Add the CA Certificate
2. An operator enables TLS in the tile configuration while installing Redis for Tanzu Application Service. See Enable TLS in Redis for Tanzu Application Service.
3. An app developer edits their app to communicate securely with the Redis for Tanzu Application Service server. See Using TLS.

Providing Your Own CA Certificate

To provide your own CA certificate instead of using the one that Tanzu Operations Manager generates, follow this workflow to enable TLS for Redis for VMware Tanzu Application Service:

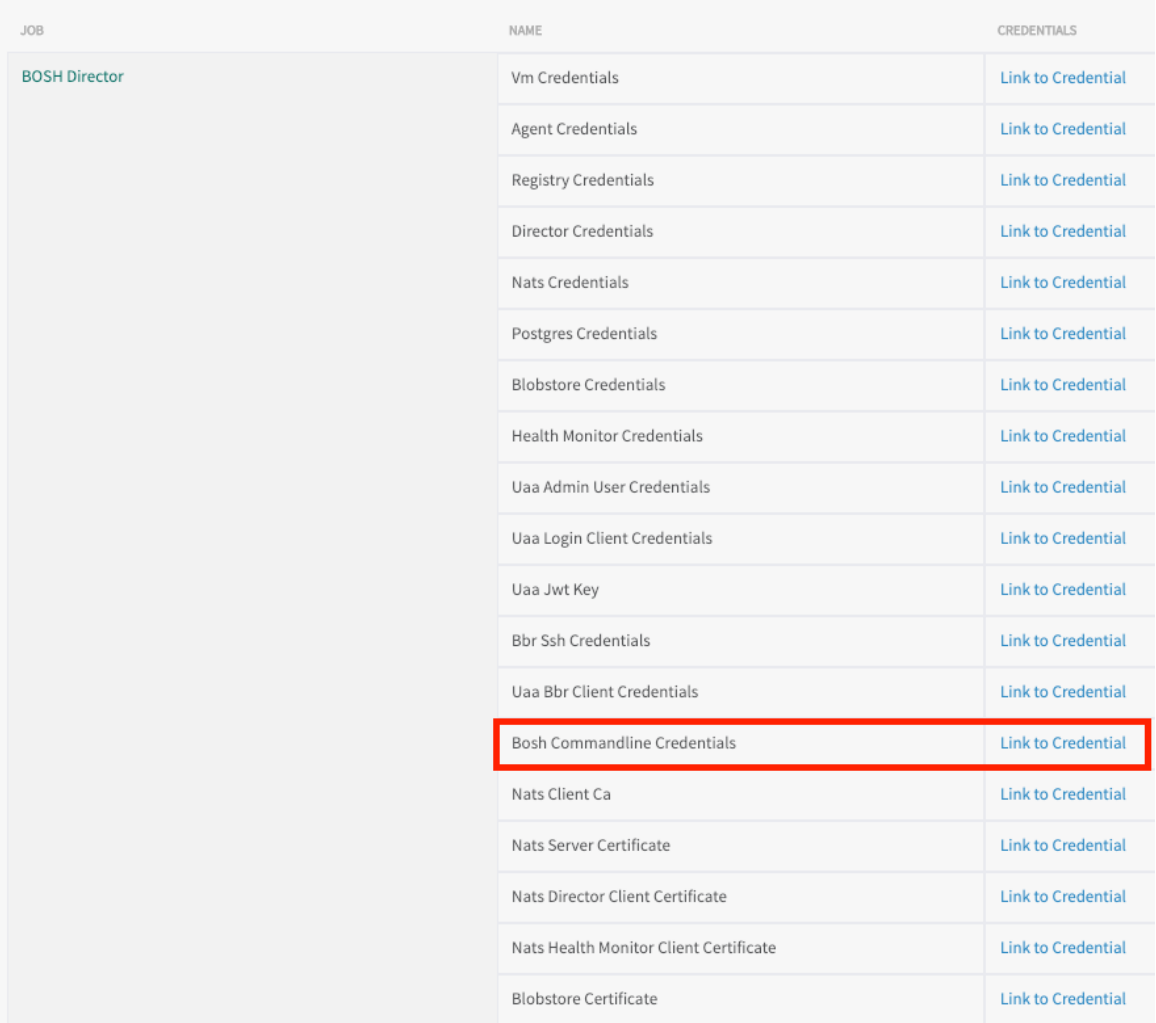
1. An operator provides a CA certificate to CredHub by performing the procedures:
    1. Find the CredHub Credentials in Tanzu Operations Manager.
    2. Set a Custom CA Certificate.
    3. Add the CA Certificate.
2. An operator enables TLS in the tile configuration while installing Redis for Tanzu Application Service. See Enable TLS in Redis for Tanzu Application Service.
3. An app developer edits their app to communicate securely with the Redis for Tanzu Application Service server. See Using TLS.

Find the CredHub Credentials in Tanzu Operations Manager

To find the BOSH CredHub client name and client secret:

1. In the Tanzu Operations Manager Installation Dashboard, click the BOSH Director tile.
2. Click the **Credentials** tab.

3. In the BOSH Director section, click the link to the **BOSH Commandline Credentials**.



JOB	NAME	CREDENTIALS
BOSH Director	Vm Credentials	Link to Credential
	Agent Credentials	Link to Credential
	Registry Credentials	Link to Credential
	Director Credentials	Link to Credential
	Nats Credentials	Link to Credential
	Postgres Credentials	Link to Credential
	Blobstore Credentials	Link to Credential
	Health Monitor Credentials	Link to Credential
	Uaa Admin User Credentials	Link to Credential
	Uaa Login Client Credentials	Link to Credential
	Uaa Jwt Key	Link to Credential
	Bbr Ssh Credentials	Link to Credential
	Uaa Bbr Client Credentials	Link to Credential
	Bosh Commandline Credentials	Link to Credential
	Nats Client Ca	Link to Credential
	Nats Server Certificate	Link to Credential
	Nats Director Client Certificate	Link to Credential
	Nats Health Monitor Client Certificate	Link to Credential
	Blobstore Certificate	Link to Credential

Click here to view a larger version of this image

4. Record the values for `BOSH_CLIENT` and `BOSH_CLIENT_SECRET`.

   Here is an example of the credentials page:

```
{"credential":"BOSH_CLIENT=ops_manager
BOSH_CLIENT_SECRET=abCdE1FgHIjkL2m3n-3PqrsT4EUVwXy5
BOSH_CA_CERT=/var/tempest/workspaces/default/root_ca_certificate
BOSH_ENVIRONMENT=10.0.0.5 bosh "}
```

   The `BOSH_CLIENT` is the BOSH CredHub client name and the `BOSH_CLIENT_SECRET` is the BOSH CredHub client secret.

Set a Custom CA Certificate

Prerequisite: To complete this procedure, you must have the CredHub CLI. For installation instructions, see credhub-cli on GitHub.

Do this procedure if you are providing your own custom CA certificate instead of using the one generated by Tanzu Operations Manager or CredHub.

To add a custom CA Certificate to CredHub:

1. Record the information needed to log in to the BOSH Director VM by following the procedure in Gather Credential and IP Address Information.

2. Log in to the Tanzu Operations Manager VM by following the procedure in Log in to the Tanzu Operations Manager VM with SSH.

3. Set the API target of the CredHub CLI as your CredHub server by running:

   ```
   credhub api  \
   https://BOSH-DIRECTOR-IP:8844 \
   --ca-cert=/var/tempest/workspaces/default/root_ca_certificate
   ```

   Where `BOSH-DIRECTOR-IP` 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
   ```

4. Log in to CredHub by running:

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

   Where:

   - `CREDHUB-CLIENT-NAME` is the value you recorded for `BOSH_CLIENT` in Find the CredHub Credentials in Tanzu Operations Manager.
   - `CREDHUB-CLIENT-SECRET` is the value you recorded for `BOSH_CLIENT_SECRET` in Find the CredHub Credentials in Tanzu Operations Manager.

   For example:

   ```
   $ credhub login \

   --client-name=credhub \

   --client-secret=abcdefghijklm123456789
   ```

5. Use the CredHub CLI to provide a CA certificate. Your deployment can have multiple CA certificates. VMware recommends a dedicated CA certificate for services. Create a new file called `root.pem` with the contents of the certificate. Then, run the following command, specifying the path to `root.pem` and the private key for the certificate. For example:

   ```
   $ credhub set \

   --name="/services/tls_ca" \

   --type="certificate" \
   ```

```
--certificate=./root.pem \

--private=ERKSOSMFF...
```

Add the CA Certificate

Prerequisite: To complete this procedure, you must have the CredHub CLI. For installation instructions, see credhub-cli on GitHub.

To add the CA Certificate to Tanzu Operations Manager:

1. Record the CA certificate by running:

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

2. Go to Tanzu Operations Manager **Installation Dashboard** > **BOSH Director** > **Security**.
3. Append the contents of the CA certificate you recorded in an earlier step into **Trusted Certificates**.
4. Click **Save**.
5. Ensure relevant app security groups are open for port 16379. This can be done through the Cloud Foundry Command Line Interface (cf CLI). For more information, see Managing ASGs with the cf CLI.

Enable TLS in Redis for Tanzu Application Service

To enable TLS in the Redis for Tanzu Application Service tile:

1. Enable TLS by doing one of the following:
   - **If you are configuring TLS for an existing installation:** Follow the procedure in Upgrade Redis for VMware Tanzu Application Service.
   - **If you are configuring TLS for a new installation:** Follow the procedures in Configure On-Demand Service Settings, including enabling TLS in the **On-Demand Service Settings** tab.
2. Navigate to **Tanzu Operations Manager Installation Dashboard** > **Review Pending Changes**.
3. Ensure that the CA certificate is deployed to all VMs by selecting:
   - VMware Tanzu Application Service for VMs
   - Redis for VMware Tanzu Application Service
   - The **Upgrade All On-Demand Service Instances** errand
4. Click **Apply Changes**. This restarts all the VMs in your deployment and applies your CA certificate.

Installing Redis for VMware Tanzu Application Service

This topic for operators provides instructions about how to install Redis for VMware Tanzu Application Service. It covers tasks from downloading the file from Broadcom's Customer Support Portal through verifying the installation after configuration.

Role-based access in Tanzu Operations Manager

Tanzu Operations Manager admins can use Role-Based Access Control (RBAC) to manage which operators can make deployment changes, view credentials, and manage user roles in Tanzu Operations Manager. Therefore, your role permissions might not allow you to follow every procedure in this operator guide.

For more information about roles in Tanzu Operations Manager, see Understand roles in Tanzu Operations Manager.

Download and install the tile

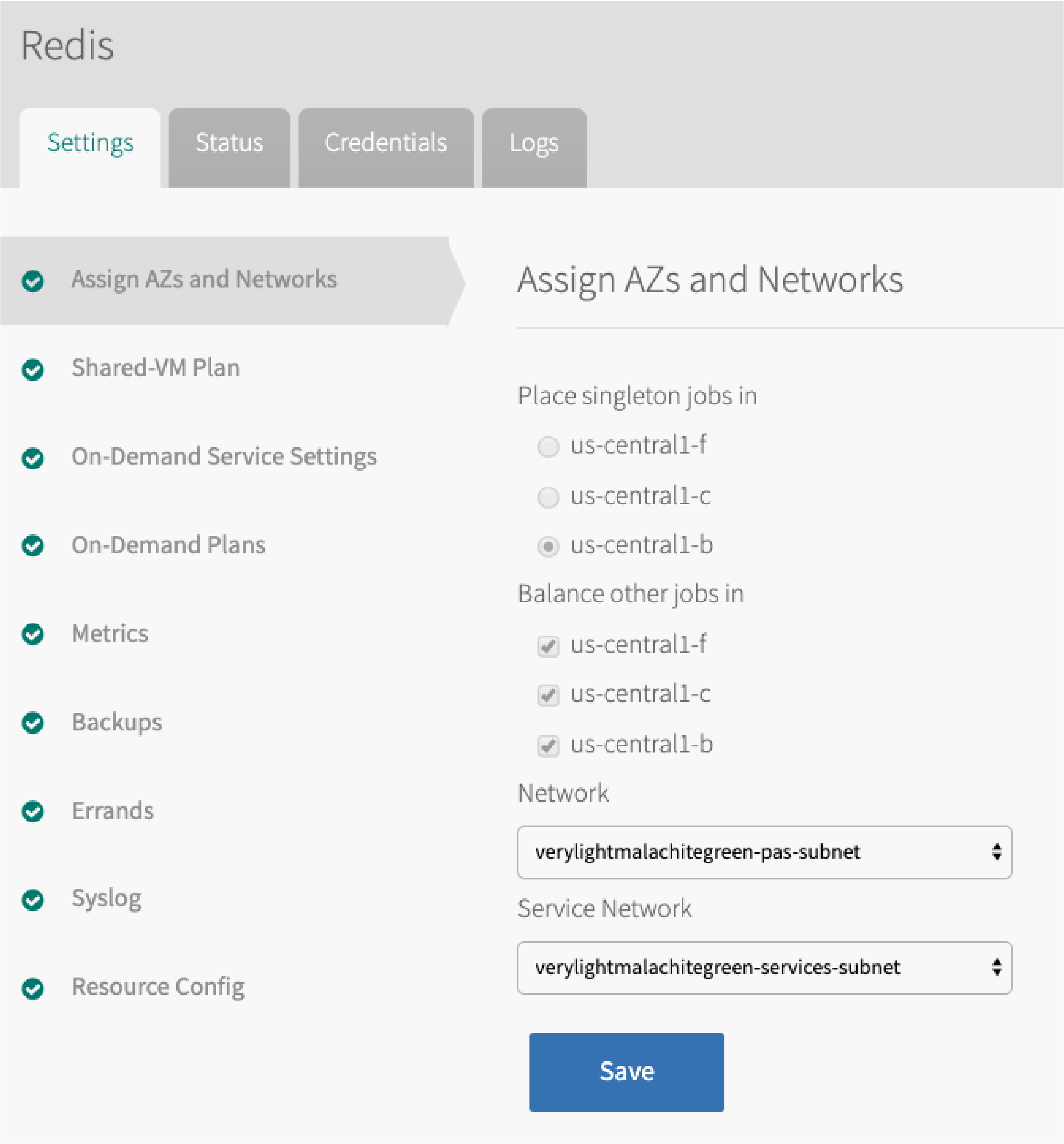
To add Redis for Tanzu Application Service to Tanzu Operations Manager, follow the procedure for adding Tanzu Operations Manager tiles:

1. Download the Redis for Tanzu Application Service file from Broadcom's Customer Support Portal. Select the latest release from the **Releases** dropdown.
2. In the Tanzu Operations Manager Installation Dashboard, click **Import a Product** to upload the Redis for Tanzu Application Service file.
3. Click the **+** sign next to the uploaded product description to add the tile to your staging area.
4. To configure Redis for Tanzu Application Service, click the newly added tile. See configuration instructions in the sections below.
5. After completing the required configuration, in the Tanzu Operations Manager Dashboard, do the following to complete the installation:
    1. If you are using Tanzu Operations Manager v2.3 or later, click **Review Pending Changes**. For more information about this Tanzu Operations Manager page, see Reviewing pending product changes.
    2. Click **Apply Changes**.

For guidance on ports and ranges used in the Redis service, see Select networks below.

Assign AZs and networks

To assign AZs and networks, click the **Assign AZs and Networks** settings tab.



Assign AZs and Networks

Place singleton jobs in

○ us-central1-f

○ us-central1-c

● us-central1-b

Balance other jobs in

☑ us-central1-f

☑ us-central1-c

☑ us-central1-b

Network

verylightmalachitegreen-pas-subnet

Service Network

verylightmalachitegreen-services-subnet

Assign AZs

You can assign multiple availability zones (AZs) to Redis jobs, however, this does not ensure high availability. You must select AZs that are in the service network you configured in your BOSH Director. For more information, see Availability Zones.

To assign AZs:

1. Select **Assign AZs and Networks**.
2. Under **Place singleton jobs in**, select an AZ for the on-demand or shared service broker VM, and any shared instances.
3. Under **Balance other jobs in**, select the AZs that you want the broker to balance on-demand service instances across.

4. Click **Save**.

Select networks

You can use Redis for Tanzu Application Service with or without using the on-demand service. To use the Redis for Tanzu Application Service on-demand service, you must select a network in which the service instances are created. For more information, see Networking for On-Demand Services.

To select networks:

1. In the **Assign AZs and Networks** tab, select a **Network**.
   - VMware recommends that each type of service run in its own network.
   - Typically the service broker network and service instance networks are the same.
2. If using the on-demand service, select a **Service Network**. Otherwise, select an empty service network. For more information, see Creating an empty Services Network when using on-demand Service Tiles for Non-On-Demand usage only in the VMware Tanzu Support knowledge base.

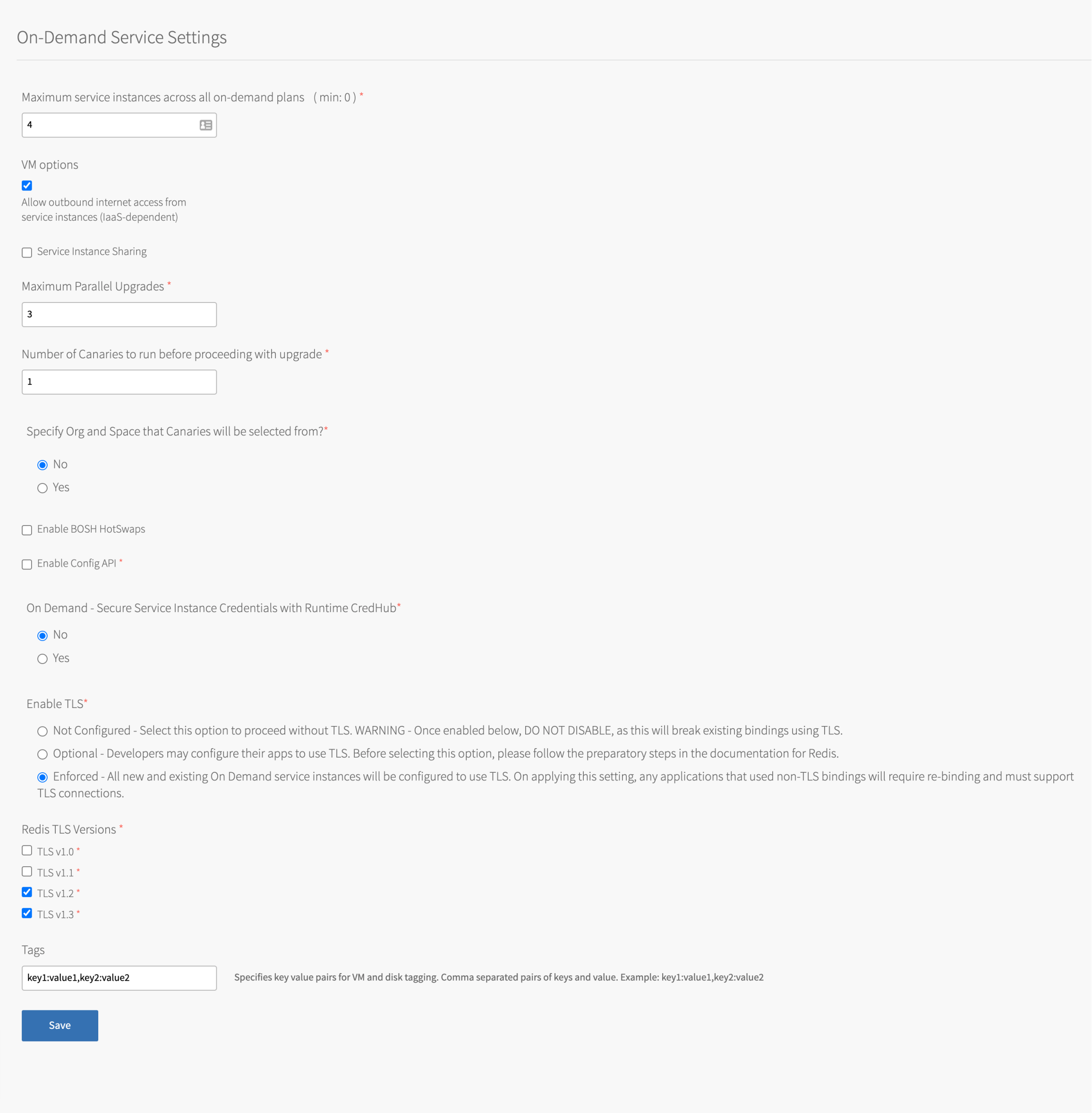
The following ports and ranges are used in Redis for Tanzu Application Service:

Port	Protocol	Direction and network	Purpose
8202	TCP	Inbound to the Cloud Foundry network Outbound from the service broker and service instance networks	Allows Redis `metron_agent` to forward metrics to the Cloud Foundry Loggregator
12350	TCP	Outbound from Cloud Foundry to the `cf-redis-broker` service broker network	Allows the cloud controllers to access the `cf-redis-broker`
8080	TCP	Outbound from Cloud Foundry to the on-demand service broker network	Allows the cloud controllers to access the on-demand service broker when using an on-demand service
6379	TCP	Outbound from Cloud Foundry to any on-demand service instance networks	Allows the Diego Cell and Diego Brain networks to access all on-demand service instances
16379	TCP	Outbound from Cloud Foundry to any on-demand service instance networks	This port allow the Diego Cell and Diego Brain networks to access all on-demand service instances. This access is only required if TLS is set to **optional**.
32768–61000	TCP	Outbound from Cloud Foundry to the `cf-redis-broker` service broker network	These ports allow Diego Cell and Diego Brain networks to access the service broker VM. This access is only required for the shared service plan.
80	http	Outbound from any service instance networks	Gives access to the backup blobstore when using service backups
443	https	Outbound from any service instance networks	Gives access to the backup blobstore when using service backups
8443 and 25555	TCP	Outbound from any on-demand service broker network to the BOSH Director network	Allows the on-demand service broker to communicate with the BOSH Director

Configure on-demand service settings

To configure settings that apply across the whole on-demand service offering:

1. In the Redis for Tanzu Application Service tile, select **On-Demand Service Settings**.



On-Demand Service Settings

Maximum service instances across all on-demand plans (min: 0) *

4

VM options

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

□ Service Instance Sharing

Maximum Parallel Upgrades *

3

Number of Canaries to run before proceeding with upgrade *

1

Specify Org and Space that Canaries will be selected from?*

● No

○ Yes

□ Enable BOSH HotSwaps

□ Enable Config API *

On Demand - Secure Service Instance Credentials with Runtime CredHub*

● No

○ Yes

Enable TLS*

○ Not Configured - Select this option to proceed without TLS. WARNING - Once enabled below, DO NOT DISABLE, as this will break existing bindings using TLS.

○ Optional - Developers may configure their apps to use TLS. Before selecting this option, please follow the preparatory steps in the documentation for Redis.

● Enforced - All new and existing On Demand service instances will be configured to use TLS. On applying this setting, any applications that used non-TLS bindings will require re-binding and must support TLS connections.

Redis TLS Versions *

□ TLS v1.0 *

□ TLS v1.1 *

☑ TLS v1.2 *

☑ TLS v1.3 *

Tags

key1:value1,key2:value2

Specifies key value pairs for VM and disk tagging. Comma separated pairs of keys and value. Example: key1:value1,key2:value2

Click here to view a larger version of this image

2. Enter the **Maximum service instances across all on-demand plans**. The maximum number of instances you set for all your on-demand plans combined cannot exceed this number.
   For more information, see Setting Limits for On-Demand Service Instances.

3. Select the **Allow outbound internet access from service instances** checkbox. You must select this checkbox to allow external log forwarding, send backup artifacts to external destinations, and communicate with an external BOSH blobstore. Outbound network traffic rules also depend on your IaaS settings. Consult your network or IaaS admin to ensure that your IaaS allows outbound traffic to the external networks you need.

4. (Optional) Select the checkbox to enable **Service Instance Sharing**. Turning on sharing enables this feature for all on-demand instances. To enable this feature a user with admin

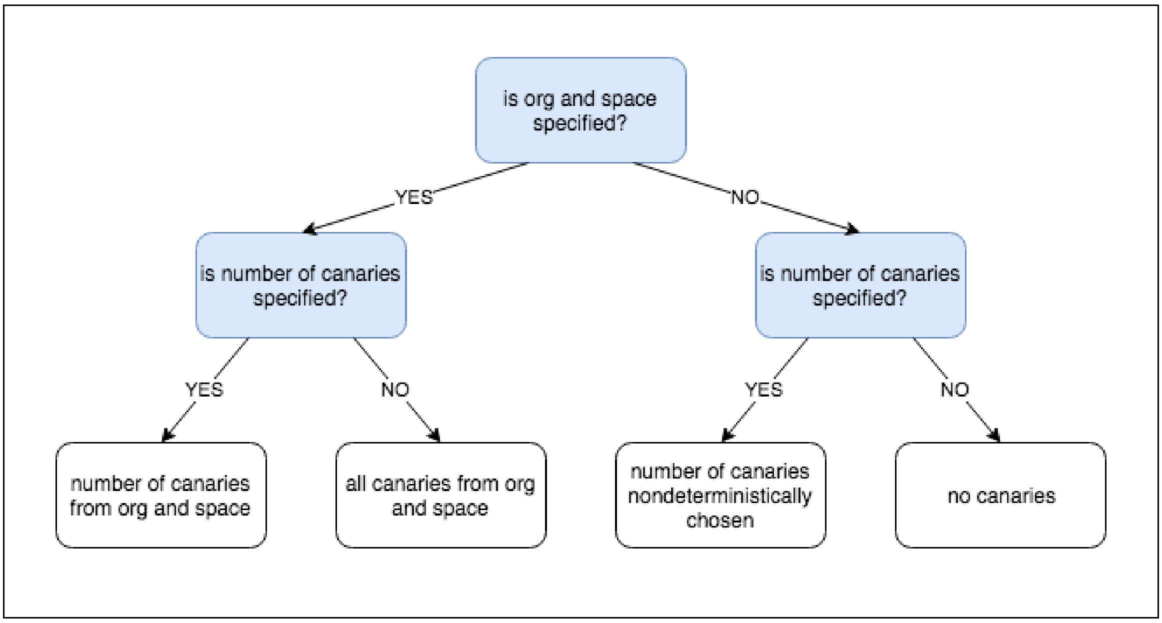
privileges must run `cf enable-feature-flag service_instance_sharing`. For information about this feature, see Sharing a Redis Instance with Another Space.

5. (Optional) Use the **Maximum Parallel Upgrades** field to configure the maximum number of Redis service instances that can be upgraded at the same time.

   When you click **Apply Changes**, the on-demand broker upgrades all service instances. By default, each instance is upgraded serially. Allowing parallel upgrades reduces the time taken to apply changes. Multiple Redis service instances are unavailable during the upgrade.

6. (Optional) Use the **Number of Canaries to run before proceeding with upgrade** field and the **Specify Org and Space that Canaries will be selected from?** options to specify settings for upgrade canaries. Canaries are service instances that are upgraded first. The upgrade fails if any canaries fail to upgrade.

   You can limit canaries by number and by org and space. To use all service instances in an org and space as canaries, set the number of canaries to zero. This upgrades all service instances in the selected org and space first.

   
   is org and space specified?
YES
NO
is number of canaries specified?
is number of canaries specified?
YES
NO
YES
NO
number of canaries from org and space
all canaries from org and space
number of canaries nondeterministically chosen
no canaries

   The flowchart above has the following information:

   - Is the org and space specified?
     - **Yes**: Is the number of canaries specified?
       - **Yes**: The number of canaries is limited by org and space.
       - **No**: All canaries from the org and space.
     - **No**: Is the number of canaries specified?
       - **Yes**: The number of canaries is chosen non-deterministically.
       - **No**: No canaries.

   Note

> If you specify that canaries should be limited to an org and space that has no service instances, the upgrade fails. Also, Canary upgrades comply with the Maximum Parallel Upgrades settings. If you specify three canaries and a Maximum Parallel Upgrades of two, then two canaries upgrade, followed by the third.

For information about this feature, see `canaries` in Upgrade all Service Instances.

7. (Optional) Select the check box to enable **BOSH HotSwaps**. This reduces downtime during upgrades. For how this feature works, see Changing VM Update Strategy in the BOSH documentation.
8. (Optional) Select **Yes** to enable **On Demand - Secure Service Instance Credentials with Runtime CredHub**. If you do select **Yes**, you must also follow the steps in Enable Secure Service Instance Credentials for On-Demand Redis later on this page.
9. (Optional) Select the **Not Configured** option under **Enable TLS** if you do not want to allow TLS connections to on-demand service instances. TLS support is optional in new installations by default. If TLS is configured in Redis for Tanzu Application Service v2.2, follow the procedures in Preparing for TLS before enabling TLS.

> ⚠ **Caution**
>
> After TLS is activated for the on-demand Redis service, deactivating TLS causes downtime and service outage for all apps that connect to Redis through TLS. If you deactivate TLS, you must unbind all apps bound to on-demand instances from the TLS port, rebind to the non-TLS port, and then restage to resume service access.

10. (Optional) If you selected the **Optional** option under **Enable TLS**, select the check box next to each TLS version you want to support.

> **Note**
>
> Selecting **TLS optional** does not enforce the use of TLS. After deploying the tile, notify developers that they must unbind, bind, and restage existing service instances to ensure Spring and Steeltoe apps use TLS. Further configuration might be needed for other frameworks and languages to ensure use of the TLS port.

11. (Optional) If you selected the **Enforced** option under **Enable TLS**, enable the checkbox next to each TLS version you want to support. The **Enforced** option requires TLS to be enabled.

> **Breaking change**
>
> If TLS is set to **Enforced** then all existing service instances use TLS after changes from the **Upgrade All Service Instances** errand are applied. Any

> apps not using TLS are no longer able to communicate with their service instances. Such apps require a new binding and must be configured to communicate with their Redis for Tanzu Application Service service instance through TLS.

12. (Optional) If you selected the **Optional** or **Enforced** option under **Enable TLS** then select the TLS versions to support. TLS v1.3 and TLS v1.2 are enabled by default. VMware recommends supporting TLS v1.1 and later. VMware does not recommend supporting TLS v1.0 because it is less secure than later versions, but it is an option for apps that only support this protocol. After selecting a TLS version, VMware recommends generating a new service key and then rebinding the service instance with the new service key. This makes the service key's `tls_versions` field reflect the new TLS version, which can help developers who use the service key to see the supported TLS version. To create a new service key, follow the steps in Check Availability. To rebind the instance, follow the steps in Bind Existing Apps with TLS.

13. (Optional) To add an endpoint to service instances for developers to query Redis configuration parameters, select the **Enable Config API** checkbox. For more information, see Using the Config API.

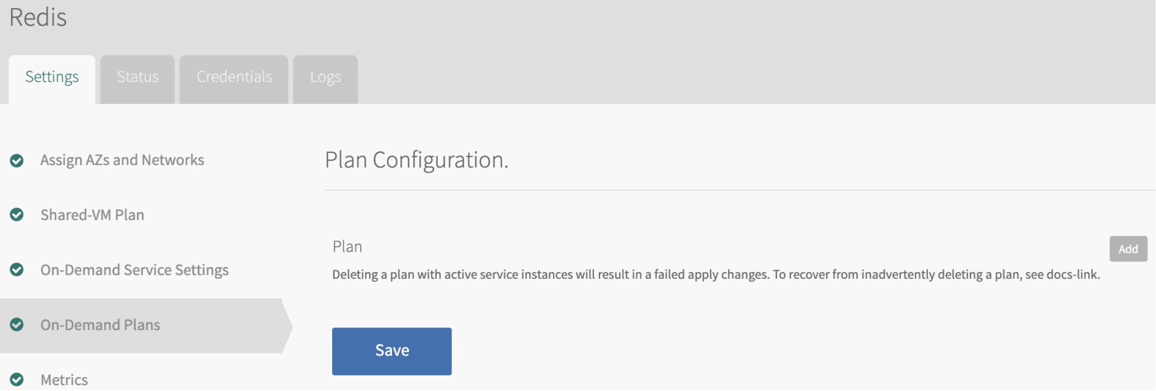
14. (Optional) In the **Tags** field, write a comma-separated list of key-value pairs for tagging service-instance VMs. Ensure the list is in a style that the underlying cloud provider accepts. For example, Google Cloud Platform (GCP) does not permit uppercase characters.

Configure on-demand plan settings

You can configure multiple on-demand plans with memory and disk sizes suited to different use cases. The configuration of resources varies depending on your IaaS.

To add and configure each on-demand service plan:

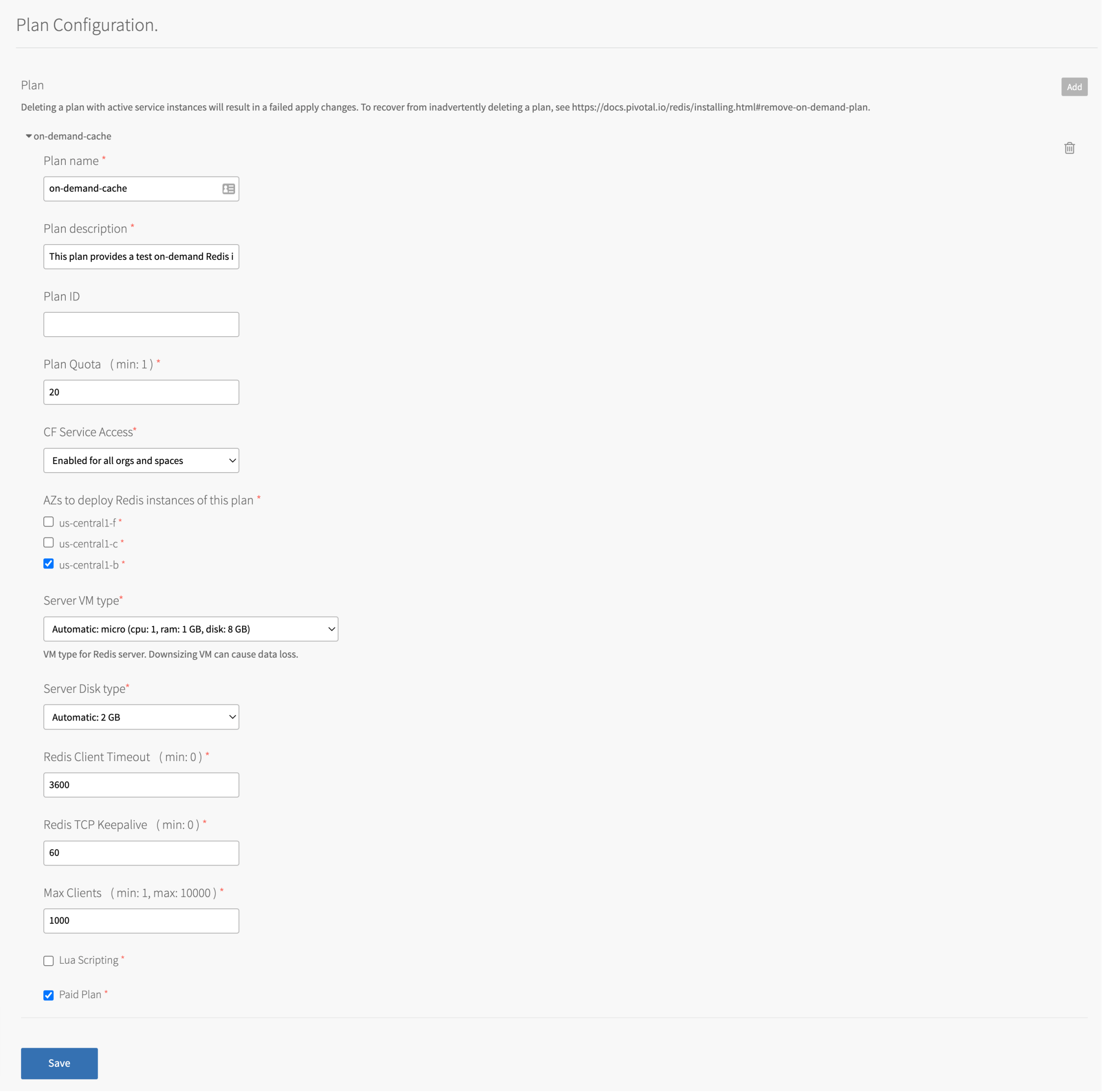
1. In the Redis for Tanzu Application Service tile, select **On-Demand Plans**.



Deleting a plan with active service instances will result in a failed apply changes. To recover from inadvertently deleting a plan, see docs-link.

Click here to view a larger version of this image

2. Click **Add** to add an on-demand plan.



Plan Configuration.

Plan

Deleting a plan with active service instances will result in a failed apply changes. To recover from inadvertently deleting a plan, see https://docs.pivotal.io/redis/installing.html#remove-on-demand-plan.

▾ on-demand-cache

Plan name *
on-demand-cache

Plan description *
This plan provides a test on-demand Redis i

Plan ID

Plan Quota (min: 1) *
20

CF Service Access*
Enabled for all orgs and spaces

AZs to deploy Redis instances of this plan *
□ us-central1-f *
□ us-central1-c *
☑ us-central1-b *

Server VM type*
Automatic: micro (cpu: 1, ram: 1 GB, disk: 8 GB)
VM type for Redis server. Downsizing VM can cause data loss.

Server Disk type*
Automatic: 2 GB

Redis Client Timeout (min: 0) *
3600

Redis TCP Keepalive (min: 0) *
60

Max Clients (min: 1, max: 10000) *
1000

□ Lua Scripting *

☑ Paid Plan *

Click here to view a larger version of this image

3. Configure the settings in the following table for your on-demand plans and then click **Save**.

> **Caution**
>
> Do not downsize the VMs or disk size. Doing so can cause data loss in pre-existing instances.

Field	Default	Description
Plan name	on-demand-cache	The name that you choose for the plan. This is displayed in the Marketplace. VMware recommends that you give your plans descriptive names based on their configuration.
Plan Description	This plan provides an on-demand Redis instance, tailored for caching use cases with persistence to disk enabled.	The description that you write for your plan. This is displayed in the Marketplace. Include details that are relevant to app developers.

Field	Default	Description
Plan ID	Empty	An ID that you configure when recovering deleted plans. Leave this field blank unless it is already configured or you are recovering a deleted plan.
Plan Quota	20	The maximum number of instances of this plan that app developers can create. For more information, see Setting Limits for On-Demand Service Instances.
CF Service Access	Enabled for all orgs and spaces	This setting does not modify the permissions that have been previously set, and allows for manual access to be configured from the CLI.
AZ to deploy Redis instances of this plan	None selected	The AZs in which to deploy the Redis instances from the plan. These must be AZs of the service network, which are configured in the BOSH Director tile. If you select multiple AZs, instances are distributed randomly between them.
Server VM type	Varies depending on IaaS	VMware recommends that the persistent disk is at least 2.5x the VM memory for on-demand service instances.
Server Disk type	Varies depending on IaaS	VMware recommends that the persistent disk is at least 2.5x the VM memory for on-demand service instances.
Redis Client Timeout	3600	The server timeout for an idle client specified in seconds. Adjust this setting as needed.
Redis TCP Keepalive	60	The interval in seconds at which TCP ACKs are sent to clients. Adjust this setting as needed.
Max Clients	1000	The maximum number of clients that can be connected at any one time. Adjust this setting as needed.
Lua Scripting	Deactivated	VMware recommends keeping Lua scripting deactivated unless developers are running apps that require Lua scripting, such as .Net Steeltoe apps. Verify that your apps are using a language that does not require Lua scripting.
Paid Plan	Deactivated	Select this check box to indicate that this service plan is paid. The plan is marked with an asterisk in the `cf marketplace` list and labeled "paid" in the "free or paid" column when individual plans are listed.

Enable secure service instance credentials for on-demand Redis

If you enabled **On Demand - Secure Service Instance Credentials with Runtime CredHub** in step 8 of Configure On-Demand Service Settings above, you must follow this procedure.

To secure your on-demand binding credentials in runtime CredHub instead of the Cloud Controller database (CCDB):

1. On the **CredHub** pane of VMware Tanzu Application Service for VMs (TAS for VMs) select **Secure service instance credentials**.

   For instructions, see Configure CredHub in *Configuring TAS for VMs*.

2. After deploying the tile, notify developers that they must unbind and rebind existing service instances to secure their credentials with CredHub.

Updating on-demand service plans

Operators can update certain settings after the plans have been created. If the operator updates the VM size, disk size, or the Redis configuration settings (enabling Lua Scripting, max-clients, timeout and TCP keepalive), these settings are implemented in all instances that are already created.

Operators should not downsize the VMs or disk size because this can cause data loss in pre-existing instances. Additionally, operators cannot make a plan that was previously active, inactive, until all instances of that plan have been deleted.

Remove an on-demand service plan

> ⚠ **Caution**
>
> Do not remove an on-demand service plan with service instances deployed. Doing so causes **Apply Changes** to fail. For how to recover a deleted plan, see Recover a deleted Redis for Tanzu Application Service On-Demand Plan in the VMware Tanzu Support knowledge base.

To remove an on-demand service plan from your tile:

1. Ensure that there are no deployed service instances of the plan by running:

   ```
   cf services
   ```

   For example:

   ```
   $ cf services
   Getting services in org my-org / space my-space as user@example.com...
   OK
   name           service     plan               bound apps    last operatio
   n
   my-instance    p.redis     on-demand-cache                  create succee
   ded
   ```

2. In the Redis for Tanzu Application Service tile, select **On-Demand Plan Settings**.
3. Delete the plan by clicking the trash can icon next to the plan name.
4. Click **Save**.
5. Go to the **Errands** page on the Redis for Tanzu Application Service tile, and set the **Register On-Demand Broker** errand to **on**. This updates the plans available in the Marketplace.

Remove all on-demand service plans

> ⚠ **Caution**
>
> Do not remove an on-demand service plan with service instances deployed. Doing so causes **Apply Changes** to fail. For how to recover a deleted plan, see Recover a

deleted Redis for Tanzu Application Service on-demand plan in the VMware Tanzu Support knowledge base.

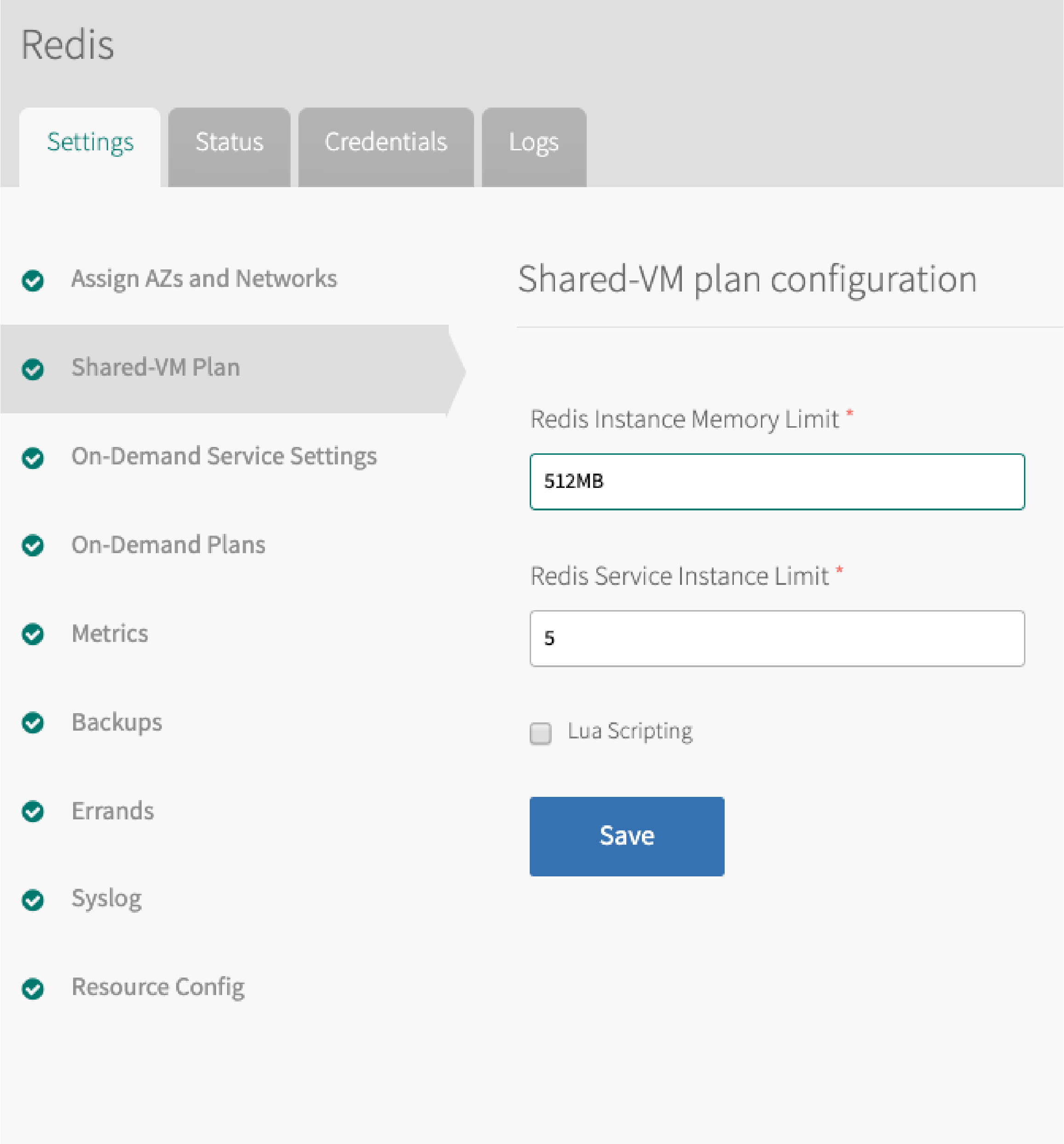
To remove the on-demand service from your tile:

1. In the Redis for Tanzu Application Service tile, select **Resource Config**
2. Set the **Redis On-Demand Broker** job instances to 0.
3. Go to the **Errands** page on the Redis for Tanzu Application Service tile, and set the following errands to **off**:
   - Register On-Demand Broker
   - On-Demand Broker Smoke Tests
   - Upgrade All On-Demand Service Instances
   - Delete All Service Instances and Deregister On-Demand Broker
4. Create an empty service network. For instructions, see Creating an Empty Services Network when using on-demand Service Tiles for Non-On-Demand Usage Only in the VMware Tanzu Support knowledge base.
5. Go to each **On-Demand Plans** page on the Redis for Tanzu Application Service tile, and delete each plan by clicking the trash can icon next to the plan name.

Configure shared-VM plan settings

To configure shared-VM service plans:

1. In the Redis for Tanzu Application Service tile, select **Shared-VM Plan**.



Redis Instance Memory Limit *: 512MB

Redis Service Instance Limit *: 5

□ Lua Scripting

2. Configure these fields:
   - **Redis Instance Memory Limit**—Enter the maximum memory used by a shared-VM instance, for example 512 MB.
   - **Redis Service Instance Limit**—Enter the maximum number of shared-VM instances.
   - **Lua Scripting**—Activate or deactivate Lua Scripting as needed using this checkbox. VMware recommends that Lua Scripting is deactivated unless developers need it to be enabled.

   Memory and instance limits depend on the total system memory of your Redis broker VM and require some additional calculation. For more information, see Memory Limits for Shared-VM Plans below.

3. Click **Save**.

4. If you do not want to use the on-demand service, you must make all of the on-demand service plans inactive. Click the tab for each on-demand plan, and select **Plan Inactive**. See the example in Step 4 of Remove on-demand service plans above.

5. To change the allocation of resources for the Redis broker, click the **Resource Config** tab.

   The Redis broker server runs all of the Redis instances for your shared-VM plan. From the **Resource Config** page, you can change the CPU, RAM, Ephemeral Disk, and Persistent Disk made available, as needed.

Configure memory limits for shared-VM plans

Additional calculation is required to configure memory limits for shared-VM plans. With these plans, several service instances share the VM, and the Redis broker also runs on this same VM. Therefore, the memory used by all the shared-vm instances combined should be at most 45% of the memory of the Redis broker VM.

To configure the limits in these fields:

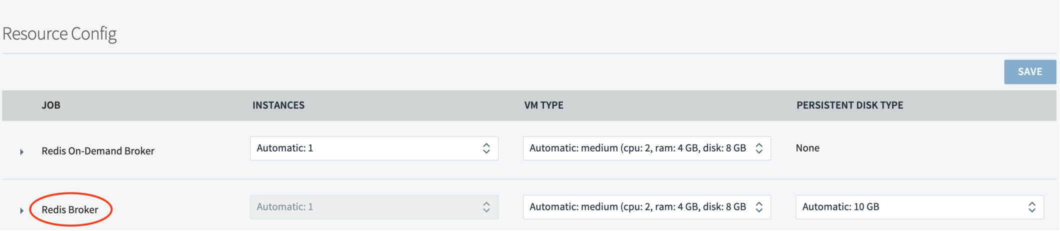
1. Estimate the combined maximum memory that all your Redis shared-VM instances can use.

2. If your estimate is higher than 45% of the Redis broker VM's total system memory, do any of the following:

   - Decrease the **Redis Instance Memory Limit** in the **Shared-VM Plan** tab.

   - Decrease the number of instances in **Redis Service Instance Limit** in the **Shared-VM Plan** tab.

   - Increase the RAM for the Redis Broker in the **Resource Config** tab as shown below.



JOB	INSTANCES	VM TYPE	PERSISTENT DISK TYPE
Redis On-Demand Broker	Automatic: 1	Automatic: medium (cpu: 2, ram: 4 GB, disk: 8 GB	None
Redis Broker	Automatic: 1	Automatic: medium (cpu: 2, ram: 4 GB, disk: 8 GB	Automatic: 10 GB

Click here to view a larger version of this image

Here are some examples for setting these limits:

Redis Broker VM total memory	Redis instance memory limit	Redis service instance limit
16 GB	512 MB	14
16 GB	256 MB	28
64 GB	512 MB	56

Note

You can configure a larger **Redis Service Instance Limit** if you are confident that the majority of the deployed instances do not use a large amount of their allocated memory, for example, in development or test environments.

> However, this practice is not supported and can cause your server to run out of memory, preventing users from writing any more data to any Redis shared-VM instance. Do not use shared-VM instances in production environments.

Configure resources for shared-VM plans

To configure resources for the shared-VM plans, click the **Resource Config** settings tab on the Redis for Tanzu Application Service tile. The shared-VM plan is on the **Redis Broker** resource.

The following are the default resource and IP requirements for Redis for Tanzu Application Service when using the shared-VM plans:

Product	Resource	Instances	CPU	Ram	Ephemeral	Persistent	Static IP	Dynamic IP
Redis	Redis Broker	1	2	3072	4096	9216	1	0
Redis	Broker registrar	1	1	1024	2048	0	0	1
Redis	Broker de-registrar	1	1	1024	2048	0	0	1
Redis	Compilation	2	2	1024	4096	0	0	1

VMware recommends that the persistent disk is at least 3.5x the VM memory for the shared-VM.

Deactivate shared VM plans

You can deactivate shared-VM plans by doing the following while configuring the Redis tile:

1. Ensure at least one on-demand plan is active.
2. Click the **Shared-VM** tab.
3. Set **Redis Service Instance Limit** to 0.
4. Click **Save**.
5. Click the **Errands** tab and configure the settings as follows:
    1. Set **Broker Registrar** to Off.
    2. Set **Smoke Tests** to Off.
    3. Set **Broker Deregistrar** to Off.
    4. Leave all four on-demand errands On.
6. Click **Save**.
7. Click the **Resource Config** tab.
8. For **VM Type** and **Persistent Disk Type**, leave the configurations as they are or increase the sizes. It is not possible to decrease the sizes.
9. Click **Save** if you changed the sizes.

Configure syslog forwarding

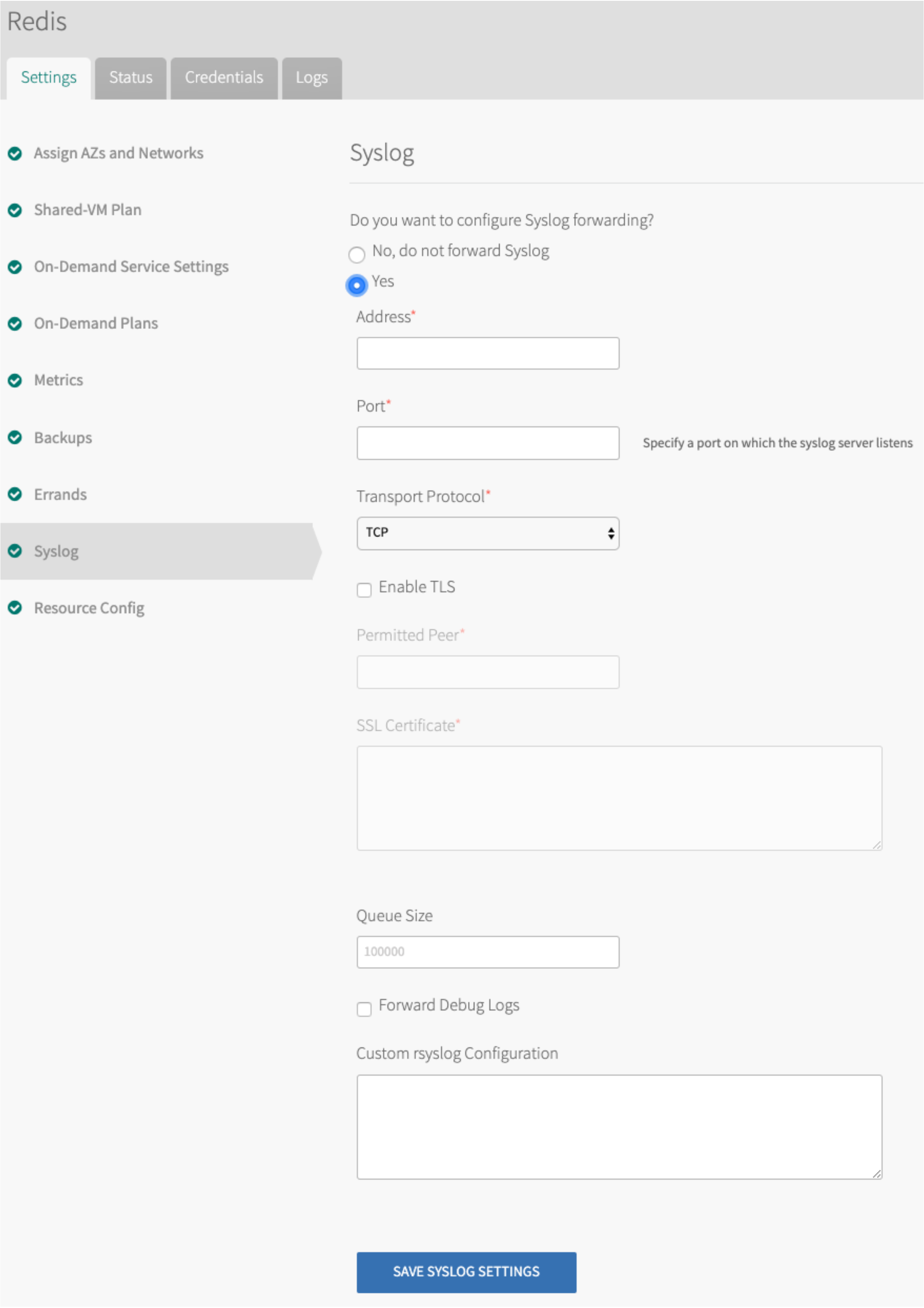
VMware recommends that operators configure syslog forwarding to a remote destination. Forwarding your system logs to a remote destination lets you:

- View logs from every VM in the Redis for Tanzu Application Service deployment in one place.
- Effectively troubleshooting when logs are lost on the source VM.
- Set up alerts for important error logs to monitor the deployment.

All logs follow RFC5424 format.

To configure syslog forwarding:

1. Click **Syslog**.



Syslog

Do you want to configure Syslog forwarding?

○ No, do not forward Syslog

● Yes

Address*

Port*

Specify a port on which the syslog server listens

Transport Protocol*

TCP

□ Enable TLS

Permitted Peer*

SSL Certificate*

Queue Size

100000

□ Forward Debug Logs

Custom rsyslog Configuration

2. (Optional) Select **Yes** to send Redis for Tanzu Application Service system logs to a remote server.

3. Enter the IP address or DNS name for the remote server in **Address**.

4. Enter the port number that the remote server listens on in **Port**.

5. Select **TCP** or **UDP** from the **Transport Protocol** drop-down menu to specify the transport protocol to use to send the logs to the remote server.
6. (Optional) Select the **Enable TLS** check box to send encrypted logs to remote server with TLS. After you select the check box:
    1. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
    2. Enter the SSL certificate for the remote server in **SSL Certificate**.

> **Important**
>
> VMware recommends that you enable TLS encryption when you are forwarding logs. Logs can contain sensitive information, such as cloud provider credentials.

7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log entries held in the buffer. The default value is 100,000.
8. (Optional) Select the check box to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you might generate a large amount of log data.
9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** text box. This text box requires the rainerscript syntax.
10. Click **Save**.

Verify the stemcell

To verify that you have the correct stemcell, follow the procedure in Importing and managing Stemcells.

Apply changes from your configuration

To apply your configuration changes:

1. Return to the Tanzu Operations Manager Installation Dashboard.
2. In the Tanzu Operations Manager Dashboard, do the following to complete the installation:
    1. If you are using Tanzu Operations Manager v2.3 or later, click **Review Pending Changes**. For more information about this Tanzu Operations Manager page, see Reviewing pending product changes.
    2. Click **Apply Changes**.

Create app security groups

To allow this service to have network access, you must create App Security Groups (ASGs). Ensure your security group allows access to the Redis Service Broker VM configured in your deployment. You can obtain the IP addresses for these VMs in Tanzu Operations Manager under the **Resource Config** section for the Redis for Tanzu Application Service tile.

Note

Without ASGs, this service is unusable.

App container network connections

App containers that use instances of Redis for Tanzu Application Service require the following outbound network connections:

Destination	Ports	Protocol	Reason
`ASSIGNED_NETWORK`	32768-61000	TCP	To enable apps to access shared-VM service instances
`ASSIGNED_NETWORK`	6379	TCP	To enable apps to access on-demand service instances
`ASSIGNED_NETWORK`	16379	TCP	To enable apps to have TLS encrypted access to on-demand service instances

Create an ASG called `redis-app-containers` with the above configuration and bind it to the appropriate space or, to give all started apps access, bind to the `default-running` ASG set and restart your apps. Example:

```
[
  {
    "protocol": "tcp",
    "destination": "ASSIGNED_NETWORK",
    "ports": "6379,16379"
  }
]
```

Validating the installation

Smoke tests run as part of Redis for Tanzu Application Service installation to verify that the installation succeeded. For more information, see Redis for VMware Tanzu Application Service Smoke Tests.

Uninstall Redis for Tanzu Application Service

To uninstall Redis for Tanzu Application Service:

1. In the Tanzu Operations Manager Installation dashboard, click the trash can icon in the lower-right corner of the Redis for Tanzu Application Service tile.
2. Confirm the product was deleted.
3. If you are using Tanzu Operations Manager v2.3 or later, click **Review Pending Changes**. For more information about this Tanzu Operations Manager page, see Reviewing pending product changes.
4. Click **Apply Changes**.

Upgrading Redis for VMware Tanzu Application Service

This topic gives you information about the upgrade paths and how to upgrade Redis for VMware Tanzu Application Service.

Compatible upgrade paths

For product versions and upgrade paths, see Upgrade Planner.

Upgrade Redis for Tanzu Application Service

> ⚠ **Caution**
>
> After TLS is activated for the on-demand Redis service, deactivating TLS causes downtime and service outage for all apps that connect to Redis through TLS. If you deactivate TLS, you must unbind all apps bound to on-demand instances from the TLS port, rebind to the non-TLS port, and then restage to resume service access.

This product enables a reliable upgrade experience between versions of the product deployed through Tanzu Operations Manager.

For information about the upgrade paths for each released version, see Compatible upgrade paths.

Upgrade procedure

To upgrade to the latest version of Redis for Tanzu Application Service:

1. Download the latest version of the product from Broadcom's Customer Support Portal.
2. Upload the new `.pivotal` file to Tanzu Operations Manager.
3. If required, upload the stemcell associated with the update.
4. If required, update any new mandatory configuration parameters.
5. (Optional) To enable TLS:
    1. Follow the procedures in Preparing for TLS.

        > **Note**
        >
        > In most cases, enabling TLS does not noticeably reduce performance. Performance impact depends on the health of resources, such as network infrastructure and application architecture.

    2. In the Redis for Tanzu Application Service tile, select **On-Demand Service Settings**.
    3. Under **Enable TLS**, select **Optional**.
    4. Enable the checkbox next to each TLS version you want to support. VMware recommends supporting TLS v1.1 and later. VMware does not recommend

supporting TLS v1.0 because it is less secure than later versions, but it is an option for apps that only support this protocol.

> **Note**
>
> After selecting a TLS version, VMware recommends generating a new service key and then rebinding the service instance with the new service key. This makes the service key's `tls_versions` field reflect the new TLS version, which can help developers who use the service key to see the supported TLS version. To create a new service key, follow the steps in Check Availability. To rebind the instance, follow the steps in Bind Existing Apps with TLS.

    5. Click **Save**.

6. (Optional) Enable developers to upgrade service instances individually. For instructions, see Enable Individual Service Instance Upgrades below.
   When this is feature is not enabled, the `upgrade-all-service-instances` errand runs by default after each upgrade. For more information, see Upgrading all Service Instances.
7. Go to the Tanzu Operations Manager **Installation Dashboard**. Click **Review Pending Changes** and **Apply Changes**.

Enable individual service instance upgrades

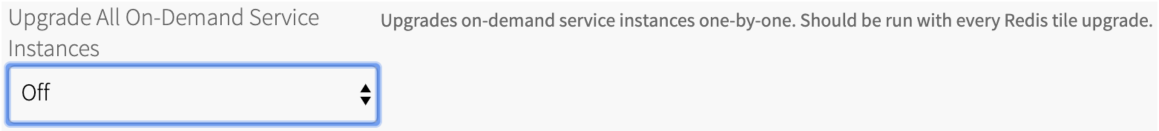
Until you upgrade service instances, they do not benefit from any security fixes or new features included in the tile upgrade. The default upgrade path automatically upgrades all on-demand service instances when you upgrade the tile. This operation can take a long time.

To expedite upgrades, in Redis for Tanzu Application Service v2.3 and later you can enable on-demand service instances to be upgraded individually. This allows developers to upgrade their own service instances after you have upgraded the tile.

> **Note**
>
> This feature is only available for upgrades from Redis for Tanzu Application Service v2.3.0 to later versions. You cannot upgrade individual service instances from v2.2 to v2.3.

To enable upgrading individual service instances:

1. Ensure that all service instances have been upgraded to Redis for Tanzu Application Service v2.3.0 or later. If not, click **Apply changes** to run the `upgrade-all-service instances` errand.
2. In Redis for Tanzu Application Service tile, navigate to the **Errands** page.
3. Select **Off** for the **Upgrade All On-Demand Service Instances** errand:



Upgrade All On-Demand Service Instances

Off

Upgrades on-demand service instances one-by-one. Should be run with every Redis tile upgrade.

Click here to view a larger version of this image

4. Click **Save**.
5. Click **Apply changes**.

After you enable individual service instance upgrades, developers can upgrade individual service instances following the instructions in Upgrading an individual Redis Service Instance.

Downtime during upgrades

During the upgrade each Redis instance experiences a small period of downtime as each instance is updated with the new software components. This downtime is because Redis instances are single VMs operating in a non-high availability (HA) setup. To reduce downtime, you can enable the BOSH HotSwaps feature. Compared to traditional BOSH upgrades, this feature has been shown to reduce downtime by 75%. For instructions on how to enable this feature, see Enable BOSH HotSwaps to Reduce Downtime below.

The length of downtime depends on whether there is a stemcell update to replace the operating system image, or whether the Redis software is updated on the existing VM. Stemcell updates incur additional downtime while the IaaS creates the new VM, whereas updates without a stemcell update are faster.

Tanzu Operations Manager ensures the instances are updated with the new packages and any configuration changes are applied automatically.

Upgrading to a newer version of the product does not cause any loss of data or configuration.

Causes of downtime

A redeploy causes downtime for the Redis for Tanzu Application Service tile. This section clarifies what events trigger a redeploy.

Changes in Tanzu Operations Manager

In Tanzu Operations Manager, any field that changes the manifest causes a redeploy of the Redis for Tanzu Application Service tile.

Changes in VMware Tanzu Application Service for VMs

In the VMware Tanzu Application Service for VMs tile, changes to any of the following properties can trigger downtime:

- `$runtime.system_domain`—Runtime System Domain
- `..cf.ha_proxy.skip_cert_verify.value`—Deactivate SSL certificate verification for this environment in TAS for VMs
- `$runtime.apps_domain`—Runtime Apps Domain
- `..cf.nats.ips`—NATS Resource Config

- `$self.service_network`—Service Networks in Tanzu Operations Manager

When the operator applies any of the above changes to TAS for VMs, downtime is triggered for:

- The Redis on-demand broker
- Shared-VM Services

Upgrading all service instances

Downtime for service instances occurs only after the operator runs the `upgrade-all-service-instances` BOSH errand, after all tile upgrades are completed successfully. Any change to a field on the Redis for Tanzu Application Service tile causes BOSH to redeploy the on-demand Redis broker and can cause service instance downtime when the operator runs the `upgrade-all-service-instances` errand.

Enable BOSH HotSwaps to reduce downtime

Enabling BOSH HotSwaps reduces the downtime for on-demand service instances when upgrading. Benchmarking shows that enabling BOSH HotSwaps can reduce service instance downtime by 75% when upgrading. For how it works, see Changing VM update strategy in the BOSH documentation. To use this feature, all service bindings must use BOSH DNS instead of IP addresses.

To enable BOSH HotSwaps:

1. Ensure all service bindings use BOSH DNS. To do so, tell developers to unbind, bind, and restage any apps created while Redis for Pivotal Cloud Foundry v1.14 or earlier was installed. For instructions, see the solution in Apps fail to connect to the Service Instance.

   > **Note**
   >
   > You must do this before enabling BOSH HotSwaps. Any apps with service bindings that do not use BOSH DNS fail to connect to the Redis service instance.

2. Select the **BOSH HotSwaps** check box in the **On-Demand Service Settings** tab.
3. Click **Save** and then **Apply Changes**.

Network changes after deployment

This section explains how changing the network after deploying Redis for Tanzu Application Service affects instances and apps.

Shared VMs

To change the network for shared-VM services, click **Assign AZs and Networks** in the Redis for Tanzu Application Service tile configuration and use the **Network** dropdown.

You can also change the network by altering the CIDR in the BOSH Director tile.

VMware discourages changing the network that a pre-existing shared-VM deployment works with.

If the network is changed, app bindings for existing shared-VM instances might stop working.

On-demand service instances

To change the service network for on-demand service instances, click **Assign AZs and Networks** in the Redis tile configuration and use the **Service Network** dropdown. The service network applies to on-demand service instances.

You can also change the service network by altering the CIDR in the BOSH Director tile.

If you change the service network, you must unbind and rebind existing apps to the on-demand Redis instance.

New on-demand service instances are placed into the new service network, but existing on-demand service instances are not moved. To move the data in on-demand Redis instances to a new service network, you must create a new instance, migrate the data manually, and delete the old instance.

Similarly, changing the availability zone (AZ) for an on-demand plan only applies to new on-demand instances and does not alter existing instances.

Release policy

When a new version of Redis is released, a new version of Redis for Tanzu Application Service is released soon after. For more information, see the Release Policy.

Setting limits for On-Demand Redis service instances

This topic tells you how operators can set resource quotas for Redis for VMware Tanzu Application Service services.

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 many 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.

You set 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 might 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:

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

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

   Where:

   - `QUOTA-NAME`—A name for this quota
   - `TOTAL-MEMORY`—Maximum memory used by all service instances combined
   - `INSTANCE-MEMORY`—Maximum memory used by any single service instance
   - `ROUTES`—Maximum number of routes allowed for all service instances combined
   - `SERVICE-INSTANCES`—Maximum number of service instances allowed for the org

   For example:

   ```
   $ cf create-org-quota myquota -m 1024mb -i 16gb -r 30 -s 50 --allow-pai
   d-service-plans
   ```

2. Associate the quota that you created with a specific org by running:

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

For example:

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

For more information about managing org-level quotas, see Creating and modifying quota plans.

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:

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`—A name for this quota
   - `TOTAL-MEMORY`—Maximum memory used by all service instances combined
   - `INSTANCE-MEMORY`—Maximum memory used by any single service instance
   - `ROUTES`—Maximum number of routes allowed for all service instances combined
   - `SERVICE-INSTANCES`—Maximum number of service instances allowed for the org

   For example:

   ```
   $ cf create-space-quota myspacequota -m 1024mb -i 16gb -r 30 -s 50 --allow-paid-service-plans
   ```

2. Associate the quota you created with a specific space by run:

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

   For example:

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

For more information about managing space-level quotas, see Creating and modifying quota plans.

View Current Org and Space-Level Quotas

To view **org** quotas, run:

```
cf org ORG-NAME
```

To view **space** quotas, run:

```
cf space SPACE-NAME
```

For more information about managing org and space-level quotas, see the Creating and modifying quota plans.

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 are the listed metrics:

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

Important

Quota metrics are not emitted if no quota was set.

You can also view service instance use information in Apps Manager. For more information, see Reporting instance usage with Apps Manager.

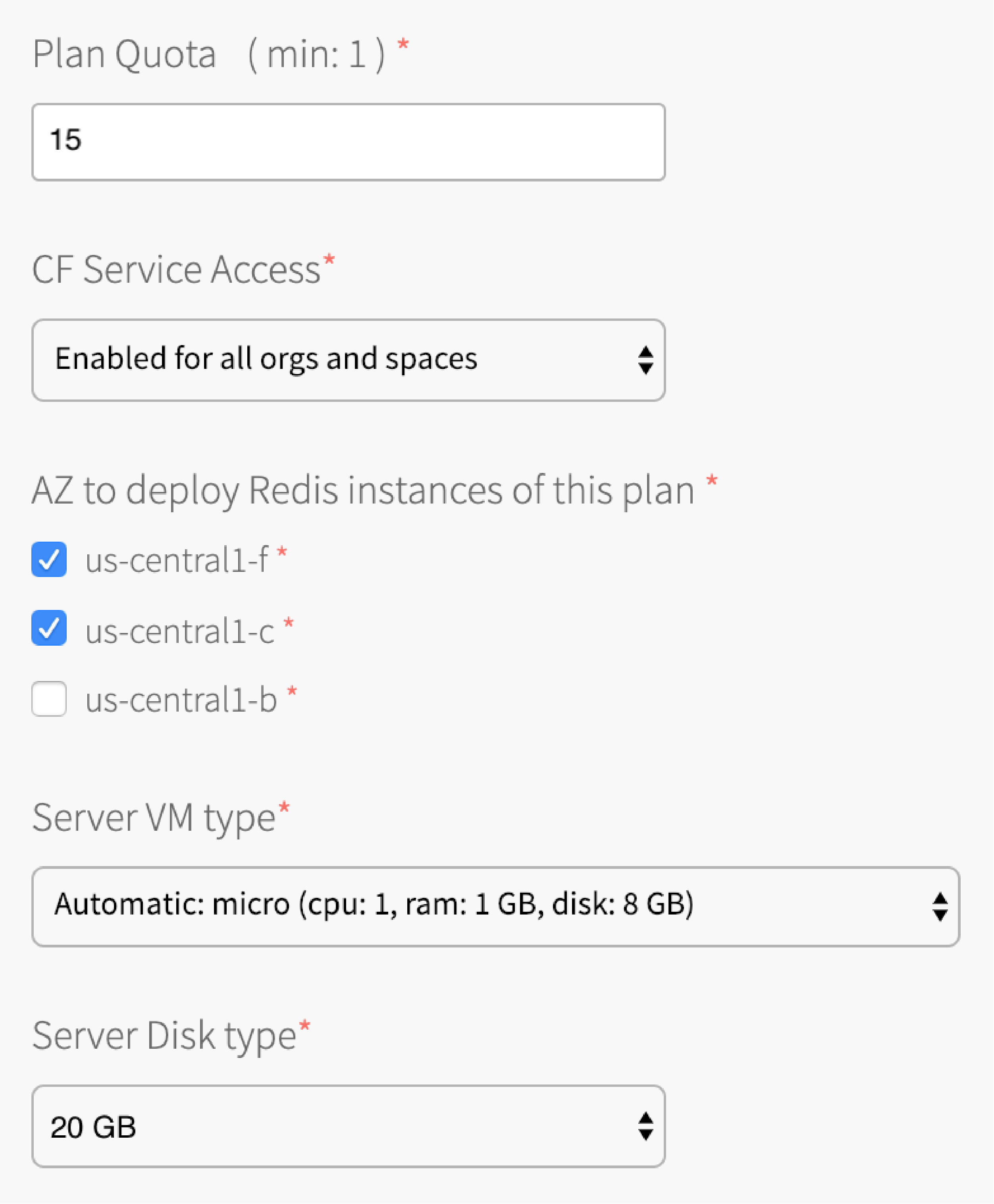
Calculate Resource Costs for On-Demand Plans

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, you multiply the quota by the cost of the resources selected in the plan configurations. The costs depend on your IaaS.

To view configurations for your Redis for Tanzu Application Service on-demand plan:

1. Go to **Tanzu Operations Manager Installation Dashboard** > **Redis for VMware Tanzu Application Service** > **Settings**.
2. Click **On-Demand Plans**.
3. Click the drop-down menu for the plan you want to view. For example, **on-demand-cache**.

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



Plan Quota (min: 1) *

15

CF Service Access*

Enabled for all orgs and spaces

AZ to deploy Redis instances of this plan *

☑ us-central1-f *

☑ us-central1-c *

□ us-central1-b *

Server VM type*

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

Server Disk type*

20 GB

Although operators can limit on-demand instances with plan quotas and a global quota, as described earlier, IaaS resource use 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 calculation as shown here:

plan quota x cost of selected resources

For example, if you selected the options shown in the image, you 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 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. Ensure 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 your on-demand plans:

1. Find the number of instances 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

Configuring automated service backups in Redis for Tanzu Application Service

This topic tells you how to configure automated backups in Redis for VMware Tanzu Application Service.

Comparison of available backup methods

Redis for Tanzu Application Service provides two backup methods, which you can use together or alone.

They are:

- BOSH Backup and Restore (BBR) - (preferred)
- Automated service backups

If you have already set up BBR for your VMware Tanzu Application Service for VMs deployment, you might find it easier to use BBR to back up your on-demand Redis service instances, in addition to or instead of, using automated service backups.

The following table summarizes the differences between the two methods:

Backup method	Supported services	What is backed up
BBR	On-demand	- Data stored in Redis - Manifest used to deploy service instance - Certain additional configuration including: plan settings such as **Redis Client Timeout** and arbitrary parameters such as `maxmemory-policy`
Automated service backups	- On-demand - Shared-VM	Data stored in Redis

Neither backup method backs up other manual changes made to service instances, either using SSH or with the Redis client `config` command.

For more information, see BOSH Backup and Restore (BBR) for On-Demand Redis for VMware Tanzu Application Service.

Automated service backups

You can configure automatic backups for both on-demand and shared-VM plan types.

Automated backups have the following features:

- Backups run on a configurable schedule.
- Every instance is backed up.
- The Redis broker state file is backed up.
- Data from Redis is flushed to disk before the backup is started by running a `BGSAVE` on each instance.
- You can configure Amazon Web Services (AWS) S3, SCP, Azure, or Google Cloud Storage (GCS) as your destination.

Backup files

When Redis for Tanzu Application Service runs an automated backup, it labels the backups in the following ways:

- For shared-VM plans, backups are labeled with timestamp, instance GUID, and plan name. Files are stored by date.
- For on-demand plans, backups are labeled with timestamp and plan name. Files are stored by deployment, then date.

For each backup artifact, Redis for Tanzu Application Service creates a file that contains the MD5 checksum for that artifact. This can be used to check that the artifact is not corrupted.

Configuring backups

Redis for Tanzu Application Service automatically backs up databases to external storage.

- **How and where**: There are four options for how automated backups transfer backup data and where the data saves to:
    - Option 1: Back Up with AWS: Redis for Tanzu Application Service runs an Amazon S3 client that saves backups to an S3 bucket.
    - Option 2: Back Up with SCP: Redis for Tanzu Application Service runs an SCP command that secure-copies backups to a VM or physical machine operating outside the deployment. SCP stands for secure copy protocol, and offers a way to securely transfer files between two hosts. The operator provisions the backup machine separately from their installation. This is the fastest option.
    - Option 3: Back Up to GCS: Redis for Tanzu Application Service runs an GCS SDK that saves backups to an Google Cloud Storage bucket.
    - Option 4: Back Up to Azure: Redis for Tanzu Application Service runs an Azure SDK that saves backups to an Azure storage account.
- **When**: Backups follow a schedule that you specify with a cron expression.

  For general information about cron, see package cron.

To configure automated backups, follow these procedures according to the option you choose for external storage.

Option 1: Back up with AWS

To back up your database to an Amazon S3 bucket:

- Create a Policy and Access Key
- Configure Backups in Tanzu Operations Manager

Create a policy and access key

Redis for Tanzu Application Service accesses your S3 store through a user account. VMware recommends that this account be solely for Redis for Tanzu Application Service. You must apply a minimal policy that lets the user account upload backups to your S3 store.

Do the following to create a policy and access key:

1. Go to the AWS Console and log in.
2. To create a new custom policy, go to **IAM > Policies > Create Policy > Create Your Own Policy** and paste in the following permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::MY-BUCKET-NAME",
                "arn:aws:s3:::MY-BUCKET-NAME/*"
            ]
        }
    ]
}
```

Where `MY-BUCKET-NAME` is the name of your S3 bucket.

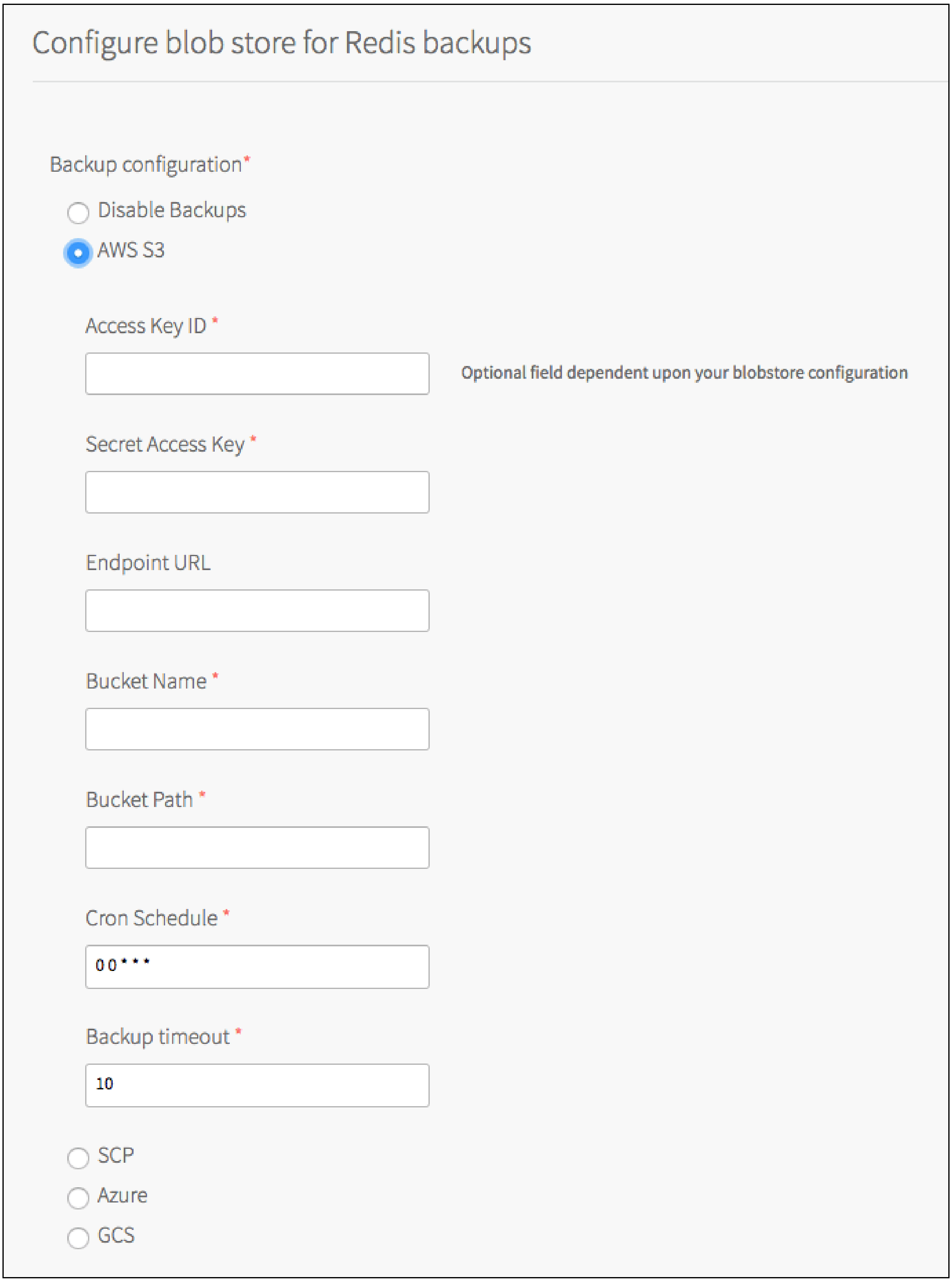
If the S3 bucket does not already exist, add `s3:CreateBucket` to the `Action` list to create it.

3. (Recommended) Create a new user for Redis for Tanzu Application Service and record its Access Key ID and Secret Access Key, the user credentials.
4. (Recommended) Attach the policy you created to the AWS user account that Redis for Tanzu Application Service will use to access S3. Go to **IAM > Policies > Policy Actions > Attach**.

Configuring backups in Tanzu Operations Manager

Do the following to connect Redis for Tanzu Application Service to your S3 account:

1. Go to the Tanzu Operations Manager Installation Dashboard and click the **Redis for Tanzu Application Service** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **AWS S3**.



Configure blob store for Redis backups

Backup configuration*

○ Disable Backups

● AWS S3

Access Key ID *

Optional field dependent upon your blobstore configuration

Secret Access Key *

Endpoint URL

Bucket Name *

Bucket Path *

Cron Schedule *

0 0 * * *

Backup timeout *

10

○ SCP

○ Azure

○ GCS

4. Fill in the fields as follows:

Field	Description	Mandatory/optional
Access Key ID	The access key for your S3 account	Mandatory
Secret Access Key	The Secret Key associated with your Access Key	Mandatory

Field	Description	Mandatory/optional
Endpoint URL	The endpoint of your S3 account, such as `http://s3.amazonaws.com`	Optional, defaults to `http://s3.amazonaws.com` if not specified
Bucket Name	Name of the bucket where to store the backup	Mandatory
Bucket Path	Path inside the bucket to save backups to	Mandatory
CA Certificate	CA certificate used to verify the connection to the S3 bucket	Optional
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is `* 2 * * *`. This field also accepts a pre-defined schedule, such as `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, or `@every TIME`, where `TIME` is any supported time string, such as `1h30m`. For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in minutes, that the backup process waits for the `BGSAVE` command to complete on your instance before transferring the RDB file to your configured destination. If the timeout is reached, `BGSAVE` continues but backups fail and are not uploaded.	Mandatory

5. Click **Save**.

The backup service uses Amazon S3 with only support of virtual-hosted-style addressing and does not use path-style-addressing. If you plan to use Amazon S3's API compatible services for backup, for example minio, it must be configured to use virtual-hosted-addressing.

Option 2: Back Up with SCP

Option 2: Back up with SCP

To back up your database using SCP:

- (Recommended) Create a Public and Private Key Pair
- Configure Backups in Tanzu Operations Manager

(Recommended) Create a public and private key pair

Redis for Tanzu Application Service accesses a remote host as a user with a private key for authentication. VMware recommends that this user and keypair be solely for Redis for Tanzu Application Service.

Do the following to create a new public and private keypair for authenticating:

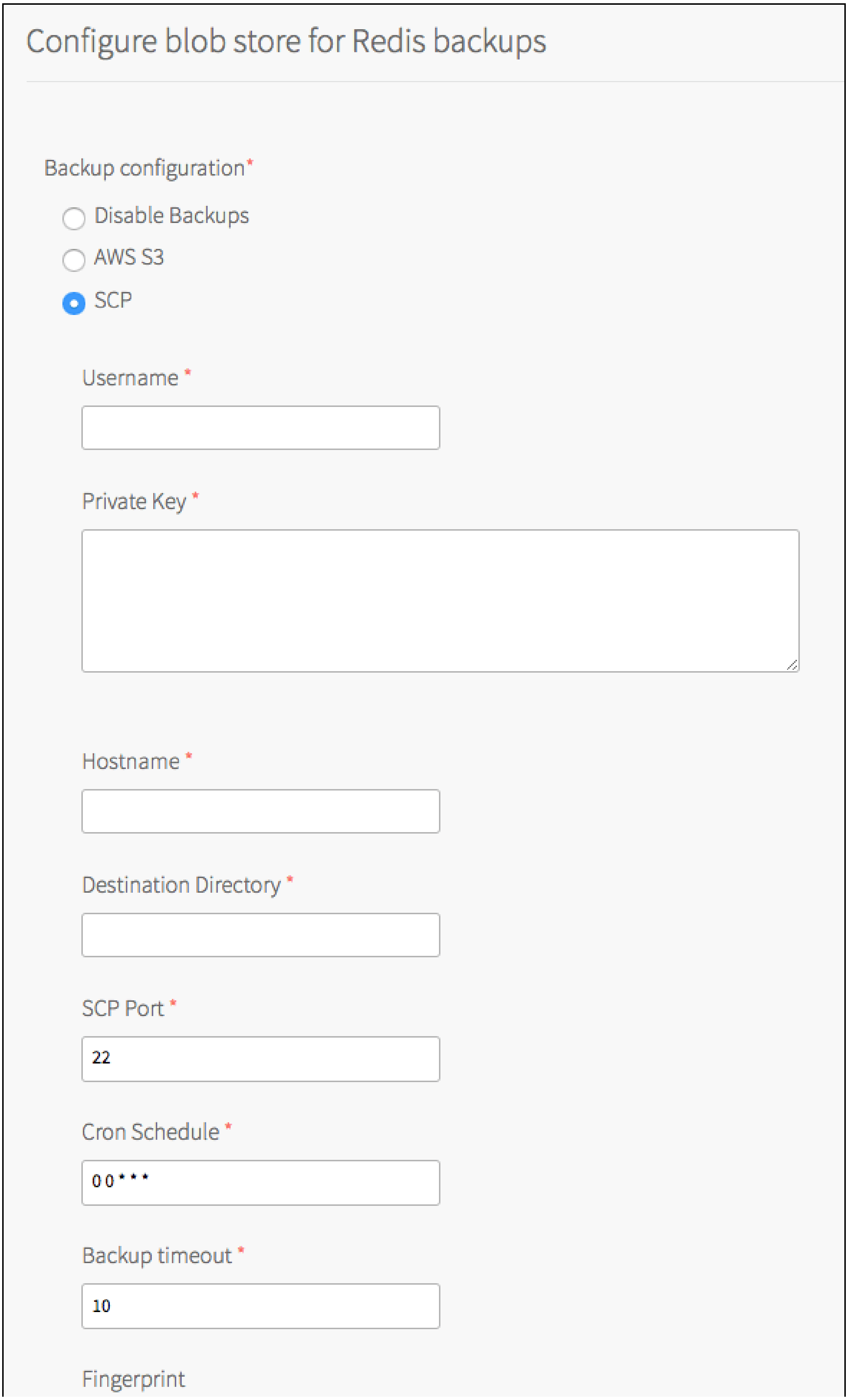
1. Determine the remote host to use to store backups for Redis for Tanzu Application Service. Ensure that the Redis service instances can access the remote host. VMware recommends using a VM outside the deployment for the destination of SCP backups. As a result, you might need to enable public IPs for the Redis VMs.
2. Create a new user for Redis for Tanzu Application Service on the destination VM.

3. Create a new public and private keypair for authenticating as the above user on the destination VM.

Configuring backups in Tanzu Operations Manager

Do the following to connect Redis for Tanzu Application Service to your destination VM:

1. Go to the Tanzu Operations Manager Installation Dashboard and click the **Redis for Tanzu Application Service** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **SCP**.



Configure blob store for Redis backups

Backup configuration*

○ Disable Backups

○ AWS S3

● SCP

Username *

Private Key *

Hostname *

Destination Directory *

SCP Port *

22

Cron Schedule *

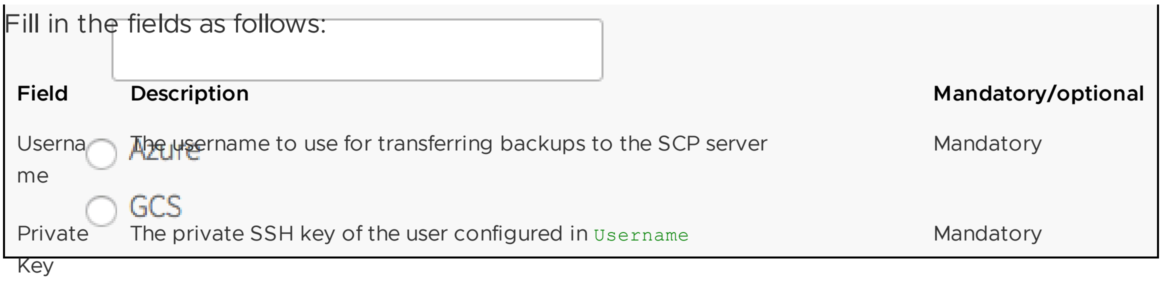
0 0 * * *

Backup timeout *

10

Fingerprint

4. Fill in the fields as follows:



○ Azure

○ GCS

Field	Description	Mandatory/optional
Username	The username to use for transferring backups to the SCP server	Mandatory
Private Key	The private SSH key of the user configured in `Username`	Mandatory
Host name	The host name or IP address of the SCP server	Mandatory
Destination Directory	The path in the SCP server, where the backups will be transferred	Mandatory
SCP Port	The SCP port of the SCP server	Mandatory
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is `* 2 * * *`. This field also accepts a pre-defined schedule, such as `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, or `@every TIME`, where `TIME` is any supported time string, such as `1h30m`. For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in minutes, that the backup process waits for the `BGSAVE` command to complete on your instance before transferring the RDB file to the SCP server. If the timeout is reached, `BGSAVE` continues but backups fail and are not uploaded.	Mandatory
Fingerprint	The fingerprint of the public key of the SCP server. To retrieve the server's fingerprint, run `ssh-keygen -E md5 -lf ~/.ssh/id_rsa.pub`.	Optional

5. Click **Save**.

Option 3: Back up with GCS

To back up your database using GCS:

- Create a Service Account
- Configure Backups in Tanzu Operations Manager

Create a service account

Redis for Tanzu Application Service accesses your GCS store through a service account. VMware recommends that this account be solely for Redis for Tanzu Application Service. You must apply a minimal policy that lets the user account upload backups to your GCS store.

Do the following to create a service account with the correct permissions:

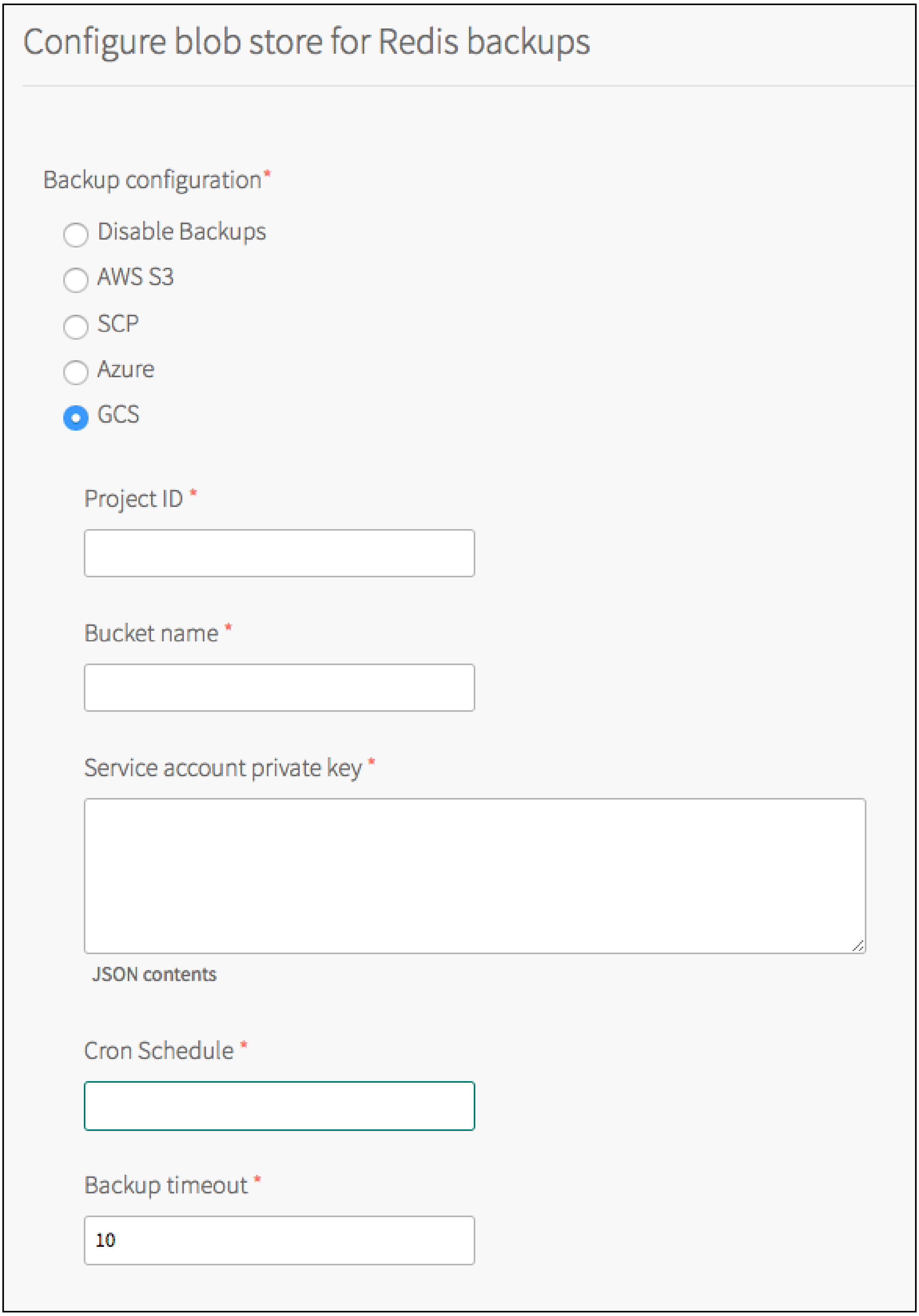
1. In the GCS console, create a new service account for Redis for Tanzu Application Service: **IAM and Admin** > **Service Accounts** > **Create Service Account**.
2. Enter a unique name in the **Service account name** field, such as `Redis-for-VMware-Tanzu`.
3. In the **Roles** dropdown, grant the new service account the **Storage Admin** role.

4. Select the **Furnish a new private key** checkbox so that a new key is created and downloaded.
5. Click **Create** and take note of the name and location of the service account JSON file that is downloaded.

Configuring backups in Tanzu Operations Manager

Do the following to connect Redis for Tanzu Application Service to GCS:

1. Go to the Tanzu Operations Manager Installation Dashboard and click the **Redis for Tanzu Application Service** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **GCS**.



Configure blob store for Redis backups

Backup configuration*

○ Disable Backups
○ AWS S3
○ SCP
○ Azure
● GCS

Project ID *

Bucket name *

Service account private key *

JSON contents

Cron Schedule *

Backup timeout *

10

4. Fill in the fields as follows:

Field	Description	Mandatory/optional
Project ID	Google Cloud Platform (GCP) Project ID	Mandatory

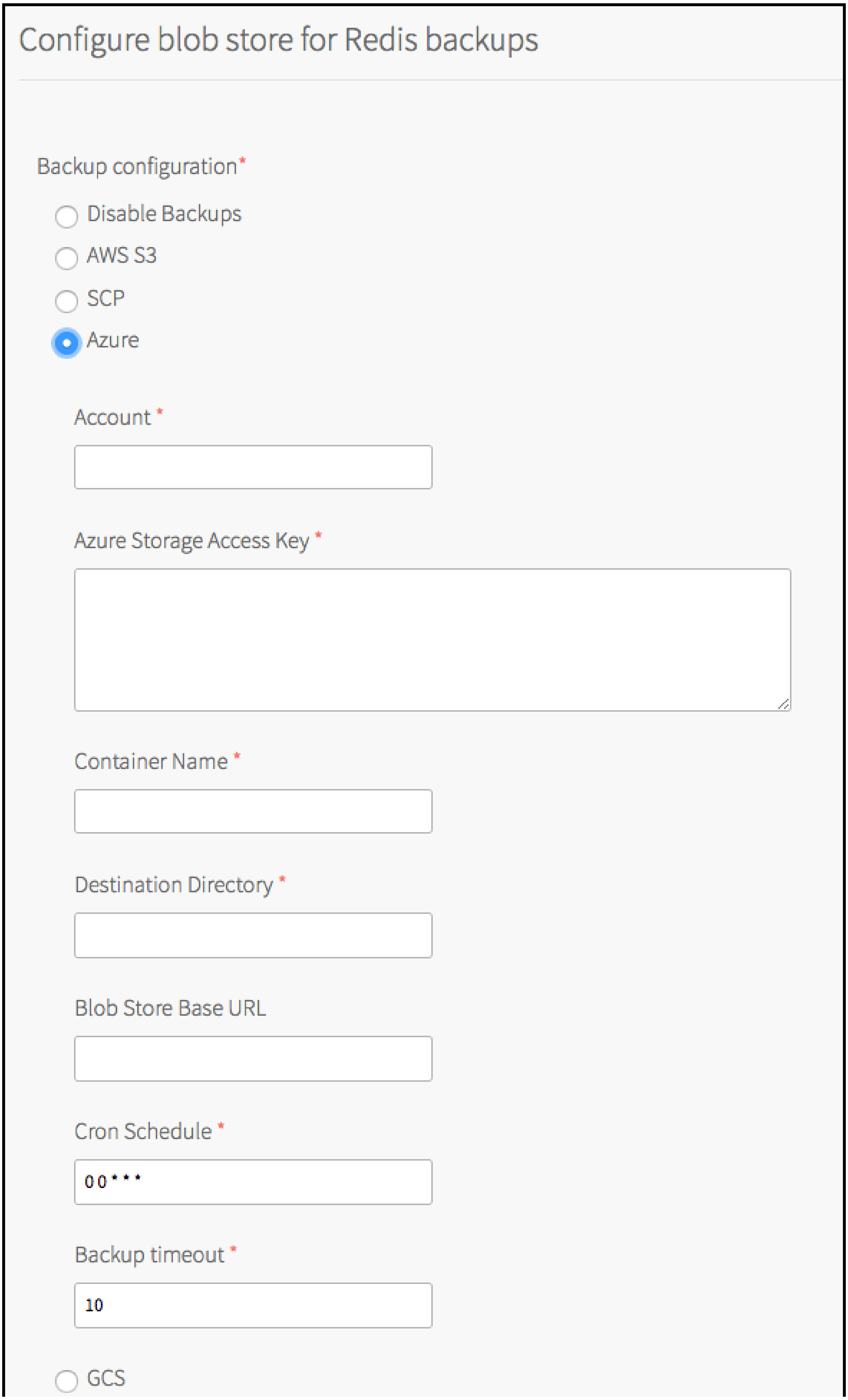
Field	Description	Mandatory/optional
Bucket name	Name of the bucket where to store the backup	Mandatory
Service account private key	The JSON secret key associated with your service account	Mandatory
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is `* 2 * * *`. This field also accepts a pre-defined schedule, such as `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, or `@every TIME`, where `TIME` is any supported time string, such as `1h30m`. For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in minutes, that the backup process waits for the `BGSAVE` command to complete on your instance before transferring the RDB file to your configured destination. If the timeout is reached, `BGSAVE` continues but backups fail and are not uploaded.	Mandatory

5. Click **Save**.

Back up to Azure

Do the following to back up your database to an Azure storage account:

1. Go to the Tanzu Operations Manager Installation Dashboard and click the **Redis for Tanzu Application Service** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **Azure**.



Configure blob store for Redis backups

Backup configuration*

○ Disable Backups

○ AWS S3

○ SCP

● Azure

Account *

Azure Storage Access Key *

Container Name *

Destination Directory *

Blob Store Base URL

Cron Schedule *

0 0 * * *

Backup timeout *

10

○ GCS

4. Fill in the fields as follows:

Field	Description	Mandatory/optional
Account	Account name	Mandatory
Azure Storage Access Key	Azure specific credentials required to write to the Azure container	Mandatory
Container Name	Name of the Azure container where to store the backup	Mandatory
Destination Directory	Directory within the Azure container to store the backup files to	Mandatory
Blob Store Base URL	URL pointing to Azure resource	Optional
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is `* 2 * * *`. This field also accepts a pre-defined schedule, such as `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, or `@every TIME`, where `TIME` is any supported time string, such as `1h30m`. For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in minutes, that the backup process waits for the `BGSAVE` command to complete on your instance before transferring the RDB file to your configured destination. If the timeout is reached, `BGSAVE` continues but backups fail and are not uploaded.	Mandatory

5. Click **Save**.

Back up and restore manually

To back up or restore Redis manually, see Manually backing up and restoring Redis for Tanzu Application Service.

Using BOSH Backup and Restore with Redis for Tanzu Application Service

You can use the BOSH Backup and Restore (BBR) command-line tool for backing up and restoring BOSH deployments and on-demand Redis for VMware Tanzu Application Service.

BBR offers a standardized way to backup and restore the BOSH Director and BOSH Deployments that support it. If you have already set up BBR for your VMware Tanzu Application Service for VMs (TAS for VMs) deployment, you might find it easier to use BBR to back up your Redis service instances, in addition to, or instead of, using automated service backups.

For more information, see Configuring automated service backups and Comparison of the available backup methods.

Note

Only on-demand Redis service instances have support for BBR. For backup and restore of shared instances, see Configuring Automated Backups for Redis for Tanzu Application Service.

Preparing to use BBR

To take a backup of BOSH deployments and on-demand Redis for Tanzu Application Service, BBR must be installed. If you do not already have it installed, follow the instructions in Preparing to create your backup in the BBR documentation.

When deciding on the disk size for the jumpbox, remember that the Redis backup artifact is roughly 1/10 of the RAM usage of the Redis instance.

Record the BOSH Director IP and path to server certificate.

Identify your Redis deployments

You need the names of your Redis service instances to back up and restore them.

To obtain the instance deployment names:

1. Run the following from your jumpbox, and record the resulting names.

   ```
   $ BOSH_CLIENT=REDIS-BOSH-CLIENT \
   BOSH_CLIENT_SECRET=REDIS-BOSH-PASSWORD \
   bosh -e BOSH-DIRECTOR-IP \
   --ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \
   --column name \
   deployments
   ```

   Where:

   - `REDIS-BOSH-CLIENT`, `REDIS-BOSH-PASSWORD`: To find these in the Tanzu Operations Manager Installation Dashboard, click the Redis for Tanzu Application Service tile, navigate to the **Credentials** tab, and click **UAA Client Credentials**. Note the `Redis BOSH UAA` credentials.
   - `BOSH-DIRECTOR-IP`: You retrieved this value in Step 1–6: Preparing to create your backup.
   - `PATH-TO-BOSH-SERVER-CERTIFICATE`: This is the path to the Certificate Authority (CA) certificate for the BOSH Director. For more information, see Set up your jumpbox.

   In the preceding command, `BOSH_CLIENT` is not a variable.

   For example:

   ```
   $ BOSH_CLIENT=p-redis-eb12345cb7a123450f08 \
   BOSH_CLIENT_SECRET=338b012345d987bb24b5f \
   bosh -e 10.0.0.5 \
   --ca-cert /var/example/workspaces/default/root_ca_certificate \
   --column name \
   deployments
   ```

Back up using BBR

To back up using BBR:

1. Back up your BOSH deployments.

   This includes backing up your Tanzu Operations Manager installation settings, BOSH Director and TAS for VMs, as detailed in Backing up your Tanzu Operations Manager deployments with BBR.

   For a full restore of Redis service instances to be valid, you must have a backup of the BOSH Director and TAS for VMs.

2. Backup each Redis service instance. From your jumpbox run the following:

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

   Where:

   - `BOSH-CLIENT`, `BOSH-CLIENT-PASSWORD`: These are the client credentials you retrieved in Preparing to use BBR.
   - `REDIS-SERVICE-INSTANCE-DEPLOYMENT-NAME`: This is the deployment name for the on-demand Redis service instance you are backing up.

   In the preceding command, `BOSH_CLIENT_SECRET` is not a variable.

   For example:

   ```
   $ BOSH_CLIENT_SECRET=KJsdgKJj12345ljk83Hufy12345b6-34n4 \
   bbr deployment \
   --target 10.0.0.5 \
   --username ops_manager \
   --ca-cert /var/example/workspaces/default/root_ca_certificate \
   --deployment service-instance_40b123e4a-be1c-1232-ad31-123e01b7d169 \
   backup
   ```

3. Follow the steps given in the After taking your backup step of the BBR documentation.

   Ensure you do this for the backup artifacts for all of your service instances and your BOSH Director and TAS for VMs.

Restore using BBR

To restore using BBR:

1. To restore on-demand Redis service instance data, follow the procedure for Restoring Tanzu Operations Manager deployments from backup with BBR in full.

Ensure that as part of Step 6: Transfer artifacts to jumpbox you transfer your Redis service instance artifacts.

2. For each Redis service instance artifact, run the following command from your jumpbox:

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

Where `PATH-TO-SERVICE-INSTANCE-ARTIFACT` is the path to the artifact for the instance that you are currently restoring. By default the artifact directory includes the deployment name and timestamp.

In the preceding command, `BOSH_CLIENT_SECRET` is not a variable.

For example:

```
$ BOSH_CLIENT_SECRET=KJsdgKJj12345jk83Hufy12345b6-34n4 \
  bbr deployment \
--target 10.0.0.5 \
--username ops_manager \
--ca-cert /var/example/workspaces/default/root_ca_certificate \
--deployment service-instance_40b12e4a-be1c-1232-ad31-12345e01b7d123 \
restore --artifact-path /tmp/service-instance_40b12e4a-be1c-1232-ad31-12345e01b7d169_1234503T141538Z
```

If a restore fails because there is no deployment of the name specified, then you are likely in the Backing up artifact for a non-existent service instance inconsistent state and can skip the restore for that artifact. For more information, see Backing up artifact for a non-existent service instance.

Note

If you have a backup artifact (a `dump.rdb` file) from any source besides a BBR backup, you can also use it in this restore procedure.

Possible inconsistent states

Because the Redis On-Demand broker is not locked during the backup process, the backups of the TAS for VMsand service instances can be out of sync if an app developer creates or deletes an on-demand Redis service between the TAS for VMsbackup and Redis service instance backups.

No backing up artifact for a service instance

If an on-demand Redis service was deleted in between the backup of the TAS for VMsand the Redis service instances, there is no backup artifact for a deployed service instance. Resolve this by deleting the service, which had already been deleted during the backup process so presumably is not wanted.

Backing up artifact for a non-existent service instance

If an on-demand Redis service was created between the backup of the TAS for VMsand the Redis service instances, there is a backup artifact which has no corresponding deployed service instance. In this case, the only action you need to take is to skip the restore of this artifact. The app developer who created the service can recreate it.

Monitoring Redis for VMware Tanzu Application Service

You can monitor the health of the Redis for VMware Tanzu Application Service service using the logs, metrics, and Key Performance Indicators (KPIs) generated by Redis for Tanzu Application Service component VMs.

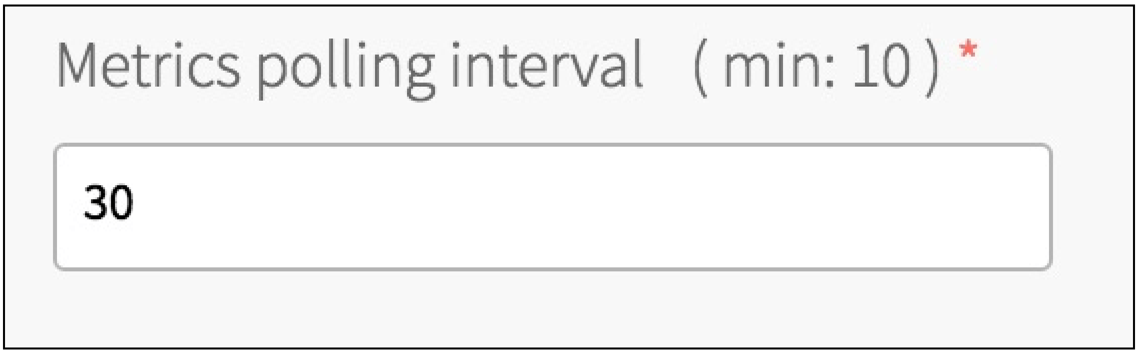
Loggregator

Redis metrics are emitted through Loggregator through the Reverse Log Proxy and Log Cache. You can use third-party monitoring tools to consume Redis metrics to monitor Redis performance and health. The Loggregator Firehose architecture endpoint is being deprecated.

As an example of how to display KPIs and metrics without the Firehose, see the CF Redis example dashboard in GitHub. This example uses Datadog. However, VMware does not endorse or support any third-party solution.

Metrics polling interval

The metrics polling interval defaults to 30 seconds. You can change this by navigating to the Metrics configuration page in Tanzu Operations Manager and entering a new value in **Metrics polling interval (min: 10)**.



Metrics polling interval (min: 10) *

30

Metrics are emitted in the following format:

```
origin:"p-redis" eventType:ValueMetric timestamp:1480084323333475533 deployme
nt:"cf-redis" job:"cf-redis-broker" index:"{redacted}" ip:"10.0.1.49" valueMe
tric:<name:"_p_redis_service_broker_shared_vm_plan_available_instances" valu
e:4 unit:"" >
```

Critical logs

VMware recommends operators set up alerts on critical logs to help prevent further degradation of the Redis service. For examples of critical logs for service backups, including log messages for failed backups, backups with errors, and backups that failed to upload to destinations, see Troubleshooting in the Service Backups documentation.

Healthwatch

The Healthwatch service monitors and alerts on the current health, performance, and capacity of your service instances. By default, the Healthwatch dashboard displays core metrics and alerts configured for recommended thresholds.

For more information, see Using Healthwatch.

Key performance indicators

Key Performance Indicators (KPIs) for Redis for VMware Tanzu Application Service are metrics that operators find most useful for monitoring their Redis service to ensure smooth operation. KPIs are high-signal-value metrics that can indicate emerging issues. KPIs can be raw component metrics or *derived* metrics generated by applying formulas to raw metrics.

VMware recommends the following KPIs for general alerting and response with typical Redis for Tanzu Application Service installations. If using Healthwatch, some core metrics are configured by default using the recommended thresholds below. VMware recommends that operators continue to fine-tune the alert measures to their installation by observing historical trends. VMware also recommends that operators expand beyond this guidance and create new, installation-specific monitoring metrics, thresholds, and alerts based on learning from their own installations.

For how to create custom service alerts for Healthwatch, see Configuring Healthwatch alerts.

For a list of all other Redis metrics, see Other Redis metrics.

Redis for Tanzu Application Service KPIs

Total instances for on-demand service

total_instances	
Description	Total instances provisioned by app developers across all on-demand services and for a specific on-demand plan **Use**: Track instance use by app developers. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Daily
Recommended alert thresholds	**Yellow warning**: N/A **Red critical**: N/A
Recommended response	N/A

Quota remaining for on-demand service

	quota_remaining
Description	Number of available instances across all on-demand services and for a specific on-demand plan. **Use**: Track remaining resources available for app developers. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Daily
Recommended alert thresholds	**Yellow warning**: 3 **Red critical**: 0
Recommended response	Increase quota allowed for the specific plan or across all on-demand services.

Total instances for shared VM service

	_p_redis_service_broker_shared_vm_plan_total_instances
Description	Total instances provisioned for shared-VM services. **Use**: Track total shared-VM instances available for app developers. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	App-specific
Recommended alert thresholds	**Yellow warning**: N/A **Red critical**: N/A
Recommended response	N/A

Redis KPIs

The metrics in this section can be used for on-demand and shared-VM service instances. You can differentiate between these service instance metrics as follows:

- **On-demand service instances:**
  - Have origin `p.redis`
- **Shared-VM service instances:**
  - Have origin `p-redis`

- Their names are pre-pended with `_p_redis_shared_vm_SHARED_INSTANCE_GUID/`. `SHARED-INSTANCE-GUID` can be retrieved by running `cf service SERVICE-NAME -guid`.

Percent of persistent disk used

disk.persistent.percent	
Description	Percentage of persistent disk being used on a VM. The persistent disk is specified as an IaaS-specific disk type with a size. For example, `pd-standard` on GCP, or `st1` on AWS, with disk size 5 GB. This is a metric relevant to the health of the VM. A percentage of disk usage approaching 100 causes the VM disk to become unusable as no more files are allowed to be written. **Use**: Redis is an in-memory datastore that uses a persistent disk to backup and restore the dataset in case of upgrades and VM restarts. **Origin**: BOSH HM **Type**: percent **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: >75 **Red critical**: >90
Recommended response	Ensure that the disk is at least 2.5x the VM memory for the on-demand broker and 3.5x the VM memory for cf-redis-broker. If it is, then contact VMware Tanzu Support. If it is not, then increase disk space.

Used memory percent

info.memory.used_memory / info.memory.maxmemory	
Description	The ratio of these two metrics returns the percentage of available memory used: • `info.memory.used_memory` is a metric of the total number of bytes allocated by Redis using its allocator (either standard libc, jemalloc, or an alternative allocator such as tcmalloc). • `maxmemory` is a configuration option for the total memory made available to the Redis instance. **Use**: This is a performance metric that is most critical for Redis instances with a `maxmemory-policy` of `allkeys-lru` **Origin**: Doppler/Firehose **Type**: percentage **Frequency**: 30s (default), 10s (configurable minimum)

	info.memory.used_memory / info.memory.maxmemory
Recommended measurement	App-specific based on velocity of data flow. Some options are: 1. Individual data points---Use if key eviction is in place, for example, in cache use cases. 2. Average over last 10 minutes---Use if this gives you enough detail. 3. Maximum of last 10 minutes If key eviction is not in place, options 1 or 3 give more useful information to ensure that high usage triggers an alert.
Recommended alert thresholds	**Yellow warning**: 80% Not applicable for cache usage. When used as a cache, Redis typically uses up to maxmemory and then evict keys to make space for new entries. A different threshold might be appropriate for specific use cases of no key eviction, to account for reaction time. Factors to consider: 1. Traffic load on app---Higher traffic means that Redis memory fills up faster. 2. Average size of data added/ transaction---The more data added to Redis on a single transaction, the faster Redis fills up its memory. **Red critical**: 90%. See warning-specific threshold information.
Recommended response	No action assuming the maxmemory policy set meets your apps needs. If the maxmemory policy does not persist data as you want, either coordinate a backup cadence or update your maxmemory policy if using the on-demand Redis service.

Connected clients

	info.clients.connected_clients
Description	Number of clients currently connected to the Redis instance. **Use**: Redis does not close client connections. They remain open until closed explicitly by the client or another script. Once the `connected_clients` reaches `maxclients`, Redis stops accepting new connections and begins producing `ERR max number of clients reached` errors. **Origin**: Doppler/Firehose **Type**: number **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: App-specific. When connected clients reaches max clients, no more clients can connect. This alert must be at the level where it can tell you that your app has scaled to a certain level and can require action. **Red critical**: App-specific. When connected clients reaches max clients, no more clients can connect. This alert must be at the level where it can tell you that your app has scaled to a certain level and can require action.

info.clients.connected_clients	
Recommended response	Increase max clients for your instance if using the on-demand service, or reduce the number of connected clients.

Blocked clients

info.clients.blocked_clients	
Description	The number of clients currently blocked waiting for a blocking request they have made to the Redis server. Redis provides two types of primitive commands to retrieve items from lists: standard and blocking. This metric concerns the blocking commands. **Standard Commands** The standard commands (LPOP, RPOP, RPOPLPUSH) immediately return an item from a list. If there are no items available the standard pop commands return nil. **Blocking Commands** The blocking commands (BLPOP, BRPOP, BRPOPLPUSH) wait for an empty list to become non-empty. The client connection is blocked until an item is added to the lists it is watching. Only the client that made the blocking request is blocked, and the Redis server continues to serve other clients. The blocking commands each take a `timeout` argument that is the time in seconds the server waits for a list before returning nil. A blocking command with timeout `0` waits forever. Multiple clients can be blocked waiting for the same list. For details of the blocking commands, see: https://redis.io/commands/blpop. **Use**: Blocking commands can be useful to avoid clients regularly polling the server for new data. This metric tells you how many clients are currently blocked due to a blocking command. **Origin**: Doppler/Firehose **Type**: number **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	App-specific. Change from baseline can be more significant than actual value.

info.clients.blocked_clients	
Recommended alert thresholds	**Yellow warning**: The expected range of the `blocked_clients` metric depends on what Redis is being used for: - Many uses have no need for blocking commands and expect `blocked_clients` to always be zero. - If blocking commands are being used to force a recipient client to wait for a required input, a raised `blocked_clients` might suggest a problem with the source clients. - `blocked_clients` might be expected to be high in situations where Redis is being used for infrequent messaging. If `blocked_clients` is expected to be non-zero, warnings could be based on change from baseline. A sudden rise in `blocked_clients` could be caused by source clients failing to provide data required by blocked clients. **Red critical**: There is no `blocked_clients` threshold critical to the function of Redis. However, a problem that is causing `blocked_clients` to rise might often cause a rise in `connected_clients`. `connected_clients` does have a hard upper limit and can be used to trigger alerts.
Recommended response	Analysis could include: - Checking the `connected_clients` metric. `blocked_clients` would often rise in concert with `connected_clients`. - Establishing whether the rise in `blocked_clients` is accompanied by an overall increase in apps connecting to Redis, or by an asymmetry in clients providing and receiving data with blocking commands - Considering whether a change in `blocked_clients` is most likely caused by oversupply of blocking requests or undersupply of data - Considering whether a change in network latency is delaying the data from source clients In general, a rise or change in `blocked_clients` is more likely to suggest a problem in the network or infrastructure, or in the function of client apps, rather than a problem with the Redis service.

Memory fragmentation ratio

info.memory.mem_fragmentation_ratio	
Description	Ratio of the amount of memory allocated to Redis by the OS to the amount of memory that Redis is using **Use**: A memory fragmentation less than one shows that the memory used by Redis is higher than the OS available memory. In other packagings of Redis, large values reflect memory fragmentation. For Redis for Tanzu Application Service, the instances only run Redis, meaning that no other processes are affected by a high fragmentation ratio (e.g., 10 or 11). **Origin**: Doppler/Firehose **Type**: ratio **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: < 1. Less than 1 indicates that the memory used by Redis is higher than the OS available memory which can lead to performance degradations. **Red critical**: Same as warning threshold.
Recommended response	Restart the Redis server to normalize fragmentation ratio.

Instantaneous operations per second

info.stats.instantaneous_ops_per_sec	
Description	The number of commands processed per second by the Redis server. The `instantaneous_ops_per_sec` is calculated as the mean of the recent samples taken by the server. The number of recent samples is hardcoded as 16 in the implementation of Redis. **Use**: The higher the commands processed per second, the better the performance of Redis. This is because Redis is single threaded and the commands are processed in sequence. A higher throughput would thus mean faster response per request which is a direct indicator of higher performance. A drop in the number of commands processed per second as compared to historical norms could be a sign of either low command volume or slow commands blocking the system. Low command volume could be normal, or it could be indicative of problems upstream. **Origin**: Doppler/Firehose **Type**: count **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	Every 30 seconds

info.stats.instantaneous_ops_per_sec	
Recommended alert thresholds	**Yellow warning**: A drop in the count compared to historical norms could be a sign of either low command volume or slow commands blocking the system. Low command volume could be normal, or it could be indicative of problems upstream. Slow commands could be due to a latency issue, a large number of clients being connected to the same instance, memory being swapped out, etc. Thus, the count is possibly a symptom of compromised Redis performance. However, this is not the case when low command volume is expected. **Red critical**: A very low count or a large drop from previous counts might indicate a downturn in performance that should result in an investigation. That is unless the low traffic is expected behavior.

	info.stats.instantaneous_ops_per_sec
Recommended response	A drop in the count can be a symptom of compromised Redis performance. The following are possible responses: 1. **Identify slow commands using the slowlog:** Redis logs all the commands that take more than a specified amount of time in slowlog. By default, this time is set to 20ms and the slowlog is allowed a maximum of 120 commands. For the purposes of slowlog, execution time is the time taken by Redis alone and does not account for time spent in I/O. So it would not log slow commands solely due to network latency. Given that typical commands, including network latency, take about 200ms, a 20ms Redis execution time is 100 times slower. This could be indicative of memory management issues wherein Redis pages have been swapped to disk. To see all the commands with slow Redis execution times, type `slowlog get` in the redis-cli. 2. **Monitor client connections:** Because Redis is single threaded, one process services requests from all clients. As the number of clients grows, the percentage of resource time given to each client decreases and each client spends an increasing time waiting for their share of Redis server time. Monitoring the number of clients can be important because there might be apps creating connections that you did not expect or your app might not be efficiently closing unused connections. The connected clients metrics can be used to monitor this. This can also be viewed from the redis-cli using the command `info clients`. 3. **Limit client connections:** This currently defaults to 10000, but depending on the app, you might want to limit this further. To do this, run `CONFIG SET maxclients NUMBER-OF-CONNECTIONS` in the redis-cli. You can configure this for On-Demand service instances in Tanzu Operations Manager. Connections that exceed the limit are rejected and closed immediately. It is important to set `maxclients` to limit the number of unintended client connections. Set `maxclients` to 110% to 150% of your expected peak number of connections. In addition, because an error message is returned for failed connection attempts, the maxclient limit warns you that a significant number of unexpected connections are occurring. This helps maintain optimal Redis performance. 4. **Improve memory management:** Poor memory can cause increased latency in Redis. If your Redis instance is using more memory than is available, the operating system swaps parts of the Redis process from out of physical memory and on to disk. Swapping significantly reduces Redis performance because reads from disk are about five orders or magnitude slower than reads from physical memory.

Keyspace hits / keyspace misses + keyspace hits

info.stats.keyspace_hits / info.stats.keyspace_misses + info.stats.keyspace_hits	
Description	Hit ratio to determine share of keyspace hits that are successful **Use:** A small hit ratio (less than 60%) indicates that many lookup requests are not found in the Redis cache and apps are being forced to revert to slower resources. This might indicate that cached values are expiring too quickly or that a Redis instance has insufficient memory allocation and is deleting volatile keys. **Origin**: Doppler/Firehose **Type**: ratio **Frequency**: 30s (default), 10s (configurable minimum)
Recommended measurement	App-specific
Recommended alert thresholds	**Yellow warning**: App-specific. In general depending how an app is using the cache, an expected hit ratio value can vary between 60% to 99% . Also, the same hit ratio values can mean different things for different apps. Every time an app gets a cache miss, it will probably go to and fetch the data from a slower resource. This cache miss cost can be different per app. The app developers might be able to provide a threshold that is meaningful for the app and its performance **Red critical**: App-specific. See the warning threshold above.
Recommended response	App-specific. See the warning threshold above. Work with app developers to understand the performance and cache configuration required for their apps.

BOSH Health Monitor metrics

The BOSH layer that underlies Tanzu Operations Manager generates `healthmonitor` metrics for all VMs in the deployment. As of Tanzu Operations Manager v2.0, these metrics are in the Loggregator Firehose by default. For more information, see BOSH System Metrics Available in Loggregator Firehose in *VMware Tanzu Application Service for VMs Release Notes*.

Other Redis metrics

Redis also exposes the following metrics. for more information, see the Redis documentation.

- `arch_bits`
- `uptime_in_seconds`
- `uptime_in_days`
- `hz`
- `lru_clock`
- `client_longest_output_list`
- `client_biggest_input_buf`
- `used_memory_rss`
- `used_memory_peak`

- `used_memory_lua`
- `loading`
- `rdb_bgsave_in_progress`
- `rdb_last_save_time`
- `rdb_last_bgsave_time_sec`
- `rdb_current_bgsave_time_sec`
- `aof_rewrite_in_progress`
- `aof_rewrite_scheduled`
- `aof_last_rewrite_time_sec`
- `aof_current_rewrite_time_sec`
- `total_connections_received`
- `total_commands_processed`
- `instantaneous_ops_per_sec`
- `total_net_input_bytes`
- `total_net_output_bytes`
- `instantaneous_input_kbps`
- `instantaneous_output_kbps`
- `rejected_connections`
- `sync_full`
- `sync_partial_ok`
- `sync_partial_err`
- `expired_keys`
- `evicted_keys`
- `keyspace_hits`
- `keyspace_misses`
- `pubsub_channels`
- `pubsub_patterns`
- `latest_fork_usec`
- `migrate_cached_sockets`
- `repl_backlog_active`
- `repl_backlog_size`
- `repl_backlog_first_byte_offset`
- `repl_backlog_histlen`
- `used_cpu_sys`

- `used_cpu_user`
- `used_cpu_sys_children`
- `used_cpu_user_children`
- `rdb_last_bgsave_status`
- `aof_last_bgrewrite_status`
- `aof_last_write_status`

Rotating certificates for Redis for Tanzu Application Service

This topic tells you how to access BOSH CredHub, check expiration dates, and rotate certificates when using Redis for VMware Tanzu Application Service.

To rotate the Services TLS CA and its leaf certificates, use one of the following procedures:

- **Tanzu Operations Manager v3.0:** See Rotate the Services TLS CA and its leaf certificates.
- **Tanzu Operations Manager v2.10:** See Rotate the Services TLS CA and its leaf certificates.

Tanzu Operations Manager v2.9 and later is compatible with CredHub Maestro. Redis for Tanzu Application Service v2.4 and later is compatible with CredHub Maestro.

Enabling service-gateway access for Redis

This topic tells you how to enable service-gateway access for Redis for VMware Tanzu Application Service.

Service-gateway access enables a Redis for VMware Tanzu Application Service on-demand service instance to connect to external components that are not on the same foundation as the service instance.

For a more detailed overview, see Service-Gateway access.

To enable service-gateway access for an on-demand offering:

- Enable TCP routing by using the <%= vars.app_runtime_abbr %> tile
- Configure the firewall to allow incoming traffic to the TCP router
- Configure the load balancer in the IaaS to redirect traffic to the TCP router
- Create a DNS record that maps to the load balancer
- Configure a service-gateway enabled plan
- Deactivate service-gateway access
- Developer workflow

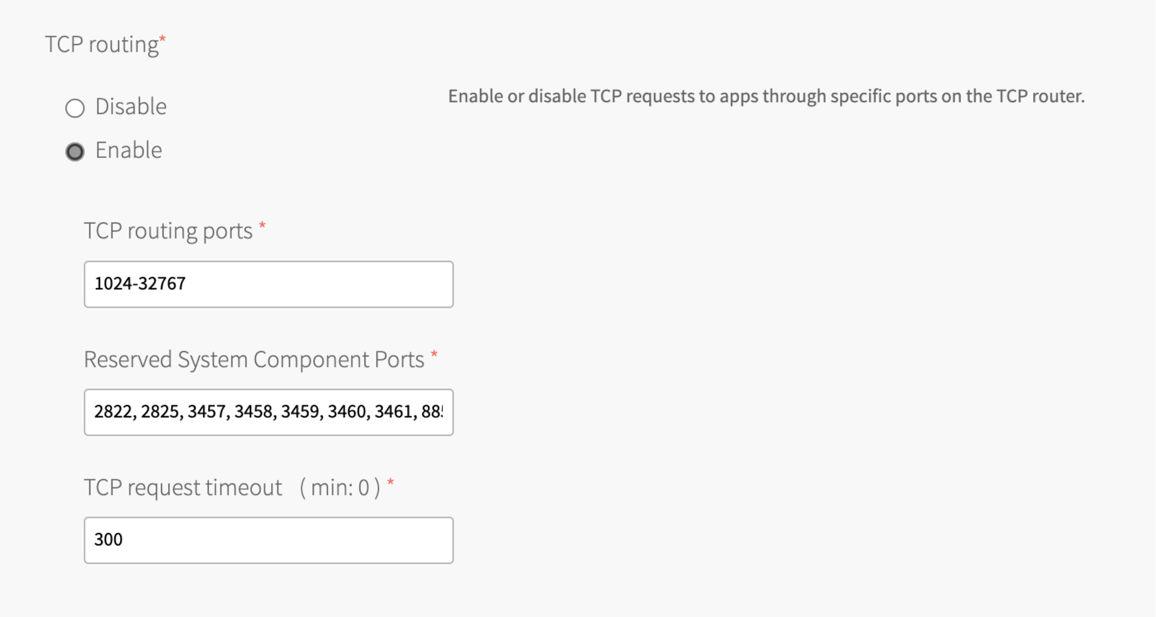
Enable TCP routing by using the TAS for VMs tile

TCP routing is deactivated by default.

To enable TCP routing:

1. Go to the **Networking** tab on the sidebar of the TAS for VMs tile.

2. Under **TCP routing**, select **Enable**.
3. For TCP routing ports, enter one or more ports to which the load balancer forwards requests. For example, `1024` for a single port or `1024-1123` for a range of ports.



TCP routing*

○ Disable

● Enable

Enable or disable TCP requests to apps through specific ports on the TCP router.

TCP routing ports *

1024-32767

Reserved System Component Ports *

2822, 2825, 3457, 3458, 3459, 3460, 3461, 88!

TCP request timeout (min: 0) *

300

4. The ports you assign must not overlap with any other application or tile.
5. Apply your changes in Tanzu Operations Manager for the TAS for VMs tile to create the TCP router.

Configure the firewall to allow incoming traffic to the TCP router

To configure the firewall:

1. Allow incoming traffic to the TCP router VM created in Enable TCP Routing using the TAS for VMs Tile earlier. For information about how to do so, see the documentation for your IaaS.

Configure the load balancer in the IaaS to redirect traffic to the TCP router

To configure the load balancer:

1. Use the IaaS console and the CID that you recorded earlier to find the VM that runs the TCP router.
2. Create an external TCP load balancer that points to the VM running the TCP router.
3. Configure a distinct external port range that does not overlap with any of the following:
   - The port range configured for service-gateway access for other service tiles, such as VMware Tanzu SQL with MySQL for VMs.
   - The TCP networking port or port range that you configured in Enable TCP Routing using the TAS for VMs Tile earlier.

For example, if your TCP routing port range is `1024-1123`, then your load balancer port range for service gateway must not overlap `1024-1123`.

Each Redis for Tanzu Application Service service instance using service-gateway access requires a unique port. Ensure that the port range you configured has enough capacity to accommodate all the service instances that you need. The start port and the end port are both inclusive.

4. Record this port range.

Create a DNS record that maps to the load balancer

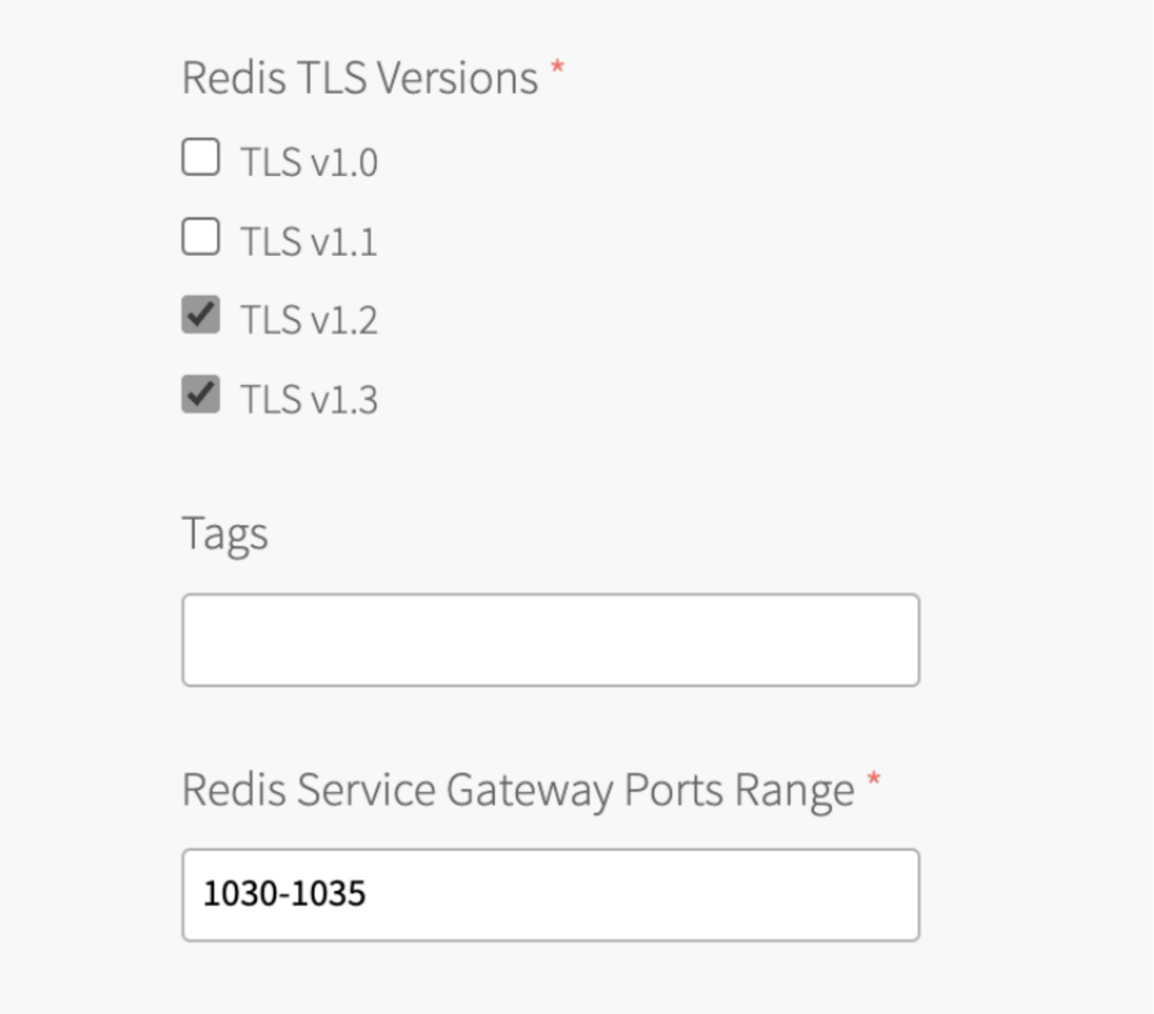
To create a DNS record and prepare to map it:

1. Following the documentation for your IaaS, create a new DNS record of type A that maps to the external IP address of the load balancer created in Configure the Load Balancer in the IaaS to Redirect Traffic to the TCP Router earlier.
2. Record the domain used for this DNS record.

Configure a service-gateway enabled plan

To configure a service-gateway-enabled plan:

1. Go to the **On-Demand Service Settings** tab in the Redis tile sidebar.
2. Scroll down to the **Redis Service Gateway Ports Range** section and enter the port range you want. This range must not overlap with TCP Router ports range or the ports for any other tile.



Redis TLS Versions *

☐ TLS v1.0

☐ TLS v1.1

☑ TLS v1.2

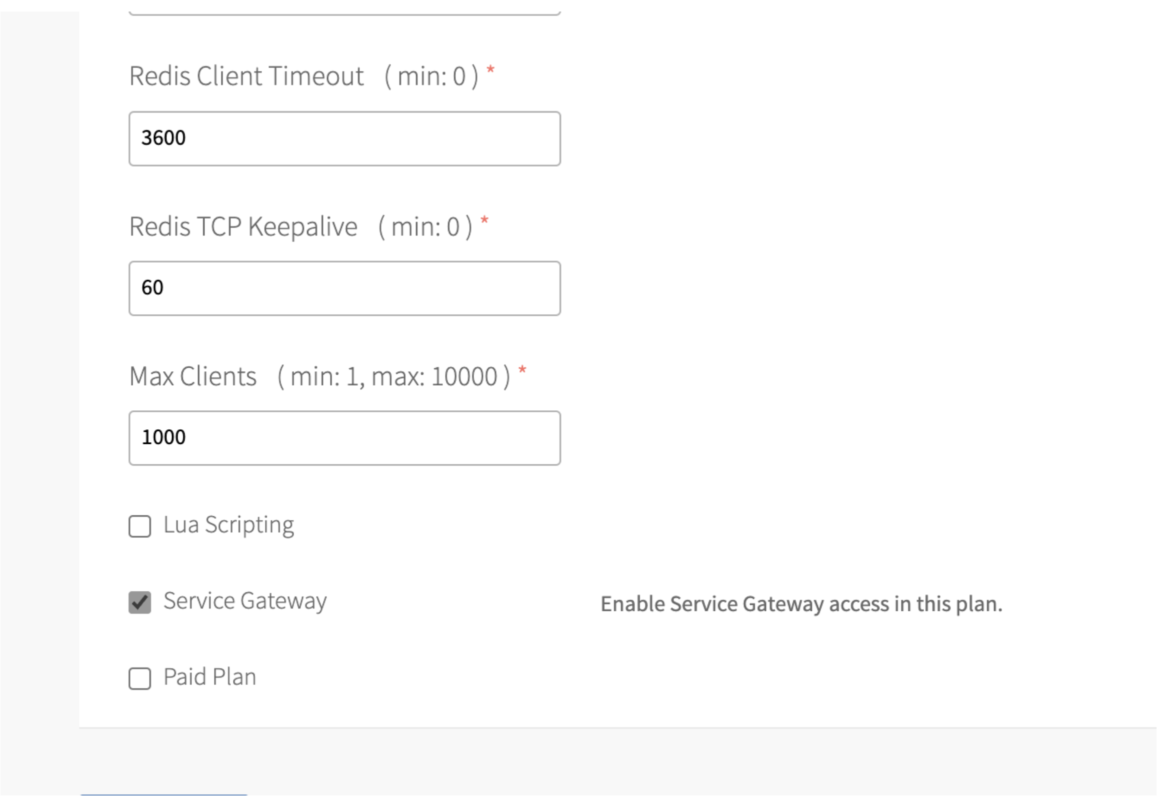
☑ TLS v1.3

Tags

Redis Service Gateway Ports Range *

1030-1035

3. Save your changes.
4. Create a new plan for the Redis tile in the **On-Demand Plans** tab.
5. In the new plan, select the **Service Gateway** check box.



Redis Client Timeout (min: 0) *

3600

Redis TCP Keepalive (min: 0) *

60

Max Clients (min: 1, max: 10000) *

1000

□ Lua Scripting

☑ Service Gateway — Enable Service Gateway access in this plan.

□ Paid Plan

6. Save your changes.
7. Go back to **Tanzu Operations Manager Installation Dashboard > Review Pending Changes**.
8. Click **Apply Changes** to apply the changes to the Redis for Tanzu Application Service tile.

> **Caution**
>
> If you already have service instances using service-gateway, any changes to this range must include ports that are already assigned to these service instances. If the port range does not contain the ports already assigned to service instances, upgrading these service instances fails. For example, if service-gateway access has the port range `1000-1005`, and there are service instances that correspond to ports `1000`, `1001`, and `1002`, then the new port range must have ports `1000`, `1001`, and `1002`.

Deactivate service-gateway access

> **Important**
>
> If service-gateway access is deactivated and then re-enabled, app developers must create new service keys to obtain a new set of credentials for service-gateway access.

To deactivate service-gateway access:

1. Go to the service plan that you want to deactivate service-gateway access for and clear the **Service-Gateway** check box. VMware recommends that you change the name or the description of the plan to indicate that service-gateway access is deactivated for that plan.
2. Go back to **Tanzu Operations Manager Installation Dashboard > Review Pending Changes**.
3. Click **Apply Changes** to apply the changes to the Redis for Tanzu Application Service tile.

Developer workflow

For instructions for app developers, see Create a Service Instance with Service-Gateway access.

Smoke tests for Redis for VMware Tanzu Application Service

This topic tells you about the set of smoke tests that Redis for VMware Tanzu Application Service runs during installation to confirm system health.

The tests run in the org `system` and in the space `tanzu-services`. The tests run as an app instance with a restrictive App Security Group (ASG).

Smoke test steps

The smoke tests perform the following tasks for each available service plan:

1. Targets the org `system` and space `tanzu-services` (creating them if they do not exist).
2. Deploys an instance of the CF Redis Example App to this space.
3. Creates a Redis instance and binds it to the CF Redis Example App.
4. Creates a service key to retrieve the Redis instance IP address.
5. Creates a restrictive security group, `redis-smoke-tests-sg`, and binds it to the space.
6. Checks that the CF Redis Example App can write to and read from the Redis instance.

Security groups

Smoke tests create a new App security group for the CF Redis Example App (`redis-smoke-tests-sg`) and delete it after the tests finish. This security group has the following rules:

```
[
    {
      "protocol": "tcp",
      "destination": "<broker IP address>",
      "ports": "32768-61000" // Ephemeral port range (assigned to shared-vm instances)
    }
]
```

This allows outbound traffic from the test app to the Redis shared-VM service instances.

Smoke test resilience

Smoke tests can fail due to reasons outside of the Redis deployment. For example, network latency causing timeouts or the Cloud Foundry instance dropping requests. They might also fail because they are being run in the wrong space.

The smoke tests implement a retry policy for commands issued to CF, for two reasons:

- To avoid smoke test failures due to temporary issues such as the ones previously mentioned.
- To ensure that the service instances and bindings created for testing are cleaned up.

Smoke tests retry failed commands against CF. They use a linear back-off with a baseline of 0.2 seconds, for a maximum of 30 attempts per command. Therefore, assuming that the first attempt is at 0s and fails instantly, subsequent retries are at 0.2s, 0.6s, 1.2s and so on until either the command succeeds or the maximum number of attempts is reached.

The linear back-off was selected as a good middle ground between:

- Situations where the system is generally unstable, such as load-balancing issues, where max number of retries are preferred.
- Situations where the system is experiencing a failure that lasts a few seconds, such as restart of a Cloud Foundry VM, where it is preferable to wait before reattempting the command.

Considerations

The retry policy does not guard against a more permanent Cloud Foundry downtime or network connectivity issues. In this case, commands fail after the maximum number of attempts and might leave claimed instances behind. VMware recommends deactivating automatic smoke tests, and manually releasing any claimed instances in case of upgrades or scheduled downtime.

Troubleshooting

If errors occur while the smoke tests are run, they are summarized at the end of the errand log output. Detailed logs can be found where the failure occurs. Here are some common failures:

Error	`Failed to target Cloud Foundry`
Cause	Your deployment is unresponsive.
Solution	Examine the detailed error message in the logs and check the Troubleshooting deployment problems for advice.

Error	`Failed to bind Redis service instance to test app.`
Cause	Your deployment's broker has not been registered.
Solution	Examine the broker-registrar installation step output and troubleshoot any problems.

Error	`protocol not supported: SSL_connect returned=1 errno=0 peeraddr= state=error: no protocols available`
Cause	This issue in the smoke tests is due to the CF app’s Ubuntu version in the smoke test. Ubuntu version 20 has deactivated TLS versions prior to v1.2.

Solution	VMware recommends to not use the TLS version v1 and v1.1 because they are deprecated. If this issue happens during upgrade, deselect TLS v1 and v1.1 from Tiles configuration and apply changes to resolve the issue.

When you encounter an error when running smoke tests, search the log for other instances of the error summary printed at the end of the tests. For example, search for `Failed to target Cloud Foundry`. Then look for `TIP: ...` in the logs next to any error output for further troubleshooting hints.

Troubleshooting Redis for VMware Tanzu Application Service

This topic for operators gives you troubleshooting techniques for Redis for VMware Tanzu Application Service.

> **Important**
>
> Some of the troubleshooting approaches in this topic suggest potentially destructive operations. VMware recommends that you back up both your Tanzu Operations Manager and deployments before attempting such operations. For more information about backing up your setup and exporting your Tanzu Operations Manager installation, see Backing Up Deployments with BBR

Useful debugging commands

Before debugging, gather the following information about your deployment:

- Current version of Redis for Tanzu Application Service, and, if upgrading, the previous version of Redis for Tanzu Application Service
- Current version of Tanzu Operations Manager, and, if upgrading, the previous version of Tanzu Operations Manager

cf CLI commands

See the following table for Cloud Foundry Command Line Interface (cf CLI) commands commonly used while debugging:

To view the...	Command
API endpoint, org, and space	`cf target`
Service offerings available in the targeted org and space	`cf marketplace`
Apps deployed to the targeted org and space	`cf apps`
Service instances deployed to the targeted org and space	`cf services`
GUID for a specific service instance	`cf service SERVICE-INSTANCE --guid`
Service instance or application logs	`cf tail SERVICE-INSTANCE/APP`

BOSH CLI commands

See the following table for BOSH CLI commands commonly used while debugging:

Purpose	Command
View the targeted BOSH Director, version, and CPI	`bosh env`
View the deployments deployed through the targeted BOSH Director	`bosh deployments`
View the VMs for a given deployment	`bosh -d DEPLOYMENT vms`
SSH into a given deployment's VM	`bosh -d DEPLOYMENT ssh VM`

You can obtain general information after you SSH into a broker or service instance as follows:

- To see system logs, go to `/var/vcap/sys/log`.
- To check process health, run `sudo monit summary`.
- To obtain a list of all processes, run `ps aux`.
- To see disk usage, run `df -h`.
- To see memory usage, run `free -m`.

You can obtain information specific to the cf-redis broker as follows:

- For shared-VMs, the redis processes are co-located with the CF-Redis broker. You can check these VMs using `ps aux | grep redis-server`.
- Shared-VM data is stored in `/var/vcap/store/cf-redis-broker/redis-data`.

About the Redis CLI

The redis-cli is a command line tool used to access a Redis server. You can use the redis-cli for create, read, update, and delete (CRUD) actions, and to set configuration values. For more information about the redis-cli, see redis-cli, the Redis command line interface in the Redis documentation.

To access the redis-cli, do the following:

1. Follow the instructions in Access the Redis Service to retrieve the password and port number for the service instance.
2. SSH into the service instance.
3. Connect to the Redis server and enter the redis-cli interactive mode by running:

   ```
   LD_LIBRARY_PATH=/var/vcap/packages/openssl/lib/ /var/vcap/packages/redis/bin/redis-cli -p PORT -a PASSWORD
   ```

   Where:

   - `PORT` is the port number retrieved in step one.
   - `PASSWORD` is the password retrieved in step one.

   For more information about the redis-cli interactive mode, see [Interactive Mode](https://redis.io/topics/rediscli#interactive-mode) in the Redis documentation.

Troubleshooting errors

Start here if you are responding to a specific error or error messages.

Common services errors

The following errors occur in multiple services:

- Failed installation
- Cannot create or delete service instances
- Broker request timeouts
- Instance does not exist
- Cannot bind to or unbind from service instances
- Cannot connect to a service instance
- Upgrade all service instances errand fails
- Missing logs and metrics

Failed installation	
Symptom	Redis for Tanzu Application Service fails to install.
Cause	Reasons for a failed installation include: • Certificate issues: The on-demand broker (ODB) requires valid certificates. • Deploy fails. There are multiple possible causes. • Networking problems: ◦ Cloud Foundry cannot reach the Redis for Tanzu Application Service broker ◦ Cloud Foundry cannot reach the service instances ◦ The service network cannot access the BOSH Director • The register broker errand fails. • The smoke test errand fails. • Resource sizing issues: These occur when the resource sizes selected for a plan are lower than Redis for Tanzu Application Service requires to function. • Other service-specific issues.
Solution	To troubleshoot: • Certificate issues: Ensure that your certificates are valid and generate new ones if necessary. To generate new certificates, contact Support. • Deploy fails: View the logs using Tanzu Operations Manager to find out why the deployment is failing. • Networking problems: For how to troubleshoot, see Networking problems. • Register broker errand fails: For how to troubleshoot, see Register broker errand. • Resource sizing issues: Verify your resource configuration in Tanzu Operations Manager and ensure that the configuration matches that recommended by the service.

Cannot create or delete service instances

Symptom

If developers report errors such as:

```
Instance provisioning failed: There was a problem completing
your request. Please contact your operations team providing
the following information: service: redis-acceptance, servic
e-instance-guid: ae9e232c-0bd5-4684-af27-1b08b0c70089, broke
r-request-id: 63da3a35-24aa-4183-aec6-db8294506bac, task-id:
442, operation: create
```

Cause

Reasons include:

- Problems with the deployment manifest
- Authentication errors
- Network errors
- Quota errors

Solution

To troubleshoot:

1. If the BOSH error shows a problem with the deployment manifest, open the manifest in a text editor to inspect it.
2. To continue troubleshooting, Log in to BOSH and target the Redis for Tanzu Application Service instance using the instructions on parsing a Cloud Foundry error message.
3. Retrieve the BOSH task ID from the error message and run:

   ```
   bosh task TASK-ID
   ```

4. See Access the broker logs and use the `broker-request-id` from the error message to search the logs for more information. Check for:
   - Authentication errors
   - Network errors
   - Quota errors

Broker request timeouts

Symptom

If developers report errors such as:

```
Server error, status code: 504, error code: 10001, message:
The request to the service broker timed out: https://BROKER-
URL/v2/service_instances/e34046d3-2379-40d0-a318-d54fc7a5b13
f/service_bindings/aa635a3b-ef6d-41c3-a23f-55752f3f651b
```

Cause

Cloud Foundry might not be connected to the service broker, or there might be a large number of queued tasks.

Broker request timeouts

Solution

To troubleshoot:

1. Confirm that Cloud Foundry (CF) is connected to the service broker.
2. Verify the BOSH queue size:
   1. Log in to BOSH as an admin.
   2. Run

      ```
      bosh tasks
      ```

   If there are a large number of queued tasks, the system might be under too much load. BOSH is configured with two workers and one status worker, which might not be enough for the level of load.
3. If the task queue is long, advise app developers to try again after the system is under less load.

Instance does not exist

Symptom

If developers report errors such as:

```
Server error, status code: 502, error code: 10001, message:
Service broker error: instance does not exist
```

Cause

The instance might have been deleted.

Solution

To troubleshoot:

1. Confirm that the Redis for Tanzu Application Service instance exists in BOSH and obtain the GUID CF by running:

   ```
   cf service MY-INSTANCE --guid
   ```

2. Using the --guid flag you obtained, run:

   ```
   bosh -d service-instance_GUID vms
   ```

If the BOSH deployment is not found, it was deleted from BOSH. Contact VMware Tanzu Support for help.

Cannot bind to or unbind from service instances

Symptom

If developers report errors such as:

```
Server error, status code: 502, error code: 10001, message:
Service broker error: There was a problem completing your re
quest. Please contact your operations team providing the fol
lowing information: service: example-service, service-instan
ce-guid: 8d69de6c-88c6-4283-b8bc-1c46103714e2, broker-reques
t-id: 15f4f87e-200a-4b1a-b76c-1c4b6597c2e1, operation: bind
```

Cannot bind to or unbind from service instances	
Cause	This might be due to authentication or network errors.
Solution	To find out the issue with the binding: 1. Access the service broker logs. 2. Search the logs for the `broker-request-id` string listed in the error message above. 3. Check for: ◦ Authentication errors ◦ Network errors 4. Contact VMware Tanzu Support for help if you are unable to resolve the problem.

Cannot connect to a service instance	
Symptom	Developers report that their app cannot use service instances that they created and bound.
Cause	The error might originate from the service or be network related.
Solution	To solve this issue, ask the user to send application logs that show the connection error. If the error originates from the service, then follow Redis for Tanzu Application Service-specific instructions. If the issue appears to be network-related, then: 1. Verify that application security groups are configured correctly. Configured access for the service network that the tile is deployed to. 2. Ensure that the network the TAS for VMs tile is deployed to has network access to the service network. You can find the network definition for this service network in the BOSH Director tile. 3. In Tanzu Operations Manager go into the service tile and see the service network that is configured in the networks tab. 4. In Tanzu Operations Manager go into the TAS for VMs tile and see the network it is assigned to. Ensure that these networks can access each other.

Upgrade all service instances errand fails	
Symptom	The `upgrade-all-service-instances` errand fails.
Cause	There might be a problem with a particular instance.
Solution	To troubleshoot: 1. Look at the errand output in the Tanzu Operations Manager log. 2. If an instance has failed to upgrade, debug and fix it before running the errand again to prevent any failure issues from spreading to other on-demand instances. 3. After the Tanzu Operations Manager log no longer lists the deployment as `failing`, re-run the errand to upgrade the rest of the instances.

Missing logs and metrics	
Symptom	No logs are being emitted by the on-demand broker.
Cause	Syslog might not be configured correctly, or you might have network access issues.
Solution	To troubleshoot: 1. Ensure that you have configured syslog for the tile. 2. Verify that your syslog forwarding address is correct in Tanzu Operations Manager. 3. Ensure that you have network connectivity between the networks that the tile is using and the syslog destination. If the destination is external, use the public ip VM extension feature available in your Tanzu Operations Manager tile configuration settings. 4. Verify that Loggregator is emitting metrics: 1. Install the `cf log-cache` plug-in. For instructions, see the Log Cache CLI Plugin GitHub repository. 2. Find logs from your service instance by running: `cf tail -f SERVICE_INSTANCE` 3. If no metrics appear within five minutes, verify that the broker network has access to the Loggregator system on all required ports. 5. If you are unable to resolve the issue, contact Support.

Redis for Tanzu Application Service-specific errors

The following troubleshooting errors are specific to Redis for Tanzu Application Service:

- AOF file corrupted, cannot start Redis instance
- Saving error
- Failed backup
- Orphaned instances: BOSH Director cannot see your instances
- Orphaned instances: Pivotal Platform cannot see your instances
- Failed to set credentials in runtime CredHub
- Service outage after deactivating TLS

AOF File Corrupted, Cannot Start Redis Instance	
Symptom	One or more VMs might fail to start the Redis server during pre-start with the error message logged in syslog: `` [ErrorLog-TimeStamp] # Bad file format reading the append only file: make a backup of your AOF file, then use ./redis-check-aof --fix `filename` `` For more information about remote syslog forwarding, see Configure syslog forwarding.

AOF File Corrupted, Cannot Start Redis Instance

Cause

In cases of hard crashes, for example, due to power loss or VM termination without running drain scripts, your AOF file might become corrupted. The error log printed out by Redis provides a clear means of recovery.

Solution

Solution for shared-VM instances:

1. SSH into your `cf-redis-broker` instance.
2. Navigate to the directory where your AOF file is stored. This is usually `/var/vcap/store/cf-redis-broker/redis-data/SERVICE-INSTANCE-GUID/`, where `SERVICE-INSTANCE-GUID` is the GUID for the affected service instance.
3. Run the following command:

   ```
   /var/vcap/packages/redis/redis-check-aof appendonly.aof --fix
   ```

4. To SSH out of the `cf-redis-broker` instance and restart, run the following command:

   ```
   bosh restart INSTANCE-GROUP/INSTANCE-ID
   ```

Solution for on-demand-VM instances:

1. SSH into your affected service instance.
2. Navigate to the directory where your AOF file is stored. This is usually `/var/vcap/store/redis/`.
3. Run:

   ```
   /var/vcap/packages/redis/redis-check-aof appendonly.aof --fix
   ```

4. SSH out of the service instance and restart it by running:

   ```
   bosh restart INSTANCE-GROUP/INSTANCE-ID
   ```

Saving Error

Symptom

One of the following error messages is logged in syslog:

```
Background saving error
```

```
Failed opening the RDB file dump.rdb (in server root dir /var/vcap/store/redis) for saving: No space left on device
```

For more information about remote syslog forwarding, see Configure syslog forwarding.

Cause

This might be logged when the configured disk size is too small, or if the Redis AOF uses all the disk space.

Saving Error	
Solution	To prevent this error, do the following: 1. Ensure the disk is configured to at least 2.5x the VM memory for the on-demand broker and 3.5x the VM memory for cf-redis-broker. 2. Check if the AOF is using too much disk space by doing the following: 1. BOSH SSH into the affected service instance VM. 2. List the size of each file by running: `cd /var/vcap/store/redis; ls -la`

Failed Backup	
Symptom	The following error message is logged: `Backup has failed. Redis must be running for a backup to run`
Cause	This is logged if a backup is initiated against a Redis server that is down.
Solution	Ensure that the Redis server being backed up is running. To do this, run `bosh restart` against the affected service instance VM.

Orphaned instances: BOSH Director cannot see your instances	
Symptom	When you run `cf curl /v2/service_instances`, some service instances are visible that are not visible to the BOSH Director. These orphaned instances can create issues. For example, they might hold on to a static IP address, causing IP conflicts.
Cause	Orphaned instances can occur in the following situations: - Both TAS for VMs and BOSH maintain state. Orphaned instances can occur if the TAS for VMs state is out of sync with BOSH. For example, the deployments or VMs have been de-provisioned by BOSH but the call to update the TAS for VMs state failed. - If a call to de-provision a service instance was made directly to BOSH rather than through the cf CLI.

Orphaned instances: BOSH Director cannot see your instances	
Solution	You can solve this issue by doing one of the following: • **If this is the first occurrence:** VMware recommends that you purge instances by running: `cf purge-service-instance SERVICE-INSTANCE` . • **If this is a repeated occurrence:** Contact VMware Tanzu Support for further help, and include the following: ∘ A snippet of your `broker.log` around the time of the incident ∘ The deployment manifest of failed instances, hiding private information like passwords ∘ Any recent logs that you can recover from the failed service instance

Orphaned instances: The deployment cannot see your instances	
Symptom	The deployment cannot see your broker or service instances. These instances exist, but cannot receive communication.
Cause	If you run `cf purge-service-instances` while your service instance or broker still exists, your service instance becomes orphaned.

Orphaned instances: The deployment cannot see your instances

Solution

If the deployment lost the details of your instances, but BOSH still has the deployment details, you can solve this issue by backing up the data on your service instance and creating a new service.

To back up your data and create a new service instance:

1. Retrieve your orphaned service instance GUID by running:

   ```
   bosh -d MY-DEPLOYMENT run-errand orphan-deployments
   ```

   Where `MY-DEPLOYMENT` is the name of your deployment.

2. SSH into your orphaned service instance by running:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT ssh VM-NAME/GUID
   ```

   Where:

   - `MY-ENV` is the name of your environment.
   - `MY-DEPLOYMENT` is the name of your deployment.
   - `VM-NAME/GUID` is the name of your service instance and GUID that you obtained in step 1.

3. Create an new RDB file by running:

   ```
   /var/vcap/jobs/redis-backups/bin/backup --snapshot
   ```

   This creates a new RDB file in `/var/vcap/store/redis-backup`.

4. Push the RDB file to your backup location by running:

   ```
   /var/vcap/jobs/service-backup/bin/manual-backup
   ```

   For information about backup locations, see Configuring Automated Service Backups.

5. Create a new service instance with the same configuration of the database you backed up.

6. Retrieve your new service instance GUID, by running:

   ```
   bosh -e MY-ENV -d MY-DEPLOYMENT vms
   ```

   Where:

   - `MY-ENV` is the name of your environment.
   - `MY-DEPLOYMENT` is the name of your deployment.

7. SSH into your new service instance by repeating step 2 above with the GUID that you retrieved in step 6.

8. Create a new directory in new service instance by running:

   ```
   mkdir /var/vcap/store/MY-BACKUPS
   ```

9. Save the RDB file in `/var/vcap/store/MY-BACKUPS/` to transfer it to the new instance. Replace `MY-BACKUPS` with the name of your backups directory.

10. Verify the RDB file has not been corrupted by running:

Orphaned instances: The deployment cannot see your instances	

	`md5sum RDB-FILE`
	Where `RDB-FILE` is the path to your RDB file.
	11. Restore your data by running:
	`sudo /var/vcap/jobs/redis-backups/bin/restore --source RDB RDB-FILE`
	Where `RDB-FILE` is the path to your RDB file.

Failed to set credentials in runtime CredHub	
Symptom	If developers report errors such as: `error: failed to set credentials in credential store: The request includes an unrecognized parameter 'mode'. Please update or remove this parameter and retry your request. error for user: There was a problem completing your request. Please contact your operations team providing the following information: service: p.redis, service-instance-guid: , broker-request-id: , operation: bind`
Cause	Your service instances might not be running the latest version of Redis for Tanzu Application Service. You might experience compatibility issues with CredHub if your service instances are running Redis for Tanzu Application Service v1.14.3 or earlier.
Solution	1. Ensure you have the latest patch version of Redis for Tanzu Application Service installed. For more information about the latest patch, see the Redis for VMware Tanzu Application Service Release Notes. 2. Run the `upgrade-all-service-instances` errand to ensure all service instances are running the latest service offering. For how to run the errand, see Upgrade All Service Instances. **Note** Running this errand causes a short period of downtime.

Service outage after deactivating TLS	
Symptom	After deactivating TLS, apps that require on-demand Redis service instances become unresponsive.

	Service outage after deactivating TLS
Cause	When TLS is first activated, all on-demand service instances are re-created with two ports. Every new or re-created app receives the new credentials. Spring and Steeltoe apps are configured for activated TLS by default, but other languages and frameworks require further configuration. When TLS is deactivated, the TLS port is removed from all on-demand instances. This prevents the apps from connecting to the instance.
Solution	First, consider activating TLS. The compliance body that oversees your apps might require TLS to be activated. Also, switching between activated and deactivated TLS incurs downtime. To activate TLS, follow these steps: 1. In your Tanzu Operations Manager home page, select the **Redis** tile. 2. Navigate to **On-Demand Service Settings.** 3. On the **Enable TLS** section, ensure it is set to **Optional**. 4. Click **Save**. 5. Navigate back to the Tanzu Operations Manager home page and click **Review Pending Changes**. 6. Ensure the Recreate All On-Demand Service Instances errand is enabled under the Redis section and then click **Apply Changes**. To continue with TLS deactivated, follow these steps: 1. Unbind, bind, and re-stage every app that was affected by deactivating TLS. For more information, see Introduction for App Developers. This makes Spring and Steeltoe apps default to non-TLS configuration. 2. Manually configure any other relevant languages and frameworks to work with TLS deactivated.

Troubleshooting components

This section provides guidance on checking for, and fixing, issues in cf-redis and on-demand service components.

BOSH problems

Large BOSH queue

On-demand service brokers add tasks to the BOSH request queue, which can back up and cause delay under heavy loads. An app developer who requests a new Redis for Tanzu Application Service instance sees `create in progress` in the Cloud Foundry Command Line Interface (cf CLI) until BOSH processes the queued request.

Tanzu Operations Manager deploys two BOSH workers to process its queue.

Configuration

Service instances in failing state

The VM or disk type that you configured in the plan page of the tile in Tanzu Operations Manager might not be large enough for the Redis for Tanzu Application Service service instance to start. See tile-specific guidance on resource requirements.

Authentication

UAA changes

If you rotated any UAA user credentials then you might see authentication issues in the service broker logs.

To resolve this, redeploy the Redis for Tanzu Application Service tile in Tanzu Operations Manager. This provides the broker with the latest configuration.

> **Caution**
>
> You must ensure that any changes to UAA credentials are reflected in the Tanzu Operations Manager `credentials` tab of the VMware Tanzu Application Service for VMs tile.

Networking

Common issues with networking include:

Issue	Solution
Latency when connecting to the Redis for Tanzu Application Service service instance to create or delete a binding.	Try again or improve network performance.
Firewall rules are blocking connections from the Redis for Tanzu Application Service service broker to the service instance.	Open the Redis for Tanzu Application Service tile in Tanzu Operations Manager and verify that the two networks configured in the **Networks** pane allow access to each other.
Firewall rules are blocking connections from the service network to the BOSH Director network.	Ensure that service instances can access the Director so that the BOSH agents can report in.
Apps cannot access the service network.	Configure Cloud Foundry application security groups to allow runtime access to the service network.
Problems accessing BOSH's UAA or the BOSH director.	Follow network troubleshooting and verify that the BOSH Director is online.

Validate service broker connectivity to service instances

To validate connectivity:

1. View the BOSH deployment name for your service broker by running:

   ```
   bosh deployments
   ```

2. SSH into the Redis for Tanzu Application Service service broker by running:

```
bosh -d DEPLOYMENT-NAME ssh
```

3. If no BOSH `task-id` appears in the error message, look in the broker log using the `broker-request-id` from the task.

Validate app access to a service instance

Use the `cf ssh` command to access to the app container, then connect to the Redis for Tanzu Application Service service instance using the binding included in the `VCAP_SERVICES` environment variable.

Quotas

Plan quota issues

If developers report errors such as:

```
Message: Service broker error: The quota for this service plan has been excee
ded.
Please contact your Operator for help.
```

1. Verify your current plan quota.
2. Increase the plan quota.
3. Log in to Tanzu Operations Manager.
4. Reconfigure the quota on the plan page.
5. Deploy the tile.
6. Find who is using the plan quota and take the appropriate action.

Global quota issues

If developers report errors such as:

```
Message: Service broker error: The quota for this service has been exceeded.
Please contact your Operator for help.
```

1. Verify your current global quota.
2. Increase the global quota.
3. Log in to Tanzu Operations Manager.
4. Reconfigure the quota on the on-demand settings page.
5. Deploy the tile.
6. Find out who is using the quota and take the appropriate action.

Failing jobs and unhealthy instances

To find out if there is an issue with the Redis for Tanzu Application Service deployment:

1. Inspect the VMs by running:

   ```
   bosh -d service-instance_GUID vms --vitals
   ```

2. For additional information, run:

   ```
   bosh -d service-instance_GUID instances --ps --vitals
   ```

If the VM is failing, follow the service-specific information. Any unadvised corrective actions (such as running BOSH `restart` on a VM) can cause issues in the service instance.

Techniques for troubleshooting

This section contains instructions on:

- Interacting with the on-demand service broker
- Interacting with on-demand service instance BOSH deployments
- Performing general maintenance and housekeeping tasks

Parse a Cloud Foundry (CF) error message

Failed operations (create, update, bind, unbind, delete) cause an error message. You can retrieve the error message later by running the cf CLI command `cf service INSTANCE-NAME`.

```
$ cf service myservice

Service instance: myservice
Service: super-db
Bound apps:
Tags:
Plan: dedicated-vm
Description: Dedicated Instance
Documentation url:
Dashboard:

Last Operation
Status: create failed
Message: Instance provisioning failed: There was a problem completing your re
quest.
     Please contact your operations team providing the following information:
     service: redis-acceptance,
     service-instance-guid: ae9e232c-0bd5-4684-af27-1b08b0c70089,
     broker-request-id: 63da3a35-24aa-4183-aec6-db8294506bac,
     task-id: 442,
     operation: create
Started: 2017-03-13T10:16:55Z
Updated: 2017-03-13T10:17:58Z
```

Use the information in the `Message` field to debug further. Provide this information to Support when filing a ticket.

The `task-id` field maps to the BOSH task ID. For more information about a failed BOSH task, use the `bosh task TASK-ID`.

The `broker-request-guid` maps to the portion of the On-Demand Service Broker log containing the failed step. Access the broker log through your syslog aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

Access broker and instance logs and VMs

Before following these procedures, log in to the cf CLI and the BOSH CLI.

Access broker logs and VMs

You can access logs using Tanzu Operations Manager by clicking on the **Logs** tab in the tile and downloading the broker logs.

To access logs using the BOSH CLI:

1. To identify the on-demand broker (ODB) deployment run:

   ```
   bosh deployments
   ```

2. To view VMs in the deployment run:

   ```
   bosh -d DEPLOYMENT-NAME instances
   ```

3. To SSH onto the VM run:

   ```
   bosh -d DEPLOYMENT-NAME ssh
   ```

4. To Download the broker logs run:

   ```
   bosh -d DEPLOYMENT-NAME logs
   ```

The archive generated by BOSH includes the following logs:

Log Name	Description
broker.stdout.log	Requests to the on-demand broker and the actions the broker performs while orchestrating the request (e.g. generating a manifest and calling BOSH). Start here when troubleshooting.
bpm.log	Control script logs for starting and stopping the on-demand broker.
post-start.stderr.log	Errors that occur during post-start verification.
post-start.stdout.log	Post-start verification.
drain.stderr.log	Errors that occur while running the drain script.

Access service instance logs and VMs

1. To target an individual service instance deployment, retrieve the GUID of your service instance with the following cf CLI command:

   ```
   cf service MY-SERVICE --guid
   ```

2. To view VMs in the deployment, run:

   ```
   bosh -d service-instance_GUID instances
   ```

3. To SSH into a VM, run:

   ```
   bosh -d service-instance_GUID ssh
   ```

4. To download the instance logs, run:

   ```
   bosh -d service-instance_GUID logs
   ```

Run service broker errands to manage brokers and instances

From the BOSH CLI, you can run service broker errands that manage the service brokers and perform mass operations on the service instances that the brokers created. These service broker errands include:

- `register-broker` registers a broker with the Cloud Controller and lists it in the Marketplace.
- `deregister-broker` deregisters a broker with the Cloud Controller and removes it from the Marketplace.
- `upgrade-all-service-instances` upgrades existing instances of a service to its latest installed version.
- `delete-all-service-instances` deletes all instances of service.
- `orphan-deployments` detects "orphan" instances that are running on BOSH but not registered with the Cloud Controller.

To run an errand:

```
bosh -d DEPLOYMENT-NAME run-errand ERRAND-NAME
```

For example:

```
bosh -d my-deployment run-errand deregister-broker
```

Register broker

The `register-broker` errand:

- Registers the service broker with Cloud Controller.
- Activates service access for any plans that are enabled on the tile.
- Deactivates service access for any plans that are deactivated on the tile.

- Does nothing for any plans that are set to manual on the tile.

Run this errand whenever the broker is re-deployed with new catalog metadata to update the Marketplace.

Plans with deactivated service access are only visible to admin Cloud Foundry users. Non-admin Cloud Foundry users, including Org Managers and Space Managers, cannot see these plans.

Deregister broker

This errand deregisters a broker from Cloud Foundry.

The errand:

- Deletes the service broker from Cloud Controller
- Fails if there are any service instances, with or without bindings

Use the Delete All Service Instances errand to delete any existing service instances.

To run the errand:

```
bosh -d DEPLOYMENT-NAME run-errand deregister-broker
```

Upgrade all service instances

The `upgrade-all-service-instances` errand:

- Collects all the service instances that the on-demand broker has registered.
- Issues an upgrade command and deploys the a new manifest to the on-demand broker for each service instance.
- Adds to a retry list any instances that have ongoing BOSH tasks at the time of upgrade.
- Retries any instances in the retry list until all instances are upgraded.

When you make changes to the plan configuration, the errand upgrades all the Redis for Tanzu Application Service service instances to the latest version of the plan.

If any instance fails to upgrade, the errand fails immediately. This prevents systemic problems from spreading to the rest of your service instances.

Delete all service instances

This errand uses the Cloud Controller API to delete all instances of your broker service offering in every Cloud Foundry org and space. It deletes only instances the Cloud Controller knows about. It does not delete orphan BOSH deployments.

> **Important**
>
> Orphan BOSH deployments do not correspond to a known service instance. While rare, orphan deployments can occur. Use the `orphan-deployments` errand to identify them.

The `delete-all-service-instances`:

1. Unbinds all apps from the service instances.
2. Deletes all service instances sequentially. Each service instance deletion includes:
    1. Running any pre-delete errands
    2. Deleting the BOSH deployment of the service instance
    3. Removing any ODB-managed secrets from BOSH CredHub
    4. Checking for instance deletion failure, which causes the errand to failfailing immediately
3. Determines whether any instances were created while the errand was running. If new instances are detected, the errand returns an error. In this case, VMware recommends running the errand again.

> ⚠ **Caution**
>
> Use extreme caution when running this errand. Use it *only* when you want to destroy all of the on-demand service instances in an environment.

To run the errand:

```
bosh -d service-instance_GUID delete-deployment
```

Detect orphaned service instances

A service instance is defined as "orphaned" when the BOSH deployment for the instance is still running, but the service is no longer registered in Cloud Foundry.

The `orphan-deployments` errand collates a list of service deployments that have no matching service instances in Cloud Foundry and return the list to the operator. It is then up to the operator to remove the orphaned BOSH deployments.

To run the errand:

```
bosh -d DEPLOYMENT-NAME run-errand orphan-deployments
```

If orphan deployments exist---The errand script does the following:

- Exit with exit code 10
- Output a list of deployment names under a `[stdout]` header
- Provide a detailed error message under a `[stderr]` header

For example:

```
[stdout]
[{"deployment\_name":"service-instance\_80e3c5a7-80be-49f0-8512-44840f3c4d1
b"}]

[stderr]
```

```
Orphan BOSH deployments detected with no corresponding service instance in Cl
oud Foundry. Before deleting any deployment it is recommended to verify the s
ervice instance no longer exists in Cloud Foundry and any data is safe to del
ete.

Errand 'orphan-deployments' completed with error (exit code 10)
```

These details are also available through the BOSH `/tasks/` API endpoint for use in scripting:

```
$ curl 'https://bosh-user:bosh-password@bosh-url:25555/tasks/task-id/output?t
ype=result' | jq .
{
  "exit_code": 10,
  "stdout": "[{"deployment_name":"service-instance_80e3c5a7-80be-49f0-8512-44
840f3c4d1b"}]\n",
  "stderr": "Orphan BOSH deployments detected with no corresponding service i
nstance in Cloud Foundry. Before deleting any deployment it is recommended to
verify the service instance no longer exists in Cloud Foundry and any data is
safe to delete.\n",
  "logs": {
    "blobstore_id": "d830c4bf-8086-4bc2-8c1d-54d3a3c6d88d"
  }
}
```

If no orphan deployments exist---The errand script:

- Exit with exit code 0
- Stdout is an empty list of deployments
- Stderr is `None`

```
[stdout]
[]

[stderr]
None

Errand 'orphan-deployments' completed successfully (exit code 0)
```

If the errand encounters an error during running---The errand script does the following:

- Exit with exit 1
- Stdout is empty
- Any error messages are under stderr

To clean up orphaned instances, run the following command on each instance:

⚠ **Caution**

Running this command might leave IaaS resources in an unusable state.

```
bosh delete-deployment service-instance_SERVICE-INSTANCE-GUID
```

Get admin credentials for a service instance

To retrieve the admin credentials for a service instance from BOSH CredHub:

1. Use the cf CLI to find the GUID associated with the service instance for which you want to retrieve credentials by running:

   ```
   cf service SERVICE-INSTANCE-NAME --guid
   ```

   For example:

   ```
   $ cf service my-service-instance --guid

   12345678-90ab-cdef-1234-567890abcdef
   ```

   If you do not know the name of the service instance, you can list service instances in the space with `cf services`.

2. Follow the steps in Gather Credential and IP Address information and Log in to the Tanzu Operations Manager VM with SSH of *Advanced Troubleshooting with the BOSH CLI* to SSH into the Tanzu Operations Manager VM.

3. From the Tanzu Operations Manager VM, log in to your BOSH Director with the BOSH CLI. See Authenticate with the BOSH Director VM in *Advanced Troubleshooting with the BOSH CLI*.

4. Find the values for `BOSH_CLIENT` and `BOSH_CLIENT_SECRET`:

   1. In the Tanzu Operations Manager Installation Dashboard, click the **BOSH Director** tile.
   2. Click the **Credentials** tab.
   3. In the **BOSH Director** section, click the link to the **BOSH Commandline Credentials** .
   4. Record the values for `BOSH_CLIENT` and `BOSH_CLIENT_SECRET`.

5. Set the API target of the CredHub CLI to your BOSH CredHub server by running:

   ```
   credhub api https://BOSH-DIRECTOR-IP:8844 \
         --ca-cert=/var/tempest/workspaces/default/root_ca_certificate
   ```

   Where `BOSH-DIRECTOR-IP` 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
   ```

6. Log in to CredHub by running:

   ```
   credhub login \
       --client-name=BOSH-CLIENT \
   ```

```
    --client-secret=BOSH-CLIENT-SECRET
```

For example:

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

7. Use the CredHub CLI to retrieve the credentials :

    - Retrieve the password for the admin user by running:

      ```
      credhub get -n /p-bosh/service-instance_GUID/admin_password
      ```

      In the output, the password appears under `value`. Record the password.
      For example:

      ```
      $ credhub get \
        -n /p-bosh/service-instance_70d30bb6-7f30-441a-a87c-05a5e4afff2
      6/admin_password

        id: d6e5bd10-3b60-4a1a-9e01-c76da688b847
        name: /p-bosh/service-instance_70d30bb6-7f30-441a-a87c-05a5e4af
      ff26/admin_password
        type: password
        value: UMF2DXsqNPPlCNWMdVMcNv7RC3Wi10
        version_created_at: 2018-04-02T23:16:09Z
      ```

Reinstall a tile

To reinstall a tile in the same environment where it was previously uninstalled:

1. Ensure that the previous tile was correctly uninstalled as follows:
    1. Log in as an admin by running:

       ```
       cf login
       ```

    2. Confirm that the Marketplace does not list Redis for Tanzu Application Service by running:

       ```
       cf m
       ```

    3. Log in to BOSH as an admin by running:

       ```
       bosh log-in
       ```

    4. Display your BOSH deployments to confirm that the output does not show the Redis for Tanzu Application Service deployment by running:

       ```
       bosh deployments
       ```

    5. Run the "delete-all-service-instances" errand to delete every instance of the service.

6. Run the "deregister-broker" errand to delete the service broker.
7. Delete the service broker BOSH deployment by running:

   ```
   bosh delete-deployment BROKER-DEPLOYMENT-NAME
   ```

8. Reinstall the tile.

View resource saturation and scaling

To view usage statistics for any service, run:

1. Run:

   ```
   bosh -d DEPLOYMENT-NAME vms --vitals
   ```

2. To view process-level information, run:

   ```
   bosh -d DEPLOYMENT-NAME instances --ps
   ```

Identify apps using a service instance

To identify which apps are using a specific service instance using the name of the BOSH deployment:

1. Take the deployment name and strip the `service-instance_` leaving you with the GUID.
2. Log in to Cloud Foundry as an admin.
3. Obtain a list of all service bindings by running::

   ```
   cf curl /v2/service_instances/GUID/service_bindings
   ```

4. The output from the curl gives you a list of `resources`, with each item referencing a service binding, which contains the `APP-URL`. To find the name, org, and space for the app, run:
   1. `cf curl APP-URL` and record the app name under `entity.name`.
   2. `cf curl SPACE-URL` to obtain the space, using the `entity.space_url` from the curl. Record the space name under `entity.name`.
   3. `cf curl ORGANIZATION-URL` to obtain the org, using the `entity.organization_url` from the curl. Record the organization name under `entity.name`.

Important

When running `cf curl` ensure that you query all pages, because the responses are limited to a certain number of bindings per page. The default is 50. To find the next page, curl the value under `next_url`.

Monitor the quota saturation and service instance count

Quota saturation and total number of service instances are available through ODB metrics emitted to Loggregator. These are the metric names:

Metric Name	Description
`on-demand-broker/SERVICE-NAME-MARKETPLACE/quota_remaining`	global quota remaining for all instances across all plans
`on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/quota_remaining`	quota remaining for a particular plan
`on-demand-broker/SERVICE-NAME-MARKETPLACE/total_instances`	total instances created across all plans
`on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/total_instances`	total instances created for a given plan

Important

s Quota metrics are not emitted if no quota was set.

VMware Tanzu Support articles

The following are VMware Tanzu Support articles about Redis for Tanzu Application Service:

- Creating an Empty Services Network when using on-demand Service Tiles for Non-On-Demand Usage Only
- Full disk scaling issue
- Tile upgrade issue
- Deploy Fails to Complete
- Instance Alive after Successful De-Provisioning
- Dedicated Instance Fails to Persist to Disk
- Redis error when saving changes after a back to AWS S3: Error: Access Denied for bucket 'pcf-redos-backup-sgp-intra-test'
- For service settings on Redis Tile, the VM options checkbox needs to be checked for GCP Environment
- Removing dedicated-vm Service Instances on CF when already deleted from BOSH
- Migrating from dedicated-vm service plans to on-demand service plans

Introduction to Redis for Tanzu Application Service for app developers

This topic for developers introduces you to Redis for VMware Tanzu Application Service and links to more information.

For instructions on creating, binding to, and deleting an instance of the On-Demand, or Shared-VM plan, see Using Redis for VMware Tanzu Application Service.

Service offerings

Redis for Tanzu Application Service packages Redis for deployment and operability.

There are two service offerings:

- **On-Demand Service**—Provides a dedicated VM running a Redis instance. The operator can configure up to three plans with different configurations, memory sizes, and quotas App developers can provision an instance for any of the On-Demand plans offered and configure certain Redis settings.
- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development purposes only, **do not use the Shared-VM service in production environments**. The Shared-VM instances are pre-provisioned by the operator with a fixed number of instances and memory size. App developers can then use one of these pre-provisioned instances.

For more information about the plans, see:

- On-Demand service offering
- Shared-VM service offering

Related software

These are descriptions of software frequently used with Redis.

Redis for Tanzu Application Service with Spring

Spring Cloud Spring Service Connectors connect to Redis for Tanzu Application Service. For more information, see the Redis section in the Spring Cloud Spring Service Connector documentation.

Spring Cloud Cloud Foundry connectors automatically connect to Redis for Tanzu Application Service. For more information, see the Redis section in the Spring Cloud Cloud Foundry Connector documentation.

To view an example Spring app using Redis as a cache with failover, see the Redis reference architectures GitHub repository.

Redis for Tanzu Application Service with Steeltoe

Steeltoe Cloud Connectors can connect to Redis for Tanzu Application Service. See the Steeltoe Cloud Connectors documentation.

To view examples of Steeltoe apps using Redis as a cache with failover, see the Example Steeltoe app repository in GitHub.

> ⚠ **Caution**
>
> The Steeltoe connector for Redis requires Redis for Tanzu Application Service to support Lua scripting. Check whether the language you are using requires Lua scripting. If it does, contact your operator. By default, Lua scripting is deactivated for Redis for Tanzu Application Service, but an operator can change the setting to enable it by selecting the **Lua Scripting** checkbox in each service plan's **On-demand plan** configuration pane.

Other software

- **Pivotal Dev** is a version of VMware Tanzu Application Service for VMs (TAS for VMs) that is small enough to run on a local machine. For more information, see VMware dev.
- **Sample Ruby code** that uses TAS for VMs is in the CF Redis Example App GitHub repository.
- **Redis** is an open-source in-memory datastore. To learn more about Redis itself, see redis.io.

Use TLS

Follow the steps below to securely bind your apps to a Redis instance with TLS.

Spring and Steeltoe apps use TLS by default when available.

Check availability

You can check if TLS has been enabled on the on-demand Redis service by inspecting the service key. To do so:

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

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

2. Display the service key by running the following command:

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

   This returns a JSON response in this format:

```
{
  "host": "q-s0.redis-instance.ENVIRONMENT-NAME-services-subnet.service-instanc
e-GUID.bosh",
  "password": YOUR-PASSWORD,
  "port": INSECURE-PORT-NUMBER,
  "tls_port": SECURE-PORT-NUMBER
}
```

If you do not see the `tls_port` field, TLS has not been enabled on your Redis service.

Bind new apps with TLS

Follow the steps below to securely bind new apps to a Redis instance.

For new apps, `cf bind-service` exposes both TLS ports and non-secure ports. Custom connectors also make both ports available. To support secure service bindings, you must specify the TLS port in your app code.

Below is an example of manually selecting the TLS port for a `redis_client` in Ruby:

```
require 'redis'
require 'cf-app-utils'

def redis_credentials
  service_name = ENV['service_name'] || "redis"
  if ENV['VCAP_SERVICES']
      all_pivotal_redis_credentials = CF::App::Credentials.find_all_by_all_service_tag
s(['redis', 'pivotal'])
      if all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
          all_pivotal_redis_credentials.first
      else
          redis_service_credentials = CF::App::Credentials.find_by_service_name(servic
e_name)
          redis_service_credentials
      end
  end
end

def redis_client
   @client ||= Redis.new(
     host: redis_credentials.fetch('host'),
     port: redis_credentials.fetch('tls_port'),
     password: redis_credentials.fetch('password'),
     ssl: true,
     timeout: 30
end
```

For Spring apps, use Java CFEnv v1.1.0 or later. See Redis Spring Boot Reference Architecture in GitHub.

For Steeltoe apps, use Steeltoe v2.3.0 or later. See Redis Steeltoe Reference Architecture in GitHub.

Bind existing apps with TLS

For each app using the Redis service with a non-TLS binding:

1. Remove the current binding by running the following command:

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

2. Re-bind to the Redis instance by running the following command:

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

3. Restage the app by running the following command:

   ```
   cf restage-app APP-NAME
   ```

Your app now communicates securely with the Redis on-demand service instance.

Introduction to Redis for Tanzu Application Service for app developers

This topic for developers introduces you to Redis for VMware Tanzu Application Service and links to more information.

For instructions on creating, binding to, and deleting an instance of the On-Demand, or Shared-VM plan, see Using Redis for VMware Tanzu Application Service.

Service offerings

Redis for Tanzu Application Service packages Redis for deployment and operability.

There are two service offerings:

- **On-Demand Service**—Provides a dedicated VM running a Redis instance. The operator can configure up to three plans with different configurations, memory sizes, and quotas App developers can provision an instance for any of the On-Demand plans offered and configure certain Redis settings.

- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development purposes only, **do not use the Shared-VM service in production environments**. The Shared-VM instances are pre-provisioned by the operator with a fixed number of instances and memory size. App developers can then use one of these pre-provisioned instances.

For more information about the plans, see:

- On-Demand service offering
- Shared-VM service offering

Related software

These are descriptions of software frequently used with Redis.

Redis for Tanzu Application Service with Spring

Spring Cloud Spring Service Connectors connect to Redis for Tanzu Application Service. For more information, see the Redis section in the Spring Cloud Spring Service Connector documentation.

Spring Cloud Cloud Foundry connectors automatically connect to Redis for Tanzu Application Service. For more information, see the Redis section in the Spring Cloud Cloud Foundry Connector documentation.

To view an example Spring app using Redis as a cache with failover, see the Redis reference architectures GitHub repository.

Redis for Tanzu Application Service with Steeltoe

Steeltoe Cloud Connectors can connect to Redis for Tanzu Application Service. See the Steeltoe Cloud Connectors documentation.

To view examples of Steeltoe apps using Redis as a cache with failover, see the Example Steeltoe app repository in GitHub.

⚠ **Caution**

The Steeltoe connector for Redis requires Redis for Tanzu Application Service to support Lua scripting. Check whether the language you are using requires Lua scripting. If it does, contact your operator. By default, Lua scripting is deactivated for Redis for Tanzu Application Service, but an operator can change the setting to enable it by selecting the **Lua Scripting** checkbox in each service plan's **On-demand plan** configuration pane.

Other software

- **Pivotal Dev** is a version of VMware Tanzu Application Service for VMs (TAS for VMs) that is small enough to run on a local machine. For more information, see VMware dev.
- **Sample Ruby code** that uses TAS for VMs is in the CF Redis Example App GitHub repository.
- **Redis** is an open-source in-memory datastore. To learn more about Redis itself, see redis.io.

Use TLS

Follow the steps below to securely bind your apps to a Redis instance with TLS.

Spring and Steeltoe apps use TLS by default when available.

Check availability

You can check if TLS has been enabled on the on-demand Redis service by inspecting the service key. To do so:

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

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

2. Display the service key by running the following command:

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

   This returns a JSON response in this format:

   ```
   {
     "host": "q-s0.redis-instance.ENVIRONMENT-NAME-services-subnet.service-instance-GUID.bosh",
     "password": YOUR-PASSWORD,
     "port": INSECURE-PORT-NUMBER,
     "tls_port": SECURE-PORT-NUMBER
   }
   ```

If you do not see the `tls_port` field, TLS has not been enabled on your Redis service.

Bind new apps with TLS

Follow the steps below to securely bind new apps to a Redis instance.

For new apps, `cf bind-service` exposes both TLS ports and non-secure ports. Custom connectors also make both ports available. To support secure service bindings, you must specify the TLS port in your app code.

Below is an example of manually selecting the TLS port for a `redis_client` in Ruby:

```
require 'redis'
require 'cf-app-utils'

def redis_credentials
  service_name = ENV['service_name'] || "redis"
  if ENV['VCAP_SERVICES']
      all_pivotal_redis_credentials = CF::App::Credentials.find_all_by_all_service_tags(['redis', 'pivotal'])
      if all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
          all_pivotal_redis_credentials.first
      else
          redis_service_credentials = CF::App::Credentials.find_by_service_name(service_name)
          redis_service_credentials
      end
  end
end

def redis_client
   @client ||= Redis.new(
     host: redis_credentials.fetch('host'),
     port: redis_credentials.fetch('tls_port'),
     password: redis_credentials.fetch('password'),
     ssl: true,
     timeout: 30
end
```

For Spring apps, use Java CFEnv v1.1.0 or later. See Redis Spring Boot Reference Architecture in GitHub.

For Steeltoe apps, use Steeltoe v2.3.0 or later. See Redis Steeltoe Reference Architecture in GitHub.

Bind existing apps with TLS

For each app using the Redis service with a non-TLS binding:

1. Remove the current binding by running the following command:

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

2. Re-bind to the Redis instance by running the following command:

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

3. Restage the app by running the following command:

   ```
   cf restage-app APP-NAME
   ```

Your app now communicates securely with the Redis on-demand service instance.

Quickstart guide for app developers

This topic provides some sample apps in various languages to demonstrate how you can get started with Redis for VMware Tanzu Application Service. It also highlights the critical components of the apps that allow them to connect to a Redis instance. Credentials to connect to a Redis for Tanzu Application Service instance are passed to the apps as environment variables under `VCAP_SERVICES`.

Additionally, this topic includes advice for setting up Spring Sessions with Redis for Tanzu Application Service.

Quickstart apps

All apps using Redis for Tanzu Application Service must parse and read the Redis for Tanzu Application Service instance credentials from the environment. The credentials are available to the app once a Redis for Tanzu Application Service instance is bound to it and are viewable by typing `$cf env {app_name}`.

Prerequisites for these examples include access to a Marketplace with `p-redis` or `p.redis`. For reference, `p.redis` refers to the Redis service that provides on-demand instances and `p-redis` refers to the Redis service that provides shared-VM instances. Any service offering and plan work with the following examples. You can view available plans and instance types in the Marketplace.

Quickstart Java app

This is a basic Java app with the capability to get and set keys in Redis and view configuration information. Prerequisites include Maven.

Here we use an on-demand-cache plan of the `p.redis` service, but a `p-redis` instance also works.

```
$ git clone git@github.com:colrich/RedisForPCF-Java-Example.git java_redis_ap
p
```

```
$ cd java_redis_app
$ mvn package
$ cf create-service p.redis on-demand-cache redis_instance
$ cf push redis_example_app -p target/RedisExample-0.0.1-SNAPSHOT.jar
$ cf bind-service redis_example_app redis_instance
$ cf restage redis_example_app
```

You can then visit the app in your browser window. The app has three entry points:

- `"/"` — Gets info about a bound Redis instance
- `"/set"` — Sets a given key to a given value. For example, `{APP_URL}/set?kn=somekeyname&kv=valuetoset`
- `"/get"` — Gets the value stored at a given key. For example, `{APP_URL}/get?kn=somekeyname`

In the application code, the snippet where `VCAP_SERVICES` is read and parsed is here:

```
@RequestMapping("/")
public RedisInstanceInfo getInfo() {
    LOG.log(Level.WARNING, "Getting Redis Instance Info in Spring controlle
r...");
    // first we need to get the value of VCAP_SERVICES, the environment varia
ble
    // where connection info is stored
    String vcap = System.getenv("VCAP_SERVICES");
    LOG.log(Level.WARNING, "VCAP_SERVICES content: " + vcap);


    // now we parse the json in VCAP_SERVICES
    LOG.log(Level.WARNING, "Using GSON to parse the json...");
    JsonElement root = new JsonParser().parse(vcap);
    JsonObject redis = null;
    if (root != null) {
        if (root.getAsJsonObject().has("p.redis")) {
            redis = root.getAsJsonObject().get("p.redis").getAsJsonArray().ge
t(0).getAsJsonObject();
            LOG.log(Level.WARNING, "instance name: " + redis.get("name").getA
sString());
        }
        else if (root.getAsJsonObject().has("p-redis")) {
            redis = root.getAsJsonObject().get("p-redis").getAsJsonArray().ge
t(0).getAsJsonObject();
            LOG.log(Level.WARNING, "instance name: " + redis.get("name").getA
sString());
        }
        else {
            LOG.log(Level.SEVERE, "ERROR: no redis instance found in VCAP_SER
VICES");
        }
    }
    // then we pull out the credentials block and produce the output
    if (redis != null) {
        JsonObject creds = redis.get("credentials").getAsJsonObject();
        RedisInstanceInfo info = new RedisInstanceInfo();
        info.setHost(creds.get("host").getAsString());
        info.setPort(creds.get("port").getAsInt());
        info.setPassword(creds.get("password").getAsString());

        // the object will be json serialized automatically by Spring web - w
e just need to return it
        return info;
    }
    else return new RedisInstanceInfo();
}
```

Quickstart Node app

This is a basic Node app with the capability to get and set keys in Redis and view configuration information. Prerequisites are the `cf cli` and access to a Marketplace with p-redis or p.redis.

Here we use an on-demand-cache plan for the `p.redis` service, but a `p-redis` instance also works.

```
$ git clone git@github.com:colrich/RedisForPCF-Node-Example.git node_redis_ap
p
$ cd node_redis_app
$ cf create-service p.redis on-demand-cache redis_instance
$ cf push redis_example_app
$ cf bind-service redis_example_app redis_instance
$ cf restage redis_example_app
```

You can then visit the app in your browser window. The app has three entry points:

- `"/"` — Gets info about bound redis instance
- `"/set"` — Sets a given key to a given value. For example, `{APP_URL}/set?kn=somekeyname&kv=valuetoset`
- `"/get"` — Gets the value stored at a given key. For example, `{APP_URL}/get?kn=somekeyname`

In the application code, the snippet where `VCAP_SERVICES` is read and parsed is here:

```
// parses the VCAP_SERVICES env var and looks for redis service instances
function getVcapServices() {
  var vcstr = process.env.VCAP_SERVICES;
  if (vcstr != null && vcstr.length > 0 && vcstr != '{}') {
    console.log("found VCAP_SERVICES: " + vcstr)

    var vcap = JSON.parse(vcstr);
    if (vcap != null) {
      if (vcap.hasOwnProperty("p.redis")) {
        console.log("found redis instance: " + vcap["p.redis"][0].name);
        return vcap["p.redis"][0]
      }
      else if (vcap.hasOwnProperty("p-redis")) {
        console.log("found redis instance: " + vcap["p-redis"][0].name);
        return vcap["p-redis"][0]
      }
      else {
        console.log("ERROR: no redis service bound!")
      }
    }
    else {
      console.log("ERROR: no redis service bound!")
    }
  }
  else {
    console.log("ERROR: VCAP_SERVICES does not contain a redis block")
  }
  return null
}

// pulls the necessary connection info out of the parsed VCAP_SERVICES block
for
// the redis connection
function getRedisInfo(vcap) {
  var info = {}
  info["host"] = vcap["credentials"]["host"]
  info["port"] = vcap["credentials"]["port"]
  info["password"] = vcap["credentials"]["password"]
  return info
}


// set the port to listen on; for apps, listen on $PORT (usually 8000)
app.set('port', (process.env.PORT || 8080))


// this method looks in VCAP_SERVICES for a redis service instance and output
s the
// host / port / password info to the response
app.get('/', function(request, response) {
  console.log("Getting Redis connection info from the environment...")

  var vcap = getVcapServices()
  if (vcap != null) {
    var info = getRedisInfo(vcap)
    console.log("connection info: " + info.host + " / " + info.port + " / " +
info.password)
    response.send("connection info: " + info.host + " / " + info.port + " / "
+ info.password)
  }
  else {
    console.log("ERROR: VCAP_SERVICES does not contain a redis block or no re
dis bound")
    response.send("ERROR: VCAP_SERVICES does not contain a redis block or no
redis bound")
  }
})
```

Quickstart Ruby app

This is a basic Ruby app with the capability to get and set keys in Redis and view configuration information. Here we use an instance of the shared-VM service, but any `p-redis` or `p.redis` instance works.

```
$ git clone git@github.com:pivotal-cf/cf-redis-example-app.git ruby_redis_app
$ cd ruby_redis_app
$ cf create-service p-redis shared-vm redis_instance
$ cf push redis_example_app --no-start
$ cf bind-service redis_example_app redis_instance
$ cf start redis_example_app"
```

You can then get, set, and delete keys:

```
$ export APP=redis-example-app.my-cloud-foundry.com
$ curl -X PUT $APP/foo -d 'data=bar'
success
$ curl -X GET $APP/foo
bar
$ curl -X DELETE $APP/foo
success
```

In the application code, the method where `VCAP_SERVICES` is read is here:

```
def redis_credentials
  service_name = ENV['service_name'] || "redis"

  if ENV['VCAP_SERVICES']
    all_pivotal_redis_credentials = CF::App::Credentials.find_all_by_all_serv
ice_tags(['redis', 'pivotal'])
    if all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
      all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
    else
      redis_service_credentials = CF::App::Credentials.find_by_service_name(s
ervice_name)
      redis_service_credentials
    end
  end
end
```

The method where `VCAP_SERVICES` is parsed is here:

```
def redis_client
  @client ||= Redis.new(
    host: redis_credentials.fetch('host'),
    port: redis_credentials.fetch('port'),
    password: redis_credentials.fetch('password'),
    timeout: 30
  )
end
```

Spring Session with Redis for Tanzu Application Service

One common use case of Redis for Tanzu Application Service is management of a user's session information with Spring Session. Spring Session provides an API and implementations with which to manage sessions.

This topic describes how to use Redis for Tanzu Application Service as the backend with Spring Session to manage user session information.

This documentation is adopted from the Spring Session docs and extends to include instructions for use with Redis for Tanzu Application Service. The document is also adopted from this Spring Session - Spring Boot guide.

Setting up Spring Session

Updating dependencies

To use Spring Session, update your dependencies to include spring-session-data-redis. The following example is for Maven.

`pom.xml`

```
<dependencies>
          <!-- ... -->
          <dependency>
                  <groupId>org.springframework.session</groupId>
                  <artifactId>spring-session-data-redis</artifactId>
                  <version>1.3.1.RELEASE</version>
                  <type>pom</type>
          </dependency>
          <dependency>
                  <groupId>biz.paluch.redis</groupId>
                  <artifactId>lettuce</artifactId>
                  <version>3.5.0.Final</version>
          </dependency>
          <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-web</artifactId>
                  <version>4.3.4.RELEASE</version>
          </dependency>
  </dependencies>
```

Spring Java Configuration

After adding the required dependencies, we can create our Spring configuration.

The Spring configuration is responsible for creating a Servlet Filter that replaces the `HttpSession` implementation with an implementation backed by Spring Session. Add the following Spring Configuration:

```
@EnableRedisHttpSession (1)
public class Config {

        @Bean
        public LettuceConnectionFactory connectionFactory() {
                return new LettuceConnectionFactory(); (2)
        }
}
```

1 The `@EnableRedisHttpSession` annotation creates a Spring Bean with the name of `springSessionRepositoryFilter` that implements Filter. The filter is what is in charge of replacing the `HttpSession` implementation to be backed by Spring Session. In this instance Spring Session is backed by Redis.

2 We create a `RedisConnectionFactory` that connects Spring Session to the Redis Server. We configure the connection to connect to localhost on the default port (6379) For more information on configuring Spring Data Redis, refer to the reference documentation.

Java Servlet container initialization

Our Spring Configuration created a Spring Bean named `springSessionRepositoryFilter` that implements `Filter`. The `springSessionRepositoryFilter` bean is responsible for replacing the `HttpSession` with a custom implementation that is backed by Spring Session.

In order for our `Filter` to do its magic:

- Spring needs to load our `Config` class.
- We need to ensure that our Servlet Container (i.e. Tomcat) uses our `springSessionRepositoryFilter` for every request.

Fortunately, Spring Session provides a utility class named `AbstractHttpSessionApplicationInitializer`, which helps us confirm that these two requirements are met.

The example below shows how to extend `AbstractHttpSessionApplicationInitializer`:

`src/main/java/sample/Initializer.java`

```
public class Initializer extends AbstractHttpSessionApplicationInitializer { (1)

        public Initializer() {
                super(Config.class); (2)
        }
}
```

The name of our class (Initializer) does not matter. What is important is that we extend `AbstractHttpSessionApplicationInitializer`. Doing this achieves the following:

- It ensures that the Spring Bean by the name `springSessionRepositoryFilter` is registered with our Servlet Container for every request.
- It provides a mechanism to easily ensure that Spring loads our `Config`.

Configuring Redis for Tanzu Application Service as a back end

At this stage, Spring Session is now configured to use a Redis instance. To use a Redis for Tanzu Application Service instance, create a `session-replication` tag for it.

```
$ cf update-service INSTANCE_NAME -t session-replication
```

Other considerations

The `RedisHttpSessionConfiguration` tries to use the Redis CONFIG command. The CONFIG command is not available due to security recommendations.

This feature can be deactivated by exposing `ConfigureRedisAction.NO_OP` as a bean:

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

However, deactivating the configuration means that Redis cannot send namespace notifications. This functionality is critical for apps that require `SessionDestroyedEvent` to be fired to clean up resources, such as for WebSocket apps to ensure open WebSockets are closed when the `HttpSession` expires.

Using Redis for VMware Tanzu Application Service

You can use Redis for VMware Tanzu Application Service both through Apps Manager and the Cloud Foundry Command Line Interface (cf CLI). Both methods are outlined in this topic.

You can find an example app has to help you get started with Redis for Tanzu Application Service. Download the example app by clicking this link.

For recommendations regarding Redis for Tanzu Application Service plans and memory allocation, see the On-Demand Service Offering and the Shared-VM Service Offering.

Prerequisites

To use Redis for Tanzu Application Service with your TAS for VMs apps, you must:

- Have an Tanzu Operations Manager installation with Redis for Tanzu Application Service installed and listed in the Marketplace.
  For how to verify availability in the Marketplace, see Confirm Service Availability.
- Have a Space Developer or Admin account on the TAS for VMs installation.
  For more information, see Manage Users and Roles.
- Have a local machine with the following installed:
  - A browser
  - A shell
  - The Cloud Foundry Command-Line Interface (cf CLI). See Installing the cf CLI.

  - The Linux watch command. See the Linux Information Project website.
- Log in to the org and space containing your app. For instructions, see Log in with the CLI.

Use Redis for Tanzu Application Service in an app

Every app and service is scoped to a space. To use a service, an app must exist in the same space as an instance of the service.

To use Redis for Tanzu Application Service in an app:

1. Use the cf CLI or Apps Manager to log in to the org and space that contains the app.
2. Make sure a Redis for Tanzu Application Service service instance exists in the same space as the app.

   - If the space does not already have a Redis for Tanzu Application Service instance, create one.
   - If the space already has a Redis for Tanzu Application Service instance, you can bind your app to the existing instance or create a new instance to bind to your app.
3. Bind the app to the Redis for Tanzu Application Service service instance, to enable the app to use Redis.

Confirm service availability

For an app to use a service, the following two things must be true:

- The service must be available in the Marketplace for its space.
- An instance of the service must exist in its space.

You can confirm both of these using the cf CLI as follows:

1. To find out if a Redis service is available in the Marketplace:
   1. Run:

      ```
      cf marketplace
      ```

   2. If the output lists `p.redis` in the `service` column, on-demand Redis for Tanzu Application Service is available. If the output lists `p-redis` in the `service` column, shared-VM Redis for Tanzu Application Service is available. If it is not available, ask your operator to install it.

      For example:

      ```
      $ cf marketplace
      Getting services from marketplace in org my-org / space my-space
      as user@example.com...
      OK
      service             plans                     description
      p-redis             shared-vm                 Redis service to pr
      ovide pre-provisioned instances configured as a datastore, runnin
      ```

```
g on a shared VM.
p.redis             on-demand-cache          Redis service to pr
ovide on-demand dedicated instances configured as a cache.
[...]
```

2. To confirm that a Redis for Tanzu Application Service instance is running in the space:
    1. Run:

        ```
        cf services
        ```

    2. Any `p.redis` listings in the `service` column are service instances of on-demand Redis for Tanzu Application Service in the space. Any `p-redis` in the `service` column are service instances of shared-VM Redis for Tanzu Application Service.

        For example:

        ```
        $ cf services
        Getting services in org my-org / space my-space as user@example.c
        om...
        OK
        name          service     plan              bound apps    last op
        eration
        my-instance   p.redis     on-demand-cache                 create
        succeeded
        ```

You can bind your app to an existing instance or create a new instance to bind to your app.

Create a service instance

To use a service you must create a service instance of a plan that is available in the Marketplace. To do so, you can use either the cf CLI or Apps Manager.

Create a service instance with the cf CLI

You can use the cf CLI to create service instances of available on-demand or shared-VM plans.

On-demand service

Unlike pre-provisioned services, on-demand instances are created asynchronously, not immediately. On-demand plans are listed under the `p.redis` service in the Marketplace.

To create a service instance of the Redis for Tanzu Application Service on-demand plan, run:

```
cf create-service p.redis CACHE_PLAN SERVICE_NAME
```

Where:

- `CACHE_PLAN` is one of the plans configured by the operator.
- `SERVICE_NAME` is a name for your service.

For example:

```
$ cf create-service p.redis on-demand-cache od-instance

Creating service my-ondemand-instance in org my-org / space my-space as user@
example.com...
OK
```

As the On-Demand instance can take longer to create, the `watch` command is helpful as a way to track when your service instance is ready to bind and use.

```
$ watch cf services

Getting services in org my-org / space my-space as user@example.com...
OK
name          service       plan              bound apps    last operation
od-instance   p.redis       on-demand-cache                 create succeeded
```

If you get an error, see Troubleshooting Instances. For information on the on-demand cache plans, see On-Demand Service Plans.

Shared-VM service

Shared-VM service instances have been pre-provisioned by the operator. This means, if an instance is available, the app developer can provision it immediately. These plans are both listed under the `p-redis` service in the Marketplace.

> **Note**
>
> Shared-VM services are designed for testing and development purposes. Shared-VMs should not be used in production environments

To create a service instance of the Redis for Tanzu Application Service shared-VM plan, run:

```
cf create-service p-redis SERVICE_TYPE SERVICE_NAME
```

Where:

- `SERVICE_TYPE` is `shared-vm`.
- `SERVICE_NAME` is a name for your service instance.

For example:

```
$ cf create-service p-redis shared-vm my-instance

Creating service my-instance in org my-org / space my-space as user@example.c
om...
OK
```

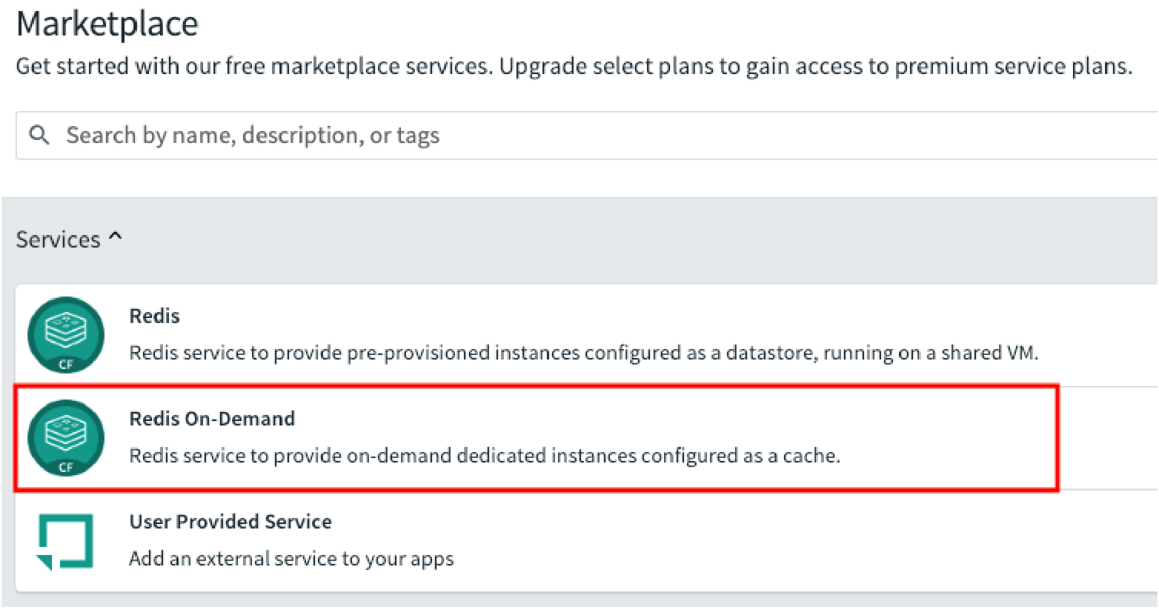
Create a service instance with Apps Manager

You can use Apps Manager to create service instances of available on-demand or shared-VM plans.

On-demand service

To create a service instance of the Redis for Tanzu Application Service on-demand plan using Apps Manager:

1. From within Apps Manager, select **Marketplace** from the left navigation menu.
2. Select **Redis On-Demand** from the displayed tiles in the Marketplace.



Marketplace

Get started with our free marketplace services. Upgrade select plans to gain access to premium service plans.

Redis

Redis service to provide pre-provisioned instances configured as a datastore, running on a shared VM.

Redis On-Demand

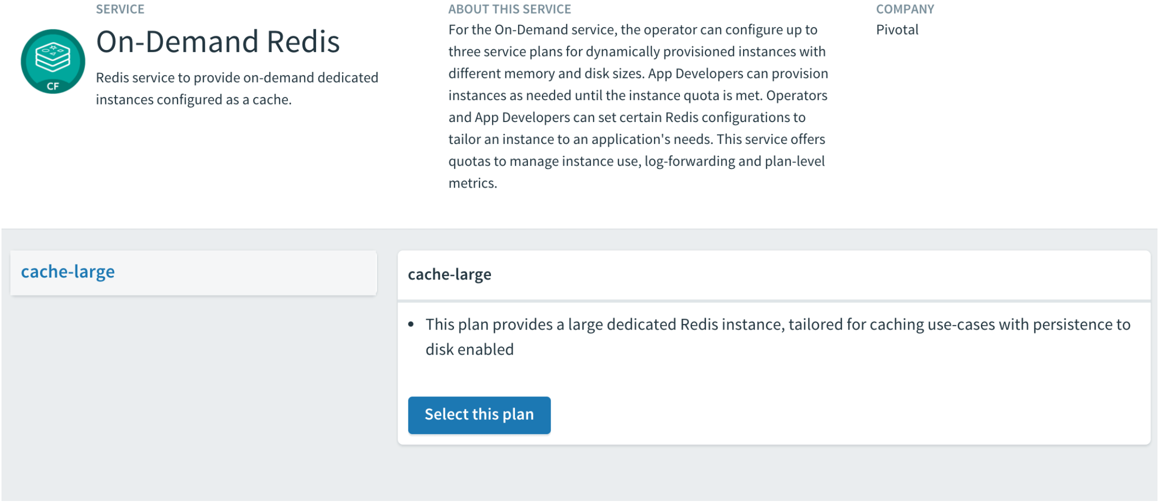
Redis service to provide on-demand dedicated instances configured as a cache.

User Provided Service

Add an external service to your apps

Click here to view a larger version of this image

3. Click the appropriate **Select this plan** button to select the required **Redis Service Plan**.



SERVICE

On-Demand Redis

Redis service to provide on-demand dedicated instances configured as a cache.

ABOUT THIS SERVICE

For the On-Demand service, the operator can configure up to three service plans for dynamically provisioned instances with different memory and disk sizes. App Developers can provision instances as needed until the instance quota is met. Operators and App Developers can set certain Redis configurations to tailor an instance to an application's needs. This service offers quotas to manage instance use, log-forwarding and plan-level metrics.

COMPANY

Pivotal

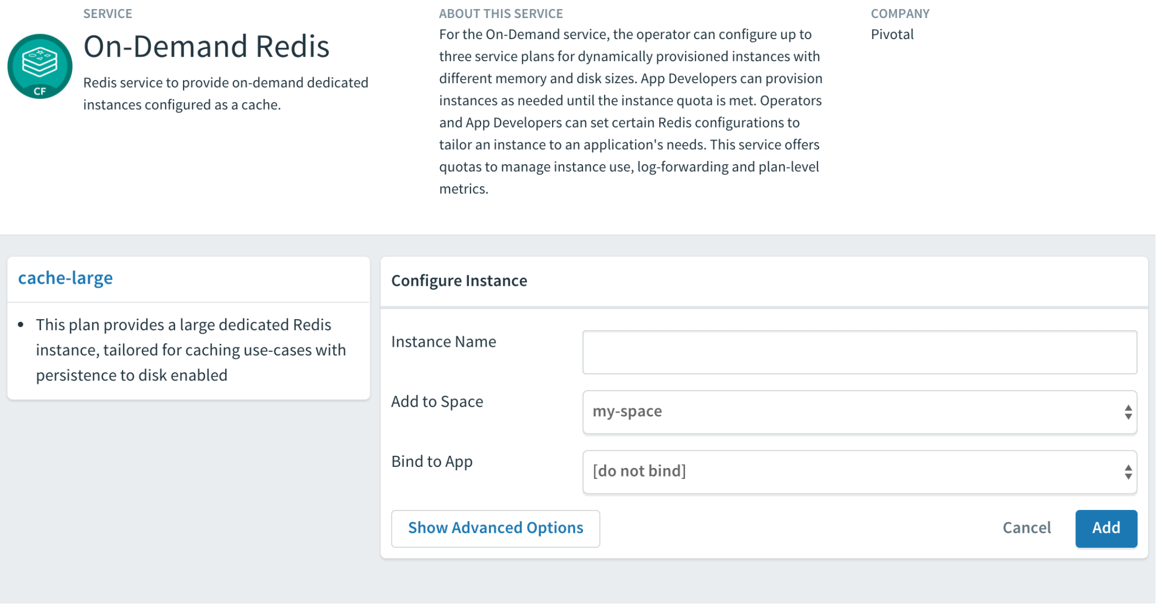
cache-large

cache-large

- This plan provides a large dedicated Redis instance, tailored for caching use-cases with persistence to disk enabled

Click here to view a larger version of this image

4. In the **Instance Name** field, enter a name that will identify this specific Redis service instance.



SERVICE

On-Demand Redis

Redis service to provide on-demand dedicated instances configured as a cache.

ABOUT THIS SERVICE

For the On-Demand service, the operator can configure up to three service plans for dynamically provisioned instances with different memory and disk sizes. App Developers can provision instances as needed until the instance quota is met. Operators and App Developers can set certain Redis configurations to tailor an instance to an application's needs. This service offers quotas to manage instance use, log-forwarding and plan-level metrics.

COMPANY

Pivotal

cache-large

- This plan provides a large dedicated Redis instance, tailored for caching use-cases with persistence to disk enabled

Configure Instance

Instance Name	
Add to Space	my-space
Bind to App	[do not bind]

Click here to view a larger version of this image

5. From the **Add to Space** dropdown, select the space where you or other users will deploy the apps that will bind to the service.

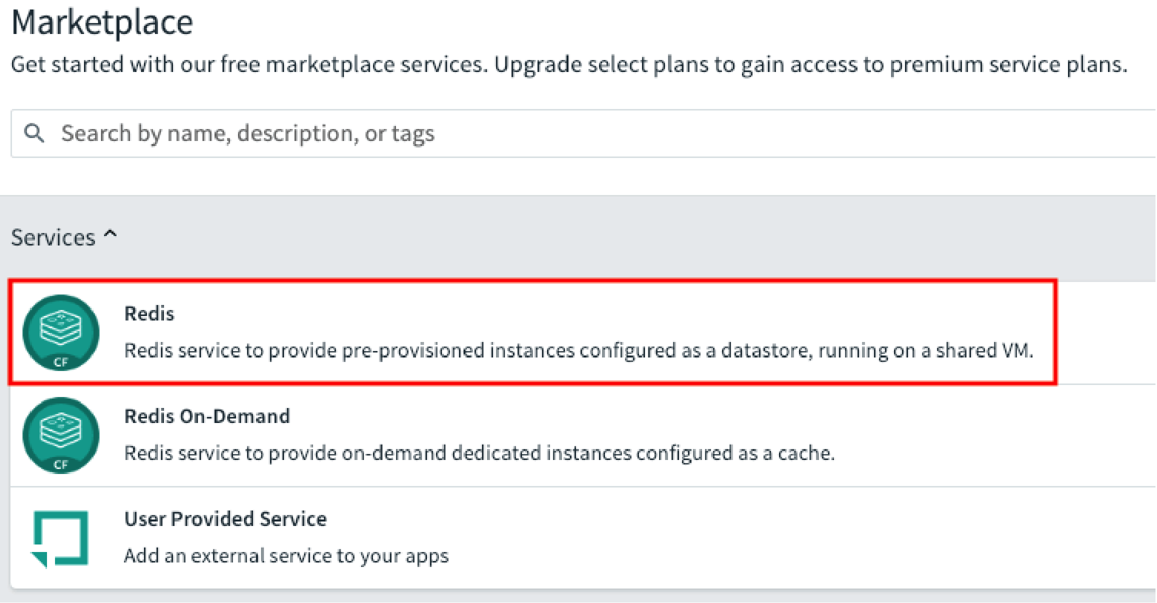
6. Click the **Add** button.

Shared-VM service

To create a service instance of the Redis for Tanzu Application Service shared-VM plan using Apps Manager:

1. From within Apps Manager, select **Marketplace** from the left navigation menu.

2. Select **Redis** from the displayed tiles in the Marketplace.



Marketplace

Get started with our free marketplace services. Upgrade select plans to gain access to premium service plans.

Redis

Redis service to provide pre-provisioned instances configured as a datastore, running on a shared VM.

Redis On-Demand

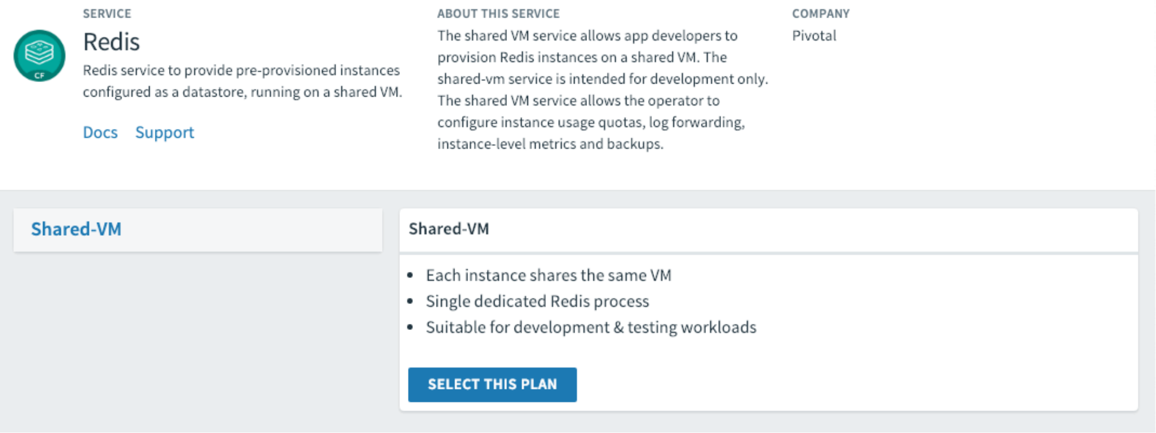
Redis service to provide on-demand dedicated instances configured as a cache.

User Provided Service

Add an external service to your apps

Click here to view a larger version of this image

3. Click on the appropriate **Select this plan** button to select the required **Redis Service Plan**.



SERVICE

Redis

Redis service to provide pre-provisioned instances configured as a datastore, running on a shared VM.

ABOUT THIS SERVICE

The shared VM service allows app developers to provision Redis instances on a shared VM. The shared-vm service is intended for development only. The shared VM service allows the operator to configure instance usage quotas, log forwarding, instance-level metrics and backups.

COMPANY

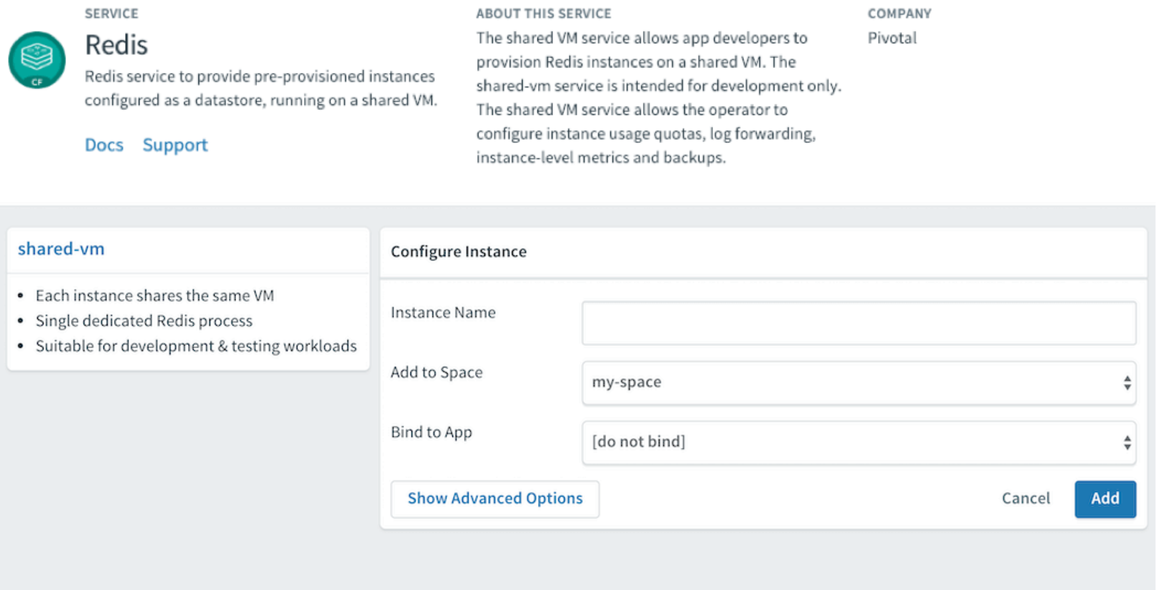
Pivotal

Shared-VM

- Each instance shares the same VM
- Single dedicated Redis process
- Suitable for development & testing workloads

Click here to view a larger version of this image

4. In the **Instance Name** field, enter a name that will identify this specific Redis service instance.



SERVICE

Redis

Redis service to provide pre-provisioned instances configured as a datastore, running on a shared VM.

ABOUT THIS SERVICE

The shared VM service allows app developers to provision Redis instances on a shared VM. The shared-vm service is intended for development only. The shared VM service allows the operator to configure instance usage quotas, log forwarding, instance-level metrics and backups.

COMPANY

Pivotal

shared-vm

- Each instance shares the same VM
- Single dedicated Redis process
- Suitable for development & testing workloads

Configure Instance

Instance Name

Add to Space: my-space

Bind to App: [do not bind]

Click here to view a larger version of this image

5. From the **Add to Space** dropdown, select the space where you or other users will deploy the apps that will bind to the service.

6. Click the **Add** button.

Bind a service instance to your app

For an app to use a service, you must bind it to a service instance. You can use either the cf CLI or Apps Manager to do this. Bind the app after you push or re-push it using `cf push`.

Bind a service instance with the cf CLI

To bind an app to a Redis for Tanzu Application Service service instance:

1. View running service instances:

```
cf services
```

For example:

```
$ cf services

Getting services in org system / space apps-manager as admin...
OK

name                service         plan        bound apps    last oper
ation
my-instance   p-redis         shared-vm                 create succeede
d
```

2. Bind the service instance to your app by running:

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

Where:

- `APP` is the app you want to use the Redis service instance.
- `SERVICE_INSTANCE` is the name you supplied when you ran `cf create-service`.

For example:

```
$ cf bind-service my-app my-instance

Binding service my-instance to my-app in org my-org / space test as use
r@example.com...
OK
TIP: Use 'cf push' to ensure your env variable changes take effect
```

Bind a service instance with Apps Manager

To bind an app to a Redis for Tanzu Application Service service instance:

1. Select the app that you want to bind to the service. A page displays showing the already bound services and instances for this app.
2. Click **Bind**. A list of available services displays.
3. Click the **Bind** button for the Redis service you want to bind to this app.
4. Start or restage your app from the command line, for example:

```
$ cf restage my-app
```

Customize an on-demand service instance

The On-Demand Service allows operators and app developers to customize certain configuration variables.

Operators can customize the memory size, org and space access, Redis Client Timeout (default 3600 seconds), Redis TCP Keepalive (default 60 seconds), Redis Max Clients (default 1000), and can enable Lua Scripting.

App developers can customize the following parameters. See the Redis documentation for more detail.

Property	Default	Options	Description
`maxmemory-policy`	`allkeys-lru`	`allkeys-lru`, `noeviction`, `volatile-lru`, `allkeys-random`, `volatile-ttl`, `volatile-lfu`, `allkeys-lfu`	Sets the behavior Redis follows when `maxmemory` is reached
`notify-keyspace-events`	""	Set a combination of the following characters (e.g., `Elg`): K, E, g, $, l, s, h, z, x, e, A	Sets the keyspace notifications for events that affect the Redis data set
`slowlog-log-slower-than`	10000	0-20000	Sets the threshhold execution time (seconds). Commands that exceed this execution time are added to the slowlog.
`slowlog-max-len`	128	1-2024	Sets the length (count) of the slowlog queue.

Customize an on-demand instance with cf CLI

> **Note**
>
> Arbitrary parameters are only supported for on-demand service instances. Shared-VM plans do not support the use of CLI commands with arbitrary parameters to configure service instances.

You can customize an instance in two ways:

- While creating the instance, run:

  ```
  cf create-service SERVICE PLAN NAME -c '{"PROPERTY":"SETTING"}'
  ```

- After creating the instance, run:

  ```
  cf update-service NAME -c '{"PROPERTY":"SETTING"}'
  ```

For both scenarios, the `-c` flag requires a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

```
$ cf update-service my-instance -c '{"maxmemory-policy":"noeviction"}'
```

You can pass through mutliple arbitrary parameters:

```
$ cf update-service my-instance -c '{"maxmemory-policy":"noeviction", "notify
-keyspace-events":"El"}'
```

If the update is not successful, an error is displayed with a description of what went wrong. Here is an example where a hyphen is added to the `noeviction` setting.

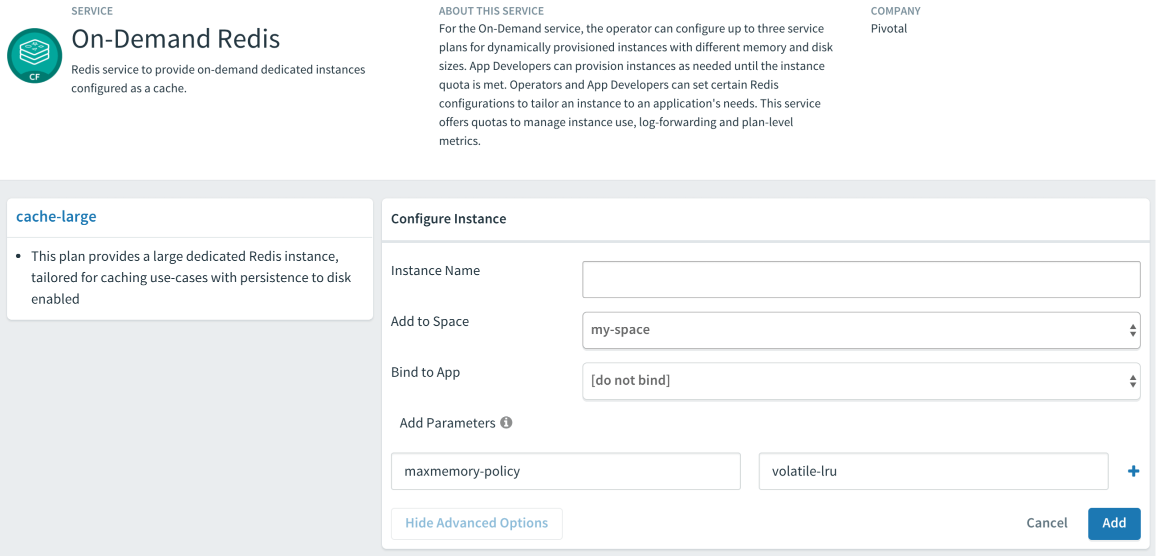
```
$ cf update-service my-instance -c '{"maxmemory-policy":"no-eviction", "notif
y-keyspace-events":"El"}'
```

```
Updating service instance my-instance as admin...
FAILED
Server error, status code: 502, error code: 10001, message: Service broker er
ror: invalid value "no-eviction" specified for maxmemory-policy
```

Customize an on-demand instance with Apps Manager

You can customize an instance in two ways:

- While creating the instance, after you select the plan, click **advanced settings**.



SERVICE

On-Demand Redis

Redis service to provide on-demand dedicated instances configured as a cache.

ABOUT THIS SERVICE

For the On-Demand service, the operator can configure up to three service plans for dynamically provisioned instances with different memory and disk sizes. App Developers can provision instances as needed until the instance quota is met. Operators and App Developers can set certain Redis configurations to tailor an instance to an application's needs. This service offers quotas to manage instance use, log-forwarding and plan-level metrics.

COMPANY

Pivotal

cache-large

- This plan provides a large dedicated Redis instance, tailored for caching use-cases with persistence to disk enabled

Configure Instance

Instance Name:

Add to Space: my-space

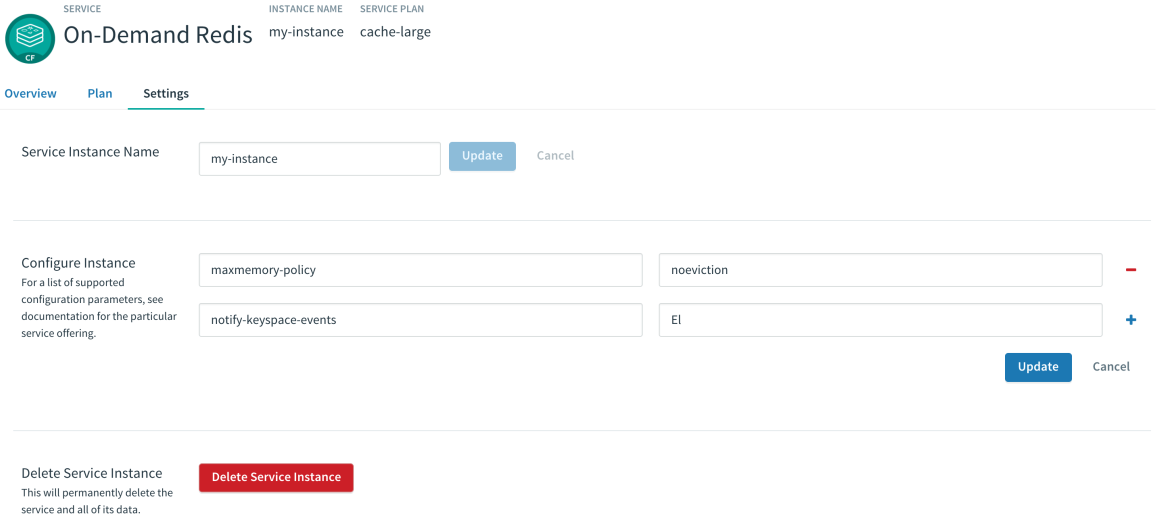
Bind to App: [do not bind]

Add Parameters

maxmemory-policy: volatile-lru

Click here to view a larger version of this image

- After creating the instance, navigate to the instance Settings page.



SERVICE: On-Demand Redis

INSTANCE NAME: my-instance

SERVICE PLAN: cache-large

Service Instance Name: my-instance

Configure Instance

For a list of supported configuration parameters, see documentation for the particular service offering.

maxmemory-policy: noeviction

notify-keyspace-events: El

Delete Service Instance

This will permanently delete the service and all of its data.

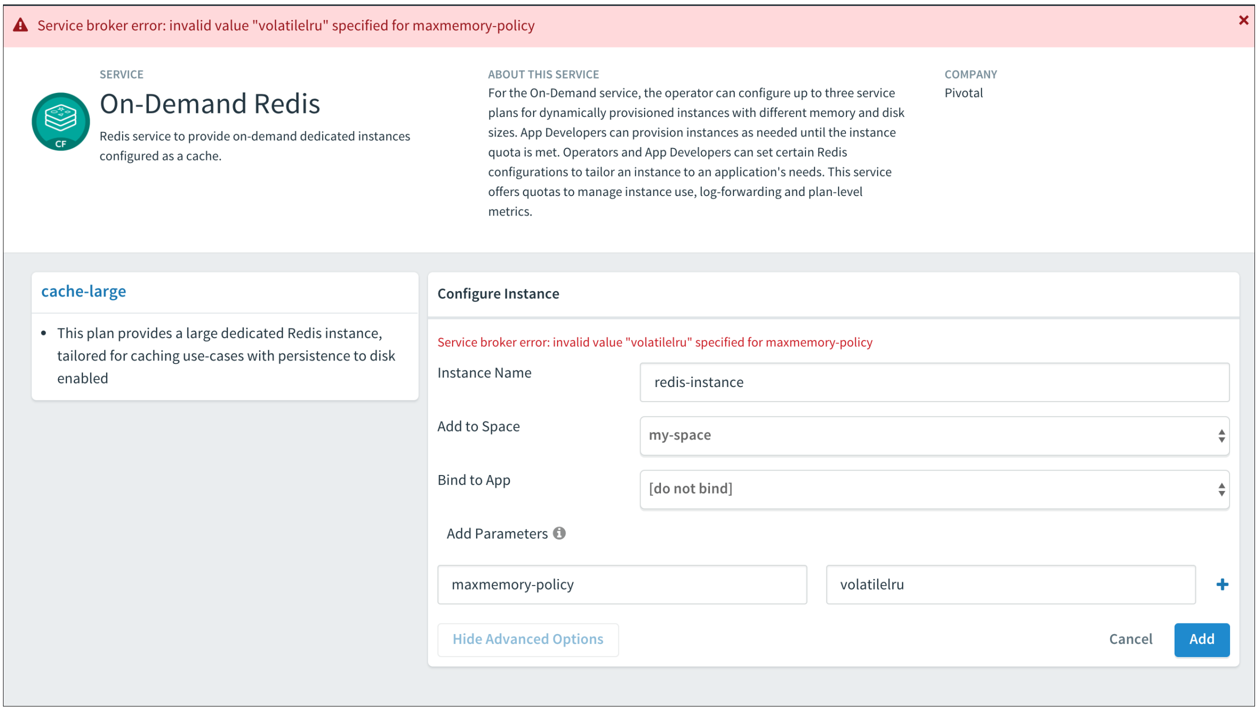
Click here to view a larger version of this image

In either of the above cases, do the following:

1. In the parameters fields enter each property you want to change and its new setting. Click the **+** sign to add more parameter fields.

2. Depending on the page you are on, click either **Add** or **Update**.

If the update is unsuccessful, Apps Manager displays an error with a description of what went wrong. The following screenshot is an example of an error caused by a missing hyphen in the `volatile-lru` setting.



Service broker error: invalid value "volatilelru" specified for maxmemory-policy

SERVICE

On-Demand Redis

Redis service to provide on-demand dedicated instances configured as a cache.

ABOUT THIS SERVICE

For the On-Demand service, the operator can configure up to three service plans for dynamically provisioned instances with different memory and disk sizes. App Developers can provision instances as needed until the instance quota is met. Operators and App Developers can set certain Redis configurations to tailor an instance to an application's needs. This service offers quotas to manage instance use, log-forwarding and plan-level metrics.

COMPANY

Pivotal

cache-large

- This plan provides a large dedicated Redis instance, tailored for caching use-cases with persistence to disk enabled

Configure Instance

Service broker error: invalid value "volatilelru" specified for maxmemory-policy

Instance Name: redis-instance

Add to Space: my-space

Bind to App: [do not bind]

Add Parameters

maxmemory-policy: volatilelru

Click here to view a larger version of this image

Retrieve the password for a Redis service instance

All Redis for Tanzu Application Service instances are password-protected and require authentication. This is enforced with the `requirepass` directive in the configuration file.

To retrieve the password, do the following:

1. Create a service-key for your Redis service instance by running:

   ```
   cf create-service-key INSTANCE-NAME SERVICE-KEY-NAME
   ```

   For example:

   ```
   $ cf create-service-key my-instance my-key
   Creating service key my-key for service instance my-instance as admi
   n...
   OK
   ```

2. Retrieve the password using the command by running:

   ```
   cf service-key INSTANCE-NAME SERVICE-KEY-NAME
   ```

   For example of this procedure, where the user is `admin`:

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

```
{
 "host": "10.0.8.4", # IP or BOSH DNS hostname for ODB instances
 "password": "admin-password",
 "port": 6379
}
```

Redis for Tanzu Application Service data is accessible from apps bound to that instance. Some Redis for Tanzu Application Service users bind the opensource cf-redis-commander app to view instance data. This app is not maintained by the Redis for Tanzu Application Service team, and VMware cannot guarantee its performance or security.

Use the Redis service in your app

Environment variables are how Cloud Foundry communicates with a deployed app about its environment. To access the environment variables, bind your app to an instance and run `cf env APP_NAME` from the cf cli.

To access the Redis service from your app:

1. Run the following command using the name of the app bound to the Redis for Tanzu Application Service instance.

   ```
   cf env APP_NAME
   ```

2. In the output, note the connection strings listed in the `VCAP_SERVICES` > `credentials` object for the app.

   Example `VCAP_SERVICES`:

   ```
    {
      "p-redis": [{
        "credentials": {
            "host": "10.0.0.11",
            "password": "",
            "port": 6379
        },
        "label": "p-redis",
        "name": "redis",
        "plan": "shared-vm",
        "provider": null,
        "syslog_drain_url": null,
        "tags": [
        "pivotal",
        "redis"
        ],
        "volume_mounts": []
      }]
    }
   ```

   You can also search for your service by its `name`, given when creating the service instance, or dynamically via the `tags` or `label` properties.

3. In your app code, call the Redis service using the connection strings.

Manage key eviction for shared-VM instances

Shared-VM plans provision Redis instances with a max-memory policy set to `no-eviction`.

It is up to the app developer to manage eviction of keys. The following are a few options for doing this:

- After setting keys, use EXPIRE to set key expiry, or use SETEX to set key value and expiry at the same time.
- Explicitly delete keys after the app is done using them.
- Add a lua script to delete keys after a specified time period.

Access Redis metrics for on-demand service instances

To access metrics for Redis for Tanzu Application Service service instances, you can use Loggregator's Log Cache feature with the Log Cache CLI plug-in. Log Cache is enabled by default.

To access metrics for on-demand service instances:

1. Install the cf CLI plug-in by running:

   ```
   cf install-plugin -r CF-Community "log-cache"
   ```

2. To access metrics for a service instance, run:

   ```
   cf tail SERVICE-INSTANCE-NAME
   ```

   Where `SERVICE-INSTANCE-NAME` is the name of your service instance.

   For example:

   ```
     $ cf tail my-instance
     Retrieving logs for service my-instance in org system / space pivotal
   -services as admin...
     2018-07-03T09:54:14.84+0100 [my-instance] GAUGE info.clients.blocked_
   clients:0.000000 metric info.clients.client_biggest_input_buf:0.000000
   metric ...
   ```

   For more information about the metrics output, see Redis KPIs.

For more information about how to enable Log Cache and about the `cf tail` command, see Enable Log Cache.

Sharing a Redis instance with another space

Sharing a service instance allows apps in different spaces to access the same Redis instance. Tile operators must enable this behavior and a cf admin must turn it on. For more information about this feature, see Sharing Service Instances in the Cloud Foundry documentation.

To share a service instance, run:

```
cf v3-share-service REDIS-SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```

Where:

- `REDIS-SERVICE-INSTANCE` is the name of the Redis instance.
- `OTHER-SPACE` is the name of the other space you want to share this instance with.
- `OTHER-ORG` is the name of another org you want to share this instance with (optional).

Unshare a Redis service instance

> **Caution**
>
> Redis only has one password and password rotation does not occur on unshare. After unsharing a service, any bound apps continue to have access to the Redis instance until the apps are restaged.

To unshare a service instance, run:

```
cf v3-unshare-service REDIS-SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```

Where:

- `REDIS-SERVICE-INSTANCE` is the name of the Redis instance.
- `OTHER-SPACE` is the name of the other space you want to share this instance with.
- `OTHER-ORG` is the name of another org you want to share this instance with (optional).

Delete a Redis instance

When you delete a Redis service instance, all apps that are bound to that service are automatically unbound and any data in the service instance is cleared.

Delete a Redis service instance with the cf CLI

To delete a service instance:

1. Run the following command and enter `y` when prompted to confirm.

   ```
   cf delete-service SERVICE-INSTANCE-NAME
   ```

   For example:

   ```
   $ cf delete-service my-redis-instance

   Really delete the service my-redis-instance?> y
   Deleting service my-redis-instance in org system / space apps-manager a
   s admin...
   OK
   ```

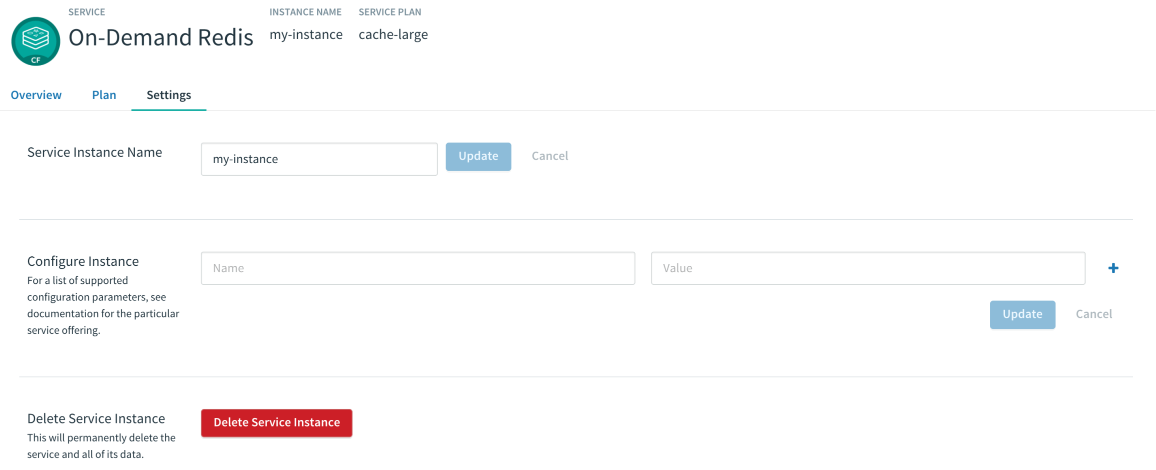
2. If you had apps that were bound to this service, you might need to restage or re-push your app for the app changes to take effect. For example:

```
$ cf restage my-app
```

Delete a Redis service instance with Apps Manager

To delete a service instance:

1. In the service instance Settings page, click **Delete Service Instance**.



SERVICE
On-Demand Redis

INSTANCE NAME
my-instance

SERVICE PLAN
cache-large

Service Instance Name: my-instance

Configure Instance
For a list of supported configuration parameters, see documentation for the particular service offering.

Delete Service Instance
This will permanently delete the service and all of its data.

Click here to view a larger version of this image

2. If you had apps that were bound to this service, you might need to restage or re-push your app for the app changes to take effect. For example:

```
$ cf restage my-app
```

Using the Config API with Redis for Tanzu Application Service

This topic tells you how to use the Config API feature for Redis for VMware Tanzu Application Service on-demand service instances.

Config API adds an endpoint to service instances for querying Redis configuration parameters. An HTTP GET request to `SERVICE-INSTANCE-BOSH-URL:8080/config/CONFIG-PARAMETER-NAME` returns the value of a setting.

Prerequisites

Before you can use the Config API, you must select the **Enable Config API** check box in the Redis for Tanzu Application Service tile. See Configure On-Demand Service settings.

Use the Config API to query Redis configuration parameters

After enabled, the Config API is available at port 8080 on the Redis service instance. You can query it from a Cloud Foundry app.

To query Redis configuration parameters:

1. Get the hostname of the service instance by running:

   ```
   cf env APP-NAME
   ```

   For example:

   ```
   $ cf env redis-example-app

   Getting env variables for app redis-example-app in org system / space p
   ivotal-services as admin...
   OK

   System-Provided:
   {
   "VCAP_SERVICES": {
     "p.redis": [
        {
           "binding_guid": "1d93f665-bb9e-493d-9c23-ea577f22a6d1",
           "binding_name": null,
           "credentials": {
              "host": "q-s0.redis-instance.pictonblue-services-subnet.serv
   ice-instance-30708f54-d8be-45f7-80a6-e587337233aa.bosh", "password": "5
   ft5I2aXZE7eXS1gjEB5DS7Izz859d",
              "port": 6379,
              "tls_port": 16379,
              "tls_versions": [
                 "tlsv1.2",
                 "tlsv1.3"
              ]
           },
        ...
        }
     ]
   }
   ```

2. Query the Redis configuration parameter by running:

   ```
   cf ssh APP-NAME -c "curl HOST-NAME:8080/config/CONFIG-PARAMETER-NAME" 2>/dev/nu
   ll
   ```

   Where: - `APP-NAME` - `HOST-NAME` - `CONFIG-PARAMETER-NAME`

   For example:

   ```
   $ cf ssh redis-example-app \
     -c "curl q-s0.redis-instance.pictonblue-services-subnet.service-insta
   nce-30708f54-d8be-45f7-80a6-e587337233aa.bosh:8080/config/port" 2>/dev/
   null
   ```

```
6379
```

Parameters you can query

You can query any parameters except for credentials such as `requirepass`, `masterauth`, or `masteruser`. The following are some configuration parameters you can query.

Redis properties

You can query the following Redis properties:

- daemonize
- port

Logging

You can query the following logging parameters:

- logfile
- syslog-enabled
- syslog-facility
- syslog-ident

Persistence

You can query the following persistence parameters:

- appendfilename
- appendonly
- dbfilename
- dir

Arbitrary parameters

You can query the following arbitrary parameters:

- maxmemory-policy
- slowlog-log-slower-than
- slowlog-max-len

Plan properties

You can query the following plan properties:

- maxclients
- tcp-keepalive
- timeout

Upgrading an individual Redis service instance

You can upgrade an individual Redis for VMware Tanzu Application Service on-demand service instance.

You can upgrade your service instances individually, if you have made a newer version of the tile available and enabled individual service instance upgrades. For the procedure, see Enabling individual Service Instance upgrades.

Prerequisites

Before you can upgrade individual Redis for Tanzu Application Service service instances, you must have the cf CLI v6.46.0 or later.

Upgrading a service instance

To upgrade a single service instance:

1. Confirm that an upgrade is available for the service instance by running:

   ```
   cf services
   ```

   The upgrade is available when the `upgrade available` column in the output says `yes`, for example:

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

   name    service    plan             last operation    broker     upgrad
   e available
   testSI  p.redis    on-demand-cache  create succeeded  p.redis    yes
   ```

2. Upgrade the service instance by running:

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

   Where `SERVICE-INSTANCE` is the name of the service instance that you want to upgrade.

3. When prompted, confirm that you want to update.

Creating a Redis service instance with service-gateway access

You can create a Redis for VMware Tanzu Application Service service instance with service-gateway access. Service-gateway access enables a Redis for VMware Tanzu Application Service on-demand service instance to connect to external components that are not on the same foundation as the service instance.

The following information assumes that you meet the prerequisites for using on-demand Redis for VMware Tanzu Application Service. For more information, see Prerequisites.

If you have enabled a service-gateway plan, you can create a service instance that can connect to components outside the your foundation. Contact your operator if you are unsure which plans are enabled for service-gateway access. For information about the architecture and use cases, see Service-gateway access.

To create a service instance that enables service-gateway access:

1. Create a service instance with the service-gateway plan by running:

```
cf create-service p.redis SERVICE-GATEWAY-PLAN SERVICE-INSTANCE-NAME
```

2. Obtain credentials by creating a service key:

```
cf create-service-key SERVICE-INSTANCE-NAME SERVICE-KEY-NAME
```

The service key looks similar to the following:

```
{
 "credentials": {
   "host": "q-s0.redis-instance.mediumcandyapplered-services-subnet.service-instance-0133e917-5cbf-432d-bab3-f4db5c603539.bosh",
   "password": "apassword",
   "port": 6379,
   "service_gateway_access_port": 1100,
   "service_gateway_enabled": true,
   "tls_port": 16379,
   "tls_versions": [
     "tlsv1.2",
     "tlsv1.3"
   ]
 }
}
```

The `service_gateway_access_port` field informs you of the port that was reserved for the created service instance. You can use this port to connect to Redis from outside your foundation.

If you deactivate and then re-activate service gateway access on a plan, you must create new service keys to obtain a new set of credentials for service gateway access.

Troubleshooting Redis instances

This topic for app developers gives you basic instructions for troubleshooting on-demand Redis for VMware Tanzu Application Service service instances.

Troubleshooting errors

Start here if you are responding to a specific error or error messages.

Common service errors

The following errors occur in multiple services:

- Apps fail to connect to the service instance

- No metrics from log cache

Apps fail to connect to the service instance	
Symptom	Apps fail to connect to the Redis service instance.
Cause	The Redis on-demand service broker now binds apps to service instances using BOSH DNS. Service bindings return the DNS address instead of the IP address. If an operator enables the BOSH HotSwaps feature, any apps with service bindings that do not use BOSH DNS will fail to connect to the Redis service instance. For more information, see Enable BOSH HotSwaps to Reduce Downtime. For more information about service bindings, see Service Bindings in the Cloud Foundry documentation.

Solution	Convert service instance bindings to use BOSH DNS. To do so, unbind, rebind, and restage all apps that were bound to a service instance using an IP address as follows:
	• Unbind the app:
	`cf unbind-service APP-NAME SERVICE-INSTANCE-NAME`
	• Rebind the app:
	`cf bind-service APP-NAME SERVICE-INSTANCE-NAME`
	• Restage the app:
	`cf restage APP-NAME`

No Metrics from log cache	
Symptom	You receive no metrics when running the `cf tail` command.
Cause	Depending on your versions of TAS for VMs and Redis for Tanzu Application Service, this can occur when the Firehose is deactivated in the TAS for VMs tile.
Solution	Ask your operator to ensure that the **V2 Firehose** checkbox is selected, and the **Enable Log Cache syslog ingestion** check box is cleared in the TAS for VMs tile. For more information about configuring these check boxes, see Enable Syslog Forwarding.

Redis for Tanzu Application Service-specific errors

The following errors are specific to Redis for Tanzu Application Service:

- Maximum Number of Clients Reached
- Maxmemory Limit Reached
- Error When Running the Save Command
- Unknown Command Error

Certain errors are returned to the Redis client instead of being recorded in the logs. The Redis protocol represents errors as simple strings beginning with a `-` character.

Maximum number of clients reached	
Symptom	You receive the following error: `-ERR max number of clients reached`
Cause	This is usually caused by apps opening multiple client connections to Redis.
Solution	Share or pool Redis connections within an app. Redis for Tanzu Application Service configures Redis to accept 10000 client connections. This can be confirmed by running the `INFO` command using the Redis CLI.

Maxmemory limit reached	
Symptom	You receive the following error: `-OOM command not allowed when used memory > 'maxmemory'.`
Cause	This occurs when the Redis server has reached its `maxmemory` limit.
Solution	Consider changing your maxmemory-policy. You can update this using the `cf update-service` parameters. For how to do this, see Customize an on-demand service instance.

Error when running the save command	
Symptom	You receive the following error message when running `redis-cli SAVE` or issuing the save command using another Redis client: `-ERR`
Cause	This might occur when the Redis server's disk is full.
Solution	A more informative message might be logged in the syslog. For more information, see Syslog Errors.

Unknown command error	
Symptom	You receive the following error message when running `redis-cli COMMAND` or issuing a command using another Redis client: `-ERR unknown command`

Unknown command error	
Cause	For security reasons, certain commands such as `CONFIG`, `SAVE`, `BGSAVE` and `ACL` are not available by default.
Solution	Talk to your operator about the availability of the command.

Techniques for troubleshooting

See the following sections for troubleshooting techniques when using the Cloud Foundry Command-Line Interface (cf CLI) to perform basic operations on a Redis for Tanzu Application Service service instance.

Basic cf CLI operations include `create`, `update`, `bind`, `unbind`, and `delete`.

Debug using the CF CLI

See the following table for Cloud Foundry Command Line Interface (cf CLI) commands commonly used while debugging:

To view the...	Command
API endpoint, org, and space	`cf target`
Service offerings available in the targeted org and space	`cf marketplace`
Apps deployed to the targeted org and space	`cf apps`
Service instances deployed to the targeted org and space	`cf services`
GUID for a specific service instance	`cf service SERVICE-INSTANCE --guid`
Service instance or application logs	`cf tail SERVICE-INSTANCE/APP`

Parse a Cloud Foundry (CF) error message

Failed operations (create, update, bind, unbind, delete) cause an error message. You can retrieve the error message later by running the cf CLI command `cf service INSTANCE-NAME`.

```
$ cf service myservice

Service instance: myservice
Service: super-db
Bound apps:
Tags:
Plan: dedicated-vm
Description: Dedicated Instance
Documentation url:
Dashboard:

Last Operation
Status: create failed
```

```
Message: Instance provisioning failed: There was a problem completing your re
quest.
     Please contact your operations team providing the following information:
     service: redis-acceptance,
     service-instance-guid: ae9e232c-0bd5-4684-af27-1b08b0c70089,
     broker-request-id: 63da3a35-24aa-4183-aec6-db8294506bac,
     task-id: 442,
     operation: create
Started: 2017-03-13T10:16:55Z
Updated: 2017-03-13T10:17:58Z
```

Use the information in the `Message` field to debug further. Provide this information to Support when filing a ticket.

The `task-id` field maps to the BOSH task ID. For more information about a failed BOSH task, use the `bosh task TASK-ID`.

The `broker-request-guid` maps to the portion of the On-Demand Service Broker log containing the failed step. Access the broker log through your syslog aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

Retrieve service instance information

To retrieve information about the service instance that you can use for debugging:

1. Log into the space containing the instance or failed instance.

   ```
   $ cf login
   ```

2. If you do not know the name of the service instance, you can view a listing of all service instances in the space by running:

   ```
   cf services
   ```

   The service instances are listed in the `name` column.

   For example:

   ```
   $ cf services
   Getting services in org my-org / space my-space as user@example.com...
   OK
   name          service      plan                 bound apps    last operat
   ion
   my-instance   p.redis      on-demand-cache                    create succ
   eeded
   ```

3. Retrieve more information about a specific service instance by running:

   ```
   cf service SERVICE-INSTANCE-NAME
   ```

4. Retrieve the GUID of the service instance by running:

```
cf service SERVICE-INSTANCE-NAME --guid
```

This is useful for debugging.

5. If the Log Cache CLI plugin is enabled, you can retrieve logs for the service instance by running:

```
cf tail SERVICE-INSTANCE-NAME/APP-NAME
```

For more information, see Log Cache CLI plug-in.

Retrieve the password for a Redis service instance

If you want to access the Redis server for troubleshooting, you can find a Redis service instance password by creating a new service key.

VMware recommends that you use this key for troubleshooting only, and that you delete the key after troubleshooting by running the command `cf delete-service-key SERVICE-INSTANCE KEY-NAME`.

For instructions on how to retrieve the password, see Retrieve the password for a Redis service instance.

Temporary outages

Redis for Tanzu Application Service service instances can become temporarily inaccessible during upgrades and VM or network failures.

Knowledge base (community)

Find the answer to your question and navigate product discussions and solutions by searching Broadcom Support.

File a support ticket

You can file a support ticket here. Include the error message from `cf service YOUR-SERVICE-INSTANCE`.

To expedite troubleshooting, provide your service broker logs, service instance logs, and BOSH task output. Your cloud operator can obtain these from your error message.

Sample Redis configuration

This topic gives you an example Redis for VMware Tanzu Application Service configuration.

The following is the default `redis.conf` file from an on-demand plan instance:

```
daemonize yes
pidfile /var/vcap/sys/run/redis.pid
port 6379
requirepass 1a1a2bb0-0ccc-222a-444b-1e1e1e1e2222
```

```
# Logging
logfile /var/vcap/sys/log/redis/redis.log
syslog-enabled yes
syslog-ident redis-server
syslog-facility local0

# Persistance
dbfilename dump.rdb
dir /var/vcap/store/redis
appendonly no
appendfilename appendonly.aof
save 900 1
save 300 10
save 60 10000

# Arbitrary Parameters
maxmemory-policy allkeys-lru
slowlog-log-slower-than 10000
slowlog-max-len 128
notify-keyspace-events ""

# Plan Properties:
timeout 3600s
tcp-keepalive 60
maxclients 10000
rename-command EVAL "EVAL"
rename-command EVALSHA "EVALSHA"

# Command Masking
rename-command CONFIG "A-B-Ab1AZec_-AaC1A2bAbB22a_a1Baa"
rename-command SAVE "SAVE"
rename-command BGSAVE "BGSAVE"
rename-command DEBUG ""
rename-command SHUTDOWN ""
rename-command SLAVEOF ""
rename-command SYNC ""
rename-command ACL "O_1awa99Ameoyzc3h7sH44XHmtvCKO_t"
maxmemory 1775550873
```