Pivotal

PRODUCT DOCUMENTATION

Redis for PCF®

Version 1.9

User's Guide

Table of Contents

Redis for PCF

Page last updated:

Note: Redis for PCF v1.9 is no longer supported. The support period for v1.9 has expired. To stay up-to-date with the latest software and security upgrades, upgrade to Redis for PCF v1.10 or later.

This is documentation for the Redis for Pivotal Cloud Foundry (PCF) service tile. This tile can be downloaded from Pivotal Network.

This documentation:

- Describes the features and architecture of Redis for PCF.
- Instructs the PCF operator on how to install, configure, maintain, and backup Redis for PCF.
- Instructs the app developer on how to choose a service plan, create and delete Redis service instances, and bind an app.

Product Snapshot

Element	Details
Version	v1.9.9
Release date	March 7, 2018
Software component version	Redis OSS v3.2.11
Compatible Ops Manager version(s)	v1.10.9+, v1.11.0*
Compatible Elastic Runtime version(s)	v1.10.9+, v1.11.0
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	Yes

** Ops Manager v1.9+ have support for securing metrics and logs through loggregator with TLS — this is enabled in Redis for PCF v1.9.*

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 is easy to install and configure and is popular with engineers as a straightforward NoSQL data store. It is used for everything from a quick way to store data for development and testing through to enterprise-scale apps like Twitter.

About Redis for PCF

Redis for PCF packages Redis for easy deployment and operability on PCF. Redis for PCF can be used as a datastore or cache. Metrics and logging enable operators to monitor Redis health. Operators can configure scheduled backups to a variety of destinations. There are manual backup and restore scripts for individual service instances. New features are regularly introduced. Upgrading Redis for PCF is straightforward and preserves configuration and data.

On-Demand Service

Redis for PCF v1.8 and later includes on-demand provisioning of Redis instances based on the On-Demand Services SDK. This enables an app developer to create a Redis instance as needed from the CF CLI through the `cf create-service` command. The Operator configures the plan options available to the app developers.

For more information, see On-Demand Service Offering.

Dedicated-VM and Shared-VM Services

Redis for PCF still offers Dedicated-VM and Shared-VM plans. With these plans, an Operator specifies at install time how many dedicated VM Redis instances to create.

For more information about the dedicated-VM and shared-VM service, see Dedicated-VM and Shared-VM Service Offerings.

Support for Multiple AZs

As of v1.9, Redis for PCF supports configuring multiple AZs. However, assigning multiple AZs to Redis jobs does not guarantee high availability.

- On-Demand plans can be assigned to any of the configured availability zones. However, each instance still operates as a single node with no clustering.
- Shared-VM instances run on a single node in just one of the configured availability zones and are therefore not highly available.
- Dedicated-VM instances can be assigned to any of the configured availability zones. However, each instance still operates as a single node with no clustering. This separation over availability zones provides no high availability.

Upgrading to the Latest Version

For information on how to upgrade and the supported upgrade paths, see Upgrading Redis for PCF.

Enterprise-Ready Checklist

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

Plans and Instances		**More Information**
On-Demand, Dedicated-VM and Shared-VM plans	Redis for PCF provides On-Demand, dedicated VM and shared VM service plans.	Plans
Customizable plans	For the On-Demand Plan and Dedicated-VM plan, the operator can customize the VM and disk size.	Configuring
Installation and Upgrades		**More Information**
Product upgrades	Redis for PCF can be upgraded from v1.7 tiles	Upgrading Redis for PCF
Deployment Smoke Tests	Redis for PCF installation and upgrade runs a post deployment BOSH errand that validates basic Redis operations	Smoke Tests
Maintenance and Backups		**More Information**
Operational Monitoring and Logging	Redis for PCF v1.9 provides metrics for monitoring On-Demand plan usage and quotas and syslog redirection to external log ingestors.	Monitoring Redis for PCF
Backup and Restore	Redis for PCF v1.9 includes automatic backups on a configurable schedule to a variety of destinations for Dedicated-VM and Shared-VM plans, as well as scripts for backup and restore of service instances. The On-Demand plan does not offer this.	Manual Backup and Restore of Redis for PCF
Scale and Availability		**More Information**
Scale	Redis for PCF has been tested with 60 GB of data	
On-Demand Plan	Redis for PCF provides up to 50 on-demand instances across plans	
Ability to Scale Up / Down	Operators can scale VMs up, but not down	Configuring
Rolling Deployments	Redis for PCF does not support rolling deployments because it is single node; the service is unavailable during upgrades.	Upgrades
AZ Support	Assigning multiple AZs to Redis jobs does not guarantee high availability.	About Multiple AZs in Redis for PCF
Encryption		**More Information**
Encrypted Communication in Transit	Redis for PCF has been tested successfully with the BOSH IPsec Add-on	Securing Data in Transit with the IPsec Add-on

More Information

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

For more information about…	See…
Is Redis for PCF right for you?	Enterprise-Ready Checklist
Product compatibility	Product Version Matrix
How to upgrade Redis for PCF	Upgrading Redis for PCF
How to use Redis	Redis Documentation

Redis for PCF and Other PCF Services

Some PCF services offer *on-demand* service plans. These plans let developers provision service instances when they want.

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

The following PCF services offer on-demand service plans:

- MySQL for PCF v2.0 and later
- RabbitMQ for PCF
- Redis for PCF
- Pivotal Cloud Cache (PCC)

These services package and deliver their on-demand service offerings differently. For example, some services, like Redis for PCF, have one tile, and you configure the tile differently depending on whether you want on-demand service plans or pre-provisioned service plans.

For other services, like PCC, you install one tile for on-demand service plans and a different tile for pre-provisioned service plans.

The following table lists and contrasts the different ways that PCF services package on-demand and pre-provisioned service offerings.

PCF service tile	Standalone product related to the service	Versions supporting on demand	Versions supporting pre-provisioned
RabbitMQ for PCF	Pivotal RabbitMQ	v1.8 and later	All versions
Redis for PCF	Redis	v1.8 and later	All versions
MySQL for PCF	MySQL	v2.x (based on Percona Server)	v1.x (based on MariaDB and Galera)
PCC	Pivotal GemFire	All versions	*NA*
GemFire for PCF	Pivotal GemFire	*NA*	All versions

Feedback

Please provide any bugs, feature requests, or questions to the Pivotal Cloud Foundry Feedback list.

Release Notes

Page last updated:

Redis for PCF v1.9.9

March 7, 2018

This release updates the stemcell verison.

Compatibility

- Compatible with stemcells 3468.25+.
- Compatible with Service Metrics v1.5.11, Service Backups for PCF v18.1.9, and On-Demand Service Broker v0.19.0.

Known Issues

- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.8

February 7, 2018

Bug Fixes

- Fixes a bug that blocks tile upgrades in some Pivotal Cloud Foundry (PCF) installations.

Compatibility

- Compatible with stemcells 3445.22+.
- Compatible with Service Metrics v1.5.11, Service Backups for PCF v18.1.9, and On-Demand Service Broker v0.19.0.

Known Issues

- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.7

February 2, 2018

Compatibility

- Compatible with stemcells 3445.22+.
- Compatible with Service Metrics v1.5.11, Service Backups for PCF v18.1.9, and On-Demand Service Broker v0.19.0.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.6

January 22, 2018

Compatibility

- Compatible with stemcells 3445.22+.
- Compatible with Service Metrics v1.5.11, Service Backups for PCF v18.1.9, and On-Demand Service Broker v0.19.0.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.5

November 10, 2017

Bug Fixes

- Logs for service-backup and service-metrics releases are no longer truncated when syslog is not configured.

Compatibility

- Compatible with OS Redis v3.2.11
- Compatible with stemcells 3445.16+.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.4

September 22, 2017

Bug Fixes

- Lua Scripting is disabled by default for the on-demand service. It was previously enabled by default. Pivotal recommends that Lua Scripting be disabled.
- Allows the on-demand broker resource to be set to 0 or 1. This eases the ability of operators to avoid setting up the on-demand service.

Compatibility

- Compatible with PCF v1.10.9+.
- Compatible with stemcells 3445.11+. This is a change from the previous stemcell line of 3363.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- `lua-timeout-limit`, a Redis configuration that is exposed to app developers through arbitrary parameters in v1.9, does not have any effect due to a bug in Redis v3.2.8.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Logs for service-backup and service-metrics releases are truncated when syslog is not configured.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.3

August 24, 2017

Bug Fixes

- Ensures all eviction policies are enforced when on-demand instance memory is full.

Compatibility

- Uses On-Demand Service Broker v0.17.0
- Compatible with PCF v1.10.9+.
- Uses a Syslog Migration Release that allows the operator to configure the log format to be either the pre-existing one or RFC 5424. RFC 5424 is the standard formatting across PCF services. The previous format will eventually be deprecated.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The on-demand broker cannot be set to 0. This means that the instructions for disabling the on-demand service cannot be followed.
- Tile states it is compatible with OM v1.10.3+ but the minimum is v1.10.9+
- `lua-timeout-limit`, a Redis configuration that is exposed to app developers through arbitrary parameters in v1.9, does not have any effect due to a bug in Redis v3.2.8.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- Logs for service-backup and service-metrics releases are truncated when syslog is not configured.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.2

July 20, 2017

Compatibility

- Compatible with PCF v1.10.9+.

Bug Fixes

- On-demand service instances correctly picks up new Redis releases and stemcells when upgraded. This addresses a prior bug where pre-existing instances were not upgrading Redis Releases or Stemcells.
- On-demand smoke tests are skipped if the quota for on-demand service instances has been met. This addresses a prior bug where the smoke tests would fail if the quota had been met.
- Includes a bug fix in the control script for process-destroyer which prevents process-watcher from restarting periodically (every 10,000 seconds).

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The on-demand broker cannot be set to 0. This means that the instructions for disabling the on-demand service cannot be followed.
- Tile states it is compatible with OM 1.10.3+ but the minimum is 1.10.9+
- `lua-timeout-limit` a Redis configuration that is exposed to app developers through arbitrary parameters in v1.9, do not have any effect due to a bug in Redis v3.2.8.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services and will be addressed in a future release.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.
- On-demand instances may not correctly enforce eviction policies when memory is full.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

Redis for PCF v1.9.0

Release Date: June 30, 2017

Compatibility

- This release is compatible with PCF v1.10.9+.
- The on-demand service leverages the On-Demand Broker v0.16.0
- There is a slight change in how syslog forwarding is turned off and on via a new toggle.

Known Issues

- Due to a bug with the properties migration logic, tile upgrades in some PCF installations may be blocked.
- The on-demand broker cannot be set to 0. This means that the instructions for disabling the on-demand service cannot be followed.
- Tile states it is compatible with OM 1.10.3+ but the minimum is 1.10.9+
- New on-demand service instances does not pick up new stemcells beyond 3363.26.
- A bug exists in the control script for process-destroyer which causes process-watcher to restart periodically (every 10,000 seconds). The bug won't cause problems outside of making it more difficult to distinguish real issues reported by monit.
- New on-demand service instances do not pick up new stemcells beyond 3363.24.
- Upgrades of on-demand service instances from v1.8.0, v1.8.1 and v1.8.2 to v1.9.0 do not take effect.
- `lua-timeout-limit` a Redis configuration that is exposed to app developrs through arbitrary parameters in v1.9, does not have any effect due to a bug in Redis v3.2.8.
- Redis smoke tests fail if the number of instances of a certain plan have reached the global quota. Either remove the smoke tests errand or increase your quota.
- The redis-odb service broker listens on port `12345`. This is inconsistent with other services and will be addressed in a future release.
- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run-rule. For more information about this unexpected behavior, see Errand Run Rules.

- On-demand instances may not correctly enforce eviction policies when memory is full.
- Default persistence is set to full persistence using an AOF file. If an instance is restarted frequently (for example, for upgrades), this file can grow significantly, leading to very large persistent disk usage. If your Redis instance has significantly larger persistent disk usage than expected, check the size of your `appendonly.aof` file (usually at `/var/vcap/store/redis`) to verify if this is the source of the usage. If so, you can mitigate this by running the BGREWRITEAOF command.

On-Demand Service Offering

Page last updated:

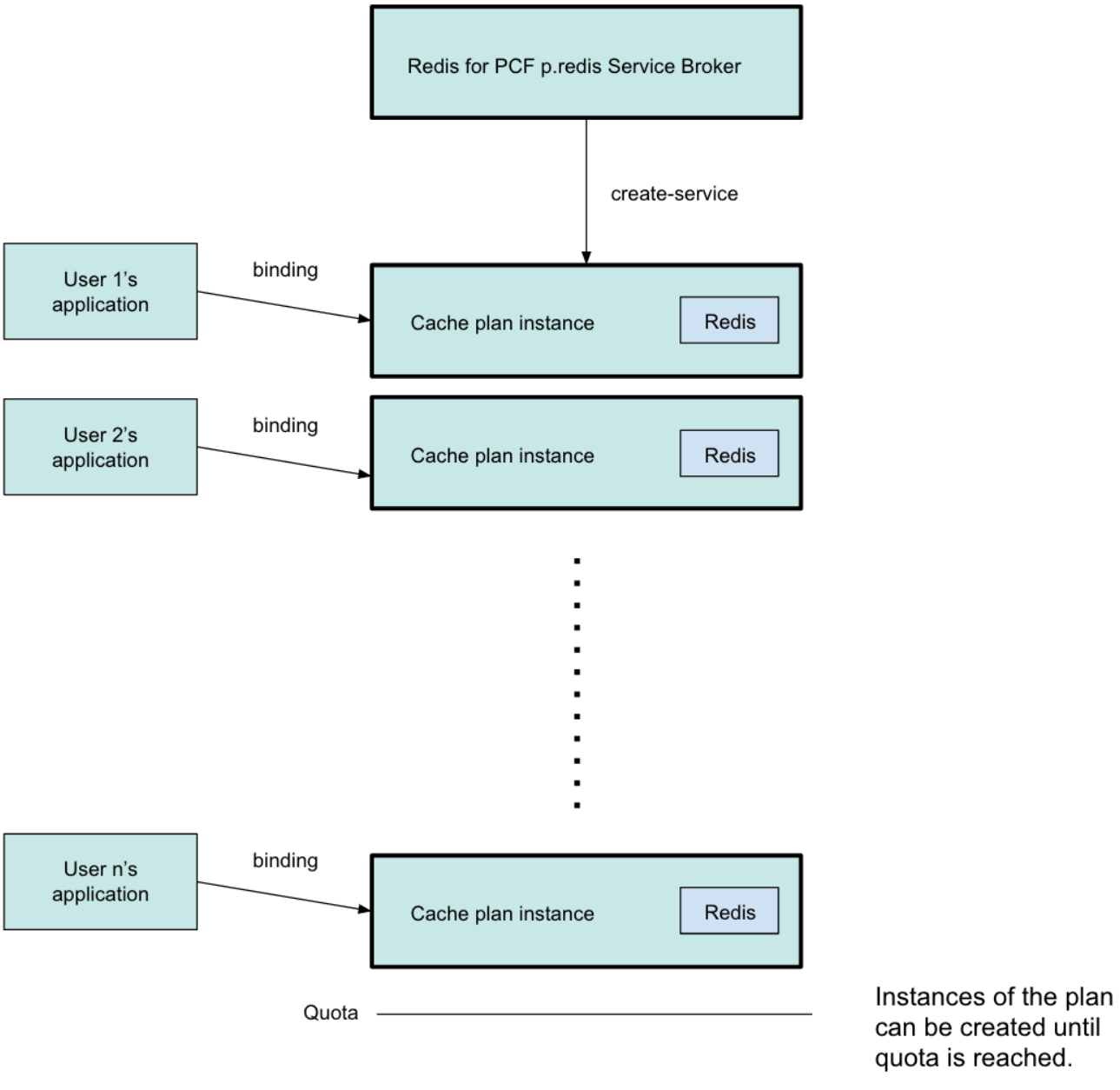
Redis for PCF offers On-Demand, Dedicated-VM, and Shared-VM service plans. This section describes the architecture, lifecycle, and configurations of the on-demand plan, as well as networking information for the on-demand service. For similar information for the Dedicated-VM and Shared-VM plans, see Dedicated-VM and Shared-VM Service Offerings.

Architecture Diagram for On-Demand Plan

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

The p.redis service broker manages the On-Demand service plan instances.

Plans are configured by the operator in Ops Manager. Instances of the plans are created by the App Developer on-demand up to a set quota. The per-plan and global quota is specified by the operator.



On-Demand Service Plans

Three On-Demand Cache Plans

On-demand plans are best fit for cache use cases and are configured as such by default.

Redis for PCF offers three on-demand plans as the `p.redis` service within the PCF Redis tile. Below is a description of each plan as it appears in the cf marketplace and its intended use case.

- **Small Cache Plan**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~1 GB of memory and >3.5 GB of persistent disk.
- **Medium Cache Plan**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~2 GB of memory and >10 GB of persistent disk.
- **Large Cache**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~4 GB of memory and >14 GB of persistent disk.

For each service plan, the operator can configure the **Plan name**, **Plan description**, **Server VM type** and **Server Disk type**, or choose to disable the plan completely.

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. Once 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. See Configuration for On-Demand Service Plans for more information.
- The default `maxmemory-policy` is `allkeys-lru` and can be updated for other cache policies.
- The maximum number of instances is managed by a per-plan and global quota. The maximum number of instances cannot surpass 50. For information on 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 Additional Redis Configurations.

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 on 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 disabled from the plan page in Ops Manager.

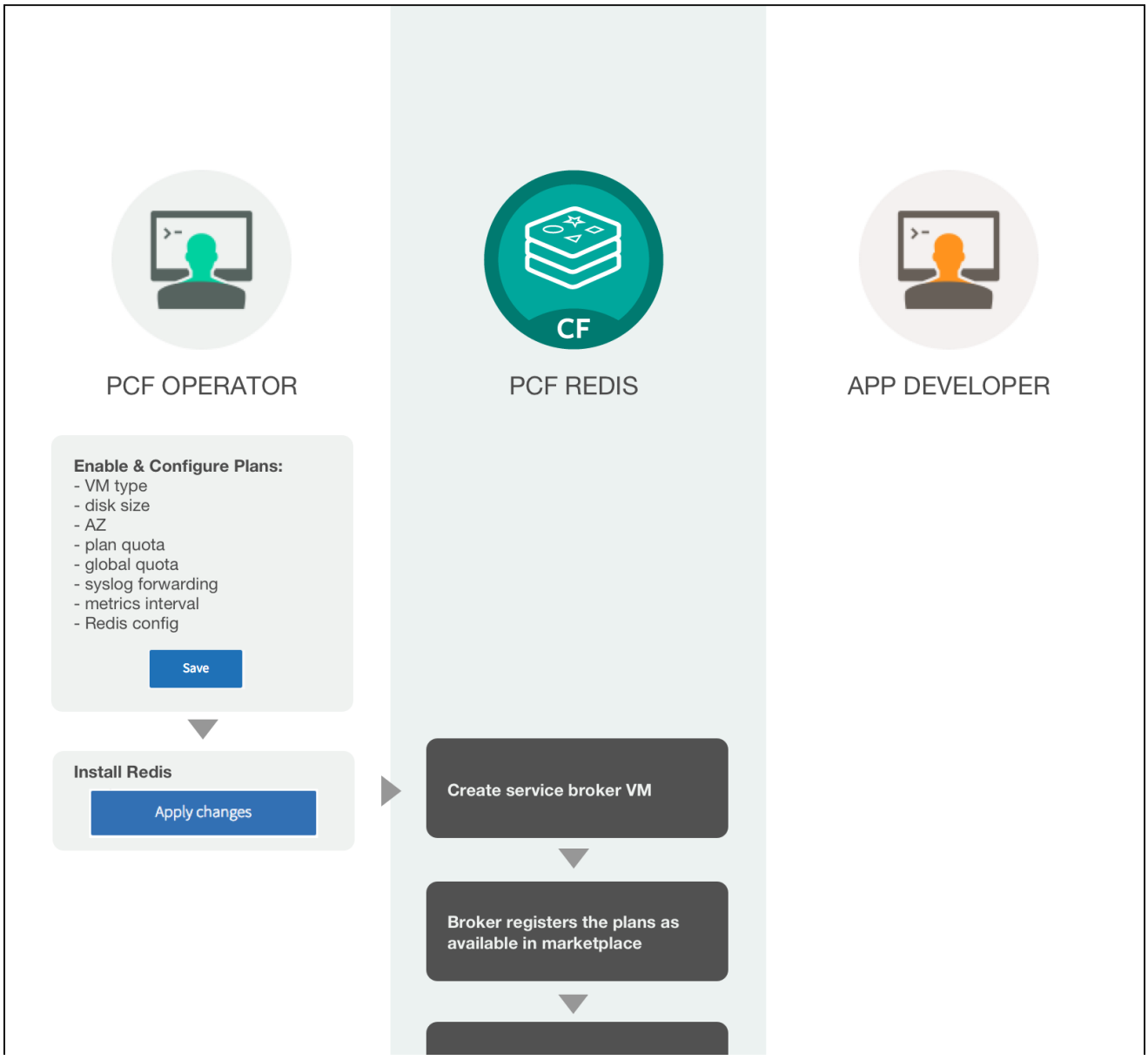
Known Limitations for On-Demand Service Plans

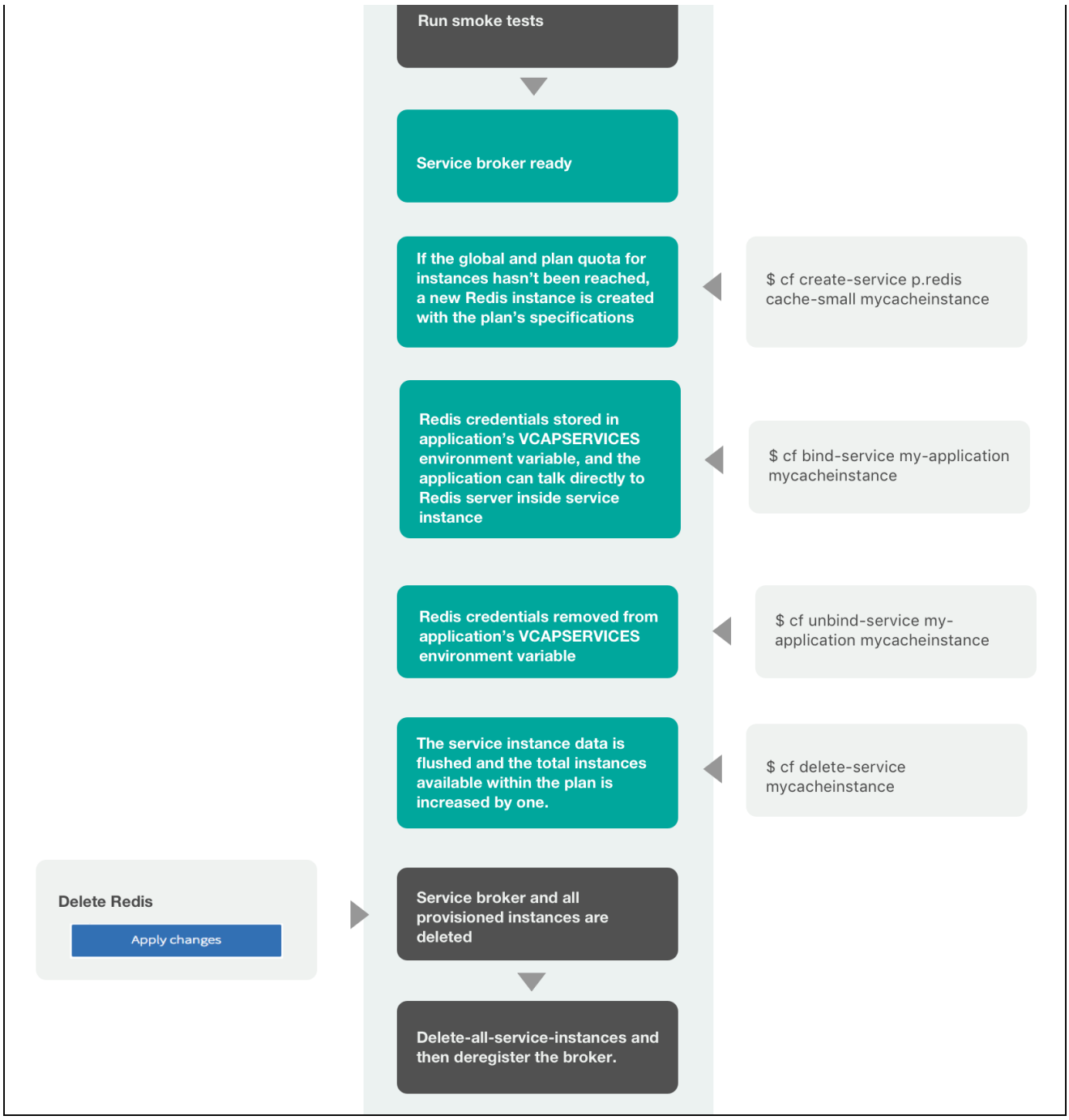
Limitations for the on-demand service include:

- 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 the operator updates the VM size, disk size, or the Redis configuration settings (enabling Lua Scripting, max-clients, timeout, and TCP keep-alive), these settings are implemented in all instances already created.
- Automatic backups are not available for on-demand plans.

Lifecycle for On-Demand Service Plan

The image below shows the lifecycle of Redis for PCF, from an operator installing the tile, through an app developer using the service, to an operator deleting the tile.





Dedicated-VM and Shared-VM Service Offerings

Page last updated:

Redis for Pivotal Cloud Foundry (PCF) offers On-Demand, Dedicated-VM, and Shared-VM service plans. This section describes the architecture, lifecycle, and configurations of Dedicated-VM and Shared-VM plans. For similar information for the On-Demand service plan, see On-Demand Service Offering.

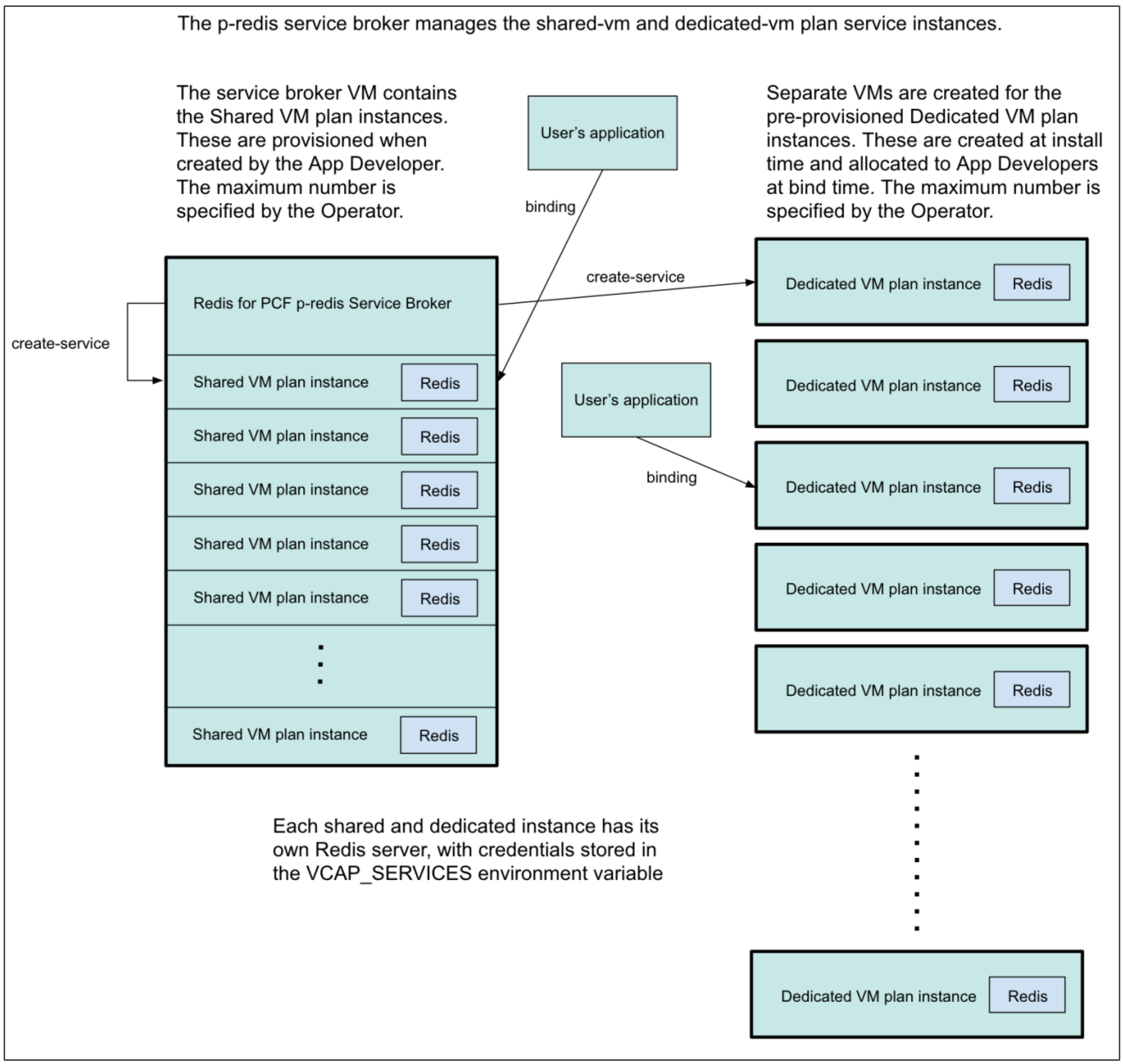
About the Pre-Provisioned Plans

Redis for PCF includes two pre-provisioned service plans:

- **Dedicated-VM Plan**
  An instance of this plan, provisions a single Redis process, on a single **dedicated VM**. This plan is suitable for production workloads, and workloads that require isolation or dedicated hardware.
- **Shared-VM Plan**
  An instance of this plan provisions a single Redis process on a single **shared VM**. This plan is suitable for workloads which do not require dedicated hardware.

Architecture Diagram for Shared and Dedicated Plans

This diagram shows how the architecture of the service broker and Shared-VM and Dedicated-VM plans and how the user's app binds to a Redis instance.



Configuration for Dedicated-VM and Shared-VM Service Plans

For Dedicated-VM and Shared-VM plans, the default Redis configurations cannot be changed. A sample `redis.conf` from a Dedicated-VM plan instance is provided here.

- 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 is configured for both `RDB` and `AOF`.
- The default maximum number of connections, `maxclients`, is set at 10000 but this number is adjusted by Redis according to the number of file handles available.
- Replication and event notification are not configured.

Configuration for the Dedicated-VM Service Plan

An instance of this plan, provisions a single Redis process, on a single **dedicated VM**. This plan is suitable for production workloads, and workloads that require isolation or dedicated hardware.

Operator Notes for the Dedicated-VM Service Plan

- The following Redis commands are enabled:
  - `MONITOR`
  - `SAVE`
  - `BGSAVE`
  - `BGREWRITEAOF`
- The `maxmemory` value for the Redis process is set to be 45% of the RAM for that instance.
- The persistent disk should be set to be at least the size of the RAM available to the VM or greater, in order to account for the final and temporary RDB file generated by the Redis background save.
- This plan deploys the operator-configured number of dedicated Redis VMs alongside a single service broker VM.
- These instances are pre-provisioned during the deployment of the tile from Ops Manager into a **pool**. The VMs are provisioned and configured with a Redis process ready to be used when an instance of the `dedicated-vm` plan is requested.
- A default deployment will provision `5 instances` of the `dedicated-vm` plan into the **pool**. This number can be increased on the `Resource Config` tab in Ops Manager, either in the initial deployment, or subsequently thereafter. The number of VMs **cannot** be decreased once deployed.
- When a user provisions an instance, it is marked as in use and taken out of the **pool**.
- When a user deprovisions an instance, the instance is cleansed of any data and configuration to restore it to a fresh state and placed back into the pool, ready to be used again.
- This plan can be disabled by setting the number of instances of the `Dedicated node` job in Ops Manager to `0`.
- The number of Dedicated-VM plan instances available to developers is set by the operator. Configurations of up to 100 Dedicated-VM plan instances have been tested.
- You can disable this plan by setting the number of instances of the `Dedicated node` job in Ops Manager to `0`.

Known Limitations of the Dedicated-VM Service Plan

Limitations of the `dedicated-vm` plan include:

- No ability to change the Redis configuration. The `CONFIG` command is disabled.
- Cannot scale down the number of VMs on the plan once deployed.
- Cannot scale down the size of VMs on the plan once deployed (this protects against data loss).

Configuration for the Shared-VM Service Plan

An instance of this plan provisions a single Redis process on a single **shared VM**. This plan is suitable for workloads which do not require dedicated hardware.

Operator Notes for the Shared-VM Plan

- This plan deploys a Redis instance in a shared VM and a single service broker VM.
- This plan can be disabled by setting the `Max instances limit` on the `Shared-VM Plan` tab in Ops Manager to be `0`.
- The maximum number of instances can be increased from the default 5 to a value of your choosing. If you increase the number of instances that can be run on this single VM, you should consider increasing the resources allocated to the VM. In particular RAM and CPU. You can overcommit to some extent, but may start to see performance degradations.
- You can also increase the maximum amount of RAM allocated to each Redis process (service instance) that is running on this VM
- If you decrease the service instance limit, any instances that are running where the count is now greater than the limit are not terminated. They are left to be removed naturally, until the total count drops below the new limit you cannot create any new instances.

  For example if you had a limit of 10 and all were used and reduced this to 8, the two instances will be left running until you terminate them yourself.
- 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. See Configuring Service Plans for details.

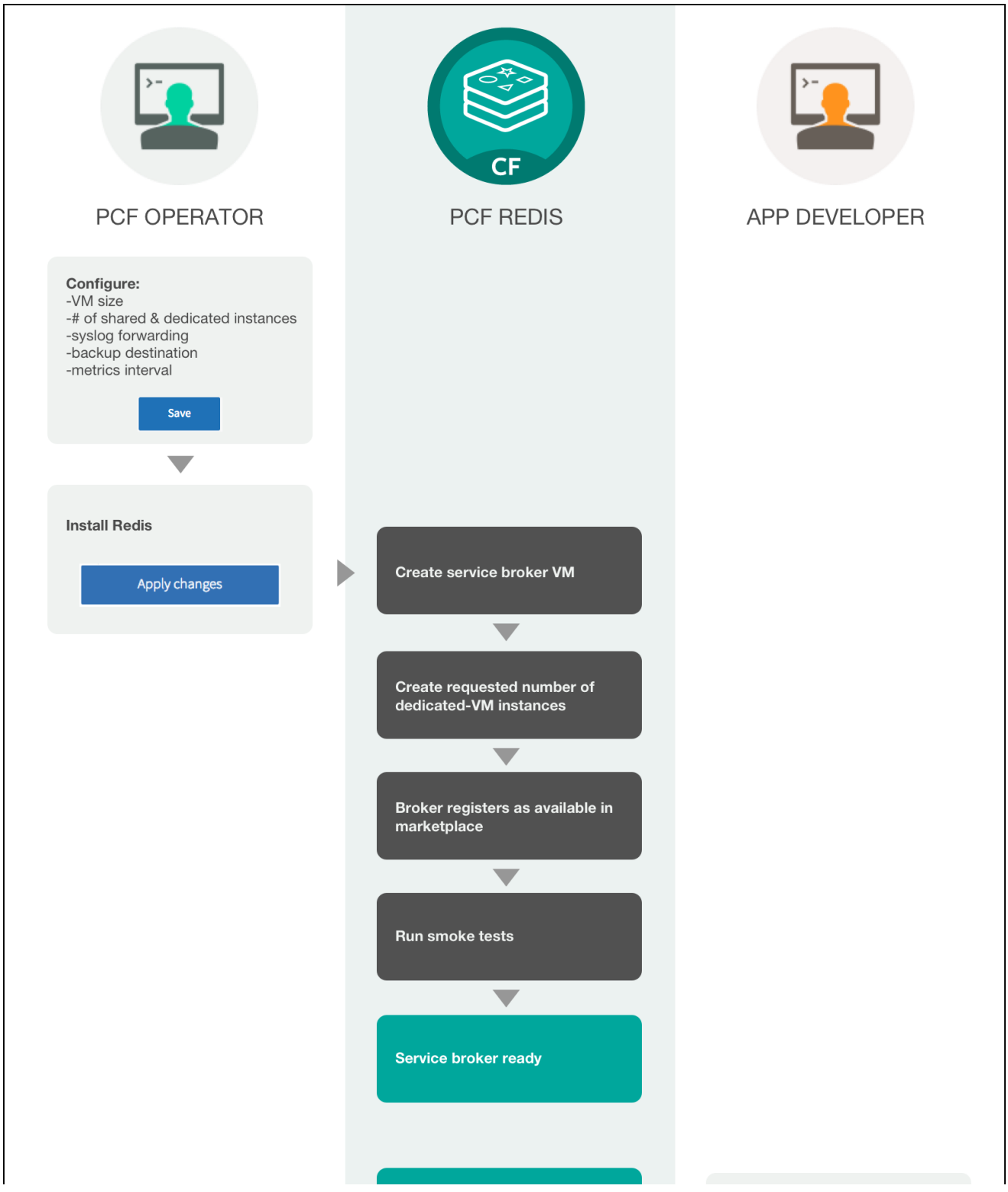
Known Limitations of the Shared-VM Plan

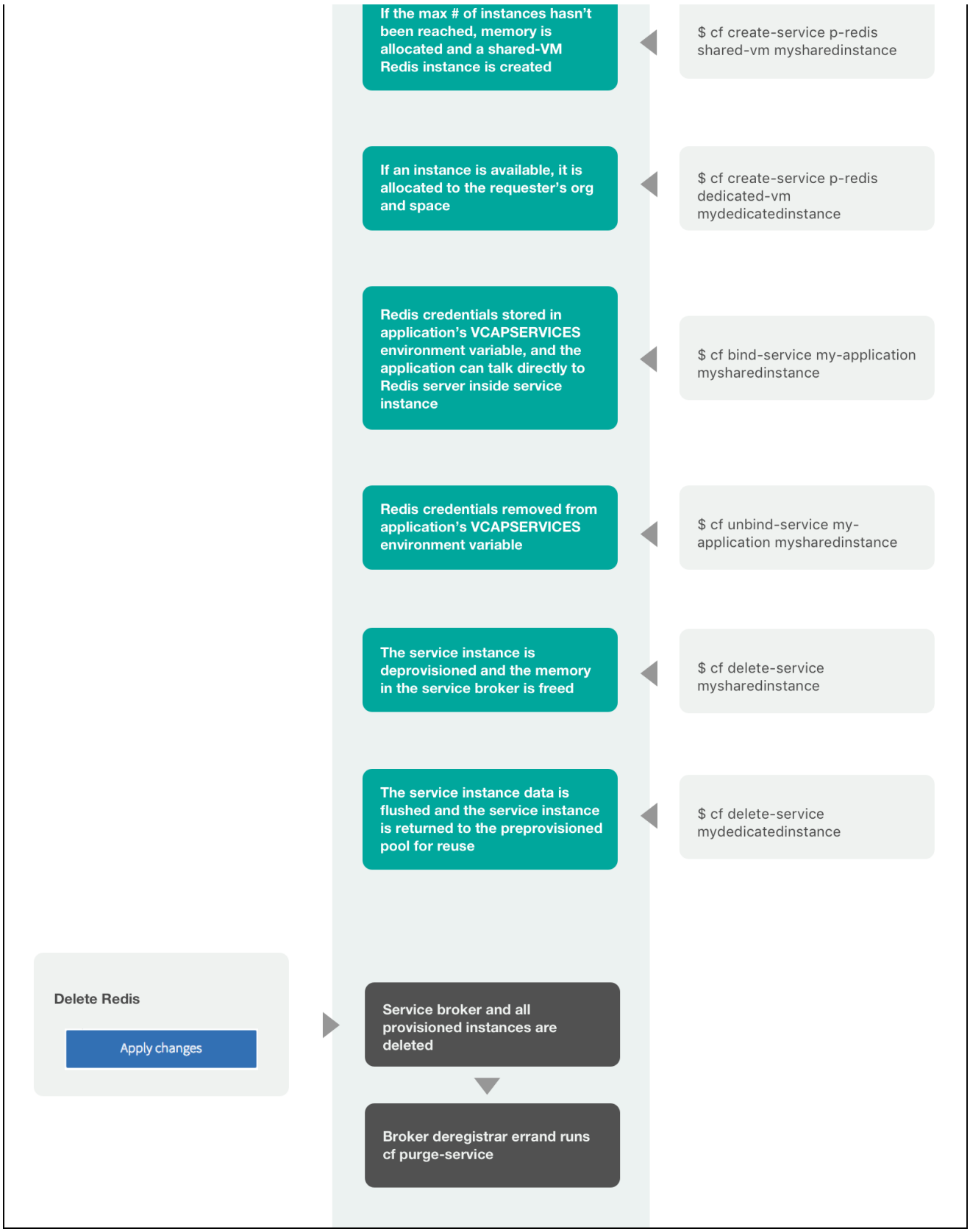
Limitations of the `shared-vm` plan include:

- It cannot be scaled beyond a single VM.
- The following commands are disabled: `CONFIG`, `MONITOR`, `SAVE`, `BGSAVE`, `SHUTDOWN`, `BGREWRITEAOF`, `SLAVEOF`, `DEBUG`, and `SYNC`.
- Constraining CPU and/or disk usage is not supported.
- The Shared-VM plan does not manage ‘noisy neighbor’ problems so it is not recommended for production apps.

Lifecycle for Dedicated-VM and Shared-VM Service Plans

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





Networking for On-Demand Services

Page last updated:

This section describes networking considerations for the Redis for Pivotal Cloud Foundry (PCF) on-demand service.

BOSH 2.0 and the Service Network

Before BOSH 2.0, cloud operators pre-provisioned service instances from Ops Manager. In the Ops Manager Director **Networking** pane, they allocated a block of IP addresses for the service instance pool, and under **Resource Config** they provisioned pool VM resources, specifying the CPU, hard disk, and RAM they would use. All instances had to be provisioned at the same level. With each `create-service` request from a developer, Ops Manager handed out a static IP address from this block, and with each `delete-service` it cleaned up the VM and returned it to the available pool.

With BOSH 2.0 dynamic networking and Cloud Foundry asynchronous service provisioning, operators can now define a dynamically-provisioned service network that hosts instances more flexibly. The service network runs separate from the PCF default network. While the default network hosts VMs launched by Ops Manager, the VMs running in the service network are created and provisioned on-demand by BOSH, and BOSH lets the IaaS assign IP addresses to the service instance VMs. Each dynamic network attached to a job instance is typically represented as its own Network Interface Controller in the IaaS layer.

Operators enable on-demand services when they deploy PCF, by creating one or more service networks in the Ops Manager Director **Create Networks** pane and selecting the **Service Network** checkbox. Designating a network as a service network prevents Ops Manager from creating VMs in the network, leaving instance creation to the underlying BOSH.

When they deploy an on-demand service, operators select the service network when configuring the tile for that on-demand service.

Default Network and Service Network

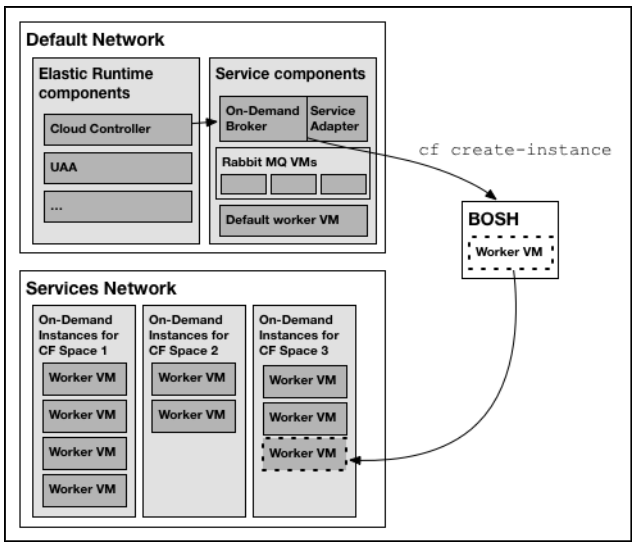
Like other on-demand PCF services, on-demand Redis for PCF relies on the BOSH 2.0 ability to dynamically deploy VMs in a dedicated network. The on-demand service broker uses this capability to create single-tenant service instances in a dedicated service network.

On-demand services use the dynamically-provisioned service network to host the single-tenant worker VMs that run as service instances within development spaces. This architecture lets developers provision IaaS resources for their service instances at creation time, rather than the operator pre-provisioning a fixed quantity of IaaS resources when they deploy the service broker.

By making services single-tenant, where each instance runs on a dedicated VM rather than sharing VMs with unrelated processes, on-demand services eliminate the “noisy neighbor” problem when one application hogs resources on a shared cluster. Single-tenant services can also support regulatory compliance where sensitive data must be compartmentalized across separate machines.

An on-demand service splits its operations between the default network and the service network. Shared components of the service, such as executive controllers and databases, run centrally on the default network along with the Cloud Controller, UAA, and other PCF components. The worker pool deployed to specific spaces runs on the service network.

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



Required Networking Rules for On-Demand Services

Prior to deploying any service tile that uses the on-demand broker (ODB), the operator must request the network connections needed to allow various components of Pivotal Cloud Foundry (PCF) to communicate with ODB. The specifics of how to open those connections varies for each IaaS.

The following table shows the responsibilities of the key components in an on-demand architecture.

Key Components	Their Responsibility
BOSH Director	Creates and updates service instances as instructed by ODB
BOSH Agent	BOSH includes an Agent on every VM that it deploys. The Agent listens for instructions from the Director and carries out those instructions. The Agent receives job specifications from the Director and uses them to assign a role, or Job, to the VM.
BOSH UAA	As an OAuth2 provider, BOSH UAA issues tokens for clients to use when they act on behalf of BOSH users.
ERT	Contains the apps that are consuming services
ODB	Instructs BOSH to create and update services, and connects to services to create bindings
Deployed service instance	Runs the given data service (for example, the deployed Redis for PCF service instance runs the Redis for PCF data service)

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

This component…	Must communicate with…	Default TCP Port	Communication direction(s)	Notes
ODB	• **BOSH Director** • **BOSH UAA**	• 25555 • 8443	One-way	The default ports are not configurable.
ODB	**ERT**	8443	One-way	The default port is not configurable.
Errand VMs	• **ERT** • **ODB** • **Deployed Service Instances**	• 8443 • 8080 • 6379 • 12345	One-way	The default ports are not configurable.
BOSH Agent	**BOSH Director**	4222	Two-way	The BOSH Agent runs on every VM in the system, including the BOSH Director VM. The BOSH Agent initiates the connection with the BOSH Director.

				The default port is not configurable.
Deployed apps on ERT	**Deployed service instances**	6379	One-way	This is the default port where Redis is deployed.
ERT	**ODB**	12345	One-way	The default port is not configurable.

For a complete list of ports and ranges used in Redis for PCF, see Network Configuration.

Redis for PCF Security

Page last updated:

Security

Pivotal recommends that Redis for PCF is run in its own network. For more information about creating service networks, see Creating Networks in Ops Manager.

Redis for PCF works with the IPsec Add-on for PCF. For information about the IPsec Add-on for PCF, see Securing Data in Transit with the IPsec Add-on.

To allow this service to have network access you must create Application Security Groups. For more information, see Networks, Security, and Assigning AZs.

Introduction for Operators

Page last updated:

This topic is for PCF operators. It introduces some best practices, but does not provide details about operation.

Recommended Use Cases

Redis for PCF can be used as a cache or as a datastore. On-Demand plans, introduced in Redis for PCF v1.8, are configured by default for cache use cases. Dedicated-VM and Shared-VM plans are designed for datastore use cases.

Redis can be used in many different ways, including:

- Key/value store for strings and more complex data structures including Hashes, Lists, Sets, Sorted Sets
- Session cache: persistence enables preservation of state
- Full page cache: persistence enables preservation of state
- Database cache: cache 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/Counting: increments and decrements of sets and sorted sets using `ZRANGE`, `ZADD`, `ZREVRANGE`, `ZRANK`, `INCRBY`, `GETSET`
- Pub/Sub: built in publish and subscribe operations: `PUBLISH`, `SUBSCRIBE`, `UNSUBSCRIBE`

For information on Redis for PCF service offerings, see the On-Demand Plan and the Dedicated and Shared Plans pages.

Best Practices

Pivotal recommends that operators follow these guidelines:

- **Resource Allocation**—Work with app developers to anticipate memory requirements and to configure VM sizes. Instances of Dedicted-VM and Shared-VM services 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 Plan 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.
- **Monitoring**—Set up a monitoring dashboard for metrics to track the health of the installation. On-Demand instances do not yet have instance-level metrics in v1.8+. See Monitoring Redis for PCF.
- **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. On-Demand instances are configured for cache uses and are not intended for backups. See Configuring Automated Backups and Manually Backing up and Restoring.
- **Using**—Instances of the On-Demand and Dedicated-VM services run on dedicated VMs. Apps in production should have a dedicated instance to prevent performance issues caused by sharing an instance. The Shared-VM service does not provide a Dedicated-VM per instance, and Pivotal recommends that you only use it for development and testing. See the service offerings for the On-Demand Plan and the Dedicated and Shared Plans.

Backing up Dedicated-VM and Shared-VM Instances

You can back up Redis for PCF instances in two ways for Dedicated-VM and Shared-VM instances:

- Configure automatic backups to be run for each instance, across both service plans. For information about setting up automatic backups, see Configuring Automated Backups.
- Create manual backups of individual instances. For information about how to make manual backups of instances, see Manually Backing up and Restoring.

Backups are not available for On-Demand instances.

Monitoring Redis for PCF

Redis Metrics

Redis for PCF emits Redis metrics via the firehose. For details, see Monitoring and Metrics.

Metrics are not currently available at the instance-level for On-Demand instances.

Logging

Syslog can be forwarded to an external log service.

Syslog for On-Demand instances conforms to RFC5424 standards. This format is as follows:

```
<$PRI>$VERSION $TIMESTAMP $HOST $APP_NAME $PROC_ID $MSG_ID
[instance@47450 deployment="$DEPLOYMENT" group="$INSTANCE_GROUP"
 az="$AVAILABILITY_ZONE" id="$ID"] $MESSAGE`
```

An example:

```
<13>1 2017-03-28T15:20:31.490350+00:00 10.0.16.35 redis-server 4951 - [instance@47450 director="us-pws" deployment="service-instance_16c95f89-8d5a-4cb7-839d-79c5d026bd15" grou
```

The following example shows syslogs for Dedicated-VM and Shared-VM instances:

```
Nov 15 17:05:01 10.0.24.10 audispd: [job=dedicated-node index=4]  node=7bfe8b1b-6c
fd-4d33-b704-c9214ce6bb3e type=USER_ACCT msg=audit(1479229501.290:86): pid=6655 ui
d=0 auid=4294967295 ses=4294967295 msg='op=PAM:accounting acct="root" exe="/usr/sbi
n/cron" hostname=? addr=? terminal=cron res=success'
```

For information about how to set up syslog output, see Configure Syslog Output.

Downtime During Redeploys

Downtime is caused by a redeploy of the Redis tile. This section clarifies what changes will trigger a redeploy.

In Ops Manager, any field that changes the manifest will cause a redeploy of the Redis tile. Any property changes in ERT listed here will trigger downtime for the Redis On-Demand Broker. Note that this list is current at the time of writing (May 2017) but that dependencies with changing products mean that it will change.

- Consul Server Resource Config (..cf.consul_server.ips)
- Runtime System Domain ($runtime.system_domain)
- Disable SSL certificate verification for this environment" in ERT (..cf.ha_proxy.skip_cert_verify.value)
- Runtime Apps Domain ($runtime.apps_domain)
- NATS Resource Config (..cf.nats.ips)
- Service Networks in Ops Manager ($self.service_network)

For Redis for PCF v1.8+, downtime for service instances will only occur once the Operator runs `upgrade-all-service-instances` BOSH errand after all tile upgrades have completed successfully. For Redis for PCF v1.7 and earlier, and the Dedicated-VM and Shared-VM Service in Redis for PCF v1.8 and v1.9, downtime is enforced as soon as the operator applies the referenced changes to ERT. Any changes to a field on the Redis tile, will cause BOSH to redeploy the legacy Redis Broker and the On-Demand Broker once `upgrade-all-service-instances` is run.

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.

Smoke Tests

Ops Manager runs Redis for PCF smoke tests as a post-install errand. The operator can also run the smoke tests errand. For more information see Redis for PCF Smoke Tests.

Installing Redis for PCF

Page last updated:

Download and Install the Tile

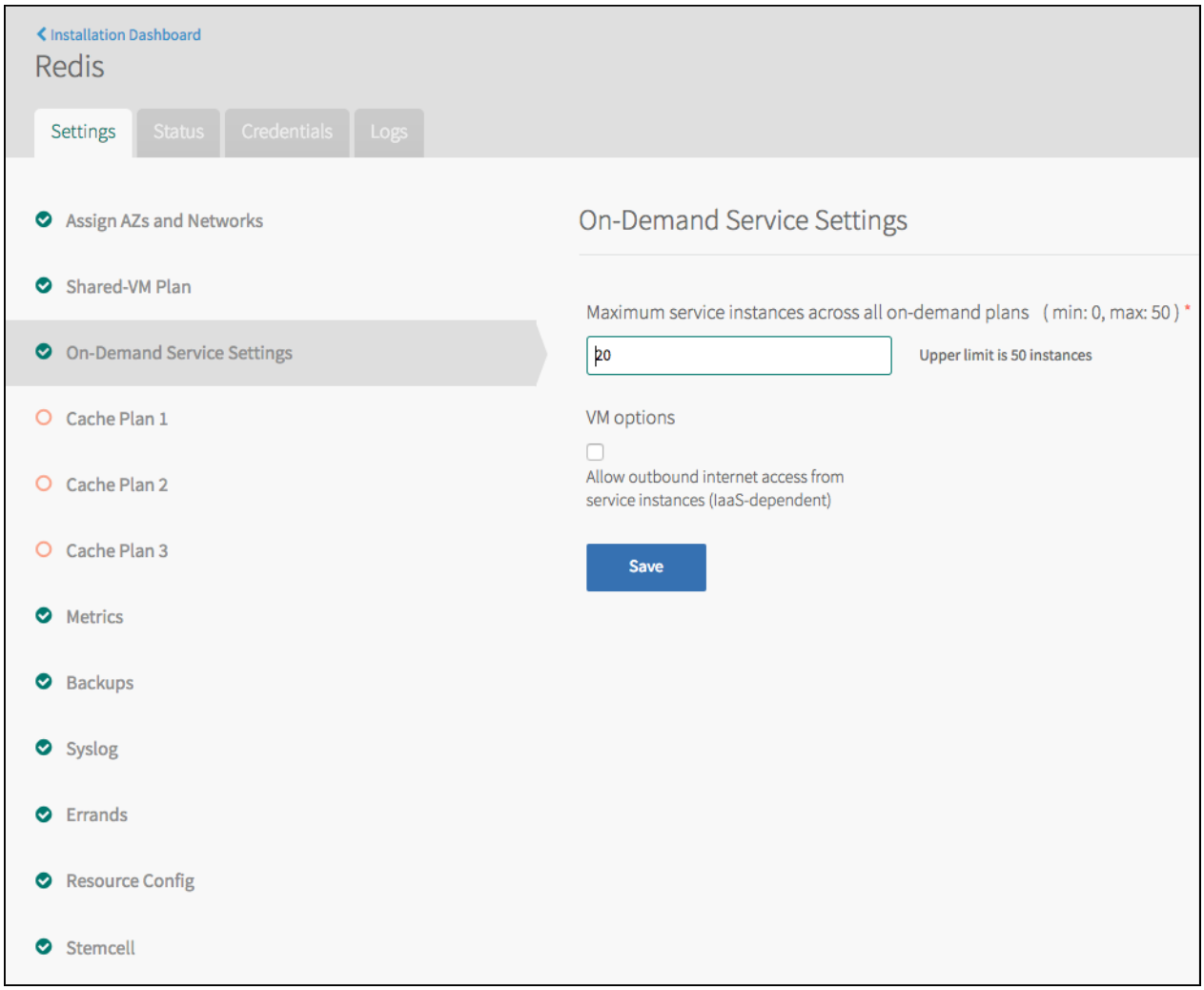
To add Redis for PCF to Ops Manager, follow the procedure for adding PCF Ops Manager tiles:

1. Download the product file from Pivotal Network. Select the latest release from the **Releases:** drop-down menu.
2. Upload the product file to your Ops Manager installation.
3. Click **Add** next to the uploaded product description in the Available Products view to add this product to your staging area.
4. (Optional) Click the newly added tile to configure your possible service plans, syslog draining, and backups.
5. Click **Apply Changes** to install the service.

See the network configuration section for guidance on the ports and ranges used in the Redis service.

Configure Redis for PCF Service Plans

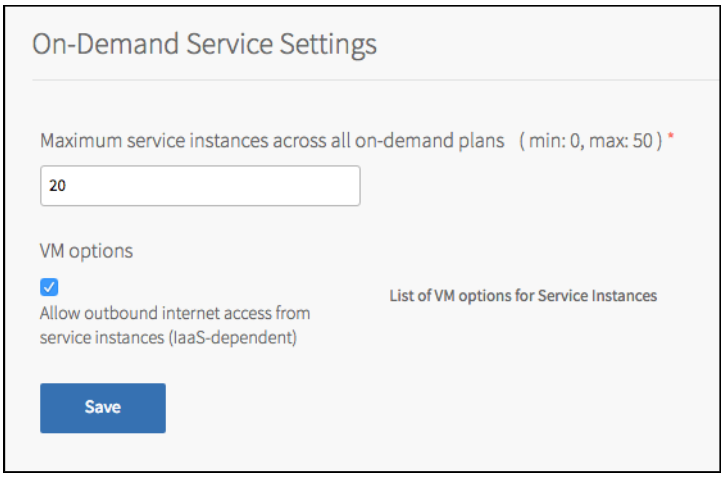
Select the **Redis** tile in the Ops Manager Installation Dashboard to display the configuration page, and allocate resources to Redis service plans.



On-Demand Service

1. Create a service network. From an IAAS perspective, creation of a service network is identical to any other network previously created for tiles on Ops Manager. The only change is that the Operator needs to mark the network as a “Service Network” in Ops Manager to instruct Ops Manager to not perform IP management in that network. If you want to use Redis for PCF without the On-Demand service, you will still need to create an empty service network to install the tile.

2. Click **On-Demand Service Settings**, and then enter the **Maximum service instances across all on-demand plans**. The maximum number of instances you set for all your cache plans combined cannot exceed this number.



   Review the guidance to understand the resource implications for on-demand instances. See Resource Usage Planning for On-Demand plans.

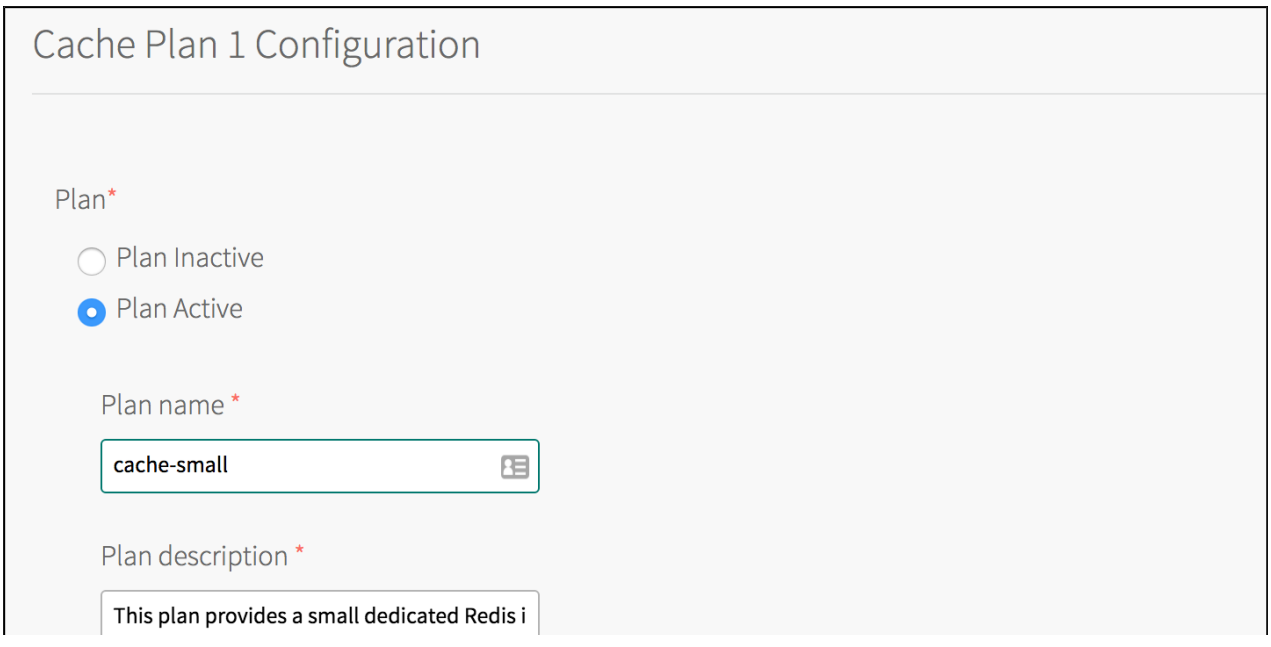
3. Enable the **Allow outbound internet access from service instances** checkbox. This checkbox must be ticked to allow external log forwarding, sending backup artifacts to external destinations, and communicating with an external BOSH blob store.

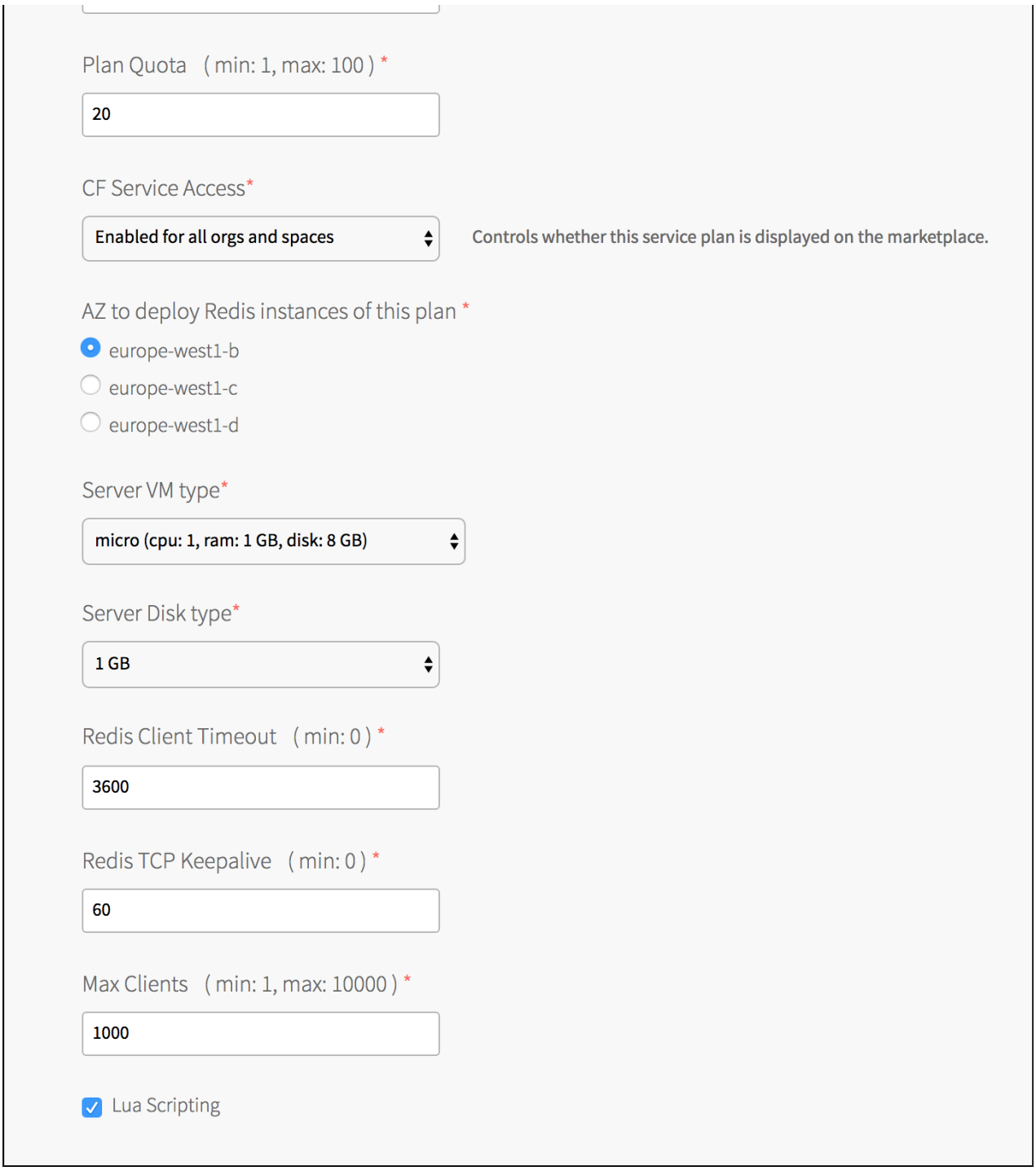
4. Click **Cache Plan 1**, **2**, or **3** to configure it.

   You can configure up to three cache plans with appropriate memory and disk sizes for your use case(s). Resource configuration options may vary on different IAASs.

   The default names of the three cache plans provided reflect that instances of these plans are intended to be used for different cache sizes, as follows:

   - **cache-small**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~1 GB of memory and >3.5 GB of persistent disk
   - **cache-medium**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~2 GB of memory and >7 GB of persistent disk
   - **cache-large**: A Redis instance deployed to a dedicated VM, suggested to be configured with ~4 GB of memory and >14 GB of persistent disk





Configure the following settings for your cache plan(s). Any pre-populated default settings have been pre-configured according to the memory/disk size of each plan.

Field	Description
Plan	Select Active or Passive. An inactive plan does not need any further configuration.
Plan Name	Enter a name that will appear in the service catalog.
Plan Description	Enter a description that will appear in the service catalog. Specify details that will be relevant to app developers.
Plan Quota	App developers can create instances until this quota is reached. For more information, see Setting Limits for On-Demand Service Instances.
CF Service Access	Select a service access level. 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	This is the AZ in which to deploy the Redis instances from the plan. This must be one of the AZs of the service network (configured in the Ops Manager Director tile).
Server VM type	Select the VM type. Pivotal recommends that the **persistent disk should be at least 3.5x the VM memory**

Server Disk type	Select the disk type. Pivotal recommends that the **persistent disk should be at least 3.5x the VM memory**
Redis Client Timeout	Redis Client Timeout refers to the server timeout for an idle client specified in seconds. The default setting is 3600. Adjust this setting as needed.
Redis TCP Keepalive	Redis TCP Keepalive refers to the interval (in seconds) at which TCP ACKS are sent to clients. The default setting is 60. Adjust this setting as needed.
Max Clients	Max Clients refers to the maximum number of clients that can be connected at any one time. Per plan, the default setting is 1000 for small, 5000 for medium and 10000 for large. Adjust this setting as needed.
Lua Scripting	Enable or disable Lua Scripting as needed. Pivotal recommends that Lua Scripting be disabled.

5. Click the **Save** button.

Updating On-Demand Service Plans

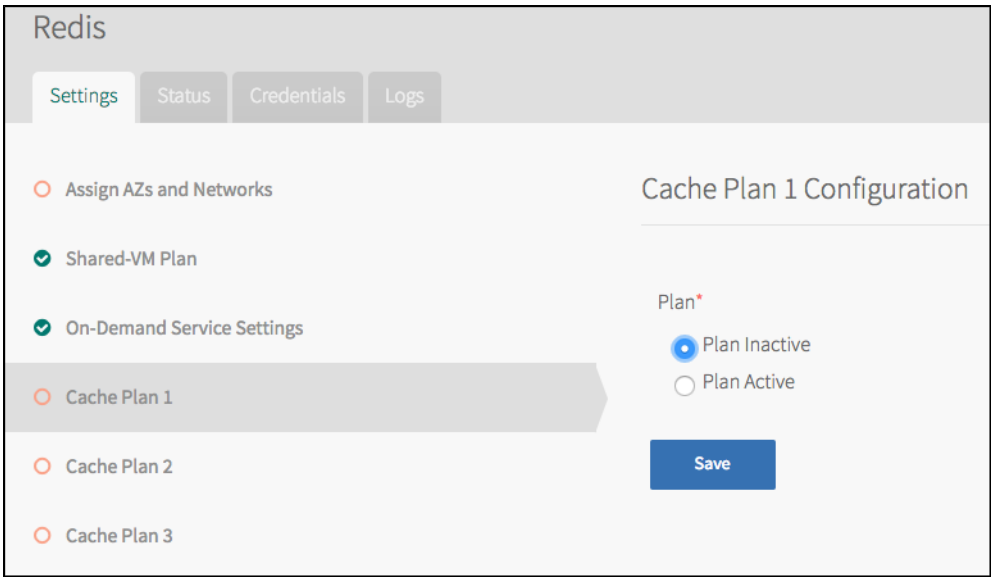
Operators can update certain settings after the plans have been created. If the Operator updates the VM size, disk size, or the Redis configuration settings (enabling Lua Scripting, max-clients, timeout and TCP keep-alive), these settings will be implemented in all instances that are already created.

Operators should not downsize the VMs or disk size as 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.

Removing On-Demand Service Plans

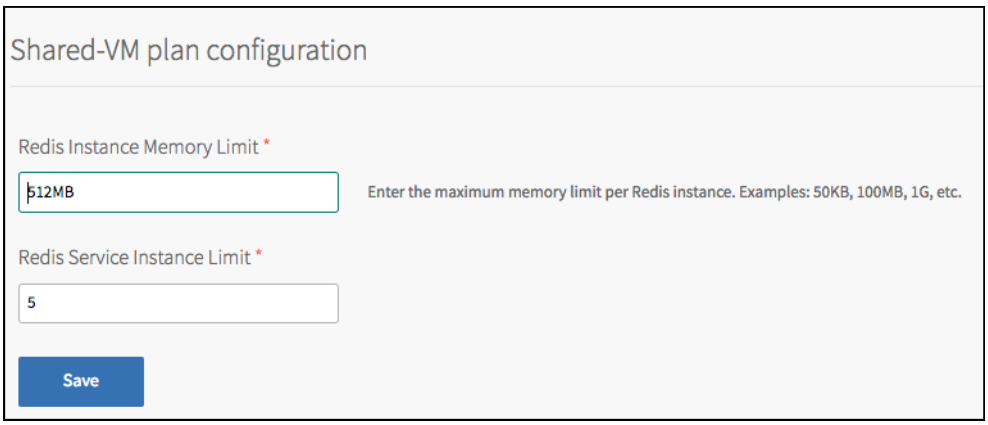
If you want to remove the On-Demand Service from your tile, do the following:

1. Go to the **Resource Config** page on the Redis for PCF tile, and set the **Redis On-Demand Broker** job instances to 0.
2. Navigate to the **Errands** page on the Redis for PCF tile, and set the following errands to **off**:
    - Register On-demand Redis Broker
    - On-demand Broker Smoke Tests
    - Upgrade all On-demand Redis Service Instances
    - Deregister On-demand Redis Broker
3. Create an empty service network. For instructions, see this Knowledge Base article.
4. Go to each of the three Cache Plan pages on the Redis for PCF tile, and set each cache plan to **Plan Inactive**. For example:



Shared-VM Plan

1. Select the **Shared-VM Plan** tab.



2. Configure these fields:

    - **Redis Instance Memory Limit**—Maximum memory used by a shared-VM instance
    - **Redis Service Instance Limit**—Maximum number of shared-VM instances

    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 Removing On-Demand Service Plans above.

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

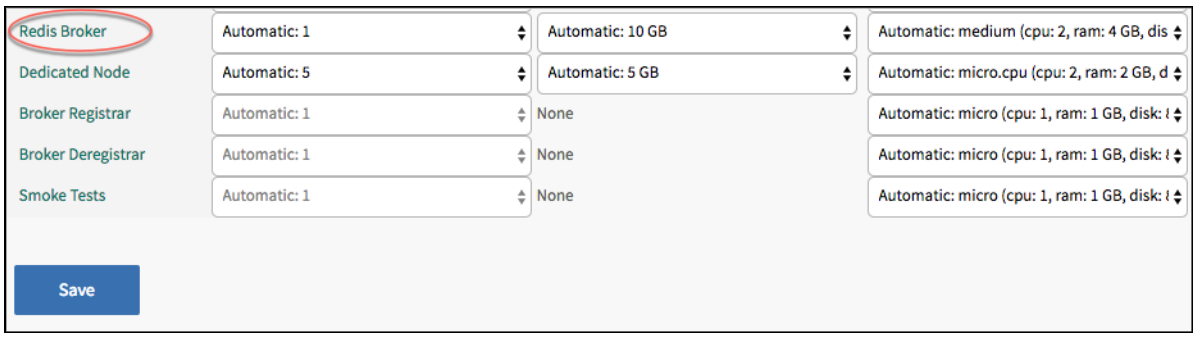
    The Redis broker server runs all of the Redis instances for your Shared-VM plan. From the **Resource Config** page, you can change the CPU, RAM, Ephemeral Disk, and Persistent Disk made available, as needed.

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, estimate the maximum memory that could be used by all your Redis shared-VM instances combined. If that figure is higher than 45% of the Redis broker VM's total system memory, you can do one of the following:

- Decrease the **Redis Instance Memory Limit**.
- Decrease the number of instances in **Redis Service Instance Limit**.
- Increase the RAM for the Redis Broker in the **Resource Config** tab as shown below.



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

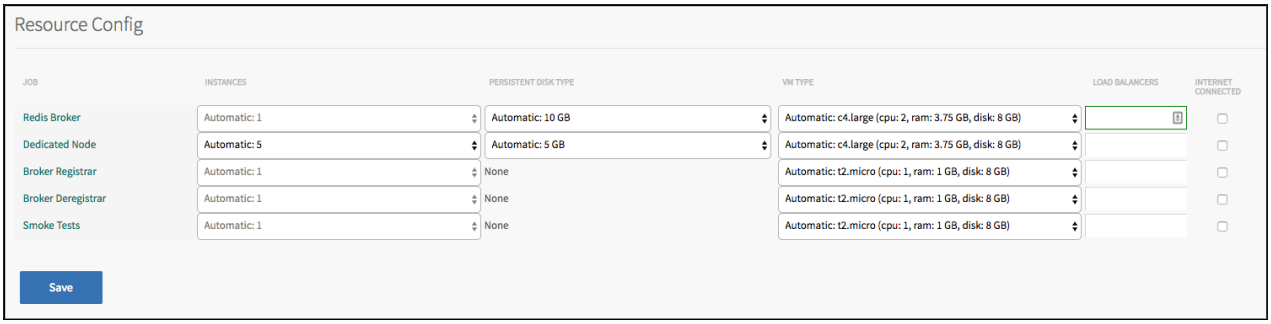
Note: It is possible to configure a larger **Redis Service Instance Limit**, if you are confident that the majority of the deployed instances will not use a large amount of their allocated memory, for example in development or test environments.

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.

Dedicated-VM Plan

Note: As of Redis for PCF v1.11, the on-demand service is at feature parity with the dedicated-VM service. The dedicated-VM service plan will be deprecated. Pivotal recommends using the on-demand service plan.

1. To configure the Dedicated-VM plan, click the **Resource Config** tab to change the allocation of resources for the **Dedicated Node**.



   - The default configuration creates five dedicated nodes (VMs). Each node can run one Redis dedicated-VM instance.
   - You can change the number of dedicated nodes, and configure the size of the persistent and ephemeral disks, and the CPU and RAM for each node.
   - The default VM size is small. It is important that you set the correct VM size to handle anticipated loads.
   - With dedicated-VM plans, there is one Redis service instance on each VM. The maximum memory an instance can use should be at most 45% of the total system RAM on the VM. You can set this with the `maxmemory` configuration. The app can use 100% of `maxmemory` –that is, up to 45% of the system RAM.
   - Pivotal recommends the persistent disk be set to 3.5x the amount of system RAM.

2. Click **Save**.

3. You must disable the On-Demand Service if you do not wan1. 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 Removing On-Demand Service Plans above.t to use it. Please see the directions here.

Configure Resources for Dedicated-VM and Shared-VM Plans

To configure resources for the Shared-VM and Dedicated-VM plans, click the **Resource Config** settings tab on the Redis for PCF tile.

- The Shared-VM plan is on the **Redis Broker** resource.
- The Dedicated-VM plan is on the **Dedicated Node** resource.

The following are the default resource and IP requirements for Redis for PCF when using the Shared-VM or Dedicated-VM plans:

Product	Resource	Instances	CPU	Ram	Ephemeral	Persistent	Static IP	Dynamic IP
Redis	Redis Broker	1	2	3072	4096	9216	1	0
Redis	Dedicated Node	5	2	1024	4096	4096	1	0
Redis	Broker Registrar	1	1	1024	2048	0	0	1
Redis	Broker De-Registrar	1	1	1024	2048	0	0	1
Redis	Compliation	2	2	1024	4096	0	0	1

Disable Shared and Dedicated VM Plans

You can disable Shared and Dedicated VM Plans by doing the following while configuring Redis tile:

1. Ensure at least one On-Demand plan is active.
2. Configure the following tabs:
    - **Shared-VM Plan**:
      a. Set **Redis Service Instance Limit** to 0.
      b. Click **Save**.
    - **Errands**:
      a. Set **Broker Registrar** to Off.
      b. Set **Smoke Tests** to Off.
      c. Set **Broker Deregistrar** to Off.
      d. Leave all four On-Demand errands On.
      e. Click **Save**.
    - **Resource Config**:
      a. Decrease **Redis Broker** Persistent disk type to the smallest size available.
      b. Decrease **Redis Broker** VM type to the smallest size available.
      c. Set **Dedicated Node** Instances to 0.
      d. Click **Save**.

Additional Redis Configurations

The operator can configure further properties per plan beyond memory and disk sizes. Appropriate defaults have been pre-configured according to the memory/disk size of each plan.

Operators can update certain plan 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 keep-alive), these settings will be implemented in all instances already created. Operators should not downsize the VMs or disk size as this can cause data loss in pre-existing instances.

The following table describes properties that operators can update in the configuration page.

Property	Default	Description
Redis Client Timeout	3600	Server timeout for an idle client specified in seconds (e.g., 3600)
Redis TCP Keepalive	60	The max number of connected clients at the same time
Max Clients	1000/5000/10000 (small/medium/large)	The max number of connected clients at the same time
Lua Scripting	Enabled	Enable/Disable Lua scripting
Plan Quota	20	Maximum number of Redis service instances for this plan, across all orgs and spaces. For more information, see Setting Limits for On-Demand Service Instances.

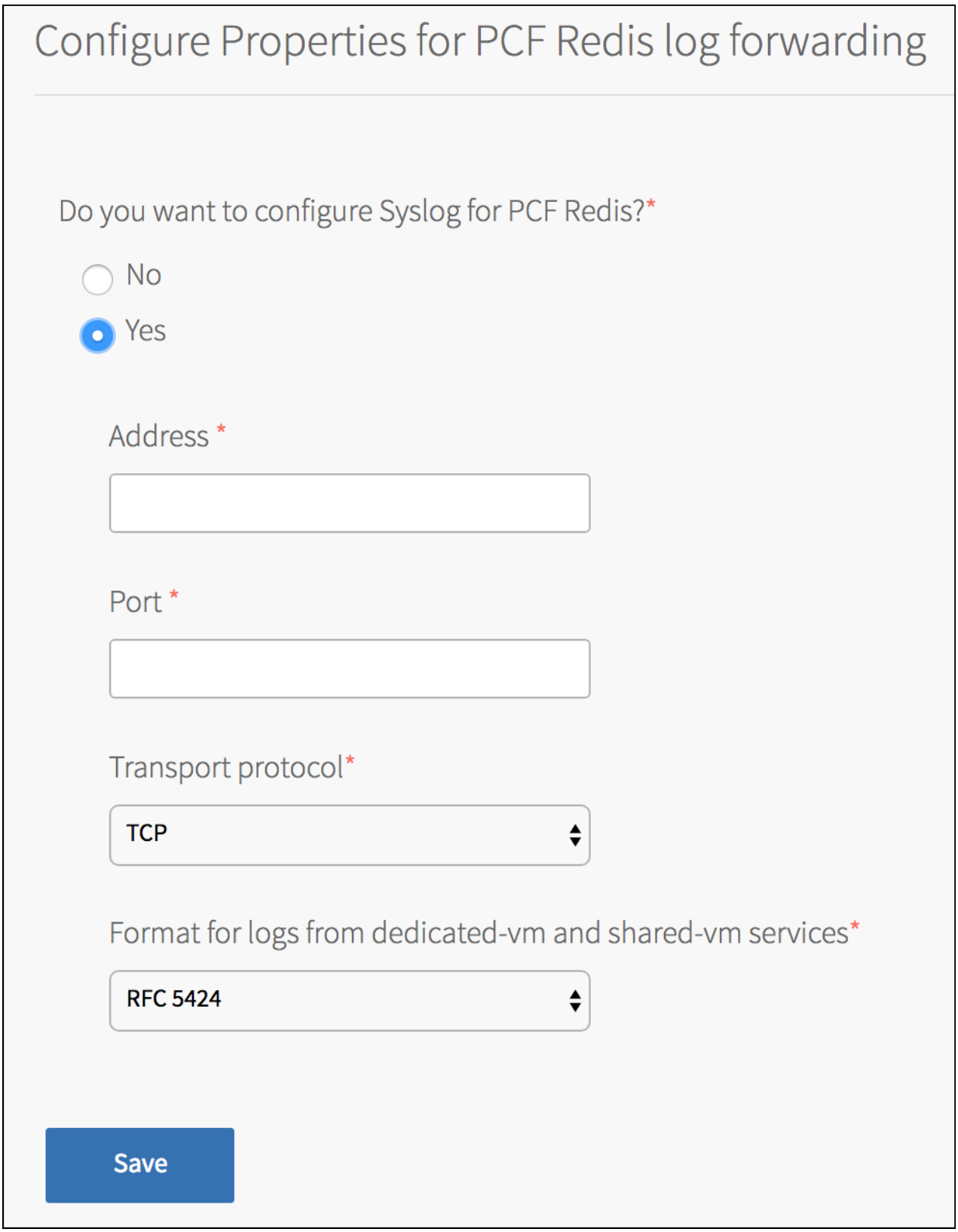
For settings that app developers can configure, see Customize an On-Demand Service Instance.

Configure Syslog Output

Pivotal recommends that operators configure a syslog output. For On-Demand instances, all logs follow RFC5424 format. Dedicated-VM and Shared-VM plan instances are consistent with their previous format.

Follow these steps to configure syslog output:

1. Click the Redis for PCF tile to display the configuration page, and then click **Syslog** on the Settings tab.



2. Select the **Yes** radio button.

3. Enter the Syslog address and port, and select the transport protocol of your log management tool. The information required for these fields is provided by your log management tool.

4. Select the format for your logs. RFC 5424 is the suggested format and is standard across PCF services.

   If you have pre-existing systems that may break with a new format, you may select the legacy format. However, the legacy format will eventually be deprecated.

5. Click **Save**.

Networks, Security, and Assigning AZs

Network Configuration

Pivotal recommends that each type of Redis for PCF service run in its own network. For example, run a Redis for PCF on-demand service on a separate network from a Redis for PCF shared-VM service.

The following ports and ranges are used in this service:

Port	Protocol	Direction and Network	Reason
8300 8301	TCP TCP and UDP	Inbound to CloudFoundry network, outbound from service broker and service instance networks*	Communication between the CF consul_server and consul_agents on Redis deployment; used for metrics
4001	TCP	Inbound to CloudFoundry network, outbound from service broker and service instance networks*	Used by the Redis metron_agent to forward metrics to the CloudFoundry etcd server
12350	TCP	Outbound from CloudFoundry to the cf-redis-broker service broker network	(Only if using a cf-redis-broker) Access to the cf-redis-broker from the cloud controllers.
12345	TCP	Outbound from CloudFoundry to the on-demand service broker network	(Only if using an On-Demand service) For access to the on-demand service broker from the cloud controllers
6379	TCP	Outbound from CloudFoundry to any service instance networks (dedicated-node and on-demand)	Access to all dedicated nodes and on-demand nodes from the Diego Cell and Diego Brain network(s)
32768-61000	TCP	Outbound from CloudFoundry to the cf-redis-broker service broker network	From the Diego Cell and Diego Brain network(s) to the service broker VM. This is only required for the shared service plan.
80 or 443 (Typically)	http or https respectively	Outbound from any service instance networks	Access to the backup blobstore
8443 25555	TCP	Outbound from any on-demand service broker network to the bosh director network	For the on-demand service, the on-demand service broker needs to talk to `bosh director`

* Typically the service broker network and service instance network(s) are the same.

Application Security Groups

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

Note: Without ASGs, this service is unusable.

Application Container Network Connections

Application containers that use instances of the Redis service require the following outbound network connections:

Destination	Ports	Protocol	Reason
`ASSIGNED_NETWORK`	32768-61000	tcp	Enable application to access shared vm service instance
`ASSIGNED_NETWORK`	6379	tcp	Enable application to access dedicated vm service instance

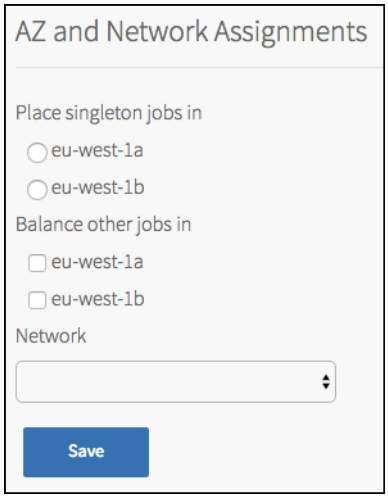
Create an ASG called `redis-app-containers` with the above configuration and bind it to the appropriate space or, to give all started apps access, bind to the `default-running` ASG set and restart your apps. Example:

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

Assigning AZs

As of Redis for PCF 1.9, you can assign multiple AZs to Redis jobs, however this will not guarantee high availability.

For more information, see About Multiple AZs in Redis for PCF.



Validating Installation

Smoke tests are run as part of Redis for PCF installation to validate that the install succeeded. Smoke tests are described here.

Uninstalling Redis for PCF

To uninstall Redis for PCF, do the following:

1. In the PCF Ops Manager Installation dashboard, click the trash can icon in the lower right hand corner of the Redis for PCF tile.
2. Confirm deletion of the product, and then click **Apply Changes**.

Upgrading Redis for PCF

Page last updated:

This section contains the upgrade procedure and upgrade paths for Redis for PCF.

Compatible Upgrade Paths

Consider the following compatibility information before upgrading Redis for PCF.

For more information, see the Product Version Matrix.

Ops Manager Version	Supported Upgrades for Redis Installations	
	From	To
v1.5.x, v1.6.x	v1.4.0 – v1.4.3	v1.4.4 – latest v1.4.x
		v1.5.0 – v1.5.7
	v1.4.4 – latest v1.4.x	Next v1.4.x – latest v1.4.x
		v1.5.0 – latest v1.5.x
	v1.5.0 – latest v1.5.x	Next v1.5.x – latest v1.5.x
v1.7.x	v1.5.0 – latest version	v1.5.1 – latest version
v1.8.x	v1.5.17 – latest version	v1.6.0 – latest version
v1.9.x – latest version	v1.6.0 – latest version	v1.8.0 – latest version
v1.10.0 – v1.10.2	v1.7.2 – latest version	v1.8.0 – latest version
v1.10.3 – latest version	v1.7.2 – latest version	v1.9.0 – latest version
v1.11.x	v1.8.0 – latest version	1.9.0 – latest version

Upgrade Redis for PCF

This product enables a reliable upgrade experience between versions of the product that is deployed through Ops Manager.

For information on the upgrade paths for each released version, see the above table.

To upgrade Redis for PCF, do the following:

1. Download the latest version of the product from Pivotal Network.
2. Upload the new `.pivotal` file to Ops Manager.
3. If required, upload the stemcell associated with the update.
4. If required, update any new mandatory configuration parameters.
5. Click **Apply changes**. The rest of the process is automated.

During the upgrade deployment each Redis instance experiences a small period of downtime as each Redis instance is updated with the new software components. This downtime is because the Redis instances are single VMs operating in a non HA setup.

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

Ops 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.

Release Policy

When a new version of Redis is released, Pivotal aims to release a new version of Redis for PCF containing the new Redis version soon after.

When a new version of Redis or another dependent software component, such as the stemcell, is released due to a critical CVE, Pivotal's goal is to release a new version of the Redis for PCF within 48 hours.

Setting Limits for On-Demand Service Instances

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

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

- Global
- Plan
- Org
- Space

After you set quotas, you can:

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

Create Global-level Quotas

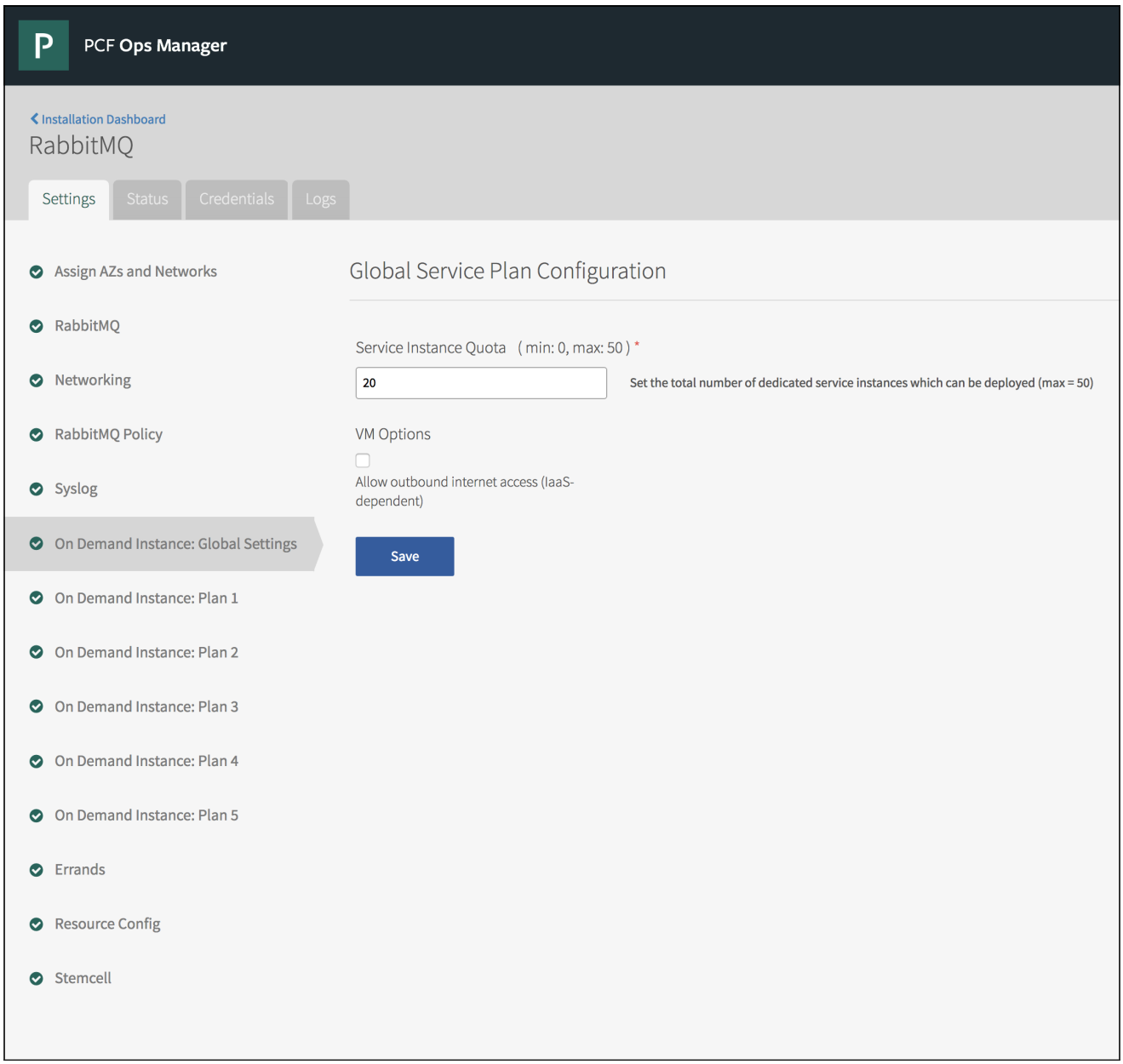
Each Pivotal Cloud Foundry (PCF) service has a separate service broker. A global quota at the service level sets the maximum number of service instances that can be created by a given service broker. If a service has more than one plan, then the number of service instances for all plans combined cannot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

The operator sets a global quota for each PCF service independently. For example, if you have Redis for PCF and RabbitMQ for PCF, 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.

The global quota is set in the service tile in Ops Manager, shown for an example service below.

Note: This is an example image only. The following screen may look slightly different for your service or release version.



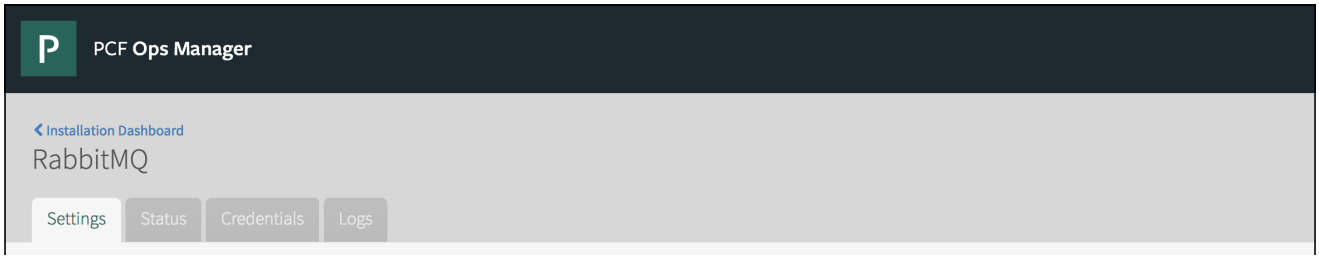
Create Plan-level Quotas

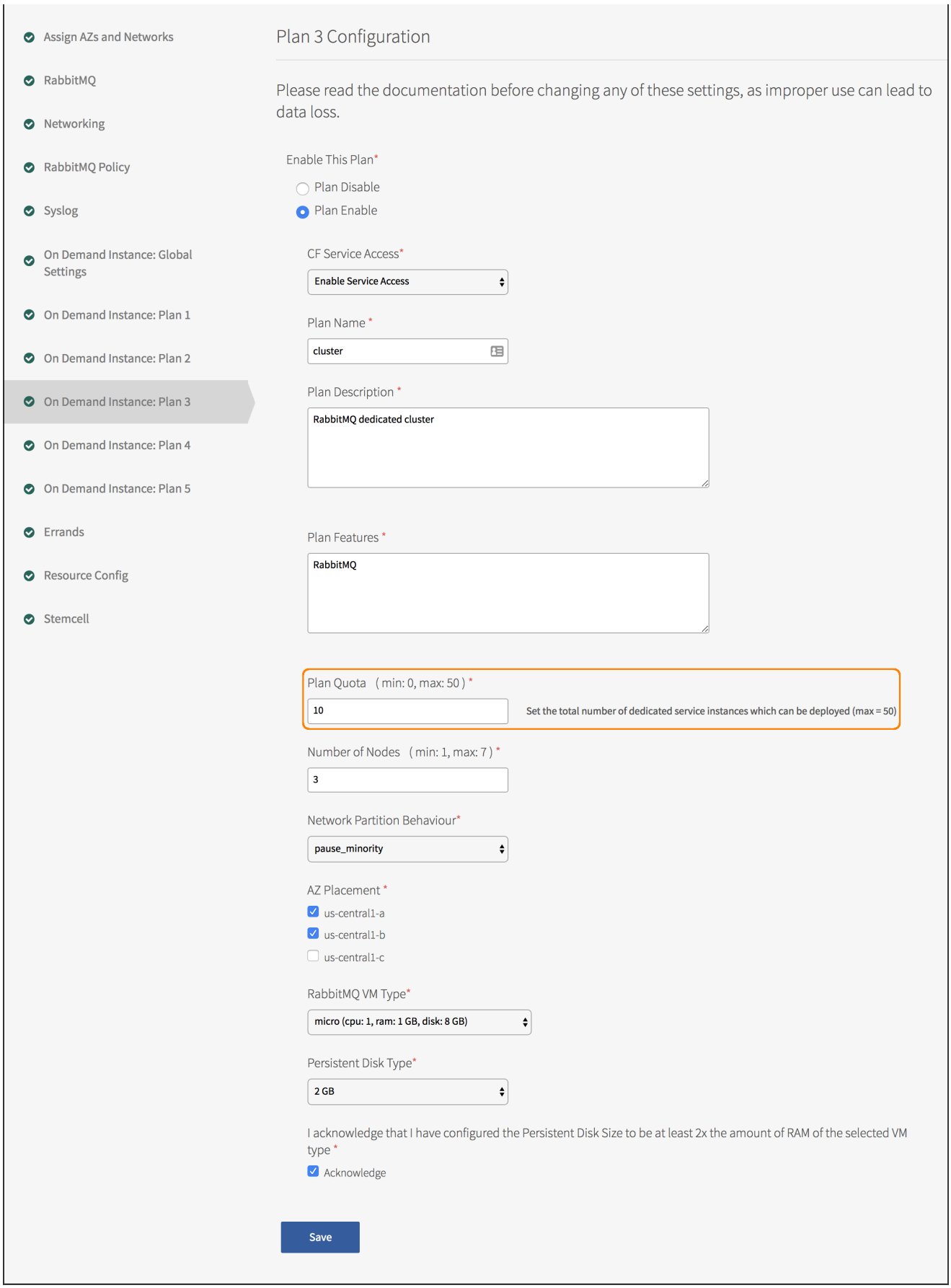
A service may offer one or more plans. You can set a separate quota per plan so that instances of that plan cannot exceed the plan quota. For a service with multiple plans, the total number of instances created for all plans combined cannot exceed the global quota for the service.

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

The plan quota is set in the service tile in Ops Manager, shown for an example service plan below.

Note: This is an example image only. The following screen may look slightly different for your service or release version.





Create and Set Org-level Quotas

An org-level quota applies to all PCF services and sets the maximum number of service instances an organization can create within PCF. 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 PCF services.

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

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

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

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

   where these variables are:

   `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-quota myquota -m 1024mb -i 16gb -r 30 -s 50 --allow-paid-service-plans`

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

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

   For example: `cf set-quota dev_org myquota`

For more information on managing org-level quotas, see Creating and Modifying Quota Plans.

Create and Set Space-level Quotas

A space-level service quota applies to all PCF services and sets the maximum number of service instances that can be created within a given space in PCF. 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 PCF services.

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

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

1. Run the following command to create the quota:

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

   where these variables are:

   `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 above with a specific space by running the following command:

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

   For example:
   `cf set-space-quota myspace myspacequota`

For more information on managing space-level quotas, see Creating and Modifying Quota Plans.

View Current Org and Space-level Quotas

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

```
cf org ORG-NAME
```

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

```
cf space SPACE-NAME
```

For more information on 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 metrics are listed below:

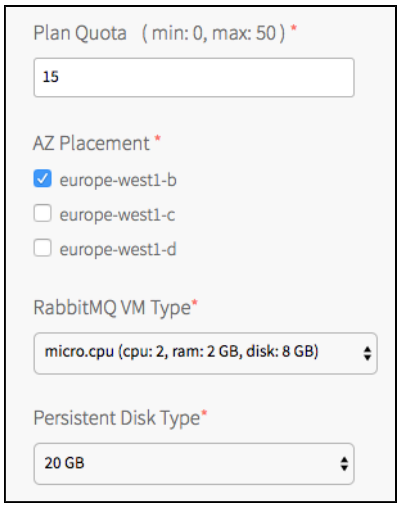
Metric Name	Description
on-demand-broker/SERVICE-NAME/quota_remaining	Quota remaining for all instances across all plans
on-demand-broker/SERVICE-NAME/PLAN-NAME/quota_remaining	Quota remaining for a specific plan
on-demand-broker/SERVICE-NAME/total_instances	Total instances created across all plans
on-demand-broker/SERVICE-NAME/PLAN-NAME/total_instances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

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

Calculate Resource Costs for On-Demand Plans

RabbitMQ on-demand plans use dedicated VM, 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 VM and Persistent Disk types selected in the plan configuration(s). The specific costs depend on your IaaS.

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



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

Calculate Maximum Resource Cost Per On-Demand Plan

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

plan quota x cost of selected resources

For example, if you selected the options in the above image, you have selected a VM type **micro.cpu** 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.cpu VM type) + (15 x cost of 20 GB persistent disk)

Calculate Maximum Resource Cost for All On-Demand Plans

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

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

For example:

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

Configuring Automated Backups for Redis for PCF

Page last updated:

Creating Backups of Redis Instances

You can configure backups to be run for all instances, across Dedicated-VM and Shared-VM service plans. **Backups are not available for On-Demand instances.**

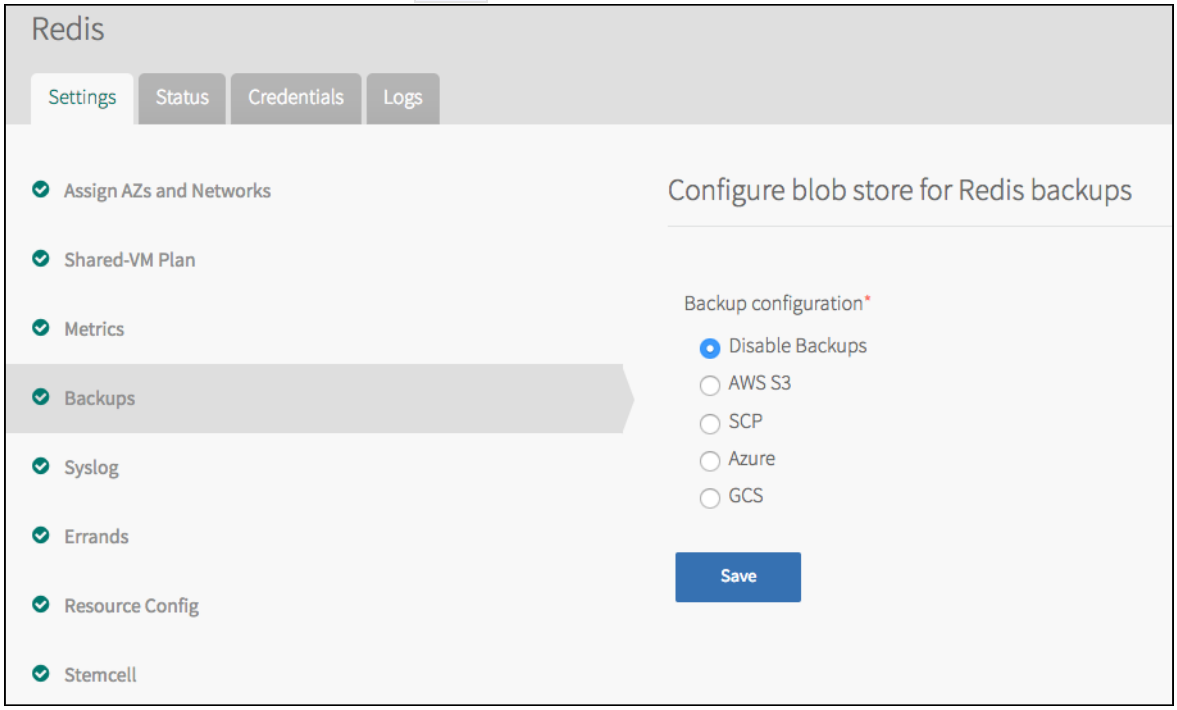
The key features are:

- Runs on a configurable schedule
- Every instance is backed up, across both service plans
- The Redis broker statefile is backed up
- For each backup artefact, a file is created that contains the MD5 checksum for that artifact. This can be used to validate that the artefact is not corrupted.
- You can configure AWS S3, SCP, Azure or Google Cloud Storage as your destination
- Data from Redis is flushed to disk, before the backup is started by running a `BGSAVE` on each instance
- Backups are labelled with timestamp, instance GUID and plan name

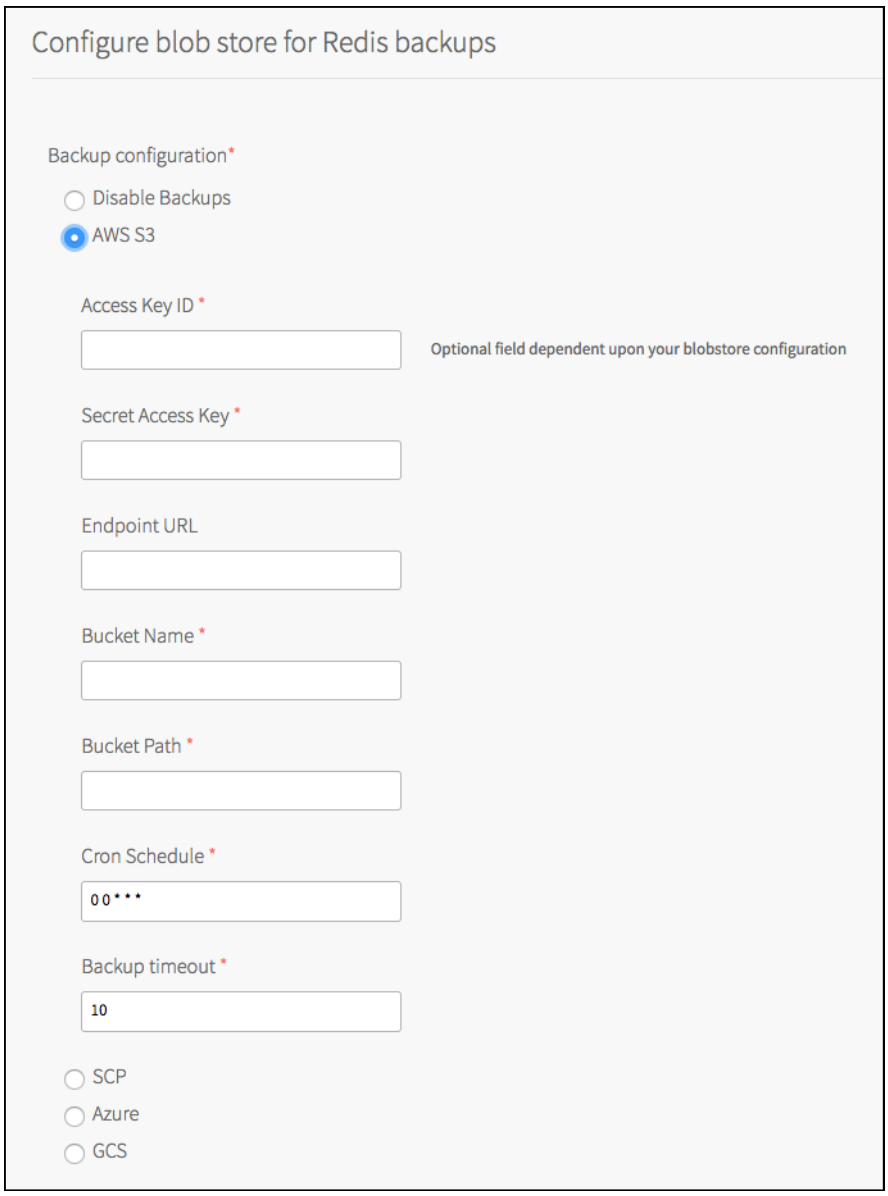
Configuring Backups

To enable backups, you will first need to choose your backup destination type - AWS S3, SCP, Azure or Google Cloud Storage.

Click on the tile in Ops Manager, followed by the `Backups` link on the left-hand menu.



S3 Backup Fields



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
Endpoint URL	The endpoint of your S3 account, e.g. `http://s3.amazonaws.com`	Optional, defaults to `http://s3.amazonaws.com` if not specified
Bucket Name	Name of the bucket you wish the files to be stored in.	Mandatory
Path	Path inside the bucket to save backups to.	Mandatory
Backup timeout	The amount of time, in seconds, that the backup process waits for the BGSAVE command to complete on your instance, before transferring the RDB file to your configured destination	Mandatory
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is `* 2 * * *`. Also accepts a pre-defined schedule: any of `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, or `@every <time>`, where `<time>` is any supported time string (e.g. `1h30m`). For more information, see the cron package documentation.	Mandatory

AWS IAM Policy

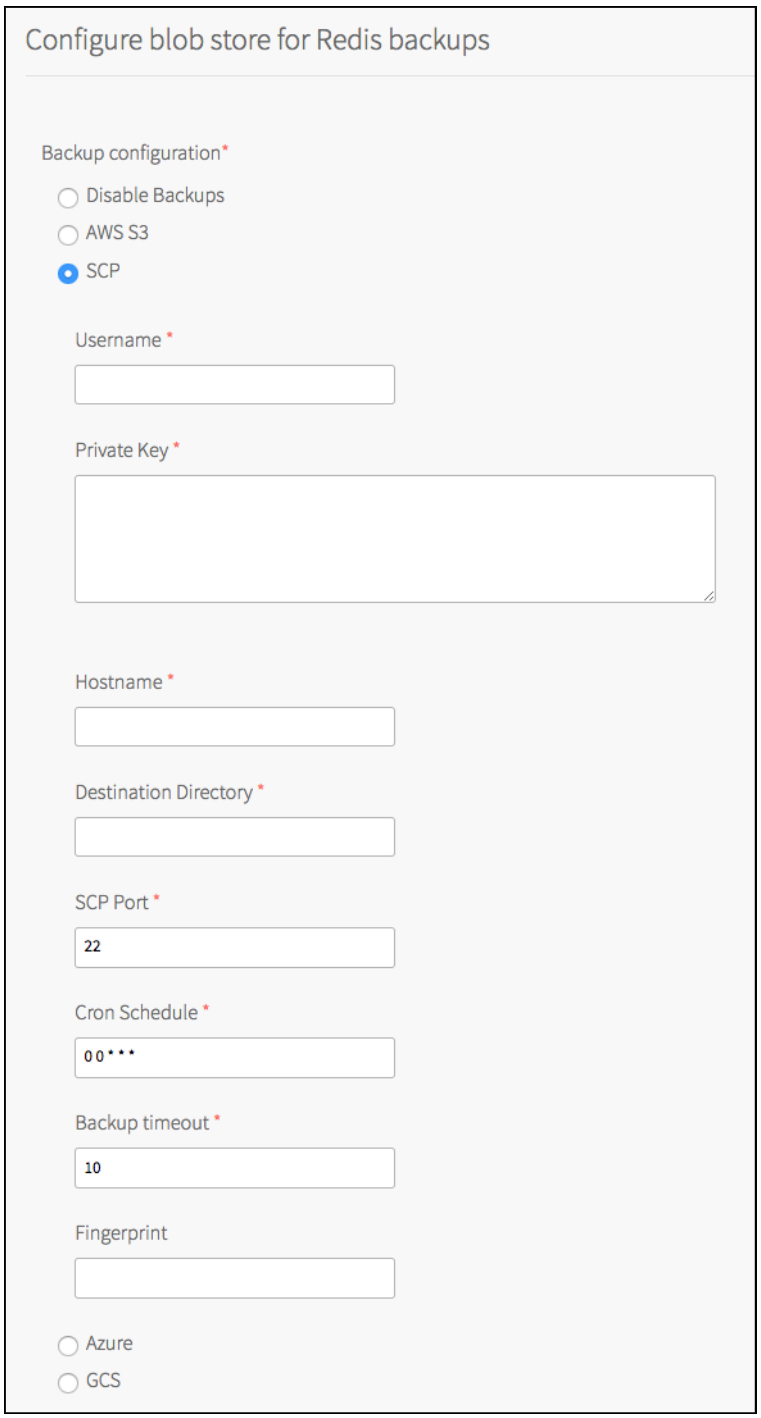
An AWS IAM policy describes the permissions related to your bucket. The minimum set of policies required in order to upload the backup files are:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
      ]
    }
  ]
}
```

Notes:

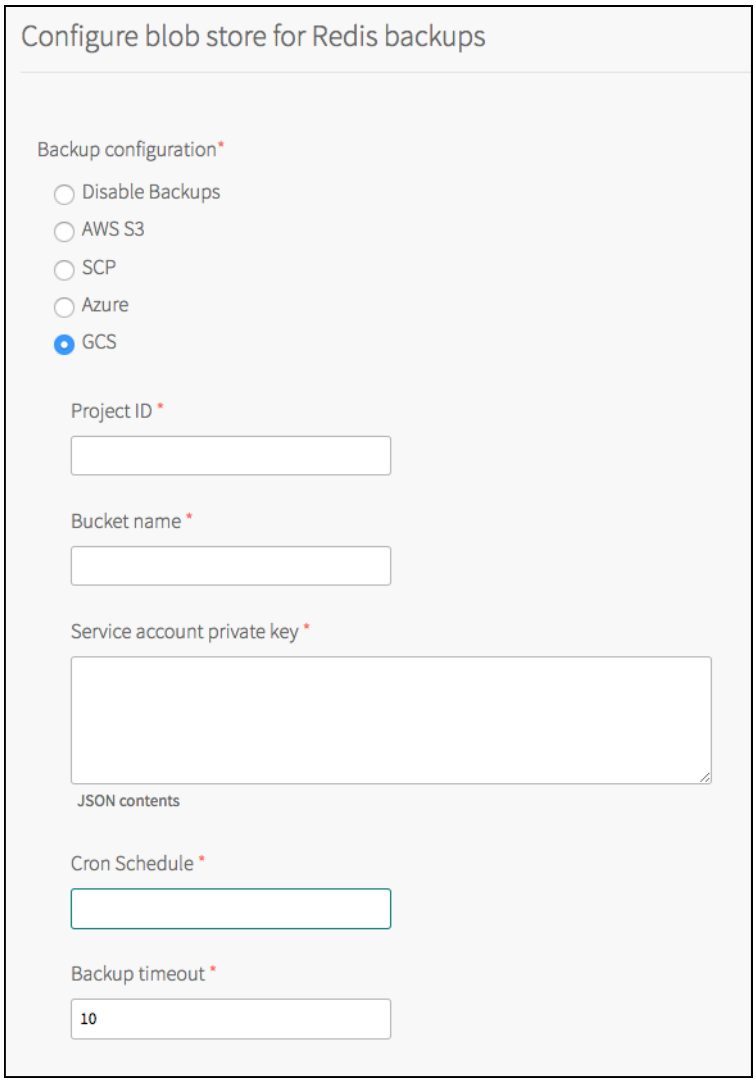
- Make sure to replace `<bucket-name>` with your correct values.
- `s3:CreateBucket` is only required if the S3 bucket does not exist.
- The additional `s3:CreateBucket` action is also required if the S3 bucket does not exist.

SCP Backup Fields



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
Hostname	The hostname 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. Refer to table for S3 backups for details	Mandatory
Backup timeout	The amount of time, in seconds, that the backup process waits for the BGSAVE command to complete on your instance, before transferring the RDB file to the scp server	Mandatory

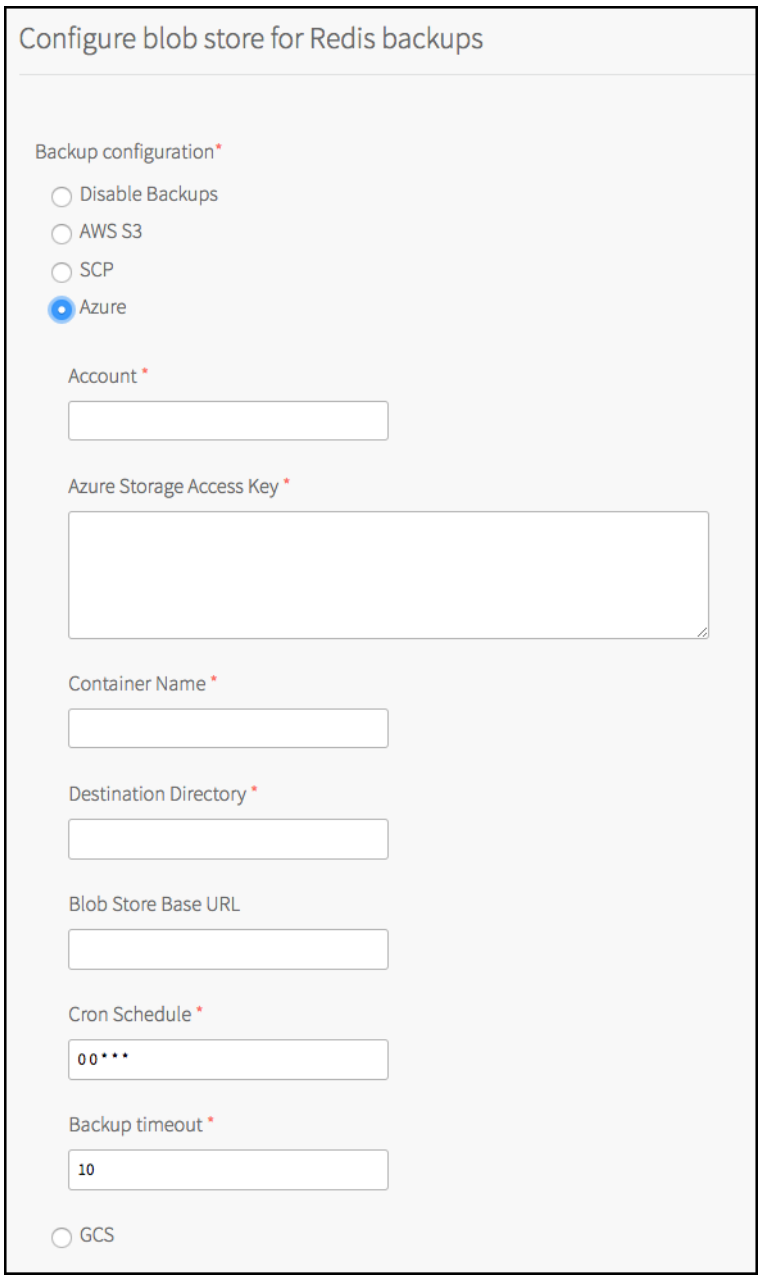
GCS Backup Fields



Redis for PCF uses service account credentials to upload backups to Google Cloud Storage. The service account should have `Storage Admin` permissions. Please refer to the documentation for details on how to set up a GCP service account.

Field	Description	Mandatory/Optional
Project ID	GCP Project ID	Mandatory
Bucket name	Name of the bucket you wish the files to be stored in.	Mandatory
Service account private key	The JSON Secret Key associated with your Service Account. See documentation for details on how to set up service account keys.	Mandatory
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is * 2 * * *. Also accepts a pre-defined schedule: any of @yearly, @monthly, @weekly, @daily, @hourly, or @every , where is any supported time string (e.g. 1h30m). For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in seconds, that the backup process waits for the BGSAVE command to complete on your instance, before transferring the RDB file to your configured destination	Mandatory

Azure Backup Fields



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 which will store backup files.	Mandatory
Destination Directory	Directory where the backup files will be stored within the Azure container.	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 * * *. Also accepts a pre-defined schedule: any of @yearly, @monthly, @weekly, @daily, @hourly, or @every , where is any supported time string (e.g. 1h30m). For more information, see the cron package documentation.	Mandatory
Backup timeout	The amount of time, in seconds, that the backup process waits for the BGSAVE command to complete on your instance, before transferring the RDB file to your configured destination	Mandatory

Notes

For each backup destination, the field `Backup timeout` causes backups to fail after a configured timeout. Redis' BGSAVE will continue but backups will not be uploaded to destinatons if this timeout is hit.

Manually Backing up and Restoring Redis for PCF

Page last updated:

Note: This topic applies only to Dedicated-VM and Shared-VM service plans.

About the BOSH CLI

The BOSH CLI is available in two major versions, v1 and v2. Pivotal recommends that you use the BOSH CLI v2 when possible.

This topic provides examples of using each version of the BOSH CLI. While all versions of the BOSH CLI work with Redis 1.9, your Ops Manager version may affect which version of the BOSH CLI you can use. Consult the table below to determine which version of the CLI is supported for your installation.

Ops Manager Version	BOSH CLI Version
1.10	CLI v1
1.11	CLI v1 or CLI v2 (Pivotal recommends CLI v2)

Triggering a Manual Backup

Backups of your Redis deployment will automatically occur per the Cron Schedule you set in your deployment. You can trigger manual backups at any time by following the steps below. The backup artifacts will be sent to the destination configured in Ops Manager for automatic backups.

1. Log in to Ops Manager and target the Redis tile deployment.

2. Identify the VM which holds your instance by running one of the following commands:
    - **For Ops Manager 1.10 or earlier:** `bosh vms`
    - **For Ops Manager 1.11 or later:** `bosh2 -d DEPLOYMENT-NAME ssh`

    a. For the `shared-vm` plan this will be the job name containing `cf-redis-broker`. Running `manual-backup` will back up all of the shared-VM instances and the broker state in the `statefile.json` file.
    b. For the `dedicated-vm` plan this will be the job name containing `dedicated-node`. Running `manual-backup` will back up the Redis dump.rdb file for that dedicated-VM instance.
    c. You can identify the exact node for your `dedicated-vm` service instance by comparing the IP Address from your application bindings.

An example output from BOSH `vms`:

```
Deployment `p-redis-9dacffffa493b5e5a386'

Director task 129

Task 129 done

+-----------------------------------------+---------+-----------------------------------------+-----------+
| Job/index                               | State   | Resource Pool                           | IPs       |
+-----------------------------------------+---------+-----------------------------------------+-----------+
| cf-redis-broker-partition-default_az_guid/0 | running | cf-redis-broker-partition-default_az_guid | 10.0.0.58 |
| dedicated-node-partition-default_az_guid/0  | running | dedicated-node-partition-default_az_guid  | 10.0.0.59 |
+-----------------------------------------+---------+-----------------------------------------+-----------+
```

3. **For BOSH CLI v1 only**, target the manifest of your deployed Redis with `bosh deployment PATH-TO-MANIFEST.yml`. If you do not have this file, you can download it by running `bosh download manifest DEPLOYMENT-NAME`.

4. Connect to the node you want to back up using one of the following commands:
    - **For Ops Manager 1.10 or earlier:** `bosh ssh`
    - **For Ops Manager 1.11 or later:** `bosh2 -d DEPLOYMENT-NAME ssh`

5. Switch to root using `sudo -i`.

6. Run `/var/vcap/jobs/service-backup/bin/manual-backup` to trigger a manual backup.

Notes

Triggering a manual backup of a large dump.rdb could take sufficiently long that your SSH connection will timeout. Ensure that you have given yourself enough of a timeout to complete the backup.

Back ups are currently not available for On-Demand instances.

As of v1.8.2, backups can be to all AWS S3 regions (previously limited to the standard region, us-east-1). This requires specifying the region of your backup and the signature version used for your region. You can find this information here. In most cases the default signature version will be sufficient.

Making Your Own Backups

It is possible to create a back up of a Redis instance by hand, bypassing the automated backup tool altogether.

Persistence is enabled on these plans through the use of `RDB` files, using the following Redis config rules:

```
save 900 1
save 300 10
save 60 10000
```

Shared-VM Plan

You can either take the latest RDB file held on disk, which is generated by the above the rules, or trigger a recent update by using the `redis-cli` to trigger a `BGSAVE`. Credentials to log in to the `redis-cli` can be obtained from `VCAP_SERVICES` for your bound application.

The `redis-cli` is located in `/var/vcap/packages/redis/bin/redis-cli`.

On this plan, the `BGSAVE` command is aliased to a random string. This can be obtained from Ops Manager in the credentials tab.

Steps to Backup

1. BOSH `ssh` into your desired node. See the above section on identifying the correct VM.
2. Change to root using `sudo -i`.
3. Copy the contents of the `/var/vcap/store/cf-redis-broker` directory to a .zip or .tar file.
4. Backup the folder / compressed file to your chosen location.

The `/var/vcap/store/cf-redis-broker` has sub-directories for each instance created of this plan. The backup file for each instance is called `dump.rdb`.

For example, here are two instances:

```
root@66358f3e-3428-46df-9bb3-9acc7770b188:/var/vcap/store/cf-redis-broker# find -type f | xargs ls -1
./redis-data/3124f373-e9e2-44e1-ad12-a8865d8978b0/db/dump.rdb
./redis-data/3124f373-e9e2-44e1-ad12-a8865d8978b0/redis.conf
./redis-data/3124f373-e9e2-44e1-ad12-a8865d8978b0/redis-server.pid
./redis-data/62333bf9-f023-4566-b233-6686f26b8f4d/db/dump.rdb
./redis-data/62333bf9-f023-4566-b233-6686f26b8f4d/redis.conf
./redis-data/62333bf9-f023-4566-b233-6686f26b8f4d/redis-server.pid
./statefile.json
```

Note: This procedure is the same for v1 and v2 of the BOSH CLI, except that v2 of the CLI requires commands to start with `bosh2`, rather than `bosh`.

Dedicated-VM Plan

You can either take the latest RDB file on disk, as generated by the above rules, or trigger a more recent RDB file by executing the `BGSAVE` command using the `redis-cli`. Credentials can be obtained from the `VCAP_SERVICES` from your bound application. The `redis-cli` can be found in `/var/vcap/packages/redis/bin/redis-cli`.

1. BOSH `ssh` into your desired node. See the above section on identifying the correct VM.
2. Change to root using `sudo -i`.
3. Copy the contents of the `/var/vcap/store/redis` directory to a .zip or .tar file.
4. Backup the folder or compressed file to your chosen location.

The backup file will be named `dump.rdb`.

Note: This procedure is the same for v1 and v2 of the BOSH CLI, except that v2 of the CLI requires commands to start with `bosh2`, rather than `bosh`.

Restore Redis Instance from a Backup

To a Local System

You can choose to restore the RDB file to a local Redis instance.

The steps to do this depend on your configuration and setup. Refer to the Redis documentation for more details.

To Pivotal Cloud Foundry

You can also restore your backup file to another instance of the `Redis for PCF` tile.

Prerequisites

- Same resource configuration as the instance from which you backed up.
- Ensure that the persistent disk is large enough to accommodate the temporary files used during the restore process. It should be **3.5x the amount of RAM in the VM**.

To restore your backup file to another instance of a `Redis for PCF` tile service instance:

1. Create a new instance of the plan that you wish to restore to.
2. Identify the VM which the instance of your plan is located on by following the steps from the `Manual Backups` section above. If you are restoring an instance of `shared-vm`, this VM is the broker VM.
3. BOSH `ssh` into the identified VM. This is the broker VM if restoring a `shared-vm` instance.

Redis for PCF v1.7 and later provides a script to automatically restore data in a newly provisioned Redis instance.

Preparation

1. Transfer your backup RDB file to a local path on the VM (`PATH-TO-RDB-BACKUP-ON-VM` has to be under `/var/vcap/store`.
2. To verify that the RDB file hasn't been corrupted, run `md5sum PATH-TO-RDB-BACKUP-ON-VM` and compare it against the contents of the `.md5` file named after the backup file. The values should be the same. The `.md5` file is located in the same bucket as the original backup file.
3. Switch to root user `sudo su`

Dedicated-VM Plan

The restore script will restore the data for the specified dedicated-VM instance.

Execution

1. Run `/var/vcap/jobs/redis-backups/bin/restore --sourceRDB PATH-TO-RDB-BACKUP-ON-VM`
2. Tail the script logs at `/var/vcap/sys/log/redis-backups/redis-backups.log` to see progress. When the data restore has been successfully completed, you will see the message `Redis data restore completed successfully`

Debugging

The data restore script runs the steps listed below. It logs its process along the way and provides helpful messages in case of failure.

The script logs at `/var/vcap/sys/log/redis-backups/redis-backups.log`.
If a step has failed, resolve the reason that caused it to fail and execute the failed step and every next step manually. You can retrieve `{instance_password}` through the binding to your service instance: `cf service-key {instance_name} {key_name}`

1. **`StopAll`**
   Run `monit stop all`
2. **`WaitForStop`**
   Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`
3. **`DeleteExistingPersistenceFiles`**
   Clean up existing Redis data files:
   - `rm -f /var/vcap/store/redis/appendonly.aof`
   - `rm -f /var/vcap/store/redis/dump.rdb`
4. **`CopyBackupFileWithCorrectPermissions`**
   Restore your Redis backup file to `/var/vcap/store/redis/dump.rdb` and correct the owner and permissions with
   `chown vcap:vcap /var/vcap/store/redis/dump.rdb && chmod 660 /var/vcap/store/redis/dump.rdb`
5. **`SetAppendOnly`**
   `Edit the template Redis config file with` vim $(find /var/vcap/data/jobs/ -name redis.conf)` and make the following line changes:
   - `appendonly yes` -> `appendonly no`
6. **`StartAll`**
   Run `monit start all`
7. **`WaitForStart`**
   Wait for monit services to enter the `running` state, you can watch this with `watch monit summary`
8. **`RewriteAOF`**
   Run `/var/vcap/packages/redis/bin/redis-cli -a {instance_password} BGREWRITEAOF`
9. **`RewriteAOF`**
   Run `watch "/var/vcap/packages/redis/bin/redis-cli -a {instance_password} INFO | grep aof_rewrite_in_progress"` until `aof_rewrite_in_progress` is `0`
10. **`StopAll`**
    Run `monit stop all`
11. **`WaitForStop`**
    Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`
12. **`ChownToUserAndGroup`**
    Set correct owner on `appendonly.aof` by running `chown vcap:vcap /var/vcap/store/redis/appendonly.aof`
13. **`SetAppendOnly`**
    Edit the template Redis config file with `vim $(find /var/vcap/data/jobs/ -name redis.conf)` and make the following line changes:
    - `appendonly no` -> `appendonly yes`
14. **`StartAll`**
    Run `monit start all`

Shared-VM Plan

The restore script will restore the data for the specified shared-VM instance.

Execution

1. Retrieve `{instance_guid}` by running: `cf service {instance_name} --guid`
2. Run `/var/vcap/jobs/redis-backups/bin/restore --sourceRDB {path-to-rdb-backup-on-vm} --sharedVmGuid {instance_guid}`
3. Tail the script logs at `/var/vcap/sys/log/redis-backups/redis-backups.log` to see progress. When the data restore has been successfully completed, you will see the message `Redis data restore completed successfully`

Debugging

The data restore script runs the steps listed below. It logs its process along the way and provides helpful messages in case of failure.

The script logs at `/var/vcap/sys/log/redis-backups/redis-backups.log`.
If a step has failed, resolve the reason that caused it to fail and execute the failed step and every next step manually. You can retrieve `{instance_password}` and `{port}` through the binding to your service instance: `cf service-key {instance_name} {key_name}`

1. **`StopAll`**
   Run `monit stop all`
2. **`WaitForStop`**
   Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`
3. **`SetConfigCommand`**
   Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and comment out the line:
   - `rename-command CONFIG "configalias"` -> `#rename-command CONFIG "configalias"`
4. **`SetRewriteCommand`**
   Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and comment out the line:
   - `rename-command BGREWRITEAOF ""` -> `#rename-command BGREWRITEAOF ""`
5. **`DeleteExistingPersistenceFiles`**
   Clean up existing Redis data files if they exist:
   - `rm -f /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/appendonly.aof`
   - `rm -f /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb`
6. **`CopyBackupFileWithCorrectPermissions`**
   Restore your Redis backup file to `/var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb` and correct the owner and permissions with
   ```
   chown vcap:vcap /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb && chmod 660 /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb
   ```
7. **`SetAppendOnly`**
   Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and make the following line changes:
   - `appendonly yes` -> `appendonly no`
8. **`StartAll`**
   Run `monit start all`
9. **`WaitForStart`**
   Wait for monit services to enter the `running` state, you can watch this with `watch monit summary`
10. **`RewriteAOF`**
    Run `/var/vcap/packages/redis/bin/redis-cli -a {instance_password} BGREWRITEAOF`

11. **`RewriteAOF`**
    Run `watch "/var/vcap/packages/redis/bin/redis-cli -a {instance_password} INFO | grep aof_rewrite_in_progress"` until `aof_rewrite_in_progress` is `0`

12. **`StopAll`**
    Run `monit stop all`

13. **`WaitForStop`**
    Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`

14. **`ChownToUserAndGroup`**
    Set correct owner on `appendonly.aof` by running `chown vcap:vcap /var/vcap/store/redis/appendonly.aof`

15. **`SetAppendOnly`**
    Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and make the following line changes:
    - `appendonly no` -> `appendonly yes`

16. **`SetConfigCommand`**
    Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and uncomment the line: `#rename-command CONFIG "configalias"` -> `rename-command CONFIG "configalias"`

17. **`SetRewriteCommand`**
    Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and uncomment the line: `#rename-command BGREWRITEAOF ""` -> `rename-command BGREWRITEAOF ""`

18. **`StartAll`**
    Run `monit start all`

Recovering Redis Instances

In the event of a recovery of Cloud Foundry, it is possible to recover bound Redis instances to healthy states that are in sync with Cloud Foundry. There are a few caveats to being able to recover previous instance state fully that depend on your plan.

Shared-VM Plan Caveats

- You need a backed up RDB Redis dump file - this would be stored in your S3 buckets if you have backups configured.
- You need a backed up `/var/vcap/store/cf-redis-broker/redis-data` directory from the service broker node. You do not need to backup and `*.aof` or `*.rdb` files from subdirectories if you have backups configured.

Dedicated-VM Plan Caveats

- You need a backed up RDB Redis dump file - this would be stored in your S3 buckets if you have backups configured.
- You need a backed up `/var/vcap/store/redis/statefile.json` from the service broker node.

Note

This procedure assumes that a recovery of service information and service keys assigned to instances are restored with a restore of Cloud Foundry.

Recovery Procedure

After redeploying Redis, take the following steps.

Shared-VM Plan

1. BOSH `ssh` into the service broker node of your Redis deployment.
2. Run `monit stop all && pkill redis-server`
3. Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`
4. Confirm no running instances of `redis-server` with `ps aux | grep redis-server`
5. Copy the backed up `redis-data` directory into `/var/vcap/store/cf-redis-broker`
6. Follow the instructions here for your plan, skipping the first four steps described here, for restoring your backed up Redis data.
7. Your Redis instance is now recovered.

Note: This procedure is the same for v1 and v2 of the BOSH CLI, except that v2 of the CLI requires commands to start with `bosh2`, rather than `bosh`.

Dedicated-VM Plan

1. BOSH `ssh` into the service broker node of your Redis deployment.
2. Run `monit stop all`
3. Wait for monit services to enter the `not monitored` state, you can watch this with `watch monit summary`
4. Copy the backed up `/var/vcap/store/cf-redis-broker/statefile.json` and ensure ownership and permissions are correct with `chown vcap:vcap /var/vcap/store/redis/dump.rdb && chmod 660 /var/vcap/store/redis/dump.rdb`
5. Follow the instructions here for your plan, skipping the first three steps described here, for restoring your backed up Redis data.
6. Your Redis instance is now recovered.

Note: This procedure is the same for v1 and v2 of the BOSH CLI, except that v2 of the CLI requires commands to start with `bosh2`, rather than `bosh`.

Monitoring Redis for PCF

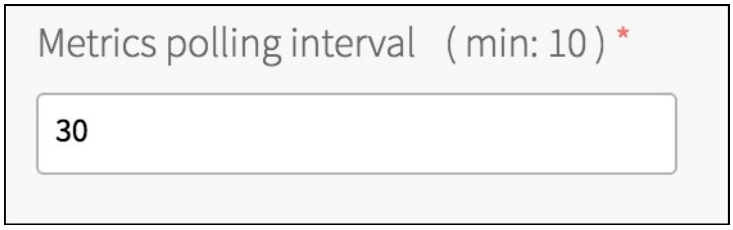
Page last updated:

The Loggregator Firehose exposes Redis metrics. You can use third-party monitoring tools to consume Redis metrics to monitor Redis performance and health. For an example Datadog configuration that displays some of the significant metrics outlined below, see the CF Redis example dashboard. Pivotal does not endorse or provide support for any third party solution.

For the On-Demand service plan, instance-level metrics are not available. The available On-Demand service metrics are listed below.

Metrics Polling Interval

The metrics polling interval defaults to 30 seconds. This can be changed by navigating to the Metrics configuration page and entering a new value in **Metrics polling interval (min: 10)**.



Monitoring Current Instances and Quotas Remaining for On-Demand Plans

The following examples shows the number of total provisioned instances for On-Demand Plans across all On-Demand plans and for a specific On-Demand plan:

```
origin:"p.redis" eventType:ValueMetric timestamp:1491922454382895846 deployment:"redis-on-demand-broker" job:"redis-on-demand-broker" index:"3d004de5-1dae-4bcf-9af8-7b18e1e5b39a
```

```
origin:"p.redis" eventType:ValueMetric timestamp:1491922454382812931 deployment:"redis-on-demand-broker" job:"redis-on-demand-broker" index:"3d004de5-1dae-4bcf-9af8-7b18e1e5b39a
```

The following examples shows the number of available instances for On-Demand Plans across all On-Demand plans and for a specific On-Demand plan:

```
origin:"p.redis" eventType:ValueMetric timestamp:1491922966211762492 deployment:"redis-on-demand-broker" job:"redis-on-demand-broker" index:"3d004de5-1dae-4bcf-9af8-7b18e1e5b39a
```

```
origin:"p.redis" eventType:ValueMetric timestamp:1491922966211730803 deployment:"redis-on-demand-broker" job:"redis-on-demand-broker" index:"3d004de5-1dae-4bcf-9af8-7b18e1e5b39a
```

The following example shows the number of available instances for the Dedicated-VM plan metric:

```
origin:"p-redis" eventType:ValueMetric timestamp:1480084323333475533 deployment:"cf-redis" job:"cf-redis-broker" index:"3be5f4b9-cdf3-45c4-a3b2-19d923d63a01" ip:"10.0.1.49" valueMe
```

Redis Metrics

Redis emits a number of metrics that can be used to monitor the health and performance of your Redis deployment. Currently these metrics are only available for dedicated-VM and shared-VM plans. On-demand broker metrics can be seen here.

keyspace_hits

Description	Number of successful lookups of keys in the main dictionary. “/p-redis/info/stats/keyspace_hits”
Significance	In conjunction with `keyspace_misses`, it can be used to calculate the hit ratio.
Notes	A successful lookup is a lookup on a key that exists.

keyspace_misses

Description	Number of unsuccessful lookups of keys in the main dictionary. “/p-redis/info/stats/keyspace_misses”
Significance	In conjunction with `keyspace_hits`, it can be used to calculate the hit ratio.
Notes	An unsuccessful lookup is a lookup on a key that does not exist.

used_memory

Description	Number of bytes allocated by Redis. “/p-redis/info/memory/used_memory”
Significance	Grows as the number of unsaved keys increases.

maxmemory

Description	Maximum number of bytes available in Redis. “/p-redis/info/memory/maxmemory”
Significance	Indicates the max memory available in Redis.

blocked_clients

Description	Number of connected clients pending on a blocking call. “/p-redis/info/clients/blocked_clients”
Significance	Can be used as an indicator to detect deadlocks.

connected_clients

Description	Number of clients connected to the Redis instance. “/p-redis/info/clients/connected_clients”

rdb_changes_since_last_save

Description	Number of keys currently in memory. “/p-redis/info/persistence/rdb_changes_since_last_save”
Significance	Memory usage grows in proportion to the number of keys in memory. If the Redis instance is stopped ungracefully, these changes may be lost.
Notes	Performing a `BGSAVE` writes these keys to disk and frees up memory.

total_commands_processed

Description	Total number of commands processed by Redis. “/p-redis/info/stats/total_commands_processed”
Significance	A crude indicator of activity. Can be used in conjunction with `uptime_in_seconds`.

mem_fragmentation_ratio

Description	Ratio of memory allocated by the operating system to the memory requested by Redis. “/p-redis/info/memory/mem_fragmentation_ratio”
Significance	A ratio in excess of 1.5 indicates excessive fragmentation, with your Redis instance consuming 150% of the physical memory it requested..

total_instances [Dedicated VM]

Description	Total number of `dedicated-vm` instances of Redis. “/p-redis/service-broker/dedicated_vm_plan/total_instances”
Significance	Used in conjunction with `available_instances`, provides information about used instances.

available_instances [Dedicated VM]

Description	Number of available `dedicated-vm` instances of Redis. “/p-redis/service-broker/dedicated_vm_plan/available_instances”
Significance	If zero, no more instances are available.

total_instances [Shared-VM]

Description	Total number of `shared-vm` instances of Redis. “/p-redis/service-broker/shared_vm_plan/total_instances”
Significance	Used in conjunction with `available_instances`, provides information about used instances.

available_instances [Shared-VM]

Description	Number of available `shared-vm` instances of Redis. “/p-redis/service-broker/shared_vm_plan/available_instances”
Significance	If zero, no more instances are available.

total_instances [Across on-demand plans]

Description	Total number of `on-demand` instances of Redis available for all plans. “/on-demand-broker/p.redis/total_instances”
Significance	Used in conjunction with `total_instances`, provides information about used instances.

available_instances [Across on-demand plans]

Description	Number of available `on-demand` instances of Redis available for all plans. “/on-demand-broker/p.redis/quota_remaining”
Significance	If zero, no more instances are available.

total_instances [Per on-demand plan]

Description	Total number of `on-demand` instances of Redis available for a specific plan. “/on-demand-broker/p.redis/{plan_name}/total_instances”
Significance	Used in conjunction with `total_instances`, provides information about used instances per plan.

available_instances [Per on-demand plan]

Description	Number of available `on-demand` instances of Redis. “/on-demand-broker/p.redis/{plan_name}/quota_remaining”
Significance	If zero, no more instances are available per plan.

BOSH Health Monitor Metrics

The BOSH layer that underlies PCF generates `healthmonitor` metrics for all VMs in the deployment. However, these metrics are not included in the Loggregator Firehose by default. To get these metrics, do either of the following:

- To send BOSH HM metrics through the Firehose, install the open-source HM Forwarder.
- To retrieve BOSH health metrics outside of the Firehose, install the JMX Bridge for PCF tile.

Other 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`
- `mem_fragmentation_ratio`
- `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`
- `connected_slaves`

- `master_repl_offset`
- `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`
- `cluster_enabled`
- `rdb_last_bgsave_status`
- `aof_last_bgrewrite_status`
- `aof_last_write_status`

Redis for PCF Smoke Tests

Page last updated:

Redis for Pivotal Cloud Foundry (PCF) runs a set of smoke tests during installation to confirm system health. The tests run in the org `system` and in the space `redis-smoke-tests`. The tests run as an application instance with a restrictive Application Security Group (ASG).

Smoke Test Steps

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

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

Security Groups

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

```
[
  {
   "protocol": "tcp",
   "destination": "<dedicated node IP addresses>",
   "ports": "6379" // Redis dedicated node port
  },
  {
   "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 and dedicated VM nodes.

Smoke Tests Resilience

Smoke tests could fail due to reasons outside of the Redis deployment; for example network latency causing timeouts or the CloudFoundry instance dropping requests.

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 mentioned above - to ensure that the service instances and bindings created for testing are cleaned up.

Smoke tests will 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 will be 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, and - situations where the system is suffering from a failure that lasts a few seconds-such as restart of a CloudFoundry VM-where it's preferable to wait before reattempting the command.

Considerations

Because of the retry policy described in Smoke tests resilience, smoke tests will generally take longer than previous versions of Redis for PCF to complete

within environments with intermittent failures. If run regularly, e.g. as part of CI, they might last longer than the frequency of the automatic runs, queueing up executions. We recommend adjusting the frequency of the automatic smoke test runs based on the amount of time they take to complete.

The above retry policy does not guard against a more permanent Cloud Foundry downtime or network connectivity issues. In this case, commands will fail after the maximum number of attempts and might leave claimed instances behind. We recommend disabling automatic smoke test runs and manually releasing any claimed instances in case of upgrades or scheduled downtimes.

Troubleshooting

If errors occur while the smoke tests run, they will be summarised at the end of the errand log output. Detailed logs can be found where the failure occurs. Some common failures are listed below.

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

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

When encountering an error when running smoke tests, it can be helpful to search the log for other instances of the error summary printed at the end of the tests, e.g. `Failed to target Cloud Foundry`. Lookout for `TIP: ...` in the logs next to any error output for further troubleshooting hints.

Troubleshooting Redis for PCF

Page last updated:

This topic lists troubleshooting information relevant to Redis for PCF.

Knowledge Base Articles

Pivotal Knowledge Base articles specifically about Redis for PCF:

- Create an Empty Service Network to use the Redis Tile without enabling the On-Demand Service
- Can't redeploy PCF Redis if shared-vm persistent disk full
- Issue with upgrading tile
- Issue with deploy failing
- Redis Instance Alive after Successful De-provisioning
- PCF Redis dedicated instance fails to persist to disk
- Redis error when saving changes after a back to AWS S3: Error: Access Denied for bucket'
- Internet access disabled for tile and instances

Other Issues

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

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

Useful Debugging Information

If you encounter an issue, here is a list of useful information to gather, especially before you perform any destructive operations like purging service offerings or deleting deployments.

- Redis for PCF version
- Previous Redis for PCF version if upgrading
- Ops Manager version, and previous if upgrading Ops Manager
- IaaS description

From Ops Manager:

- The installation logs
- A copy of all files in `/var/tempest/workspaces/default/deployments`

For All VMs, Unless Specified Otherwise:

- Copy of `/var/vcap/sys/log` (particularly the broker)
- If unable to get logs from disk, logs from a forwarded endpoint
- `monit summary`

- Full `ps aux`: Has monit done its job?
- `ps aux | grep redis-serve[r]`: Are Redis instances running?
- `df -h`: disk usage
- `free -m`: memory usage
- `cf m`
- `tree /var/vcap/store/cf-redis-broker/redis-data` (broker only)
- Copy of `/var/vcap/store/cf-redis-broker/statefile.json` (broker only)

Introduction for App Developers

Page last updated:

This section provides an introduction to Redis for Pivotal Cloud Foundry (PCF) services for developers, and links to more information.

For instructions on creating, binding to, and deleting an instance of the On-Demand, Dedicated-VM, or Shared-VM plan see Using Redis for PCF.

Redis for PCF Services

Redis for PCF v1.8+ offers On-Demand, Dedicated-VM, and Shared-VM services.

- **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.
- **Dedicated-VM Service**—Provides a dedicated VM running a Redis instance. The Dedicated-VM Service is pre-provisioned by the operator with a fixed number of VMs and memory size. App developers can then use one of those pre-provisioned VMs.
- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development. 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 on the plans see the service offerings for the on-demand plan and the dedicated and shared plans.

Getting Started

Using Redis for PCF with Spring

Spring Cloud Connectors can connect to Redis for PCF. Spring Cloud Cloud Foundry connectors automatically connect to Redis for PCF.

To view an example Spring app demonstrating Redis as a cache with failover, see the Example Spring App in GitHub.

PCF Dev

PCF Dev is a small footprint version of PCF that's small enough to run on a local developer machine. For more information, seehttps://pivotal.io/pcf-dev.

Redis Example App

Sample ruby code that uses PCF can be found here https://github.com/pivotal-cf/cf-redis-example-app.

Redis

To learn more about Redis itself, see redis.io.

Using Redis for PCF

Page last updated:

Redis for Pivotal Cloud Foundry (PCF) can be used both via Pivotal Apps Manager and the Cloud Foundry Command Line Interface (cf CLI). Both methods are outlined below.

You can find an example app has to help you get started with Redis for PCF. Download the example app by clicking this link.

For recommendations regarding Redis for PCF service plans and memory allocation, see the service offerings for the on-demand plan and the dedicated and shared plans. .

Prerequisites

To use Redis for PCF with your PCF apps, you need:

- A PCF installation with Redis for PCF installed and listed in the Marketplace. The three Redis services are listed differently in the marketplace, ensure the service you want to use is enabled.
- A Space Developer or Admin account on the PCF installation
- To use the cf CLI, you must log into the org and space containing your app and have a local machine with the following installed:
  - A browser
  - A shell
  - The Cloud Foundry Command-Line Interface (cf CLI)

Use Redis for PCF in a PCF app

Every app and service in PCF 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 PCF in a PCF app:

1. Use the cf CLI or Apps Manager to log in to the org and space that contains the app.
2. Make sure an instance of the Redis for PCF service exists in the same space as the app.
   - If the space does not already have a Redis for PCF instance, create one.
   - If the space already has a Redis for PCF 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 PCF service instance, to enable the app to use Redis.

Confirm Redis for PCF 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 for PCF service is available in the Marketplace:
   a. Enter `cf marketplace`.
   b. If the output lists `p.redis` in the `service` column, on-demand Redis for PCF is available. If the output lists `p-redis` in the `service` column, dedicated-VM and shared-VM Redis for PCF is available. If it is not available, ask your operator to install it.

```
$ 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, dedicated-vm   Redis service to provide pre-provisioned instances configured as a datastore, running on a shared or dedicated VM.
p.redis         cache-small, cached-med   Redis service to provide on-demand dedicated instances configured as a cache.
[...]
```

2. To confirm that a Redis for PCF instance is running in the space:

    a. Enter `cf services`.

    b. Any `p.redis` listings in the `service` column are service instances of on-demand Redis for PCF in the space. Any `p-redis` in the `service` column are service instances of dedicated-VM and shared-VM Redis for PCF.

```
$ cf services
Getting services in org my-org / space my-space as user@example.com...
OK
name          service    plan         bound apps    last operation
my-instance   p.redis    cache-small                create succeeded
```

You can bind your app to an existing instance or create a new instance to bind to your app.

Create a Redis for PCF Service Instance

Create a Service Instance with the cf CLI

Dedicated-VM and Shared-VM Service

Dedicated-VM and Shared-VM service instances have been pre-provisioned by the operator. This means, if an instance is available, the app developer can provision it immeditately. These plans are both listed under the `p-redis` service in the Marketplace.

To create an instance of the Redis for PCF Dedicated-VM of Shared-VM service, run this command:

```
cf create-service p-redis SERVICE_TYPE
SERVICE_NAME
```

where:

- `SERVICE_TYPE` is `dedicated-vm` or `shared-vm`.
- `SERVICE_NAME` is a name for your service instance.

```
$ cf create-service p-redis dedicated-vm dedicated-instance

Creating service dedicated-instance in org my-org / space my-space as user@example.com...
OK
```

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 an instance of the Redis for PCF On-Demand service, run this command:

```
cf create-service p.redis CACHE_PLAN
SERVICE_NAME
```

where:

- `CACHE_PLAN` is `cache-small`, `cache-medium`, or `cache-large`.
- `SERVICE_NAME` is a name for your service.

```
$ cf create-service p.redis cache-small od-instance

Creating service my-ondemand-instance in org my-org / space my-space as user@example.com...
OK
```

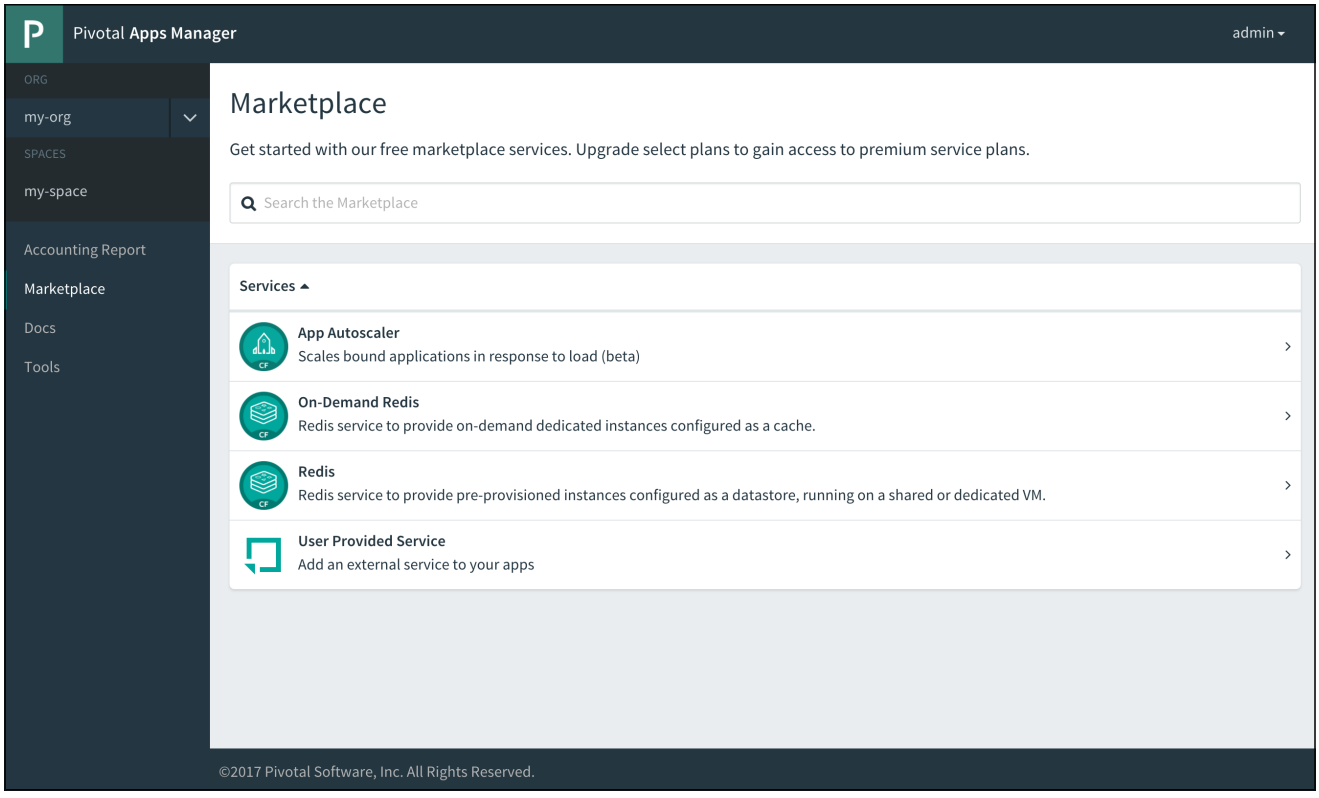
As the On-Demand instance can take longer to create, the `watch` command is helpful as a way to track when your service instance is ready to bind and use.

```
$ 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    cache-small                create succeeded
```

If you get an error, see Troubleshooting Instances. For information on the on-demand cache plans, see On-Demand Service Plans.

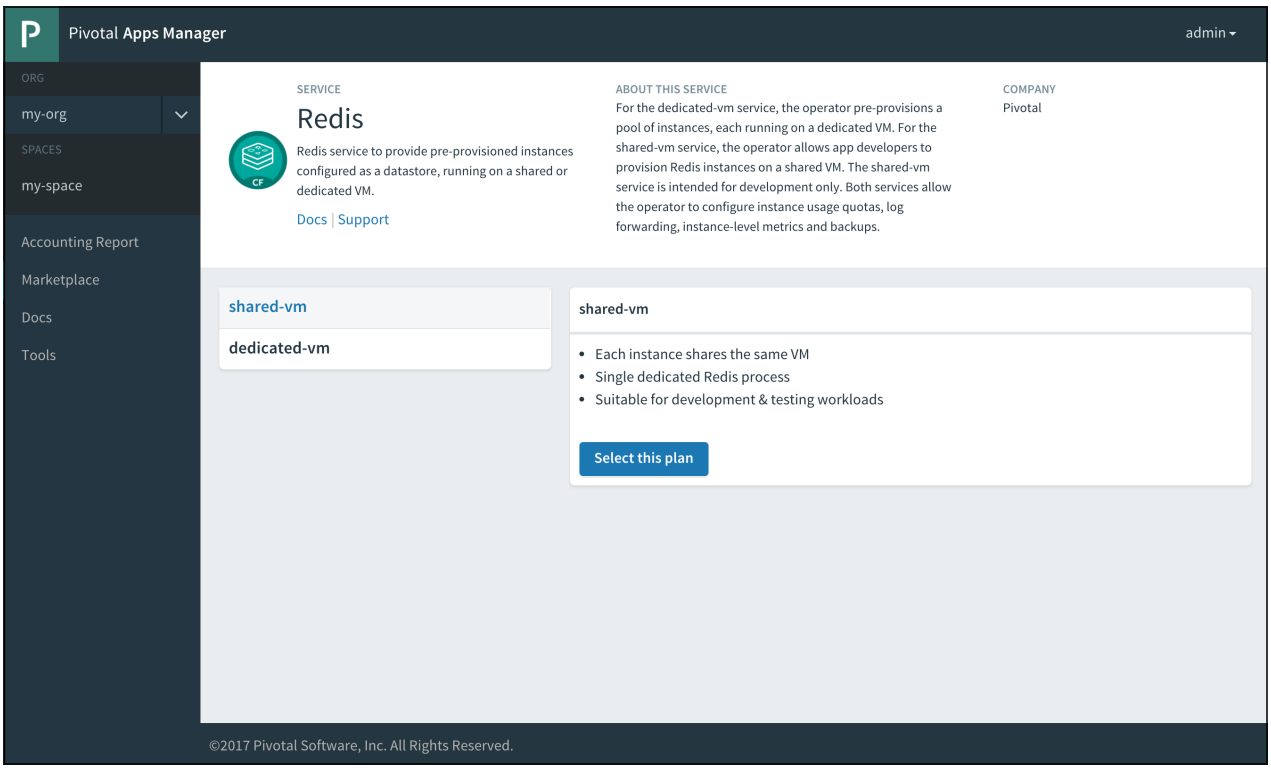
Create a Service Instance with Apps Manager

From within Pivotal Apps Manager, select **Marketplace** from the left navigation menu under Spaces.

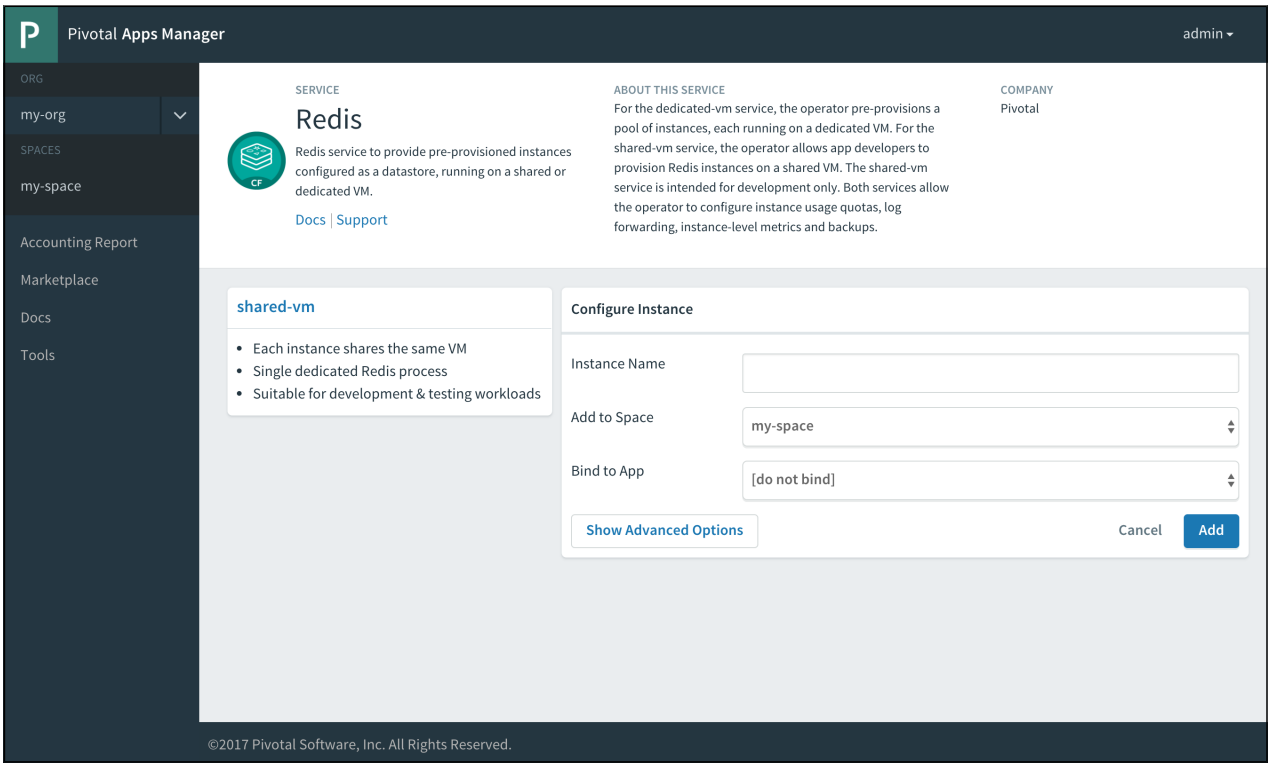


Dedicated-VM and Shared-VM Service

1. Select **Redis** from the displayed tiles in the Marketplace.



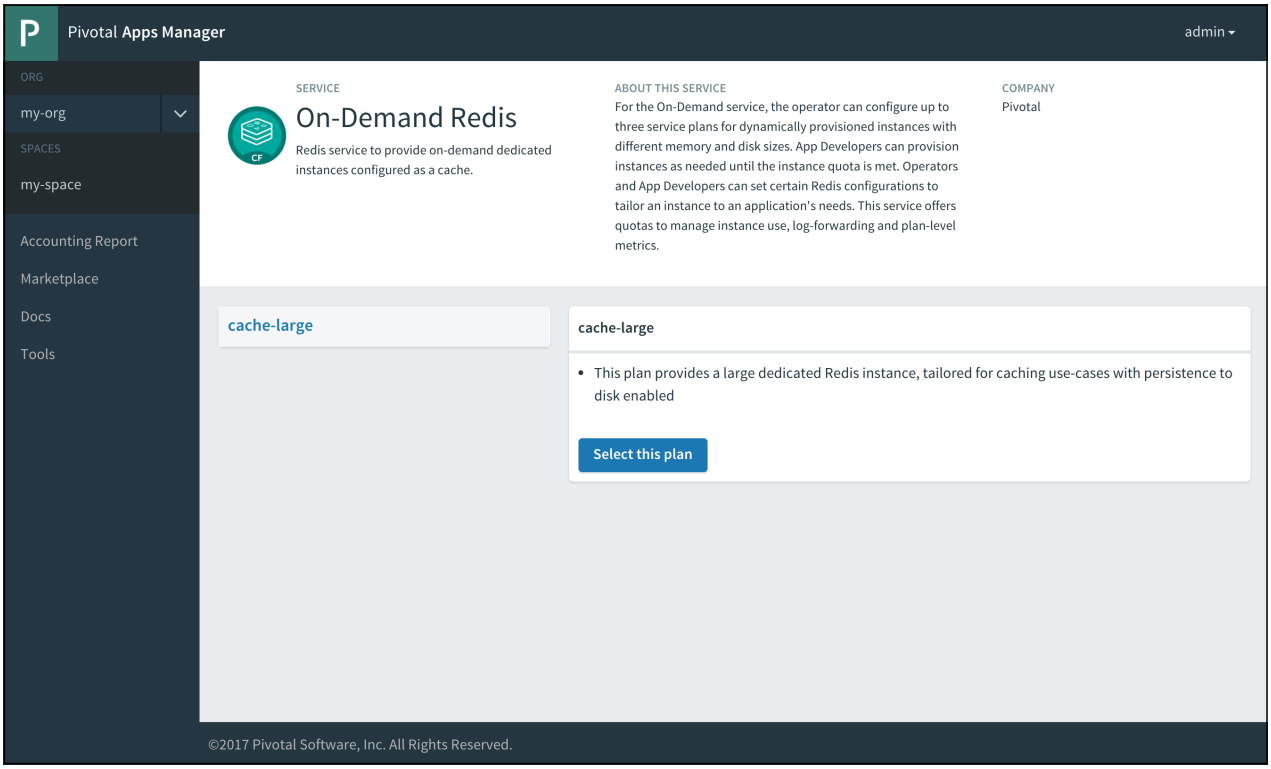
2. Click on the appropriate **Select this plan** button to select the required **Redis Service Plan**.



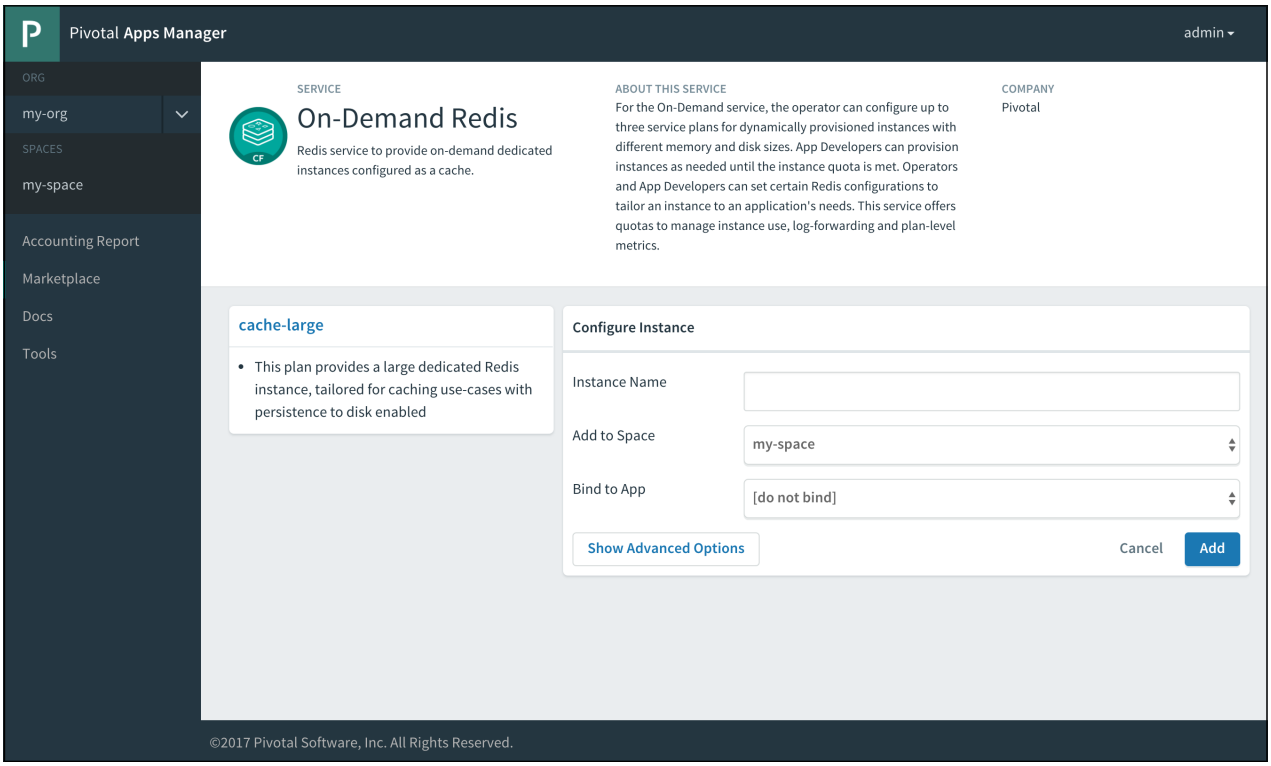
3. In the **Instance Name** field, enter a name that will identify this specific Redis service instance.
4. From the **Add to Space** drop-down list, select the space where you or other users will deploy the apps that will bind to the service.
5. Click the **Add** button.

On-Demand Service

1. Select **On-Demand Redis** from the displayed tiles in the Marketplace.



2. Click on the appropriate **Select this plan** button to select the required **Redis Service Plan**.



3. In the **Instance Name** field, enter a name that will identify this specific Redis service instance.

4. From the **Add to Space** drop-down list, select the space where you or other users will deploy the apps that will bind to the service.

5. 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. Do this after you push or re-push the app using `cf push`.

Bind a Service Instance with the cf CLI

To bind an app to a Redis for PCF instance use `$ cf bind-service`.

1. Run `cf services` to view running service instances.

```
$ cf services

Getting services in org system / space apps-manager as admin...
OK

name              service      plan       bound apps   last operation
my-instance       p-redis      shared-vm                create succeeded
```

2. Enter `cf bind-service APP 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`.

```
$ cf bind-service my-app my-instance

Binding service my-instance to my-app in org my-org / space test as user@example.com...
OK
TIP: Use 'cf push' to ensure your env variable changes take effect
```

Bind a Service Instance with Apps Manager

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`	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.
	128	1-2024	Sets the length (count) of the slowlog queue.

`slowlog-max-len`			

Customize an On-Demand Instance with the cf CLI

You can customize an instance in two ways:

- While creating the instance, run:
  `cf create-service SERVICE PLAN NAME -c '{"PROPRETY":"SETTING"}'`
- After creating the instance, run:
  `cf update-service NAME -c '{"PROPRETY":"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"}'
```

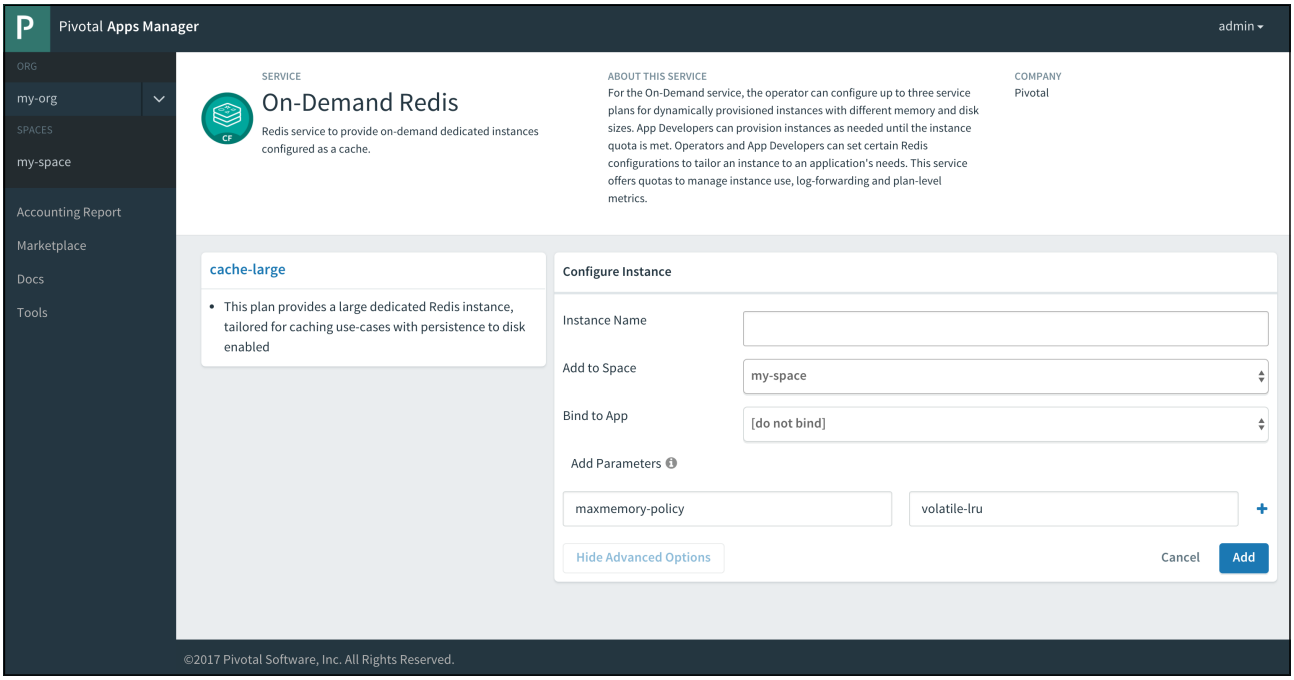
If the update is not successful, an error is displayed with a description of what went wrong. Here is an example where a hyphen is added to the `noeviction` setting.

```
$ cf update-service my-instance -c '{"maxmemory-policy":"no-eviction", "notify-keyspace-events":"El"}'
Updating service instance my-instance as admin...
FAILED
Server error, status code: 502, error code: 10001, message: Service broker error: invalid value "no-eviction" specified for maxmemory-policy
```

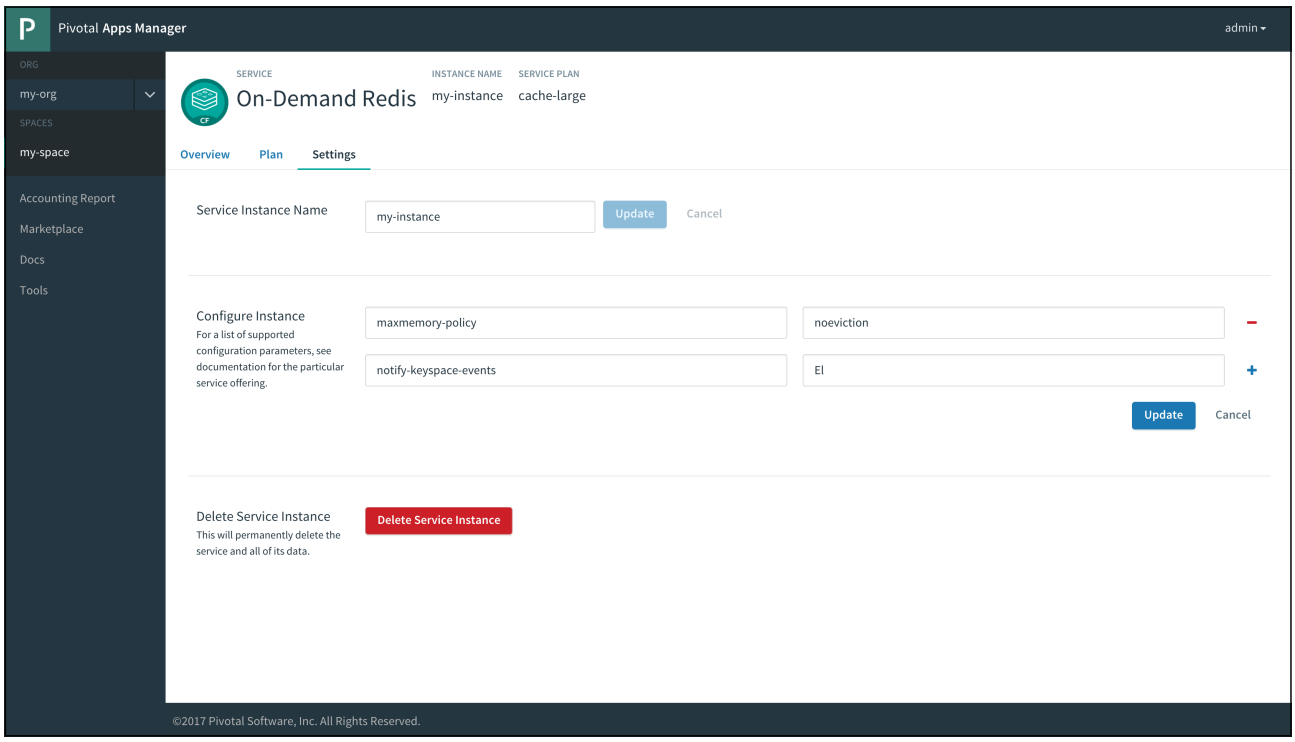
Customize an On-Demand Instance with the Apps Manager

You can customize an instance in two ways:

- While creating the instance, after you select the plan, click **advanced settings**.



- After creating the instance, navigate to the instance Settings page.

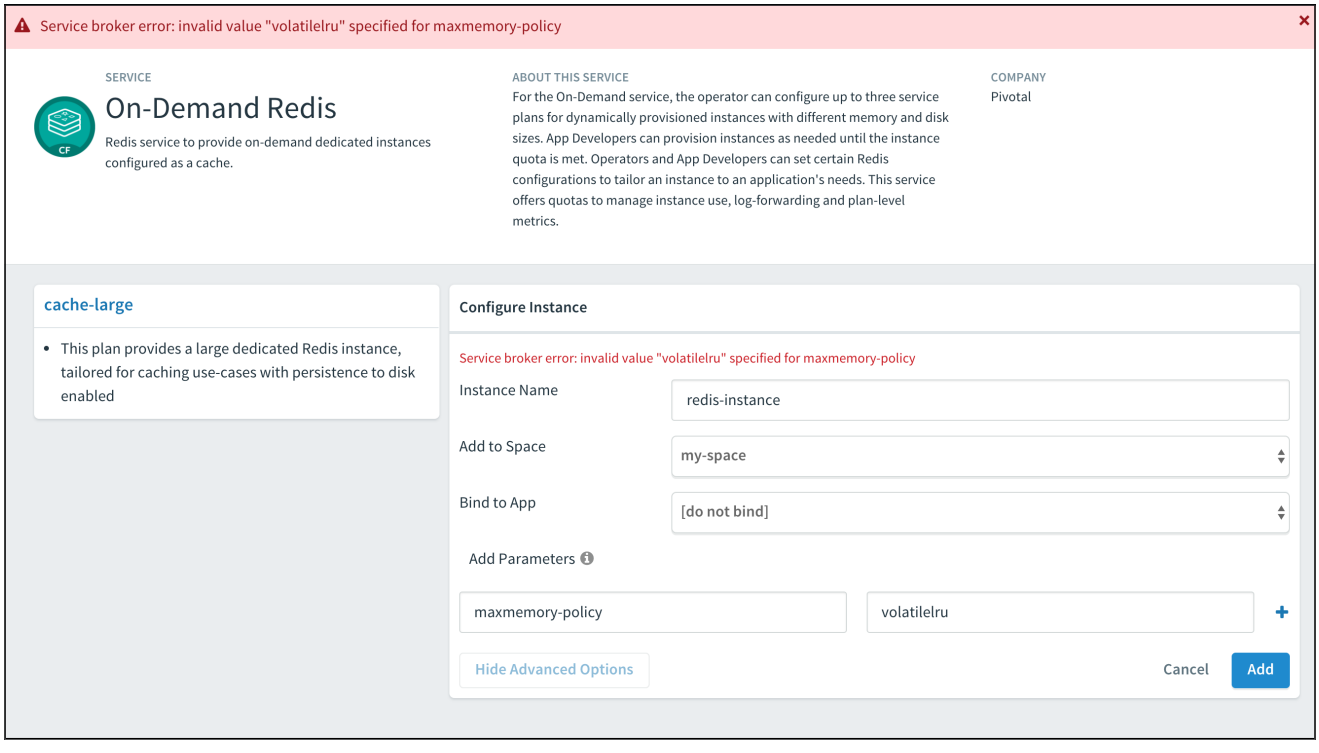


In either of the above cases, do the following:

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 not successful, an error is diplayed with a description of what went wrong. Here is an example where we forgot the hyphen in the `volatile-lru` setting.



Access the Redis Service

All Redis for PCF 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 instance using the command `cf create-service-key INSTANCE-NAME SERVICE-KEY-NAME`.
2. Retrieve the password using the command `cf service-key INSTANCE-NAME SERVICE-KEY-NAME`.

Here is an example of this procedure:

```
$ cf create-service-key my-instance my-key
Creating service key my-key for service instance my-instance as admin...
OK
$ cf service-key my-instance my-key
Getting key my-key for service instance my-instance as admin...
{
 "host": "10.0.8.4",
 "password": "",
 "port": 6379
}
```

Redis for PCF data is accessible from apps bound to that instance. Some Redis for PCF users bind the opensource cf-redis-commander app to view instance data. This app is not maintained by the Redis for PCF team, and Pivotal cannot guarantee its performance or security.

Use the Redis Service in Your App

To access the Redis service from your app:

1. Run `cf env APP_NAME` with the name of the app bound to the Redis for PCF instance.
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": "<redacted>",
     "port": 6379
  },
  "label": "p-redis",
  "name": "redis",
  "plan": "dedicated-vm",
  "provider": null,
  "syslog_drain_url": null,
  "tags": [
  "pivotal",
  "redis"
  ],
  "volume_mounts": []
 }]
}
```


> **Note**: 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.

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 Instance with the cf CLI

1. Run `cf delete-service SERVICE-INSTANCE-NAME` and enter `y` when prompted to confirm.

For example:

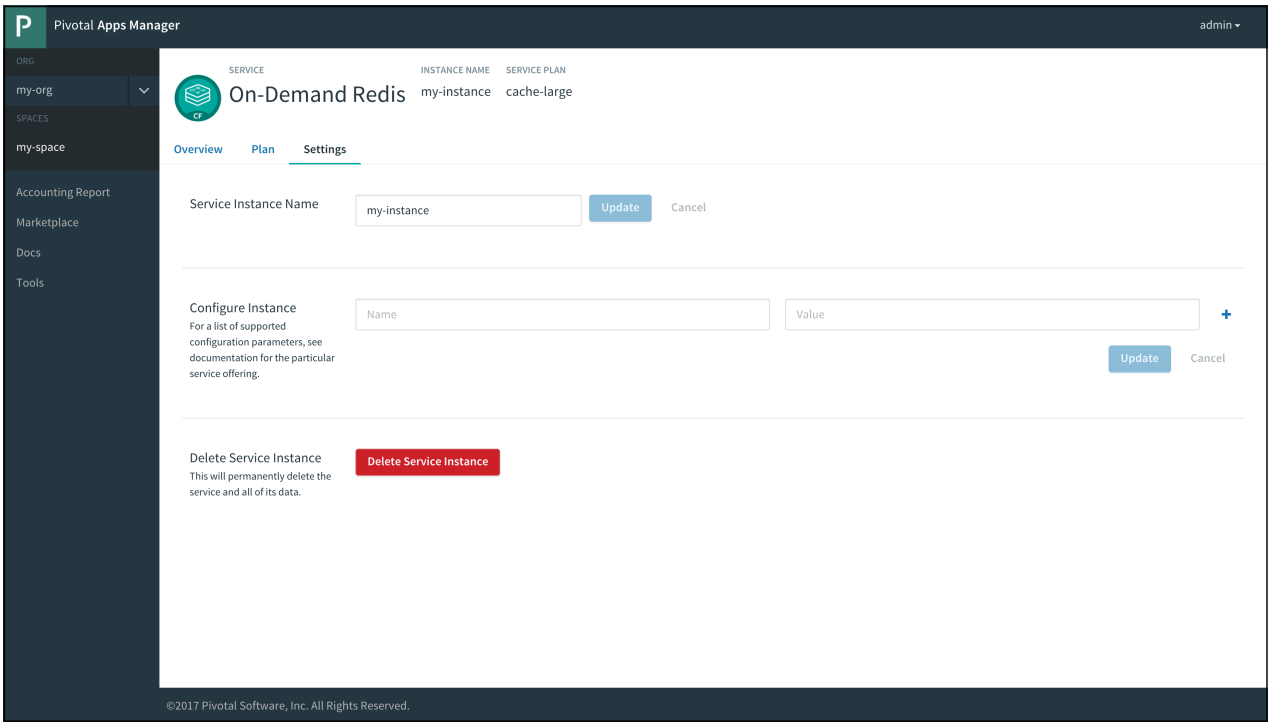
```
$ cf delete-service my-redis-instance

Really delete the service my-redis-instance?> y
Deleting service my-redis-instance in org system / space apps-manager as 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 Instance with Pivotal Apps Manager

1. In the service instance Settings page, click **Delete Service Instance**.



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
```

Troubleshooting Instances

Page last updated:

This topic provides basic instructions for app developers troubleshooting On-Demand Redis for Pivotal Cloud Foundry (PCF).

About the BOSH CLI

The BOSH CLI is available in two major versions, v1 and v2. Pivotal recommends that you use the BOSH CLI v2 when possible.

This topic provides examples of using each version of the BOSH CLI. While all versions of the BOSH CLI work with Redis 1.9, your PCF installation may affect which version of the BOSH CLI you can use. Consult the table below to determine which version of the CLI is supported for your installation.

PCF Version	BOSH CLI Version
1.10	CLI v1
1.11	CLI v1 or CLI v2 (Pivotal recommends CLI v2)

Temporary Outages

Redis for PCF service instances can become temporarily inaccessible during upgrades and VM or network failures.

Errors

You may see an error when using the Cloud Foundry Command-Line Interface (cf CLI) to perform basic operations on a Redis for PCF service instance:

- `cf create`
- `cf update`
- `cf bind`
- `cf unbind`
- `cf delete`

Parse a Cloud Foundry (CF) Error Message

Failed operations (create, update, bind, unbind, delete) result in 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 request.
    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 Pivotal Support when filing a ticket.

The `task-id` field maps to the BOSH task id. For further information on a failed BOSH task, use the `bosh task TASK-ID` command in v1 of the BOSH CLI. For v2, use `bosh2 task TASK-ID`.

The `broker-request-guid` maps to the portion of the On-Demand 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

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, run `cf services` to see a listing of all service instances in the space. The service instances are listed in the `name` column.

```
$ cf services
Getting services in org my-org / space my-space as user@example.com...
OK
name          service     plan       bound apps    last operation
my-instance   p.Redis     db-small                 create succeeded
```

3. Run `cf service SERVICE-INSTANCE-NAME` to retrieve more information about a specific instance.

4. Run `cf service SERVICE-INSTANCE-NAME --guid` to retrieve the GUID of the instance, which is useful for debugging.

Select the BOSH Deployment for a Service Instance

This is an additional troubleshooting option for **BOSH CLI v1 only**. It does not apply to the BOSH CLI v2.

1. Retrieve the GUID of your service instance with the command `cf service YOUR-SERVICE-INSTANCE --guid`.

2. To download your BOSH manifest for the service, run `bosh download manifest service-instance_SERVICE-INSTANCE-GUID myservice.yml` using the GUID you just obtained and a file name you want to use when saving the manifest.

3. Run `bosh deployment MY-SERVICE.yml` to select the deployment.

Knowledge Base (Community)

Find the answer to your question and browse product discussions and solutions by searching the Pivotal Knowledge Base.

File a Support Ticket

You can file a support ticket here. Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To help expedite troubleshooting, if possible also provide your service broker logs, service instance logs, and BOSH task output. Your cloud operator should be able to obtain these from your error message.

Sample Redis Configuration

Page last updated:

The following is a `redis.conf` file from a Dedicated-VM plan instance:

```
daemonize yes
pidfile /var/vcap/sys/run/redis.pid
port 6379
tcp-backlog 511
timeout 0
tcp-keepalive 0
loglevel notice
logfile /var/vcap/sys/log/redis/redis.log
syslog-enabled yes
syslog-ident redis-server
syslog-facility local0
databases 16
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename dump.rdb
dir /var/vcap/store/redis
slave-serve-stale-data yes
slave-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-ping-slave-period 10
repl-timeout 60
repl-disable-tcp-nodelay no
slave-priority 100
maxmemory-policy noeviction
appendonly yes
appendfilename appendonly.aof
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
lua-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-entries 512
list-max-ziplist-value 64
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
aof-rewrite-incremental-fsync yes
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 ""
requirepass 1a1a2bb0-0ccc-222a-444b-1e1e1e1e2222
maxmemory 1775550873
```