

# Redis for Tanzu Application Service v2.0

Redis for VMware Tanzu Application Service 2.0

You can find the most up-to-date technical documentation on the VMware website at:  
<https://docs.vmware.com/>

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

Copyright © 2023 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

# Contents

About Redis for PCF	12
Redis for PCF	12
Product Snapshot	12
About Redis	12
About Redis for PCF	13
Is Redis for PCF right for your enterprise?	13
Upgrading to the Latest Version	13
More Information	13
Redis for PCF and Other PCF Services	13
Feedback	14
Redis for PCF Release Notes	14
v2.0.6	15
Security Fixes	15
Resolved Issues	15
Known Issues	15
Compatibility	15
v2.0.5	16
Security Fixes	16
Known Issues	16
Compatibility	16
v2.0.4	16
Security Fixes	16
Resolved Issues	17
Known Issues	17
Compatibility	17
v2.0.2	17
Feature	18
Security Fixes	18
Resolved Issues	18
Known Issues	18
Compatibility	18
v2.0.1	19
Security Fixes	19

Known Issues	19
Compatibility	19
v2.0.0	20
Features	20
Known Issues	20
Compatibility	20
View Release Notes for Another Version	21
<b>Planning Guide</b>	<b>22</b>
<b>Is Redis for PCF right for your enterprise?</b>	<b>22</b>
Recommended Use Cases	22
SLO Benchmark	23
Service Offerings	23
Enterprise-Readiness Checklist	23
Support for Multiple AZs	24
<b>Service Offerings</b>	<b>25</b>
<b>On-Demand Service Offering</b>	<b>25</b>
Architecture Diagram for On-Demand Plan	25
On-Demand Service Plans	26
Three On-Demand Cache Plans	26
Features of On-Demand Service Plans	27
Configuration of On-Demand Service Plans	27
Operator Configurable Redis Settings	27
App Developer Configurable Redis Settings	28
Operator Notes for On-Demand Service Plans	28
Known Limitations for On-Demand Service Plans	28
Lifecycle for On-Demand Service Plan	28
<b>Shared-VM Service Offering</b>	<b>30</b>
About the Shared-VM Plan	30
Architecture Diagram for Shared Plans	30
Settings for Shared-VM Service Plans	31
Memory Policy	32
Persistence	32
Number of Connections	32
Replication and Event Notification	32
Operator Notes for the Shared-VM Plan	32

Known Limitations of the Shared-VM Plan	33
Lifecycle for Shared-VM Service Plan	33
Networking for On-Demand Services	34
Service Network Requirement	35
Default Network and Service Network	35
Required Networking Rules for On-Demand Services	36
Redis for PCF Security	37
Security	38
Operator Guide	39
Introduction for Operators	39
Best Practices	39
Redis Key Count and Memory Size	39
Errands	40
Post-Deploy Errands	40
Pre-Delete Errands	40
Turning off Post-Deploy Errands	41
Changes to Redis for PCF Tile Configuration	41
Installing Another Tile	41
Changes to Other Tiles	41
Broker Registrar Errand	41
Deprecate Dedicated-VM Plan Errand	41
Register On-Demand Broker Errand	41
Smoke Tests and On-Demand Smoke Tests Errands	41
Upgrade All On-Demand Service Instances Errand	42
Recreate All On-Demand Service Instances Errand	42
Smoke Tests	42
Installing Redis for PCF	42
Role-Based Access in Ops Manager	43
Download and Install the Tile	43
Assign AZs and Networks	43
Assign AZs	44
Select Networks	44
Port Ranges Used in Redis for PCF	45
Configure Redis for PCF Service Plans	45
On-Demand Service Settings	46
On-Demand Plan Settings	48

(Optional) Enabling Secure Service Instance Credentials for On-Demand Redis	50
Updating On-Demand Service Plans	51
Removing On-Demand Service Plans	51
Shared-VM Plan Settings	52
Configure Memory Limits for Shared-VM Plans	53
Configure Resources for Shared-VM Plans	54
Disable Shared VM Plans	54
Configure Syslog Forwarding	54
Update Stemcell	57
Apply Changes from Your Configuration	57
Create Application Security Groups	58
Application Container Network Connections	58
Validating Installation	58
Uninstalling Redis for PCF	58
<b>Upgrading Redis for PCF</b>	<b>59</b>
Compatible Upgrade Paths	59
Upgrade to Redis for PCF v2.0	59
Downtime During Upgrades and Redeploys	60
Ops Manager Changes	60
PAS Changes	60
Upgrading all Service Instances	60
Network Changes after Deployment	61
Shared VMs	61
On-Demand Service Instances	61
Release Policy	61
<b>Setting Limits for On-Demand Service Instances</b>	<b>61</b>
Create Global-level Quotas	62
Create Plan-level Quotas	62
Create and Set Org-level Quotas	63
Create and Set Space-level Quotas	63
View Current Org and Space-level Quotas	64
Monitor Quota Use and Service Instance Count	64
Calculate Resource Costs for On-Demand Plans	65
Calculate Maximum Resource Cost Per On-Demand Plan	66
Calculate Maximum Resource Cost for All On-Demand Plans	66
Calculate Actual Resource Cost of all On-Demand Plans	67

<b>Configuring Automated Service Backups</b>	<b>67</b>
Comparison of the Available Backup Methods	67
About Automated Service Backups	68
Backup Files	68
About Configuring Backups	68
Option 1: Back Up with AWS	69
Create a Policy and Access Key	69
Configure Backups in Ops Manager	70
Option 2: Back Up with SCP	72
(Recommended) Create a Public and Private Key Pair	72
Configure Backups in Ops Manager	72
Option 3: Back Up with GCS	74
Create a Service Account	74
Configure Backups in Ops Manager	75
Back Up to Azure	77
Back Up and Restore Manually	79
<b>BOSH Backup and Restore (BBR) for On-Demand Redis for PCF</b>	<b>79</b>
Prepare to Use BBR	79
Identify Your Redis Deployments	80
Back Up Using BBR	80
Restore Using BBR	81
Possible Inconsistent States	82
No Backup Artifact for a Service Instance	82
Backup Artifact for a Non-Existent Service Instance	82
<b>Monitoring Redis for PCF</b>	<b>83</b>
Metrics Polling Interval	83
Critical Logs	83
Key Performance Indicators	83
Redis for PCF Service KPIs	84
Total Instances For On-Demand Service	84
Quota Remaining For On-Demand Service	84
Total Instances For Shared-VM Service	84
Redis KPIs	85
Percent of Persistent Disk Used	85
Used Memory Percent	85
Connected Clients	86
Blocked Clients	87
Memory Fragmentation Ratio	88

Instantaneous Operations Per Second	89
Keyspace Hits / Keyspace Misses + Keyspace Hits	90
BOSH Health Monitor Metrics	91
Other Redis Metrics	91
<b>Redis for PCF Smoke Tests</b>	<b>93</b>
Smoke Test Steps	93
Security Groups	93
Smoke Tests Resilience	94
Considerations	94
Troubleshooting	94
<b>Troubleshooting Redis for PCF</b>	<b>95</b>
Useful Debugging Commands	95
cf CLI Commands	95
BOSH CLI Commands	95
About the Redis CLI	96
Troubleshooting Errors	96
Failed Installation	96
Cannot Create or Delete Service Instances	97
Broker Request Timeouts	97
Cannot Bind to or Unbind from Service Instances	98
Instance Does Not Exist	98
Other Errors	98
Cannot Connect to a Service Instance	99
Upgrade All Service Instances Errand Fails	99
Missing Logs and Metrics	99
Error Messages Logged in Syslog	100
AOF File Corrupted, Cannot Start Redis Instance	100
Saving Error	101
Failed Backup	102
Orphaned Instances	102
BOSH Director Cannot See Your Instances	102
PCF Cannot See Your Instances	103
Failed to Set Credentials in Runtime CredHub	104
Troubleshooting Components	105
BOSH Problems	105
Large BOSH Queue	105
Configuration	105



Service Instances in Failing State	105
Authentication	106
UAA Changes	106
Networking	106
Validate Service Broker Connectivity to Service Instances	106
Validate App Access to a Service Instance	106
Quotas	106
Plan Quota Issues	107
Global Quota Issues	107
Failing Jobs and Unhealthy Instances	107
Techniques for Troubleshooting	107
Parse a Cloud Foundry (CF) Error Message	108
Access Broker and Instance Logs and VMs	108
Access Broker Logs and VMs	108
Access Service Instance Logs and VMs	109
Run Service Broker Errands to Manage Brokers and Instances	110
Register Broker	110
Deregister Broker	110
Upgrade All Service Instances	111
Delete All Service Instances	111
Detect Orphaned Service Instances	112
Get Admin Credentials for a Service Instance	113
Reinstall a Tile	114
View Resource Saturation and Scaling	114
Identify a Service Instance Owner	114
Monitor the Quota Saturation and Service Instance Count	115
Pivotal Support Articles	115
<b>Application Developer Guide</b>	<b>117</b>
<b>Introduction for App Developers</b>	<b>117</b>
Redis for PCF Services	117
Getting Started	117
Using Redis for PCF with Spring	118
Using Redis for PCF with Steeltoe	118
PCF Dev	118
Redis Example App	118
Redis	118
<b>Quickstart Guide for App Developers</b>	<b>118</b>

Feedback	119
Quickstart Apps	119
Quickstart Java App	119
Quickstart Node App	120
Quickstart Ruby App	122
Spring Session with Redis for PCF	123
Setting Up Spring Session	123
Updating Dependencies	123
Spring Java Configuration	123
Java Servlet Container Initialization	124
Configuring Redis for PCF as a Backend	125
Other Considerations	125
<b>Using Redis for PCF</b>	<b>125</b>
Prerequisites	126
Use Redis for PCF in a PCF app	126
Confirm Redis for PCF Service Availability	126
Create a Redis for PCF Service Instance	127
Create a Service Instance with the cf CLI	127
Shared-VM Service	127
On-Demand Service	128
Create a Service Instance with Apps Manager	128
Shared-VM Service	129
On-Demand Service	129
Bind a Service Instance to Your App	130
Bind a Service Instance with the cf CLI	131
Bind a Service Instance with Apps Manager	131
Customize an On-Demand Service Instance	131
Customize an On-Demand Instance with the cf CLI	132
Customize an On-Demand Instance with the Apps Manager	132
Retrieve the Password for a Redis Service Instance	134
Use the Redis Service in Your App	134
Manage Key Eviction for Shared-VM Instances	135
Access Redis Metrics for On-Demand Service Instances	135
Sharing a Redis Instance with Another Space	136
Share a Redis Service Instance	136
Unshare a Redis Service Instance	136
Delete a Redis Instance	137
Delete a Redis Instance with the cf CLI	137
Delete a Redis Instance with Pivotal Apps Manager	137

Troubleshooting Instances	138
Debugging Using the CF CLI	138
Temporary Outages	139
Errors	139
Parse a Cloud Foundry (CF) Error Message	139
Retrieve Service Instance Information	140
Retrieve the Password for a Redis Service Instance	140
Error Messages from the Redis Client	141
Maximum Number of Clients Reached	141
Maxmemory Limit Reached	141
Error When Running the Save Command	142
Unknown Command Error	142
Knowledge Base (Community)	142
File a Support Ticket	142
Sample Redis Configuration	143

# About Redis for PCF

## Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

This is documentation for Redis for Pivotal Cloud Foundry (PCF). You can download the Redis for PCF tile from [Pivotal Network](#).

This documentation:

- Describes 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	2.0.6
Release date	October 7, 2019
Software component version	Redis OSS 5.0.5
Compatible Ops Manager version(s)	2.3, 2.4, 2.5, and 2.10
Compatible Pivotal Application Service (PAS) version(s)	2.3, 2.4, 2.5, and 2.10
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	Yes

## About Redis

**Redis** is an easy to use, high speed key-value store that can be used as a database, cache, and message broker. It supports a range of data structures including strings, lists, hashes, sets, bitmaps, hyperloglogs, and geospatial indexes. It 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 Pivotal Cloud Foundry (PCF).

Redis for PCF v2.0 and later offers On-Demand 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.
- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development purposes only, **do not use the Shared-VM service in production environments**. The Shared-VM instances are pre-provisioned by the operator with a fixed number of instances and memory size. App developers can then use one of these pre-provisioned instances.



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

For more information on the plans, see:

- [On-Demand Service Offering](#)
- [Shared-VM Service Offering](#)

## Is Redis for PCF right for your enterprise?

For information on recommended use cases, and the enterprise-readiness of Redis for PCF, see [Is Redis for PCF right for your enterprise?](#).

## Upgrading to the Latest Version

For information on how to upgrade and the supported upgrade paths, see [Upgrading Redis for PCF](#).

## More Information

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

For more information about...	See...
Product compatibility	<a href="#">Product Version Matrix</a>
How to upgrade Redis for PCF	<a href="#">Upgrading Redis for PCF</a>
How to use Redis	<a href="#">Redis Documentation</a>

## Redis for PCF and Other PCF Services

As well as Redis for Pivotal PCF, other Pivotal Cloud Foundry services offer *on-demand* service plans. These plans let developers provision service instances when they want.

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

The following table lists which Pivotal Cloud Foundry services offer on-demand and pre-provisioned service plans:

Pivotal Cloud Foundry service tile	Standalone product related to the service	Supports on-demand	Supports pre-provisioned
RabbitMQ for Pivotal Platform	Pivotal RabbitMQ	Yes	Yes. Only recommended for test environments.
Redis for PCF	Redis	Yes	Yes (shared-VM plan). Only recommended for test environments.
MySQL for Pivotal Platform	MySQL	Yes	No
Pivotal Cloud Cache (PCC)	Pivotal GemFire	Yes	No

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

## Feedback

Please provide any bugs, feature requests, or questions to [the Pivotal Cloud Foundry Feedback list](#).

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Redis for PCF Release Notes



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.



**Warning:** If you are upgrading PCF v2.3 to v2.4, you must run the `bosh -d YOUR-DEPLOYMENT restart redis-on-demand-broker` errand after upgrading. Failure to do so causes compatibility issues with runtime CredHub.

## v2.0.6

**Release Date: October 7, 2019**

### Security Fixes

This release includes the following security fixes:

- Updates nginx and goLang to resolve [CVE-2019-9512](#), [CVE-2019-9514](#), and [CVE-2019-9515](#).
- Uses a secured credential for S3 service backups.

### Resolved Issues

This release has the following fix:

- Configuring S3 service backups no longer results in an error due to a non-configurable property.

### Known Issues

This release has the following issue:

- The **When Changed** option for errands has unexpected behavior. Do not select this choice as an errand run rule. For more information about errand run rules, see [Errand Run Rules](#).
- Smoke tests fail when run in environments with Ruby buildpack v1.8.11 or later.

### Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	456.x
PCF	2.3.x, 2.4.x, 2.5.x, and 2.10.x
shared-redis-release	435.0.0
on-demand-service-broker	0.33.0
routing	0.188.0
service-metrics	1.12.1
service-backup	18.3.2
syslog	11.4.0
loggregator-agent	3.21.1
bpm	1.1.3

Redis OSS	5.0.5
-----------	-------

## v2.0.5

**Release Date: August 27, 2019**

### Security Fixes

This release includes the following security fixes:

- Updates bpm to v1.1.1 to resolve [CVE-2019-9893](#).

### Known Issues

This release has the following 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 option. For more information about this unexpected behavior, see [Errand Run Rules](#).
- Configuring S3 service backups results in an error due to a non-configurable property.

### Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	315.x
PCF	2.3.x, 2.4.x, and 2.5.x
cf-redis-release	434.3.12
on-demand-service-broker	0.31.1
routing	0.188.0
service-metrics	1.12.0
service-backup	18.3.1
syslog	11.4.0
loggregator-agent	3.21.0
bpm	1.1.1
Redis OSS	5.0.5

## v2.0.4

**Release Date: August 5, 2019**

### Security Fixes



This release includes the following security fixes:

- Updates open source Redis to v5.0.5, which includes fixes for CVEs and issues that caused crashes in very rare cases. For more information about Redis v5.0.5, see the [Redis 5.0 release notes](#).
- Updates golang to v1.12.6. For release notes, see the [golang documentation](#).
- Updates on-demand broker to v0.31.1.
- Updates the cf CLI to v1.16, which resolves [CVE-2019-3781: CF CLI does not sanitize user's password in verbose/trace/debug](#).

## Resolved Issues

This release has the following fix:

- Smoke tests now run in the `pivotal-services` space.

## Known Issues

This release has the following 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 option. For more information about this unexpected behavior, see [Errand Run Rules](#).
- Configuring S3 service backups results in an error due to a non-configurable property.

## Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	315.x
PCF	2.3.x, 2.4.x, and 2.5.x
cf-redis-release	434.3.12
on-demand-service-broker	0.31.1
routing	0.188.0
service-metrics	1.12.0
service-backup	18.3.1
syslog	11.4.0
loggregator-agent	3.18.0
bpm	1.0.4
Redis OSS	5.0.5

## v2.0.2

**Release Date: May 7, 2019**

## Feature

New feature in this release:

- A new errand is available to the operator: `recreate-all-service-instances`. See [Post-Deploy Errands](#).

## Security Fixes

This release includes the following security fixes:

- Redis for PCF uses open source Redis v5.0.4, which includes fixes for issues that caused crashes in very rare cases.  
For more information about Redis v5.0.4, see the [Redis 5.0 release notes](#).
- Updates goolang to v1.12.1, which includes a fix for a regression around sync
- Updates on-demand broker to v0.26.1, which contains a fix for the `recreate-all-service-instances` errand

## Resolved Issues

This release fixes the following issues:

- For PCF v2.4 and later, no metrics emitted from cf-redis-broker reached the metrics Firehose.
- Service Backups have been upgraded to v18.2, which contains a fix for an issue where up-to-date versions of `scp` no longer support path `.`
- The fields for AWS Secret Access Key pair required to set up S3 Backups were marked as text and not secret.  
These have now been updated so your S3 keys are masked. Pivotal recommends that you rotate your S3 credentials to ensure they are safe.

## Known Issues

This release has the following 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 option. For more information about this unexpected behavior, see [Errand Run Rules](#).
- Configuring S3 service backups results in an error due to a non-configurable property.

## Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	250.x
PCF	2.3.x, 2.4.x, and 2.5.x

cf-redis-release	434.2.7
on-demand-service-broker	0.26.1
routing	0.188.0
service-metrics	1.10.0
service-backup	18.2.0
syslog	11.4.0
loggregator-agent	3.9
bpm	1.0.3
Redis OSS	5.0.4

## v2.0.1

**Release Date: February 12, 2019**

### Security Fixes

This release includes the following security fixes:

- Bumped Go version used to v1.10.8 for <https://github.com/golang/go/issues/29903>

### Known Issues

This release has the following issues:

- For PCF 2.4 and later, metrics emitted from cf-redis-broker do not reach the Loggregator Firehose.
- 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 option. For more information about this unexpected behavior, see [Errand Run Rules](#).
- Service backups fail to upload backups when running on stemcell v170.30 or later.
- Configuring S3 service backups results in an error due to a non-configurable property.

### Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	170
PCF	2.3, 2.4, and 2.5
cf-redis-release	434.0.28
on-demand-service-broker	0.25.0
routing	0.185.0

service-metrics	1.9.0
service-backup	18.1.16
syslog	11.4.0
loggregator-agent	3.3
bpm	1.0.2
Redis OSS	5.0.2

## v2.0.0

**Release Date: January 4, 2019**

## Features

New features and changes in this release:

- Dedicated-VM support is removed, see the Breaking Change above. To install or upgrade the tile, operators must acknowledge the deletion of dedicated-VMs using a checkbox.
- The Redis on-demand service broker now binds apps to a service instance using BOSH DNS. Therefore, service bindings return the DNS address instead of the IP address. To use BOSH DNS with existing bindings you must rebind your apps to on-demand service instances.
- Redis for PCF uses open source Redis v5.0.2, which includes a fix for a critical issue for users that rely on streams. For more information about Redis v5.0.2, see the [Redis 5.0 release notes](#).
- On-Demand Redis supports secure manifests, which avoids plaintext secrets in manifests by passing these to the ODB to store in BOSH CredHub.
- Added error message and logging when a shared service instance is missing a PID.

## Known Issues

This release has the following issues:

- For PCF 2.4 and later, metrics emitted from cf-redis-broker do not reach the Loggregator Firehose.
- 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](#).
- Service backups fail to upload backups when running on stemcell v170.30 and later.
- Configuring S3 service backups results in an error due to a non-configurable property.

## Compatibility

The following components are compatible with this release:

Component	Version
Stemcell	170
PCF	2.3 and 2.4
cf-redis-release	434.0.27
on-demand-service-broker	0.25.0
routing	0.184.0
service-metrics	1.9.0
service-backup	18.1.16
syslog	11.4.0
loggregator-agent	3.2
bpm	1.0.0
Redis OSS	5.0.2

## View Release Notes for Another Version

To view the release notes for another product version, select the version from the dropdown at the top of this page.

[Create a pull request or raise an issue on the source for this page in GitHub](#)

# Planning Guide

## Is Redis for PCF right for your enterprise?



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

This topic provides recommended use cases for Redis for Pivotal Cloud Foundry (PCF) and information for determining the product's fit for your enterprise's use case.

## Recommended Use Cases

On-Demand plans are configured by default for cache use cases but can also be used as a datastore. Dedicated-VM and Shared-VM plans are designed for datastore use cases.



**Note:** The Shared-VM service should only be used for development and testing. Do not use for production.



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

Redis can be used in many different ways, including:

- Key/value store: For strings and more complex data structures including Hashes, Lists, Sets, and Sorted Sets
- Session cache: Persistence enabled preservation of state
- Full page cache: Persistence enabled preservation of state
- Database cache: Middle-tier database caching to speed up common queries
- Data ingestion: Because Redis is in memory, it can ingest data very quickly
- Message queues: List and set operations. `PUSH`, `POP`, and blocking queue commands.

- Leaderboards and counting: Increments and decrements sets and sorted sets using [ZRANGE](#), [ZADD](#), [ZREVRANGE](#), [ZRANK](#), [INCRBY](#), and [GETSET](#)
- Pub/Sub: Built in publish and subscribe operations: [PUBLISH](#), [SUBSCRIBE](#), and [UNSUBSCRIBE](#)

## SLO Benchmark

The Redis for PCF team maintains a monthly Service Level Objective (SLO) of 99.95% uptime for the Redis for PCF offering on Pivotal Web Services. This is provided as a benchmark. SLOs for separate offerings of the Redis for PCF service vary based on variables such as infrastructure, networking, and relevant policies around security upgrades.

## Service Offerings

For descriptions of the Redis for PCF service offerings, see:

- [On-Demand Service Offering](#)
- [Shared-VM Service Offering](#)



**Note:**

- The Shared-VM plan is not recommended for production use.
- The Dedicated-VM plan is removed in v2.0.
- If you have any dedicated-VM instances you must migrate to the on-demand service plan before upgrading to Redis for PCF v2.0. For information about migrating to on-demand service plans, see [Migrating from dedicated-vm service plans to on-demand service plans](#) in the Pivotal Support knowledge base.

## Enterprise-Readiness Checklist

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

Resilience		More Information
Availability	<p>All Redis for PCF services are single nodes without clustering capabilities. This means that planned maintenance jobs (e.g., upgrades) can result in 2 – 10 minutes of downtime, depending on the nature of the upgrade. Unplanned downtime (e.g., VM failure) also affects the Redis service.</p> <p>Redis for PCF has been used successfully in enterprise-ready apps that can tolerate downtime. Pre-existing data is not lost during downtime with the default persistence configuration. Successful apps include those where the downtime is passively handled or where the app handles failover logic.</p>	<p><a href="#">Recommended Use Cases</a></p> <p><a href="#">Support for Multiple AZs</a></p>

Failure Recovery	VM failures and process failures are handled automatically by BOSH and Redis for PCF. Manual backup and restore instructions are available for on-demand and shared-VM Redis services. Automatic backup capabilities are enabled for on-demand and shared-VM Redis services.	<a href="#">Manually Backing Up and Restoring Redis for Pivotal Cloud Foundry</a>  <a href="#">Configuring Automated Service Backups</a>
Isolation	Isolation is provided when using the On-Demand service. Individual apps and workflows should have their own Redis for PCF instance to maximize isolation.	
<b>Day 2 Operations</b>		<b>More Information</b>
Resource Planning	Operators can configure the number of VMs and the size of those VMs. For the On-Demand service, the operator does this by creating plans with specific VM sizes and quotas for each plan. For the Shared-VM service, the number and size of VMs are pre-provisioned by the operator. BOSH errands used for registration, upgrade and cleanup use short-lived VMs that cannot be configured but can be turned on or off.	<a href="#">On-Demand Resource Planning</a>  <a href="#">Shared-VM Plan</a>
Health Monitoring	The On-Demand service VMs emit metrics. These include Redis-specific metrics and Redis for PCF metrics. Guidance on critical metrics and alerting levels is captured with the Redis for PCF Key Performance Indicators (KPIs).	<a href="#">Key Performance Indicators</a>
Scalability	For the On-Demand Service, the operator can configure three plans with different resource sizes. The operator can also scale up the VM size associated with the plan. Additionally, the operator can increase the quota, which caps the number of instances allowed for each On-Demand plan. To prevent data loss, only scaling up is supported. For the Shared-VM Service, the operators can change the Redis instance memory limit as well as change the instance limit. To prevent data loss, only scaling up is supported.	<a href="#">Scaling the On-Demand Service</a>
Logging	All Redis services emit logs. Operators can configure syslog forwarding to a remote destination. This enables viewing logs from every VM in the Redis for PCF deployment in one place, effective troubleshooting when logs are lost on the source VM, and setting up alerts for important error logs to monitor the deployment.	<a href="#">Configuring syslog forwarding</a>
Customization	The On-Demand service can be configured to best fit the needs of a specific app. The Shared-VM service cannot be customized.	<a href="#">Configuring the On-Demand service</a>
Upgrades	For information about preparing an upgrade and about understanding the effects on your Redis for PCF and other services, see <a href="#">Upgrading Redis for PCF</a> . Redis for PCF upgrades run a post deployment BOSH errand called smoke tests to validate the success of the upgrade.	<a href="#">Upgrades</a>  <a href="#">Smoke Tests</a>
<b>Encryption</b>		<b>More Information</b>
Encrypted Communication in Transit	Redis for PCF has been tested with the IPsec Add-on for PCF. Beyond that Redis for PCF does not provide additional encryption on top of Redis.	<a href="#">Securing Data in Transit with the IPsec add-on</a>  <a href="#">OS Redis Security</a>



## Support for Multiple AZs

Redis for PCF supports configuring multiple availability zones (AZs). However, assigning multiple AZs to Redis instances does not guarantee high availability as clustered Redis is not supported. Redis instances operate as single nodes.

- On-Demand plans can be configured to deploy instances to any AZ.
- Shared-VM instances run on a single node in the AZ in which the tile is deployed.

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Service Offerings

### On-Demand Service Offering



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

#### Page last updated:

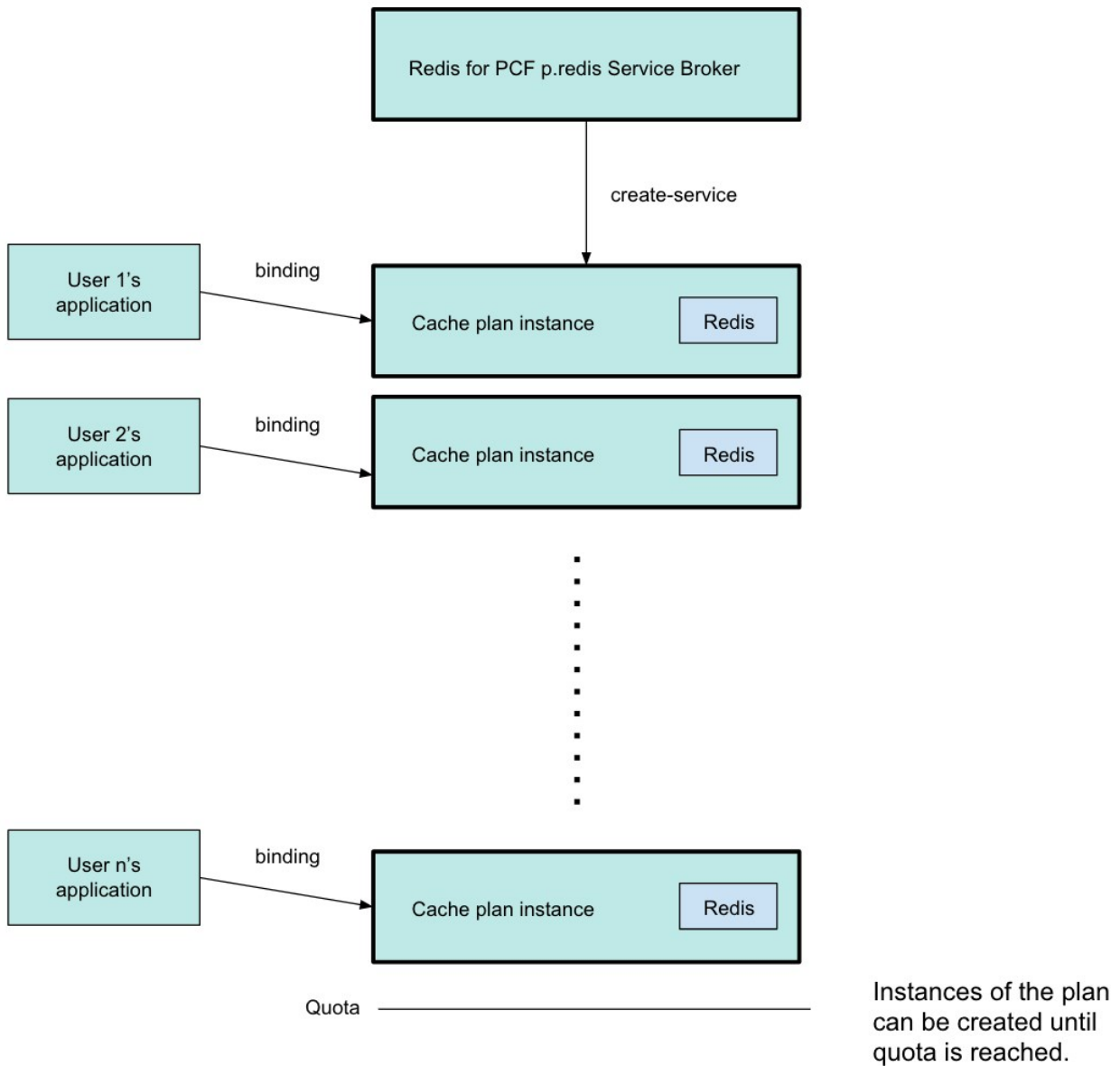
Redis for PCF offers On-Demand and Shared-VM service plans. This section describes the architecture, lifecycle, and configurations of the on-demand plan, as well as networking information for the on-demand service. For similar information for the Shared-VM plans, see [Shared-VM Service Offering](#).

## 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 Marketplace and its intended use case.

- **Small Cache Plan:** A Redis instance deployed to a dedicated VM, suggested to be configured with about 1 GB of memory and more than 2.5 GB of persistent disk.
- **Medium Cache Plan:** A Redis instance deployed to a dedicated VM, suggested to be configured with about 2 GB of memory and more than 5 GB of persistent disk.
- **Large Cache:** A Redis instance deployed to a dedicated VM, suggested to be configured with about 4 GB of memory and more than 10 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. Set the persistent disk size to least 2.5 times the memory of the instance.

## 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.
- On-Demand Redis supports Redis Database Backup (RDB) snapshots, but not Append-Only File (AOF) persistence. For more information, see [Redis Persistence](#) in the Redis documentation.
- The maximum number of instances is managed by a per-plan and global quota. For information 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 [On-Demand Plan Settings](#).

## App Developer Configurable Redis Settings

The Redis settings that an app developer can configure include:

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

For more information, see [Customize an On-Demand Service Instance](#).

## Operator Notes for On-Demand Service Plans

- Instances of the on-demand plan can be deployed until their number reaches either an operator-set per-plan quota or a global quota. For information 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.

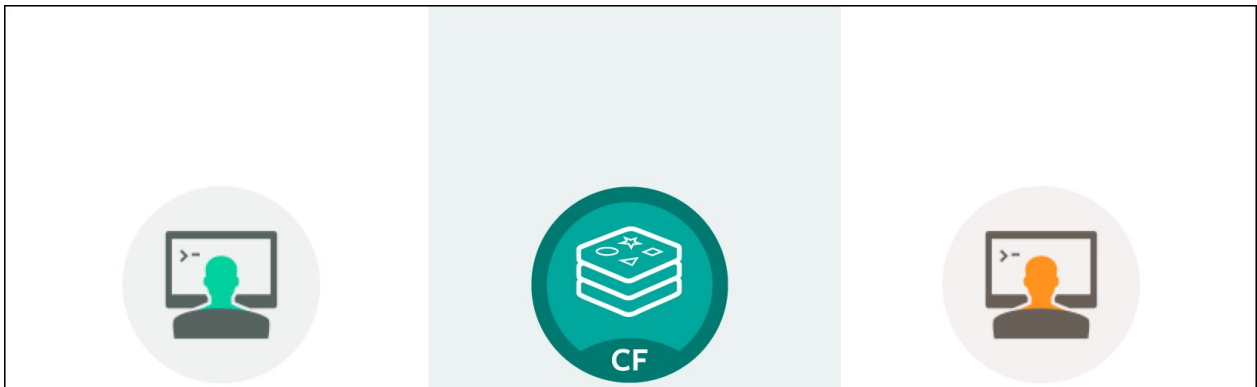
## Known Limitations for On-Demand Service Plans

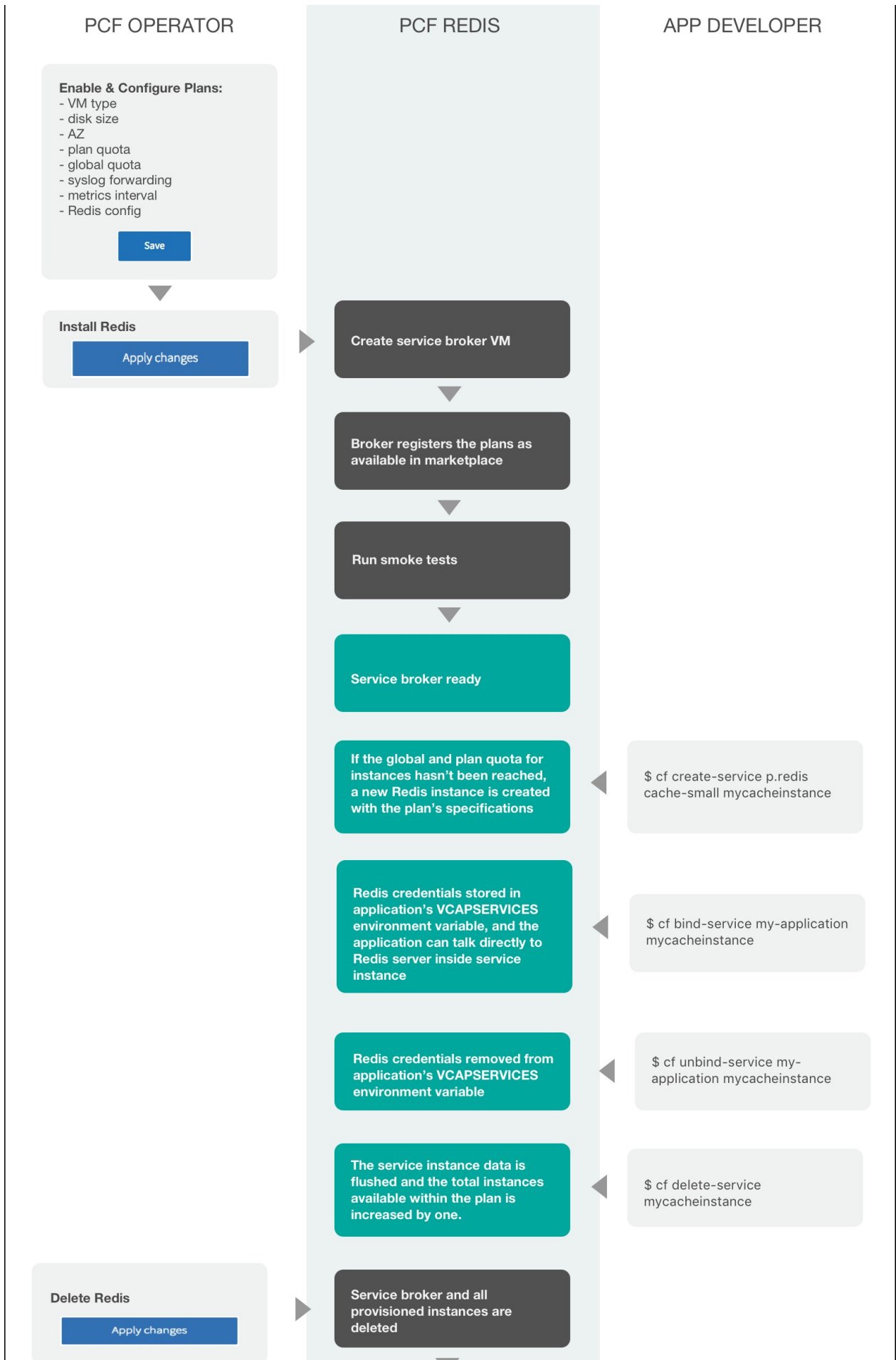
Limitations for the On-Demand Service include:

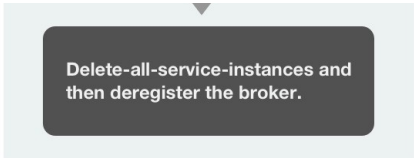
- Operators must not downsize the VMs or disk size as this can cause data loss in pre-existing instances.
- Operators can update certain plan settings after the plans have been created. To ensure upgrades happen across all instances, set the **upgrade instances** errand to **On**.
- If the operator updates 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.

## Lifecycle for On-Demand Service Plan

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.







Delete-all-service-instances and then deregister the broker.

Create a pull request or raise an issue on the source for this page in [GitHub](#)

## Shared-VM Service Offering



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

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



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

## About the Shared-VM Plan

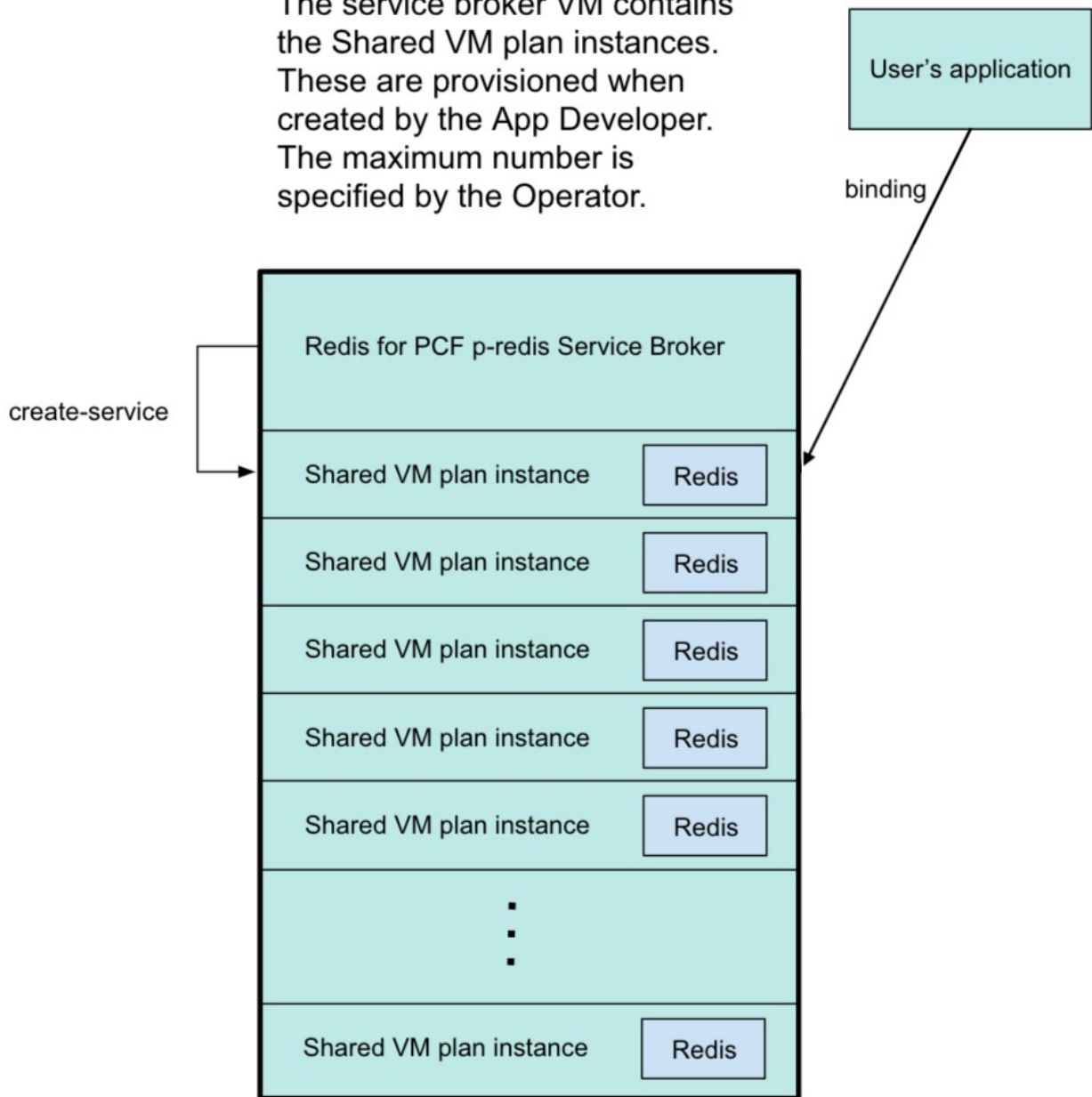
The shared-VM plan is a pre-provisioned service plan for development and testing purposes only. An instance of this plan provisions a single Redis process on a single shared VM. This plan is suitable for workloads that do not require dedicated hardware. This plan is **not** suitable for production purposes.

## Architecture Diagram for Shared Plans

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

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

The service broker VM contains the Shared VM plan instances. These are provisioned when created by the App Developer. The maximum number is specified by the Operator.



Each shared instance has its own Redis server, with credentials stored in the VCAP\_SERVICES environment variable.

## Settings for Shared-VM Service Plans

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

The default Redis settings are as follows:

## Memory Policy

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

## Persistence

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

## Number of Connections

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

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

```
$ CONFIG SET maxclients 9000
```

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

## Replication and Event Notification

- Replication and event notification are not configured.


## 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 `0`.
- You can increase the maximum number of service instances that can run on a shared VM from five, which is the default, up to 250. There is a hard maximum of 250 shared instances.
- If you increase the number of instances that can be run on a VM, consider increasing the resources allocated to the VM, in particular RAM and CPU. Failure to do so might lead to a degradation of performance.
- You can also increase the maximum amount of RAM allocated to each service instance that is running on this VM.
- If you decrease the service instance limit, any instances that are now running beyond the limit are not automatically terminated. You cannot create any new instances until the total falls below the new limit.



For example, if you use 10 service instances, and you then reduce the limit to 8, the two instances outside the limit continue to run until you terminate them.

- The number of shared-VM instances available to developers is set by the operator. The maximum number of shared-VM instances is relative to the memory allocated to each shared-VM instance and the total memory of the Redis service broker. For more information, see [Shared-VM Plan Settings](#).
- You can enable or disable Lua scripting. Changes to this configuration apply to all existing shared-VM instances. Pivotal recommends that Lua scripting is disabled unless developers need it to be enabled. This is because using it can affect the performance of other service instances on the VM.



**Warning:** The Steeltoe connector for Redis requires Redis for PCF to support Lua scripting. Check if any of your apps require Lua scripting. By default, Lua scripting is disabled for Redis for PCF, but a PCF operator can change the setting to enable it by selecting the **Lua Scripting** checkbox in the **Shared-VM Plan** configuration pane.

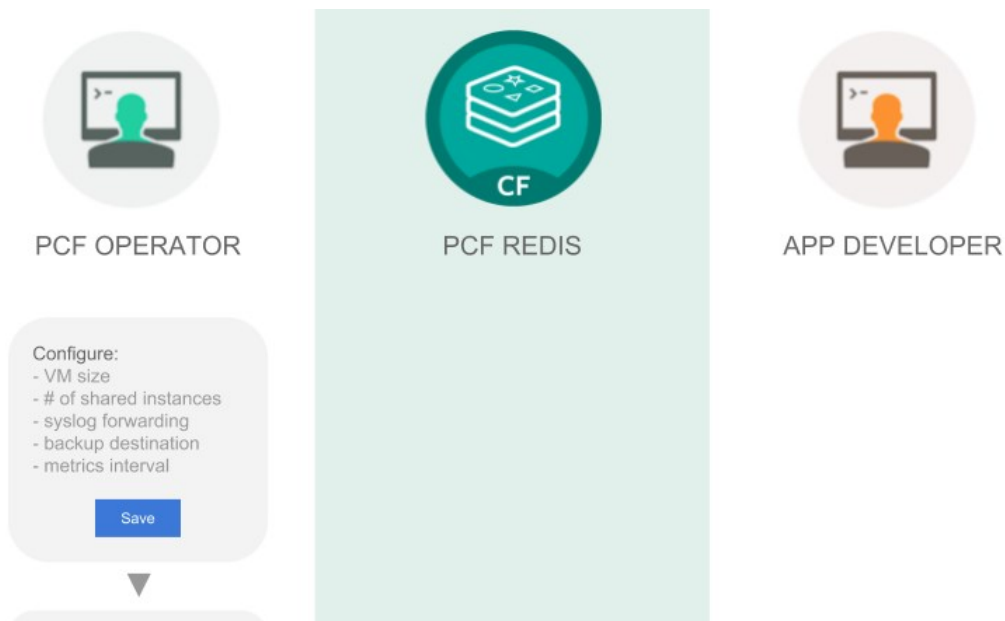
### Known Limitations of the Shared-VM Plan

The shared-vm plan cannot:

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

### Lifecycle for Shared-VM Service Plan

Below is the lifecycle of Redis for PCF, from an operator installing the tile, to an app developer using the service, to an operator deleting the tile.





Create a pull request or raise an issue on the source for this page in GitHub

## Networking for On-Demand Services



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

#### Page last updated:

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

## Service Network Requirement

When you deploy Pivotal Cloud Foundry, you must create a statically defined network to host the component virtual machines that constitute the Pivotal Cloud Foundry infrastructure.

Pivotal Cloud Foundry components, like the Cloud Controller and UAA, run on this infrastructure network. On-demand Pivotal Cloud Foundry services may require that you host them on a network that runs separately from the Pivotal Cloud Foundry default network. You can also deploy tiles on separate service networks to meet your own security requirement.

Pivotal Cloud Foundry v2.1 and later include dynamic networking. Operators can use this dynamic networking with asynchronous service provisioning to define dynamically-provisioned service networks. For more information, see [Default Network and Service Network](#).

In Pivotal Cloud Foundry v2.1 and later, on-demand services are enabled by default on all networks. Operators can create separate networks to host services in BOSH Director, but doing so is optional. Operators select which network hosts on-demand service instances when they configure the tile for that service.

## Default Network and Service Network

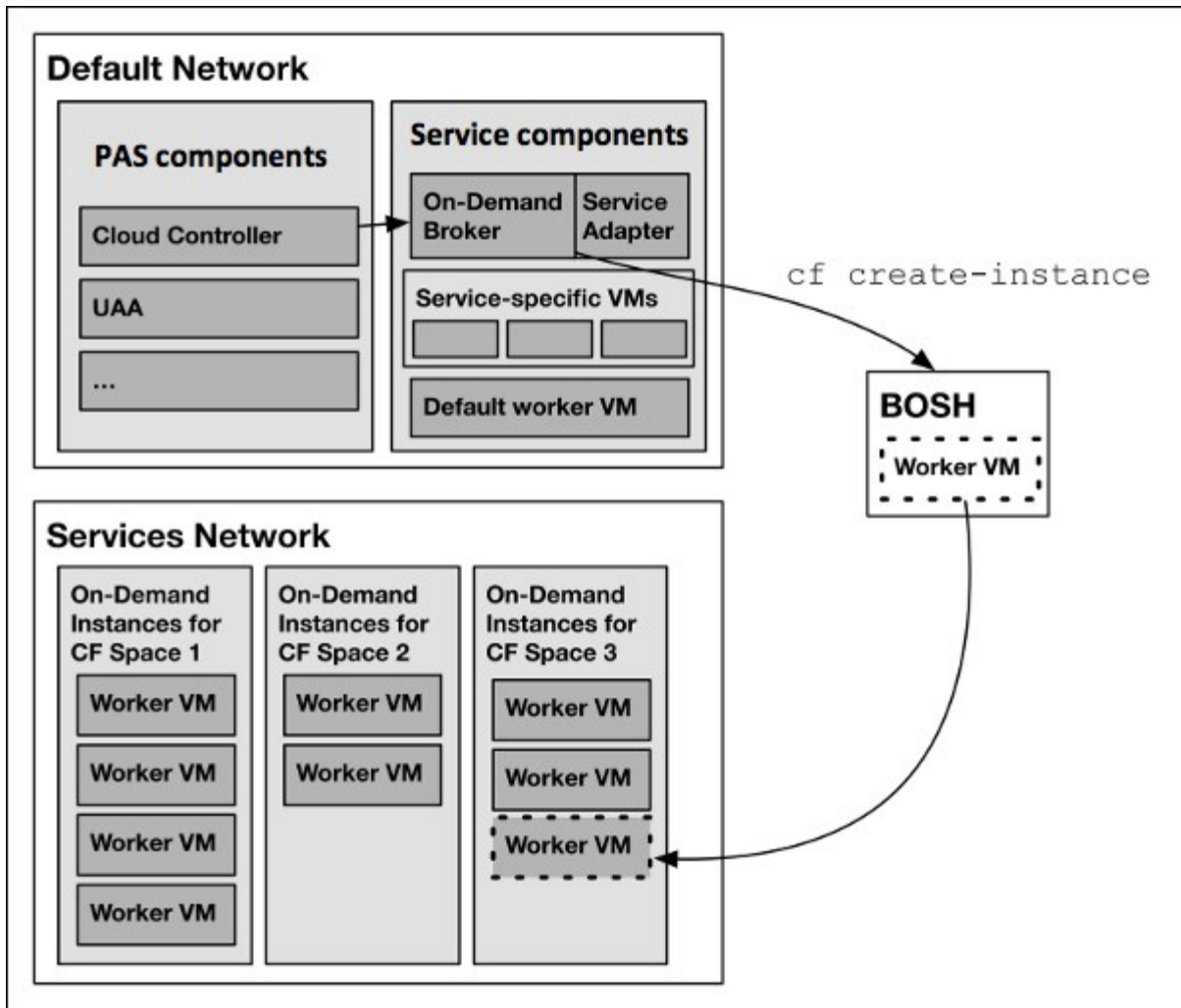
On-demand PCF services rely on the BOSH 2.0 ability to dynamically deploy VMs in a dedicated network. The on-demand service broker uses this capability to create single-tenant service instances in a dedicated service network.

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 app 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 running on a separate services network, while other components run on the default network.



## Required Networking Rules for On-Demand Services

Before deploying a service tile that uses the on-demand service broker (ODB), request the needed network connections to allow components of Pivotal Cloud Foundry to communicate with ODB.

The specifics of how to open those connections varies for each IaaS.

See the following table for key components and their responsibilities in an on-demand architecture.

Key Components	Their Responsibilities
BOSH Director	Creates and updates service instances as instructed by ODB.
BOSH Agent	Includes an agent on every VM that it deploys. The agent listens for instructions from the BOSH Director and carries out those instructions. The agent receives job specifications from the BOSH Director and uses them to assign a role, or job, to the VM.
BOSH UAA	Issues OAuth2 tokens for clients to use when they act on behalf of BOSH users.

<b>Pivotal Application Service</b>	Contains the apps that are consuming services
<b>ODB</b>	Instructs BOSH to create and update services, and connects to services to create bindings.
<b>Deployed service instance</b>	Runs the given data service. For example, the deployed Redis for Pivotal Platform service instance runs the Redis for Pivotal Platform 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.

Source Component	Destination Component	Default TCP Port	Notes
ODB	BOSH Director  BOSH UAA	25555 8443 8844	The default ports are not configurable.
ODB	PAS	8443	The default port is not configurable.
Errand VMs	PAS  ODB  Deployed service instances	8443 8080 6379 12345	The default ports are not configurable.
BOSH Agent	BOSH Director	4222	The BOSH Agent runs on every VM in the system, including the BOSH Director VM. The BOSH Agent initiates the connection with the BOSH Director. The default port is not configurable.  The communication between these components is two-way.
Deployed apps on PAS	Deployed service instances	6379	This is the default port where Redis is deployed.
PAS	ODB	12345	The default port is not configurable.

For a complete list of ports and ranges used in Redis for PCF, see [Network Configuration](#).

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Redis for PCF Security



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

**Page last updated:**

## Security

Pivotal recommends the following best practices for security:

- (Required) To allow this service to have network access you must create Application Security Groups. For more information, see [Networks, Security, and Assigning AZs](#).
- Run Redis for PCF in its own network. For more information about creating service networks, see [Creating Networks in Ops Manager](#).
- You can use Redis for PCF 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](#).
- Do not use a single Redis for PCF instance for multi-tenancy. A single Redis instance of the On-Demand service should only support a single workload.
- The Shared-VM service is designed for multi-tenancy, but you should not use it for production use cases because it is not considered adequately secure for that purpose.

[Create a pull request or raise an issue on the source for this page in GitHub](#)

# Operator Guide

## Introduction for Operators



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

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

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

## Redis Key Count and Memory Size

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

## Errands

Redis for PCF includes the errands listed below.

### Post-Deploy Errands

The following post-deploy errands are run by default when **Apply Changes** is triggered, whether or not there has been a configuration change in the Redis for PCF tile itself:

- **Broker Registrar**—Registers the cf-redis-broker with PCF to offer the `p-redis` service, that is, the shared-VM plan. The Broker Registrar errand also includes the Deprecate Dedicated-VM Plan errand. It disables service access to the dedicated-VM plan for non-admin app developers.
- **Deprecate Dedicated-VM Plan**—Disables service access to the dedicated-VM plan for non-admin app developers.
- **Cleanup Metadata if Dedicated-VM Plan Disabled**—Cleans up any metadata in PCF for any dedicated-VM service instances if zero dedicated instances are provisioned. You can run this errand to clean up metadata after you have deleted all dedicated instances.
- **Smoke Tests**—Runs lifecycle tests for shared-VM plans if these have been enabled and there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding, and deprovisioning of service instances.
- **Register On-Demand Broker**—Registers the on-demand Redis broker with PCF to offer the `p.redis` service (on-demand plans).
- **On-Demand Smoke Tests**—Runs lifecycle tests for enabled plans of the `p.redis` service if there is remaining quota available. The tests cover provisioning, binding, reading, writing, unbinding and deprovisioning of service instances.
- **Upgrade All On-Demand Service Instances**—Upgrades on-demand service instances to use the latest plan configuration, service releases, and stemcell. This causes downtime to any service instances with available upgrades.

The following post-deploy errand does not run by default when **Apply Changes** is triggered. This type of post-deploy errand helps operators troubleshoot and maintain their service fleet:

- **Recreate All On-Demand Service Instances**—Re-creates on-demand service instances one-by-one. This causes downtime to any service instances with available upgrades.

### Pre-Delete Errands

- **Broker Deregistrar**—Deregisters the `cf-redis-broker`.
- **Delete All On-Demand Service Instances and Deregister Broker**—Deletes all on-demand instances and deregisters the on-demand Redis broker.

The above pre-delete errands are run by default whenever the Redis for PCF tile is deleted.



## Turning off Post-Deploy Errands

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

### Changes to Redis for PCF Tile Configuration

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

### Installing Another Tile

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

### Changes to Other Tiles

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

### Broker Registrar Errand

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

### Deprecate Dedicated-VM Plan Errand

- Run this errand to prepare for deprecation of support for dedicated nodes.
- Only needs to be run once to disable service access to the dedicated-VM plan for non-admin app developers.
- Not needed if the Broker Registrar errand has been run.

### Register On-Demand Broker Errand

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



Pivotal **recommends** against changing the BOSH Director's network configuration in a way that changes the ranges where the Redis for PCF tile deploys VMs.

### Smoke Tests and On-Demand Smoke Tests Errands

- Required to run if their respective register broker errand is required.

- Required to run both if a newer stemcell minor version is uploaded. The Redis for PCF tile floats to the newest minor version. For more information, see [Floating Stemcells](#).
- Good practice to run both for any change in the BOSH Director or PAS tile.
- Not necessary to run either if the change only involves other tiles except PAS and BOSH Director.

### Upgrade All On-Demand Service Instances Errand

- Required to run if a newer stemcell minor version is uploaded. The Redis for PCF tile floats to the newest minor version. For more information, see [Floating Stemcells](#).
- Not necessary to run if there are no on-demand instances provisioned.

### Recreate All On-Demand Service Instances Errand

- Run this errand when it is necessary to re-create an instance with different resources, for example, when rotating CA certificates.
- It increases the time to **Apply Changes** because it follows the typical instance lifecycle.
- Do not run this errand if there are no on-demand instances provisioned. Pivotal recommends that you keep this errand off unless it is needed.

## Smoke Tests

Ops Manager runs Redis for PCF smoke tests as a post-install errand. To run the smoke tests errand manually:

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

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

For more information, see [Redis for PCF Smoke Tests](#).



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

Create a pull request or raise an issue on the source for this page in [GitHub](#)

## Installing Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported

version.

### Page last updated:

This topic describes the process of installing Redis for PCF. It covers tasks from downloading the file from the Pivotal Network through verifying the installation after configuration.

## Role-Based Access in Ops Manager

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

For more information about roles in Ops Manager, see [Understand Roles in Ops Manager](#).

## Download and Install the Tile

To add Redis for Pivotal Cloud Foundry (PCF) to Ops Manager, follow the procedure for adding PCF Ops Manager tiles:

1. Download the Redis for PCF file from [Pivotal Network](#). Select the latest release from the **Releases** dropdown.
2. In the PCF Ops Manager Installation Dashboard, click **Import a Product** to upload the Redis for PCF file.
3. Click the + sign next to the uploaded product description to add the tile to your staging area.
4. To configure Redis for PCF, click the newly added tile. See configuration instructions in the sections below.
5. After completing the required configuration, in the Ops Manager Dashboard, do the following to complete the installation:
  1. If you are using Ops Manager v2.3 or later, click **Review Pending Changes**. For more information about this Ops Manager page, see [Reviewing Pending Product Changes](#).
  2. Click **Apply Changes**.

For guidance on ports and ranges used in the Redis service, see [Select Networks](#) below.

## Assign AZs and Networks

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

The screenshot shows the 'Settings' tab for Redis for PCF. The 'Assign AZs and Networks' tab is selected. On the left, there is a list of settings with checkmarks: Assign AZs and Networks, WARNING - DEDICATED INSTANCES, Shared-VM Plan, On-Demand Service Settings, On-Demand Plan 1, On-Demand Plan 2, On-Demand Plan 3, Metrics, and Backups. On the right, the 'AZ and Network Assignments' section is expanded. It contains two sections: 'Place singleton jobs in' and 'Balance other jobs in'. Under 'Place singleton jobs in', there are three radio buttons: 'us-central1-b' (selected), 'us-central1-f', and 'us-central1-c'. Under 'Balance other jobs in', there are three checked checkboxes: 'us-central1-b', 'us-central1-f', and 'us-central1-c'. Below these are two dropdown menus: 'Network' (set to 'pas-subnet') and 'Service Network' (set to 'services-subnet'). A blue 'Save' button is at the bottom right.

## Assign AZs

As of Redis for PCF v1.9, you can assign multiple AZs to Redis jobs, however this does not ensure high availability. For more information, see [Support for Multiple AZs](#).

To assign AZs, do the following:

1. In the **Assign AZs and Networks** tab, make your selections under **Place singleton jobs in** and **Balance other jobs in**.
2. Click **Save**.

## Select Networks

You can use Redis for PCF with or without using the on-demand service. To use the Redis for PCF on-demand service, you must select a network in which the service instances are created. For more information, see [Networking for On-Demand Services](#).



**Note:** In Ops Manager v2.0 and earlier, a specific network was designated as the Service Network to reserve IPs for the on-demand service. As of Ops Manager v2.1, IPs are no longer managed in this way. All networks are now available to use as a

## Service Network.

To select networks, do the following:

1. In the **Assign AZs and Networks** tab, select a **Network**.

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

2. If using the on-demand service, select a **Service Network**. Otherwise, select an empty service network. For more information, see [Creating an Empty Services Network when using on-demand Service Tiles for Non-On-Demand Usage Only](#) in the Pivotal Support knowledge base.

## Port Ranges Used in Redis for PCF

The following ports and ranges are used in Redis for PCF:

Port	Protocol	Direction and Network	Reason
830 0 8301	TCP TCP and UDP	Inbound to Cloud Foundry network, outbound from service broker and service instance networks*	Used for metrics
8202	TCP	Inbound to Cloud Foundry network, outbound from service broker and service instance networks*	Used by the Redis metron_agent to forward metrics to the Cloud Foundry Loggregator
1235 0	TCP	Outbound from Cloud Foundry 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.
1234 5	TCP	Outbound from Cloud Foundry 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 Cloud Foundry 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)
3276 8- 6100 0	TCP	Outbound from Cloud Foundry 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 (Typi cally )	http or https respectiv ely	Outbound from any service instance networks	Access to the backup blobstore
8443 2555 5	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 the BOSH Director

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

## Configure Redis for PCF Service Plans

Click the Redis for PCF tile in the Ops Manager Installation Dashboard to display the configuration page and allocate resources to Redis service plans.

### On-Demand Service Settings

1. 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 on-demand plans combined cannot exceed this number.

The screenshot shows the Redis configuration interface. On the left is a navigation menu with options: Assign AZs and Networks, WARNING - DEDICATED INSTANCES, Shared-VM Plan, On-Demand Service Settings (highlighted), On-Demand Plan 1, On-Demand Plan 2, On-Demand Plan 3, Metrics, Backups, Syslog, Errands, and Resource Config. The main content area is titled 'On-Demand Service Settings' and contains the following fields:

- Maximum service instances across all on-demand plans (min: 0) \***: A text input field containing the value '4'. A tooltip indicates 'Configure the maximum number of service instances'.
- VM options**: A section with a checked checkbox for 'Allow outbound internet access from service instances (IaaS-dependent)' and an unchecked checkbox for 'Service Instance Sharing'.
- Maximum Parallel Upgrades \***: A text input field containing the value '2'.
- Number of Canaries to run before proceeding with upgrade \***: A text input field containing the value '3'.
- Specify Org and Space that Canaries will be selected from? \***: Radio buttons for 'No' (unselected) and 'Yes' (selected).
- Canary Org \***: A text input field containing the value 'system'.
- Canary Space \***: A text input field containing the value 'pivotal-services'.

A blue 'Save' button is located at the bottom of the configuration area.

For more information, see [Setting Limits for On-Demand Service Instances](#).

2. Select the **Allow outbound internet access from service instances** checkbox. You must select this checkbox to allow external log forwarding, send backup artifacts to external

destinations, and communicate with an external BOSH blob store.



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

3. (Optional) Select the checkbox to enable **Service Instance Sharing**. Turning on sharing enables this feature for all on-demand instances.



**Note:** To enable this feature a user with admin privileges must run `cf enable-feature-flag service_instance_sharing`. For information about this feature, see [Sharing a Redis Instance with Another Space](#).

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

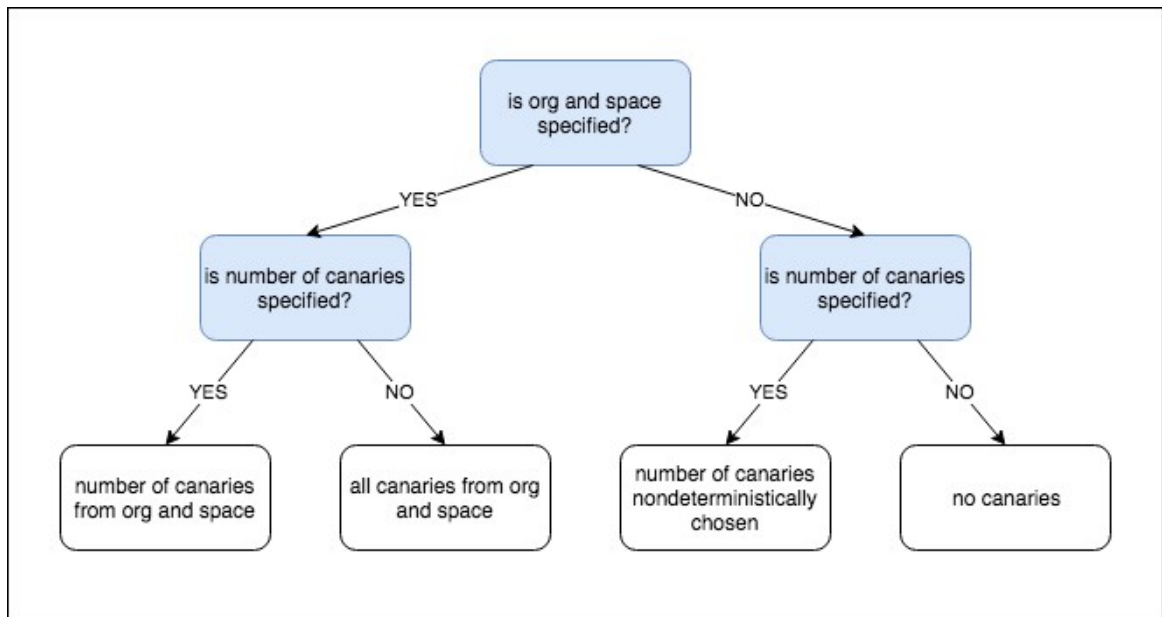
When you click **Apply Changes**, the on-demand broker upgrades all service instances. By default, each instance is upgraded serially. Allowing parallel upgrades reduces the time taken to apply changes.



**Note:** Multiple Redis service instances will be concurrently unavailable during the upgrade.

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

Canaries can be limited by number and by org and space. If you want to use all service instances in an org and space as canaries, set the number of canaries zero. This upgrades all service instances in the selected org and space first.



**Note:** If you specify that canaries should be limited to an org and space that has no service instances, the upgrade fails.



**Note:** Canary upgrades comply with the Maximum Parallel Upgrades settings. If you specify three canaries and a Maximum Parallel Upgrades of two, then two canaries upgrade, followed by the third.

For information about this feature, see [canaries](#) in [Upgrade All Service Instances](#) in the On-Demand Services SDK documentation.

- (Optional) Select the checkbox to enable **BOSH HotSwaps**. This reduces downtime during upgrades. For how this feature works, see [Changing VM Update Strategy](#).

## On-Demand Plan Settings

You can configure up to three on-demand plans with memory and disk sizes suited to different use cases. Resource configuration options may vary on different IaaSes.

For each on-demand service plan, configure it as follows:


- Click **On-Demand Plan 1**, **2**, or **3** to open the configuration pane for the plan.
- Select **Plan Active** to make the plan available.
- (Optional) For **Plan name**, assign a name to the plan. The default names for the three on-demand plans convey that they allocate different cache sizes:
  - ❖ **cache-small:** A Redis instance deployed to a dedicated VM, suggested to be configured with 1 GB of memory and more than 2.5 GB of persistent disk
  - ❖ **cache-medium:** A Redis instance deployed to a dedicated VM, suggested to be configured with 2 GB of memory and more than 5 GB of persistent disk
  - ❖ **cache-large:** A Redis instance deployed to a dedicated VM, suggested to be configured with 4 GB of memory and more than 10 GB of persistent disk



**On-Demand Plan 1 Configuration**

- Plan\*
  - Plan Inactive
  - Plan Active
- Plan name\*  The plan name.
- Plan description\*
- Plan Quota (min: 1)\*
- CF Service Access\*
- AZ to deploy Redis instances of this plan\*
  - us-central1-b
  - us-central1-f
  - us-central1-c
- Server VM type\*
- Server Disk type\*
- Redis Client Timeout (min: 0)\*
- Redis TCP Keepalive (min: 0)\*
- Max Clients (min: 1, max: 10000)\*
- Lua Scripting

- Configure the following settings for your on-demand plan(s) and then click **Save**. Any pre-populated default settings are pre-configured according to the memory and disk size of each plan.

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

Field	Default	Description
-------	---------	-------------

<b>Plan</b>	Active	Select <b>Plan Active</b> or <b>Plan Inactive</b> . An inactive plan does not need any further configuration.
<b>Plan Name</b>	cache-small cache-medium cache-large	This is a name displayed in the Marketplace.
<b>Plan Description</b>	This plan provides a small, medium, or large on-demand Redis instance, tailored for caching use-cases with persistence to disk enabled.	This is a description displayed in the Marketplace. Include details that are relevant to app developers.
<b>Plan Quota</b>	20	This is the maximum number of instances of this plan that app developers can create. For more information, see <a href="#">Setting Limits for On-Demand Service Instances</a> .
<b>CF Service Access</b>	Enabled for all orgs and spaces	This setting does not modify the permissions that have been previously set, and allows for manual access to be configured from the CLI.
<b>AZ to deploy Redis instances of this plan</b>	None selected	These are the AZs in which to deploy the Redis instances from the plan. These must be AZs of the service network, which are configured in the BOSH Director tile. If you select multiple AZs, instances are distributed randomly between them.
<b>Server VM type</b>	Varies depending on IaaS	Pivotal recommends that the persistent disk is at least 2.5x the VM memory for On-Demand Service Instances.
<b>Server Disk type</b>	Varies depending on IaaS	Pivotal recommends that the persistent disk is at least 2.5x the VM memory for On-Demand Service Instances.
<b>Redis Client Timeout</b>	3600	This is the server timeout for an idle client specified in seconds. Adjust this setting as needed.
<b>Redis TCP Keepalive</b>	60	Redis TCP Keepalive refers to the interval in seconds at which TCP ACKs are sent to clients. Adjust this setting as needed.
<b>Max Clients</b>	Small: 1000 Medium: 5000 Large: 10000	Max Clients refers to the maximum number of clients that can be connected at any one time. Adjust this setting as needed.
<b>Lua Scripting</b>	Disabled	Pivotal recommends keeping Lua scripting disabled unless developers are running apps that require Lua scripting, such as .Net Steeltoe apps. Check whether the language your apps are using requires Lua scripting.

### (Optional) Enabling Secure Service Instance Credentials for On-Demand Redis

To secure your on-demand binding credentials in runtime [CredHub](#) instead of the Cloud Controller database (CCDB), do the following:





**Note:** This is a beta feature. Use it at your own risk in non-production environments. Send comments and feedback to the [PCF Feedback List](#).

1. Configure the Pivotal Application Service (PAS) tile to support securing service instance credentials in runtime CredHub. See [Step 1: Configure the PAS Tile](#).
2. After deploying the tile, notify developers that they must unbind and rebind existing service instances to secure their credentials with CredHub.

## Updating On-Demand Service Plans

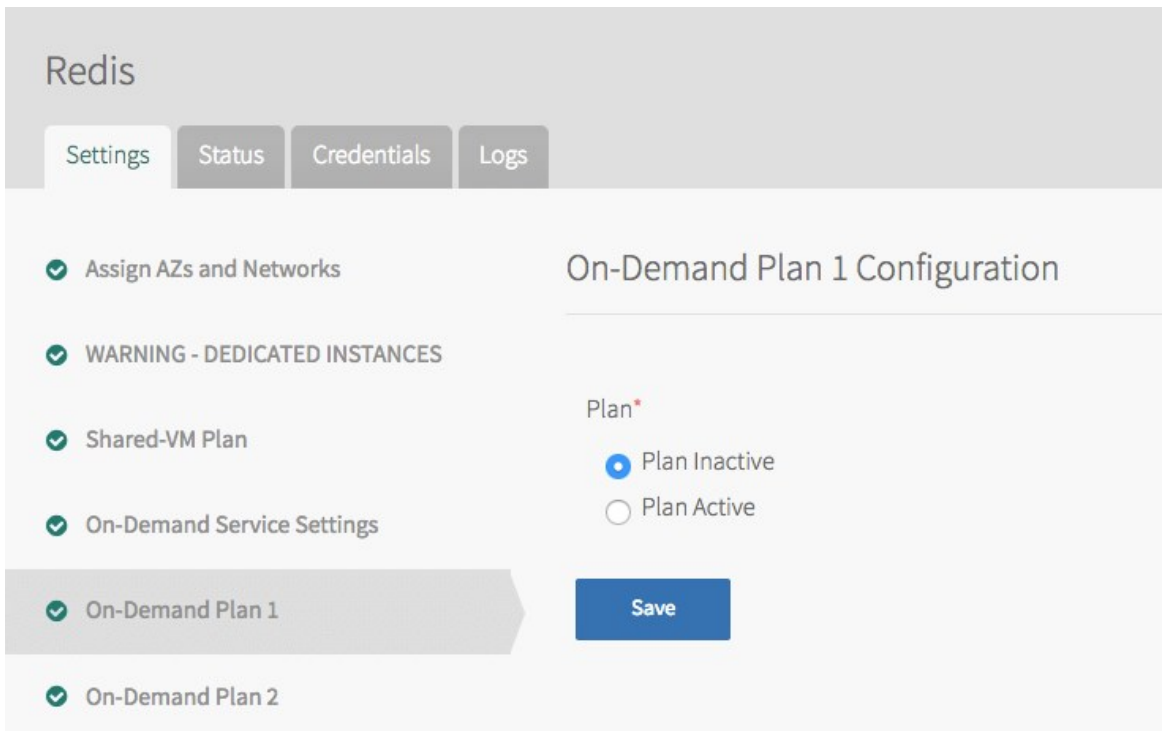
Operators can update certain settings after the plans have been created. If the operator updates the VM size, disk size, or the Redis configuration settings (enabling Lua Scripting, max-clients, timeout and TCP keep-alive), these settings are implemented in all instances that are already created.

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

## Removing On-Demand Service Plans

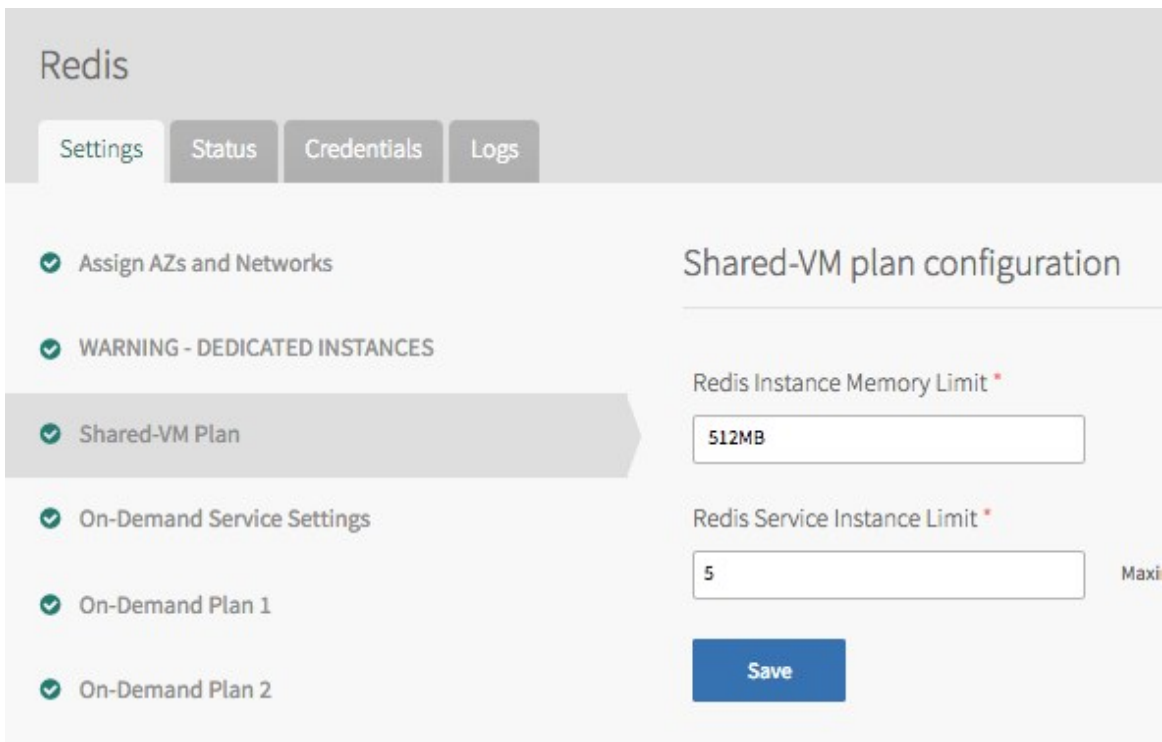
If you want to remove the On-Demand Service from your tile, do the following:

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 Broker
  - ◆ On-Demand Broker Smoke Tests
  - ◆ Upgrade All On-Demand Service Instances
  - ◆ Delete All Service Instances and Deregister On-Demand Broker
3. Create an empty service network. For instructions, see [Creating an Empty Services Network when using on-demand Service Tiles for Non-On-Demand Usage Only](#) in the Pivotal Support knowledge base.
4. Go to each of the three On-Demand Plan pages on the Redis for PCF tile, and set each plan to **Plan Inactive**. For example:



## Shared-VM Plan Settings

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.

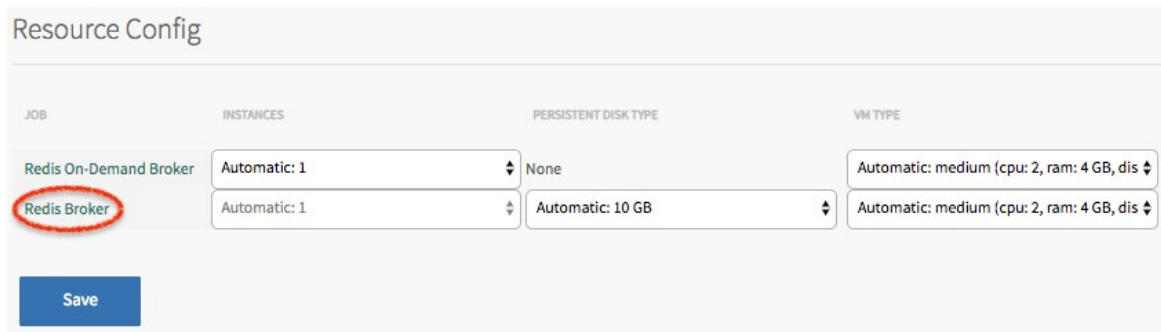
The Redis broker server runs all of the Redis instances for your Shared-VM plan. From the **Resource Config** page, you can change the CPU, RAM, Ephemeral Disk, and Persistent Disk made available, as needed.

### Configure Memory Limits for Shared-VM Plans

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


To configure the limits in these fields, 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:** You can 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. Do not use shared-VM instances in production environments.

## Configure Resources for Shared-VM Plans

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

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

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

Pivotal recommends that the persistent disk is at least 3.5x the VM memory for the Shared-VM.

## Disable Shared VM Plans

You can disable Shared VM Plans by doing the following while configuring the 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**.

## Configure Syslog Forwarding

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

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

All logs follow RFC5424 format.

To configure syslog forwarding, do the following:

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

## Configure properties for PCF Redis syslog forwarding

Do you want to configure syslog forwarding for PCF Redis?\*

No  
 Yes without encryption  
 Yes with TLS encryption

Address \*

Port \*

Transport protocol\*

Permitted peer \*

TLS CA certificate \*

```

-----BEGIN CERTIFICATE-----
MIIDYzCCAkugAwIBAgIU1anzWHYT6BEB0ZTTh4r88u8mF4wDQYJKoZIhvcNAQEL
BQAwHzELMAkGA1UEBhMCVVMxEDAoBgNVBAoMB1Bpdm90YWwwHhcNMTgwOTI3MD
Yw
NzU5WhcNMjAwOTI3MDYwNzU5WjAwMQswCQYDVOQGEwJVUzEQMA4GA1UECgwHUGI
1k3PkhDEFMAQCA1UEAwYwCkMAQomQwMURUJANBakkkkCQwQBAQFEAAQCAQ8AMIB
-----END CERTIFICATE-----

```

The CA certificate of the syslog destination

**Save**

2. Select either **Yes without encryption** or **Yes with TLS encryption**.



**Note:** To use syslog forwarding for on-demand instances, you must select the [Allow outbound internet access from service instances](#) checkbox in the **On-Demand Service Settings** tab.

3. Enter the Syslog **Address** and **Port**, and select the **Transport protocol** of your remote destination. You can only use TCP if you are using TLS encryption.



The information required for these fields is provided by your remote destination. **Address** should be something such as `logs.papertrailapp.com`, and **Port** will be a number such as `41635`.

4. If you selected **Yes with TLS encryption**, complete these fields:
  - ◊ **Permitted Peer** refers to the remote syslog destination. It allows each VM to establish an encrypted tunnel with the remote syslog destination. The Permitted Peer is either the accepted fingerprint (SHA1) or name of the remote peer, for example `*.example.com`.
  - ◊ **TLS CA certificate** refers to the trusted certificate authorities for the remote syslog destination. Large certificate chains (> 8 kb) are not supported.
5. Click **Save**.

## Update Stemcell

If required, do the following to update the stemcell for Redis for PCF:

1. Download the stemcell from [Pivotal Network](#).
2. In the Ops Manager, click **Stemcell Library**.
3. Click **Import Stemcell**, and then select the stemcell you downloaded from Pivotal Network.

Product	Required	Deployed	Staged
<b>Pivotal Container Service</b> Version 1.2.0	ubuntu-xenial 97.17	-	97.17 Latest stemcell.
<b>Pivotal Application Service</b> Version 2.3.0	ubuntu-xenial 97.16	-	97.17 Latest stemcell.

4. Click **Save**.

## Apply Changes from Your Configuration

Your installation is not complete until you apply your configuration changes. Follow the steps below:

1. Return to the Ops Manager Installation Dashboard.
2. In the Ops Manager Dashboard, do the following to complete the installation:
  1. If you are using Ops Manager v2.3 or later, click **Review Pending Changes**. For more information about this Ops Manager page, see [Reviewing Pending Product Changes](#).

2. Click **Apply Changes**.

## Create 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 configured in your deployment. You can obtain the IP addresses for these VMs in Ops Manager under the **Resource Config** section for the Redis for PCF tile.



**Note:** Without ASGs, this service is unusable.

## Application Container Network Connections

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

Destination	Ports	Protocol	Reason
<code>ASSIGNED_NETWORK</code>	32768-61000	tcp	Enable application to access shared vm service instance
<code>ASSIGNED_NETWORK</code>	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"
  }
]
```

## Validating Installation

Smoke tests run as part of Redis for PCF installation to validate that the install succeeded. For more information, see [Redis for PCF Smoke Tests](#).

## 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 do the following:
  1. If you are using Ops Manager v2.3 or later, click **Review Pending Changes**. For more information about this Ops Manager page, see [Reviewing Pending Product Changes](#).

2. Click **Apply Changes**.

Create a pull request or raise an issue on the source for this page in [GitHub](#)

## Upgrading Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

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

## Compatible Upgrade Paths

Before upgrading Redis for PCF, for compatibility information, see the [Product Version Matrix](#).

## Upgrade to Redis for PCF v2.0



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

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

For information on the upgrade paths for each released version, see [Compatible Upgrade Paths](#) above.

To upgrade Redis for PCF v2.0:

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. Select the deprecation warning checkbox in the **WARNING - DEPRECATED INSTANCES** tab. This removes any dedicated-VM instances in your deployment.



**Note:** This checkbox is mandatory when upgrading to Redis for PCF v2.0. Ensure that you have backed up or migrated data to on-demand instances before upgrading to Redis for PCF v2.0.

5. If required, update any new mandatory configuration parameters.

- Pivotal recommends that you ensure that the **Upgrade All On-Demand Service Instances** errand checkbox is selected on the **Review Pending Changes** page.



**Note:** Existing service instances are not upgraded if you do not run this errand. These instances do not benefit from any security fixes or new features included in the upgrade.

- Click **Apply changes**. The rest of the process is automated.

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

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

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.

## Downtime During Upgrades and Redeploys

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

### Ops Manager Changes

In Ops Manager, any field that changes the manifest causes a redeploy of the Redis for PCF tile.

### PAS Changes

In the Pivotal Application Service (PAS), changes to any of the following properties can trigger downtime:

- `$runtime.system_domain`—Runtime System Domain
- `..cf.ha_proxy.skip_cert_verify.value`—Disable SSL certificate verification for this environment in PAS
- `$runtime.apps_domain`—Runtime Apps Domain
- `..cf.nats.ips`—NATS Resource Config
- `$self.service_network`—Service Networks in Ops Manager

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

- Redis on-demand broker in Redis for PCF v1.8 and later
- Shared-VM Services in Redis for PCF v1.9 and earlier

## Upgrading all Service Instances

- For Redis for PCF v1.8 and later, downtime for service instances occurs only after the operator runs the `upgrade-all-service-instances` BOSH errand, after all tile upgrades are completed successfully.
- Any change to a field on the Redis for PCF tile causes BOSH to redeploy the on-demand Redis broker and can cause service instance downtime when the operator runs the `upgrade-all-service-instances` errand.

## Network Changes after Deployment

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

### Shared VMs

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

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

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

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

### On-Demand Service Instances

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

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

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

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

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

## Release Policy

When a new version of Redis is released, a new version of Redis for Pivotal PCF is released soon after.

For more information about the Pivotal Cloud Foundry release policy, see [Release Policy](#).

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Setting Limits for On-Demand Service Instances



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

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 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 Pivotal Cloud Foundry and RabbitMQ for Pivotal Cloud Foundry, you must set a separate global service quota for each of them.

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

## Create Plan-level Quotas

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

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

## Create and Set Org-level Quotas

An org-level quota applies to all 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:

- ◆ **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-NAME -m TOTAL-MEMORY -i INSTANCE-MEMORY -r ROUTES -s SERVICE-INSTANCES --allow-paid-service-plans
```

Where:

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

For example:

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

2. Associate the quota you created 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
-------------	-------------



<code>on-demand-broker/SERVICE-NAME/quota_remaining</code>	Quota remaining for all instances across all plans
<code>on-demand-broker/SERVICE-NAME/PLAN-NAME/quota_remaining</code>	Quota remaining for a specific plan
<code>on-demand-broker/SERVICE-NAME/total_instances</code>	Total instances created across all plans
<code>on-demand-broker/SERVICE-NAME/PLAN-NAME/total_instances</code>	Total instances created for a specific plan



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

You can also view service instance usage information in Apps Manager. For more information, see [Reporting Instance Usage with Apps Manager](#).

## Calculate Resource Costs for On-Demand Plans

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

To view configurations for your Redis for Pivotal PCF on-demand plan, do the following:

1. Navigate to **Ops Manager Installation Dashboard > Redis > Settings**.
2. Click the section for the plan you want to view. For example, **On-Demand Plan 1**.

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

Plan Quota (min: 1) \*

CF Service Access \*

AZ to deploy Redis instances of this plan \*

us-central1-f \*

us-central1-c \*

us-central1-b \*

Server VM type \*

Server Disk type \*



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

## Calculate Maximum Resource Cost Per On-Demand Plan

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

**plan quota x cost of selected resources**

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

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

## Calculate Maximum Resource Cost for All On-Demand Plans

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

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**

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Configuring Automated Service Backups



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

This topic describes how to configure automated backups in Redis for Pivotal Cloud Foundry (PCF).

## Comparison of the Available Backup Methods

Redis for PCF provides two backup methods, which can be used together or alone:

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

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

The table below summarizes the differences between the two methods:

Backup Method	Supported Services	What is Backed Up
BBR	On-demand	<ul style="list-style-type: none"> <li>• Data stored in Redis</li> <li>• Manifest used to deploy service instance</li> <li>• Certain additional configuration including: plan settings such as <b>Redis Client Timeout</b> and arbitrary parameters such as <code>maxmemory-policy</code></li> </ul>

---

Automated service backups	<ul style="list-style-type: none"> <li>• On-demand Data stored in Redis</li> <li>• Shared-VM</li> </ul>
---------------------------	---

---



**Note:** Neither backup method backs up other manual changes made to service instances, either via SSH or with the redis client `config` command.

For more information, see [BOSH Backup and Restore \(BBR\) for On-Demand Redis for PCF](#).

## About Automated Service Backups

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

Automated backups have the following features:

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

## Backup Files

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

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

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

## About Configuring Backups

Redis for PCF automatically backs up databases to external storage.

- **How and where:** There are four options for how automated backups transfer backup data and where the data saves to:
  - ◊ **Option 1: Back Up with AWS:** Redis for PCF runs an AWS S3 client that saves backups to an S3 bucket.
  - ◊ **Option 2: Back Up with SCP:** Redis for PCF runs an SCP command that secure-copies backups to a VM or physical machine operating outside of PCF. SCP stands for secure copy protocol, and offers a way to securely transfer files between two hosts. The operator provisions the backup machine separately from their PCF

installation. This is the fastest option.

- [Option 3: Back Up to GCS](#): Redis for PCF runs an GCS SDK that saves backups to an Google Cloud Storage bucket.
- [Option 4: Back Up to Azure](#): Redis for PCF runs an Azure SDK that saves backups to an Azure storage account.
- **When:** Backups follow a schedule that you specify with a cron expression.

For general information about cron, see [package cron](#).

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

## Option 1: Back Up with AWS

To back up your database to an Amazon S3 bucket, complete the following procedures:

- [Create a Policy and Access Key](#)
- [Configure Backups in Ops Manager](#)

### Create a Policy and Access Key

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

Do the following to create a policy and access key:

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

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

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

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

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

## Configure Backups in Ops Manager

Do the following to connect Redis for PCF to your S3 account:

1. Navigate to the Ops Manager Installation Dashboard and click the **Redis for PCF** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **AWS S3**.

### Configure blob store for Redis backups

**Backup configuration\***

Disable Backups  
 AWS S3

**Access Key ID \***

Optional field dependent upon your blobstore configuration

**Secret Access Key \***

**Endpoint URL**

**Bucket Name \***

**Bucket Path \***

**Cron Schedule \***

00 \* \* \* \*

**Backup timeout \***

10

SCP  
 Azure  
 GCS

4. Fill in the fields as follows:

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

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

5. Click **Save**.

## Option 2: Back Up with SCP

To back up your database using SCP, complete the following procedures:

- [\(Recommended\) Create a Public and Private Key Pair](#)
- [Configure Backups in Ops Manager](#)

### (Recommended) Create a Public and Private Key Pair

Redis for PCF accesses a remote host as a user with a private key for authentication. Pivotal recommends that this user and key pair be solely for Redis for PCF.

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

1. Determine the remote host that you will be using to store backups for Redis for PCF. Ensure that the Redis service instances can access the remote host.



**Note:** Pivotal recommends using a VM outside the PCF deployment for the destination of SCP backups. As a result you might need to enable public IPs for the Redis VMs.

2. Create a new user for Redis for PCF on the destination VM.
3. Create a new public and private key pair for authenticating as the above user on the destination VM.

## Configure Backups in Ops Manager

Do the following to connect Redis for PCF to your destination VM:

1. Navigate to the Ops Manager Installation Dashboard and click the **Redis for PCF** tile.



2. Click **Backups**.
3. Under **Backup configuration**, select **SCP**.

### Configure blob store for Redis backups

Backup configuration \*

Disable Backups

AWS S3

SCP

Username \*

Private Key \*

Hostname \*

Destination Directory \*

SCP Port \*

Cron Schedule \*

Backup timeout \*

The screenshot shows a configuration form with a 'Fingerprint' label above an empty text input field. Below the input field are two radio button options: 'Azure' and 'GCS', both of which are currently unselected.

4. Fill in the fields as follows:

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 <code>Username</code>	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. For example, once daily at 2am is <code>* 2 * * *</code> . This field also accepts a pre-defined schedule, such as <code>@yearly</code> , <code>@monthly</code> , <code>@weekly</code> , <code>@daily</code> , <code>@hourly</code> , or <code>@every TIME</code> , where <code>TIME</code> is any supported time string, such as <code>1h30m</code> . For more information, see the cron package <a href="#">documentation</a> .	Mandatory
Backup timeout	The amount of time, in seconds, that the backup process waits for the <code>BGSAVE</code> command to complete on your instance before transferring the RDB file to the SCP server. If the timeout is reached, <code>BGSAVE</code> continues but backups fail and are not uploaded.	Mandatory
Fingerprint	The fingerprint of the public key of the SCP server. To retrieve the server's fingerprint, run <code>ssh-keygen -E md5 -lf ~/.ssh/id_rsa.pub</code> .	Optional

5. Click **Save**.

## Option 3: Back Up with GCS

To back up your database using GCS, complete the following procedures:

- [Create a Service Account](#)
- [Configure Backups in Ops Manager](#)

### Create a Service Account

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

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

1. In the GCS console, create a new service account for Redis for PCF: **IAM and Admin > Service Accounts > Create Service Account**.
2. Enter a unique name in the **Service account name** field, such as `Redis-for-PCF`.
3. In the **Roles** dropdown, grant the new service account the **Storage Admin** role.
4. Select the **Furnish a new private key** checkbox so that a new key is created and downloaded.
5. Click **Create** and take note of the name and location of the service account JSON file that is downloaded.

## Configure Backups in Ops Manager

Do the following to connect Redis for PCF to GCS:

1. Navigate to the Ops Manager Installation Dashboard and click the **Redis for PCF** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **GCS**.

### Configure blob store for Redis backups

Backup configuration \*

Disable Backups

AWS S3

SCP

Azure

GCS

Project ID \*

Bucket name \*

Service account private key \*

JSON contents

Cron Schedule \*

Backup timeout \*

4. Fill in the fields as follows:

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

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

5. Click **Save**.

## Back Up to Azure

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

1. Navigate to the Ops Manager Installation Dashboard and click the **Redis for PCF** tile.
2. Click **Backups**.
3. Under **Backup configuration**, select **Azure**.

The screenshot shows a configuration interface titled "Configure blob store for Redis backups". Under the "Backup configuration\*" section, there are four radio button options: "Disable Backups", "AWS S3", "SCP", and "Azure". The "Azure" option is selected. Below the radio buttons, there are two required fields: "Account\*" and "Azure Storage Access Key\*", each with an empty text input box.

The screenshot shows a configuration form with the following fields and values:

- Container Name \***: [Empty text box]
- Destination Directory \***: [Empty text box]
- Blob Store Base URL**: [Empty text box]
- Cron Schedule \***: [00\*\*\*]
- Backup timeout \***: [10]
- GCS**

4. Fill in the fields as follows:

Field	Description	Mandatory/Optional
Account	Account name	Mandatory
Azure Storage Access Key	Azure specific credentials required to write to the Azure container	Mandatory
Container Name	Name of the Azure container where to store the backup	Mandatory
Destination Directory	Directory within the Azure container to store the backup files to	Mandatory
Blob Store Base URL	URL pointing to Azure resource	Optional
Cron Schedule	Backups schedule in crontab format. For example, once daily at 2am is * 2 * * *. This field also accepts a pre-defined schedule, such as @yearly, @monthly, @weekly, @daily, @hourly, or @every TIME, where TIME is any supported time string, such as 1h30m. For more information, see the cron package <a href="#">documentation</a> .	Mandatory

Field	Description	Mandatory/Optional
Backup timeout	The amount of time, in seconds, that the backup process waits for the <code>BGSAVE</code> command to complete on your instance before transferring the RDB file to your configured destination. If the timeout is reached, <code>BGSAVE</code> continues but backups fail and are not uploaded.	Mandatory

5. Click **Save**.

## Back Up and Restore Manually

To back up or restore Redis manually, see [Manually Backing Up and Restoring Redis for Pivotal Cloud Foundry](#) in the Pivotal Support knowledge base.

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## BOSH Backup and Restore (BBR) for On-Demand Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

This topic describes how to use the BOSH Backup and Restore (BBR) command-line tool for backing up and restoring BOSH deployments.

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

For more information, see [Configuring Automated Service Backups](#) and [Comparison of the Available Backup Methods](#).



**Note:** Only on-demand Redis service instances have support for BBR, for backup and restore of dedicated and shared instances see [Configuring Automated Backups for Redis for PCF](#).

## Prepare to Use BBR

To take a backup of PCF and On-Demand Redis for PCF, BBR must be installed. If you do not already have it installed, follow the instructions in [Prepare to Create Your Backup](#) in the BBR docs.



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

Record the BOSH Director IP and path to server certificate.

## Identify Your Redis Deployments

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

To obtain the instance deployment names, do the following:

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

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

Where:

- ◆ **REDIS-BOSH-CLIENT**, **REDIS-BOSH-PASSWORD**: To find these in the Ops Manager Installation Dashboard, click the Redis for PCF tile, navigate to the **Credentials** tab, and click **UAA Client Credentials**. Note the **Redis BOSH UAA** credentials.
- ◆ **BOSH-DIRECTOR-IP**: You retrieved this value in [Step 1-6: Prepare to Create Your Backup](#).
- ◆ **PATH-TO-BOSH-SERVER-CERTIFICATE**: This is the path to the Certificate Authority (CA) certificate for the BOSH Director. For more information, see [Ensure BOSH Director Certificate Availability](#).



**Note:** In the command above, **BOSH\_CLIENT** is not a variable.

For example:

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

## Back Up Using BBR

Follow these steps:

1. Back up PCF.

This includes backing up your Ops Manager installation settings, BOSH Director and PAS, as detailed in [the Pivotal BBR Documentation](#).



**Note:** For a full restore of Redis service instances to be valid, you must have a backup of the BOSH Director and PAS.



2. Backup each Redis service instance:

From your jumpbox run the following.

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

Where:

- ◆ **BOSH-CLIENT**, **BOSH-CLIENT-PASSWORD**: These are the client credentials you retrieved in [Preparing to Use BBR](#).
- ◆ **REDIS-SERVICE-INSTANCE-DEPLOYMENT-NAME**: This is the deployment name for the on-demand Redis service instance you are backing up.



**Note:** In the above command, **BOSH\_CLIENT\_SECRET** is not a variable.

For example:

```
$ BOSH_CLIENT_SECRET=KJsdgKJj123451jk83Hufy12345b6-34n4 \  
bbr deployment \  
--target 10.0.0.5 \  
--username ops_manager \  
--ca-cert /var/example/workspaces/default/root_ca_certificate \  
--deployment service-instance_40b123e4a-be1c-1232-ad31-123e01b7d169 \  
backup
```

3. Follow the steps given in the [After Taking the Backups](#) step of the BBR documentation.

Make sure to do this for the backup artifacts for all of your service instances and your BOSH Director and PAS.

## Restore Using BBR

Follow these steps:

1. To restore on-demand Redis service instance data, follow the procedure for [Restoring PCF from Backup with BBR](#) in full.



**Note:** Ensure that as part of [Step 6: Transfer Artifacts to Jumpbox](#) you transfer your Redis service instance artifacts.

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

```
$ BOSH_CLIENT_SECRET=BOSH-CLIENT-PASSWORD \  
    bbr deployment \  
--target BOSH-DIRECTOR-IP \  
--deployment REDIS-SERVICE-INSTANCE-DEPLOYMENT-NAME \  
restore
```

```
--username BOSH-CLIENT \  
--ca-cert PATH-TO-BOSH-SERVER-CERTIFICATE \  
--deployment REDIS-SERVICE-INSTANCE-DEPLOYMENT-NAME \  
restore --artifact-path PATH-TO-SERVICE-INSTANCE-ARTIFACT
```

Where:

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



**Note:** In the above command, `BOSH_CLIENT_SECRET` is not a variable.

For example:

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

If a restore fails because there is no deployment of the name specified, then you are likely in the [Backup Artifact for a Non-Existent Service Instance](#) inconsistent state and can skip the restore for that artifact.



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

## Possible Inconsistent States

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

### No Backup Artifact for a Service Instance

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

### Backup Artifact for a Non-Existent Service Instance

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

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Monitoring Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### 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.

As an example of how to display KPIs and metrics, see the [CF Redis example dashboard](#), which uses [Datadog](#). Pivotal does not endorse or provide support for any third-party solution.

## 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)**.

Metrics polling interval (min: 10) \*

Metrics are emitted in the following format:

```
origin:"p-redis" eventType:ValueMetric timestamp:1480084323333475533 deployment:"cf-redis" job:"cf-redis-broker" index:"{redacted}" ip:"10.0.1.49" valueMetric:<name:"/p-redis/service-broker/dedicated_vm_plan/available_instances" value:4 unit:"" >
```

## Critical Logs

Pivotal recommends operators set up alerts on critical logs to help prevent further degradation of the Redis service. For examples of critical logs for service backups, including log messages for failed backups, backups with errors, and backups that failed to upload to destinations, see [Service Backups for Pivotal Cloud Foundry](#).

## Key Performance Indicators

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

Pivotal provides the following KPIs as general alerting and response guidance for typical Redis for PCF installations. Pivotal recommends that operators continue to fine-tune the alert measures to their installation by observing historical trends. Pivotal also recommends that operators expand beyond this guidance and create new, installation-specific monitoring metrics, thresholds, and alerts

based on learning from their own installations.

For a list of all other Redis metrics, see [Other Redis Metrics](#).

## Redis for PCF Service KPIs

### Total Instances For On-Demand Service

#### total\_instances

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

### Quota Remaining For On-Demand Service

#### quota\_remaining

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

### Total Instances For Shared-VM Service

/p-redis/service-broker/shared\_vm\_plan/total\_instances

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

## Redis KPIs

### Percent of Persistent Disk Used

`disk.persistent.percent`

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

### Used Memory Percent

`info.memory.used_memory / info.memory.maxmemory`

<p><b>Description</b></p>	<p>The ratio of these two metrics returns the percentage of available memory used:</p> <ul style="list-style-type: none"> <li><code>info.memory.used_memory</code> is a metric of the total number of bytes allocated by Redis using its allocator (either standard libc, jemalloc, or an alternative allocator such as tcmalloc).</li> <li><code>maxmemory</code> is a configuration option for the total memory made available to the Redis instance.</li> </ul> <p><b>Use:</b> This is a performance metric that is most critical for Redis instances with a <code>maxmemory-policy</code> of <code>allkeys-lru</code></p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> percentage  <b>Frequency:</b> 30s (default), 10s (configurable minimum)</p>
<p><b>Recommended measurement</b></p>	<p>App-specific based on velocity of data flow. Some options are:</p> <ol style="list-style-type: none"> <li>Individual data points—Use if key eviction is in place, for example, in cache use cases.</li> <li>Average over last 10 minutes—Use if this gives you enough detail.</li> <li>Maximum of last 10 minutes</li> </ol> <p>If key eviction is not in place, options 1 or 3 give more useful information to ensure that high usage triggers an alert.</p>
<p><b>Recommended alert thresholds</b></p>	<p><b>Yellow warning:</b> 80% Not applicable for cache usage. When used as a cache, Redis will typically use up to <code>maxmemory</code> and then evict keys to make space for new entries.</p> <p>A different threshold might be appropriate for specific use cases of no key eviction, to allow for reaction time. Factors to consider:</p> <ol style="list-style-type: none"> <li>Traffic load on app—Higher traffic means that Redis memory will fill up faster.</li> <li>Average size of data added/ transaction—The more data added to Redis on a single transaction, the faster Redis will fill up its memory.</li> </ol> <p><b>Red critical:</b> 90%. See warning-specific threshold information.</p>
<p><b>Recommended response</b></p>	<p>No action assuming the <code>maxmemory</code> policy set meets your apps needs. If the <code>maxmemory</code> policy does not persist data as you wish, either coordinate a backup cadence or update your <code>maxmemory</code> policy if using the on-demand Redis service.</p>

## Connected Clients

### `info.clients.connected_clients`

<p><b>Description</b></p>	<p>Number of clients currently connected to the Redis instance.</p> <p><b>Use:</b> Redis does not close client connections. They remain open until closed explicitly by the client or another script. Once the <code>connected_clients</code> reaches <code>maxclients</code>, Redis stops accepting new connections and begins producing <code>ERR max number of clients reached</code> errors.</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> number  <b>Frequency:</b> 30s (default), 10s (configurable minimum)</p>
---------------------------	---

<b>Recommended measurement</b>	Average over last 10 minutes
<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> App-specific. When connected clients reaches max clients, no more clients can connect. This alert should be at the level where it can tell you that your app has scaled to a certain level and may require action.</p> <p><b>Red critical:</b> App-specific. When connected clients reaches max clients, no more clients can connect. This alert should be at the level where it can tell you that your app has scaled to a certain level and may require action.</p>
<b>Recommended response</b>	Increase max clients for your instance if using the on-demand service, or reduce the number of connected clients.

## Blocked Clients

### info.clients.blocked\_clients

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

<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> The expected range of the <code>blocked_clients</code> metric depends on what Redis is being used for:</p> <ul style="list-style-type: none"> <li>• Many uses will have no need for blocking commands and should expect <code>blocked_clients</code> to always be zero.</li> <li>• If blocking commands are being used to force a recipient client to wait for a required input, a raised <code>blocked_clients</code> might suggest a problem with the source clients.</li> <li>• <code>blocked_clients</code> might be expected to be high in situations where Redis is being used for infrequent messaging.</li> </ul> <p>If <code>blocked_clients</code> is expected to be non-zero, warnings could be based on change from baseline. A sudden rise in <code>blocked_clients</code> could be caused by source clients failing to provide data required by blocked clients.</p> <p><b>Red critical:</b> There is no <code>blocked_clients</code> threshold critical to the function of Redis. However a problem that is causing <code>blocked_clients</code> to rise might often cause a rise in <code>connected_clients</code>. <code>connected_clients</code> does have a hard upper limit and should be used to trigger alerts.</p>
<b>Recommended response</b>	<p>Analysis could include:</p> <ul style="list-style-type: none"> <li>• Checking the <code>connected_clients</code> metric. <code>blocked_clients</code> would often rise in concert with <code>connected_clients</code>.</li> <li>• Establishing whether the rise in <code>blocked_clients</code> is accompanied by an overall increase in apps connecting to Redis, or by an asymmetry in clients providing and receiving data with blocking commands</li> <li>• Considering whether a change in <code>blocked_clients</code> is most likely caused by oversupply of blocking requests or undersupply of data</li> <li>• Considering whether a change in network latency is delaying the data from source clients</li> </ul> <p>In general, a rise or change in <code>blocked_clients</code> is more likely to suggest a problem in the network or infrastructure, or in the function of client apps, rather than a problem with the Redis service.</p>

## Memory Fragmentation Ratio

`info.memory.mem_fragmentation_ratio`

<b>Description</b>	<p>Ratio of the amount of memory allocated to Redis by the OS to the amount of memory that Redis is using</p> <p><b>Use:</b> A memory fragmentation less than one shows that the memory used by Redis is higher than the OS available memory. In other packagings of redis, large values reflect memory fragmentation. For Redis for PCF, the instances only run Redis meaning that no other processes will be affected by a high fragmentation ratio (e.g., 10 or 11).</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> ratio  <b>Frequency:</b> 30s (default), 10s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes



<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> &lt; 1. Less than 1 indicates that the memory used by Redis is higher than the OS available memory which can lead to performance degradations.</p> <p><b>Red critical:</b> Same as warning threshold.</p>
<b>Recommended response</b>	Restart the Redis server to normalize fragmentation ratio.

## Instantaneous Operations Per Second

### info.stats.instantaneous\_ops\_per\_sec

<b>Description</b>	<p>The number of commands processed per second by the Redis server. The <code>instantaneous_ops_per_sec</code> is calculated as the mean of the recent samples taken by the server. The number of recent samples is hardcoded as 16 in the implementation of Redis.</p> <p><b>Use:</b> The higher the commands processed per second, the better the performance of Redis. This is because Redis is single threaded and the commands are processed in sequence. A higher throughput would thus mean faster response per request which is a direct indicator of higher performance. A drop in the number of commands processed per second as compared to historical norms could be a sign of either low command volume or slow commands blocking the system. Low command volume could be normal, or it could be indicative of problems upstream.</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> count  <b>Frequency:</b> 30s (default), 10s (configurable minimum)</p>
<b>Recommended measurement</b>	Every 30 seconds
<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> A drop in the count compared to historical norms could be a sign of either low command volume or slow commands blocking the system. Low command volume could be normal, or it could be indicative of problems upstream. Slow commands could be due to a latency issue, a large number of clients being connected to the same instance, memory being swapped out, etc. Thus, the count is possibly a symptom of compromised Redis performance. However, this is not the case when low command volume is expected.</p> <p><b>Red critical:</b> A very low count or a large drop from previous counts may indicate a downturn in performance that should result in an investigation. That is unless the low traffic is expected behavior.</p>

---

**Recommended response** A drop in the count may be a symptom of compromised Redis performance. The following are possible responses:

1. **Identify slow commands using the slowlog:**

Redis logs all the commands that take more than a specified amount of time in slowlog. By default, this time is set to 20ms and the slowlog is allowed a maximum of 120 commands. For the purposes of slowlog, execution time is the time taken by Redis alone and does not account for time spent in I/O. So it would not log slow commands solely due to network latency.

Given that typical commands, including network latency, take about 200ms, a 20ms Redis execution time is 100 times slower. This could be indicative of memory management issues wherein Redis pages have been swapped to disk.

To see all the commands with slow Redis execution times, type `slowlog get` in the redis-cli.

2. **Monitor client connections:**

Because Redis is single threaded, one process services requests from all clients. As the number of clients grows, the percentage of resource time given to each client decreases and each client spends an increasing time waiting for their share of Redis server time.

Monitoring the number of clients may be important because there may be apps creating connections that you did not expect or your app may not be efficiently closing unused connections.

The connected clients metrics can be used to monitor this. This can also be viewed from the redis-cli using the command `info clients`.

3. **Limit client connections:**

This currently defaults to 10000, but depending on the app, you might want to limit this further. To do this, run `CONFIG SET maxclients NUMBER-OF-CONNECTIONS` in the redis-cli. You can configure this for On-Demand service instances in Ops Manager. Connections that exceed the limit are rejected and closed immediately.

It is important to set `maxclients` to limit the number of unintended client connections. Set `maxclients` to 110% to 150% of your expected peak number of connections. In addition, because an error message is returned for failed connection attempts, the maxclient limit warns you that a significant number of unexpected connections are occurring. This helps maintain optimal Redis performance.

4. **Improve memory management:**

Poor memory can cause increased latency in Redis. If your Redis instance is using more memory than is available, the operating system will swap parts of the redis process from out of physical memory and onto disk. Swapping will significantly reduce Redis performance since reads from disk are about 5 orders or magnitude slower than reads from physical memory.

---

## Keyspace Hits / Keyspace Misses + Keyspace Hits

`info.stats.keyspace_hits / info.stats.keyspace_misses + info.stats.keyspace_hits`

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

## BOSH Health Monitor Metrics

The BOSH layer that underlies Pivotal Cloud Foundry generates `healthmonitor` metrics for all VMs in the deployment. As of Redis for Pivotal Cloud Foundry v2.0, these metrics are included in the Loggregator Firehose by default. For more information, see [BOSH System Metrics Available in Loggregator Firehose](#) in *Pivotal Application Service Release Notes*.

## Other Redis Metrics

Redis also exposes the following metrics. for more information, see the [Redis documentation](#).

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

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

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

Create a pull request or raise an issue on the source for this page in GitHub

## Redis for PCF Smoke Tests



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### 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. Deploys an instance of the [CF Redis Example App](#) to this space.
3. Creates a Redis instance and binds it to the CF Redis Example App.
4. Creates a service key to retrieve the Redis instance IP address.
5. Creates a restrictive security group, `redis-smoke-tests-sg`, and binds it to the space.
6. Checks that the CF Redis Example App can write to and read from the Redis instance.

## Security Groups

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

```
[
  {
    "protocol": "tcp",
    "destination": "<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 nodes.

## Smoke Tests Resilience

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

The smoke tests implement a retry policy for commands issued to CF, for two reasons: - To avoid smoke test failures due to temporary issues such as the ones mentioned above - To ensure that the service instances and bindings created for testing are cleaned up.

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

The linear back-off was selected as a good middle ground between: - Situations where the system is generally unstable-such as load-balancing issues-where max number of retries are preferred, and - Situations where the system is suffering from a failure that lasts a few seconds-such as restart of a Cloud Foundry VM where it is preferable to wait before reattempting the command.

## Considerations

The above retry policy does not guard against a more permanent Cloud Foundry downtime or network connectivity issues. In this case, commands fail after the maximum number of attempts and might leave claimed instances behind. Pivotal recommends 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 are summarised at the end of the errand log output. Detailed logs can be found where the failure occurs. Some common failures are listed below.

<b>Error</b>	Failed to target Cloud Foundry
<b>Cause</b>	Your PCF is unresponsive.
<b>Solution</b>	Examine the detailed error message in the logs and check the <a href="#">PCF Troubleshooting Guide</a> for advice.
<b>Error</b>	Failed to bind Redis service instance to test app.
<b>Cause</b>	Your deployment's broker has not been registered with PCF.
<b>Solution</b>	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, for example, `Failed to target Cloud Foundry`. Lookout for `TIP: ...` in the logs next to any error output for further troubleshooting hints.

[Create a pull request or raise an issue on the source for this page in GitHub](#)


## Troubleshooting Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

**Page last updated:**

This topic lists troubleshooting information for Redis for PCF.



**Note:** Some of the troubleshooting approaches in this topic suggest potentially destructive operations. Pivotal recommends that you back up both your Ops Manager and deployments before attempting such operations.

For more information about backing up your setup and exporting your Ops Manager installation, see [Backing Up Pivotal Cloud Foundry with BBR](#)

## Useful Debugging Commands

Before debugging, gather the following about your PCF deployment:

- Current version of Redis for PCF, and, if upgrading, the previous version of Redis for PCF
- Current version of Ops Manager, and, if upgrading, the previous version of Ops Manager

### cf CLI Commands

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

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

### BOSH CLI Commands

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

Purpose	Command
---------	---------

View the targeted BOSH director, version, and CPI	<code>bosh env</code>
View the deployments deployed via the targeted BOSH director	<code>bosh deployments</code>
View the VMs for a given deployment	<code>bosh -d DEPLOYMENT vms</code>
SSH into a given deployment's VM	<code>bosh -d DEPLOYMENT ssh VM</code>

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

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

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

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

## About the Redis CLI

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

To access the `redis-cli`, do the following:

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

```
/var/vcap/packages/redis/bin/redis-cli -p PORT -a PASSWORD
```

Where:

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

For more information about the `redis-cli` interactive mode, see [Interactive Mode](#) in the Redis documentation.

## Troubleshooting Errors

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



## Failed Installation

1. Certificate issues: The on-demand broker (ODB) requires valid certificates. Ensure that your certificates are valid and generate new ones if necessary. To generate new certificates, contact [Pivotal Support](#).
2. Deploy fails: Deploys can fail for a variety of reasons. View the logs using Ops Manager to determine why the deploy is failing.
3. [Networking problems](#):
  - ◊ Cloud Foundry cannot reach the Redis for Pivotal Platform service broker
  - ◊ Cloud Foundry cannot reach the service instances
  - ◊ The service network cannot access the BOSH director
4. [Register broker errand](#) fails.
5. The smoke test errand fails.
6. Resource sizing issues: These occur when the resource sizes selected for a given plan are less than the Redis for Pivotal Platform service requires to function. Check your resource configuration in Ops Manager and ensure that the configuration matches that recommended by the service.
7. Other service-specific issues.

## Cannot Create or Delete Service Instances

If developers report errors such as:

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

Follow these steps:

1. If the BOSH error shows a problem with the deployment manifest, open the manifest in a text editor to inspect it.
2. To continue troubleshooting, [Log in to BOSH](#) and target the Redis for Pivotal Platform service instance using the instructions on [parsing a Cloud Foundry error message](#).
3. Retrieve the BOSH task ID from the error message and run the following command:

```
bosh task TASK-ID
```

4. If you need more information, [access the broker logs](#) and use the `broker-request-id` from the error message above to search the logs for more information. Check for:
  - ◊ [Authentication errors](#)
  - ◊ [Network errors](#)
  - ◊ [Quota errors](#)

## Broker Request Timeouts

If developers report errors such as:

```
Server error, status code: 504, error code: 10001, message: The request to the service broker timed out: https://BROKER-URL/v2/service_instances/e34046d3-2379-40d0-a318-d54fc7a5b13f/service_bindings/aa635a3b-ef6d-41c3-a23f-55752f3f651b
```

Follow these steps:

1. Confirm that Cloud Foundry (CF) is [connected to the service broker](#).
2. Check the BOSH queue size:
  1. Log into BOSH as an admin.
  2. Run `bosh tasks`.
3. If there are a large number of queued tasks, the system may be under too much load. BOSH is configured with two workers and one status worker, which may not be sufficient resources for the level of load. Advise app developers to try again once the system is under less load.

## Cannot Bind to or Unbind from Service Instances

### Instance Does Not Exist

If developers report errors such as:

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

Follow these steps:

1. Confirm that the Redis for Pivotal Platform service instance exists in BOSH and obtain the GUID CF by running:

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

2. Using the GUID obtained above, the following BOSH CLI command:

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

If the BOSH deployment is not found, it has been deleted from BOSH. Contact Pivotal support for further assistance.

### Other Errors

If developers report errors such as:

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

To find out the exact issue with the binding process:

1. [Access the service broker logs.](#)
2. Search the logs for the `broker-request-id` string listed in the error message above.
3. Contact Pivotal support for further assistance if you are unable to resolve the problem.
4. Check for:
  - ◆ [Authentication errors](#)
  - ◆ [Network errors](#)

## Cannot Connect to a Service Instance

If developers report that their app cannot use service instances that they have successfully created and bound:

Ask the user to send application logs that show the connection error. If the error is originating from the service, then follow Redis for PCF-specific instructions. If the issue appears to be network-related, then:

1. Check that [application security groups](#) are configured correctly. Access should be configured for the service network that the tile is deployed to.
2. Ensure that the network the Pivotal Application Service tile is deployed to has network access to the service network. You can find the network definition for this service network in the BOSH Director tile.
3. In Ops Manager go into the service tile and see the service network that is configured in the networks tab.
4. In Ops Manager go into the Pivotal Application Service tile and see the network it is assigned to. Make sure that these networks can access each other.

## Upgrade All Service Instances Errand Fails

If the `upgrade-all-service-instances` errand fails, look at the errand output in the Ops Manager log.

If an instance fails to upgrade, debug and fix it before running the errand again to prevent any failure issues from spreading to other on-demand instances.

Once the Ops Manager log no longer lists the deployment as `failing`, [re-run the errand](#) to upgrade the rest of the instances.

## Missing Logs and Metrics

If no logs are being emitted by the on-demand broker, check that your syslog forwarding address is correct in Ops Manager.

1. Ensure you have configured syslog for the tile.
2. Ensure that you have network connectivity between the networks that the tile is using and the syslog destination. If the destination is external, you need to use the [public ip VM extension](#) feature available in your Ops Manager tile configuration settings.
3. Verify that the Firehose is emitting metrics:
  1. Install the `cf nozzle` plugin. For instructions, see the [firehose plugin GitHub](#)

repository.

- To find logs from your service in the `cf nozzle` output, run the following:

```
cf nozzle -f ValueMetric | grep --line-buffered "on-demand-broker/MY-SERVICE"
```

If no metrics appear within five minutes, verify that the broker network has access to the Loggregator system on all required ports.

[Contact Pivotal support](#) if you are unable to resolve the issue.

## Error Messages Logged in Syslog

You can configure Redis for PCF with remote syslog forwarding. For more information, see [Configure Syslog Forwarding](#).

This section helps to troubleshoot the following errors logged in syslog:

- [AOF File Corrupted, Cannot Start Redis Instance](#)
- [Saving Error](#)

### AOF File Corrupted, Cannot Start Redis Instance

#### Symptom

One or more VMs might fail to start the redis server during pre-start with the error message:

```
[ErrorLog-TimeStamp] # Bad file format reading the append only file: make a backup of your AOF file, then use ./redis-check-aof --fix filename
```

#### Explanation

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

#### Solution for Shared-VM instances:

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

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

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

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

### Solution for On-Demand-VM instances:

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

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

4. To SSH out of the service instance and restart it, run the following command:

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

## Saving Error

### Symptom

One of the following error messages is logged:

```
Background saving error
```

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

### Explanation

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

### Solution

To prevent this error, do the following:

1. Ensure the disk is configured to at least 2.5x the VM memory for the on-demand broker and 3.5x the VM memory for `cf-redis-broker`.
2. Check if the AOF is using too much disk space by doing the following:
  1. BOSH SSH into the affected service instance VM.

2. Run `cd /var/vcap/store/redis; ls -la` to list the size of each file.

## Failed Backup

### Symptom

The following error message is logged:

```
Backup has failed. Redis must be running for a backup to run
```

### Explanation

This is logged if a backup is initiated against a Redis server that is down.

### Solution

Ensure that the Redis server being backed up is running. To do this, run `bosh restart` against the affected service instance VM.

## Orphaned Instances

When a service instance is orphaned, one of the following occurs:

- BOSH Director cannot see your service instances but PCF can. See [BOSH Director Cannot See Your Instances](#).
- PCF cannot see your broker or service instances but BOSH Director can. See [PCF Cannot See Your Instances](#).

### BOSH Director Cannot See Your Instances

#### Symptom

BOSH Director cannot see your instances but they are visible on PCF. When you run `cf curl /v2/service_instances`, some service instances are visible that are not visible to the BOSH Director. These orphaned instances can create issues. For example, they might hold on to a static IP address, causing IP conflicts.

#### Explanation

Orphaned instances can occur in the following situations:

- Both PCF and BOSH maintain state. Orphaned instances can occur if the PCF state is out of sync with BOSH. For example, the deployments or VMs have been deprovisioned by BOSH

but the call to update the PCF state failed.

- If a call to deprovision a service instance was made directly to BOSH rather than through the cf CLI.

## Solution

You can solve this issue by doing one of the following:

- **If this is the first occurrence:** Pivotal recommends that you purge instances by running `cf purge-service-instance SERVICE-INSTANCE`.
- **If this is a repeated occurrence:** Contact [Pivotal Support](#) for further help, and include the following:
  - ◊ A snippet of your `broker.log` around the time of the incident
  - ◊ The deployment manifest of failed instances, hiding private information like passwords
  - ◊ Any recent logs that you can recover from the failed service instance

## PCF Cannot See Your Instances

### Symptom

PCF cannot see your broker or service instances. These instances exist but PCF and apps cannot communicate with them.

### Explanation

If you run `cf purge-service-instances` while your service instance or broker still exists, your service instance becomes orphaned.

### Solution

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

To back up your data and create a new service instance, do the following:

1. To retrieve your orphaned service instance GUID, run the following command:

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

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

2. To SSH into your orphaned service instance, run the following command:

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

Where:

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

3. To create an new RDB file, run the following command:

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

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

4. To push the RDB file to your backup location, run the following command:

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

For information about backup locations, see [Configuring Automated Service Backups](#).

5. Create a new service instance with the same configuration of the database you backed up.
6. To retrieve your new service instance GUID, run the following command:

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

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

7. To SSH into your new service instance, repeat step 2 above with the GUID that you retrieved in step 6.
8. To create a new directory in new service instance, run the following command:

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

9. Save the RDB file in `/var/vcap/store/MY-BACKUPS/` to transfer it to the new instance. Replace `MY-BACKUPS` with the name of your backups directory.
10. To verify the RDB file has not been corrupted, run the following command:

```
md5sum RDB-FILE
```

Where `RDB-FILE` is the path to your RDB file.

11. To restore your data, run the following command:

```
sudo /var/vcap/jobs/redis-backups/bin/restore --sourceRDB RDB-FILE
```

Where `RDB-FILE` is the path to your RDB file.

## Failed to Set Credentials in Runtime CredHub

### Symptom



If developers report errors such as:

```
error: failed to set credentials in credential store: The request includes an unrecognized parameter 'mode'. Please update or remove this parameter and retry your request..
error for user: There was a problem completing your request. Please contact your operations team providing the following information: service: p.redis, service-instance-gu
id: , broker-request-id: , operation: bind
```

### Explanation

Your service instances might not be running the latest version of Redis for PCF. You might experience compatibility issues with CredHub if your service instances are running Redis for PCF v1.14.3 or earlier.

### Solution

1. Make sure you have the latest patch version of Redis for PCF installed. For more information about the latest patch, see [Redis for PCF Release Notes](#).
2. Run the `upgrade-all-service-instances` errand to ensure all service instances are running the latest service offering. For how to run the errand, see [Upgrade All Service Instances](#).



**Note:** Running this errand causes a short period of downtime.

## Troubleshooting Components

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

### BOSH Problems

#### Large BOSH Queue

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

Ops Manager currently deploys two BOSH workers to process its queue. Future versions of Ops Manager will let users configure the number of BOSH workers.

### Configuration

#### Service Instances in Failing State

You may have configured a VM / Disk type in tile plan page in Ops Manager that is insufficiently large for the Redis for Pivotal Platform service instance to start. See tile-specific guidance on

resource requirements.

## Authentication

### UAA Changes

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

To resolve this, redeploy the Redis for PCF tile in Ops Manager. This provides the broker with the latest configuration.



**Note:** You must ensure that any changes to UAA credentials are reflected in the Ops Manager `credentials` tab of the Pivotal Application Service tile.

## Networking

Common issues with networking include:

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

### Validate Service Broker Connectivity to Service Instances

To validate connectivity, do the following:

1. To SSH into the Redis for Pivotal Platform service broker, run the following command:

```
bosh -d service-instance_GUID ssh
```

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

### Validate App Access to a Service Instance

Use `cf ssh` to access to the app container, then try connecting to the Redis for Pivotal Platform service instance using the binding included in the `VCAP_SERVICES` environment variable.

## Quotas

### Plan Quota Issues

If developers report errors such as:

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

1. Check your current plan quota.
2. Increase the plan quota.
3. Log into Ops Manager.
4. Reconfigure the quota on the plan page.
5. Deploy the tile.
6. Find who is using the plan quota and take the appropriate action.

### Global Quota Issues

If developers report errors such as:

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

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

## Failing Jobs and Unhealthy Instances

To determine whether there is an issue with the Redis for Pivotal Platform service deployment, inspect the VMs. To do so, run the following command:

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

For additional information, run the following command:

```
bosh instances --ps --vitals
```

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

## Techniques for Troubleshooting

This section contains instructions on:

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

## Parse a Cloud Foundry (CF) Error Message

Failed operations (create, update, bind, unbind, delete) 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 more information on a failed BOSH task, use the `bosh 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.

## Access Broker and Instance Logs and VMs

Before following the procedures below, log into the [cf CLI](#) and the [BOSH CLI](#).

### Access Broker Logs and VMs

You can [access logs using Ops Manager](#) by clicking on the **Logs** tab in the tile and downloading the broker logs.

To access logs using the BOSH CLI, do the following:

1. Identify the on-demand broker (ODB) deployment by running the following command:

```
bosh deployments
```

2. View VMs in the deployment by running the following command:

```
bosh -d DEPLOYMENT-NAME instances
```

3. SSH onto the VM by running the following command:

```
bosh -d service-instance_GUID ssh
```

4. Download the broker logs by running the following command:

```
bosh -d service-instance_GUID logs
```

The archive generated by BOSH or Ops Manager includes the following logs:

Log Name	Description
broker.log	Requests to the on-demand broker and the actions the broker performs while orchestrating the request (e.g. generating a manifest and calling BOSH). Start here when troubleshooting.
broker_ctl.log	Control script logs for starting and stopping the on-demand broker.
post-start.stderr.log	Errors that occur during post-start verification.
post-start.stdout.log	Post-start verification.
drain.stderr.log	Errors that occur while running the drain script.

### Access Service Instance Logs and VMs

1. To target an individual service instance deployment, retrieve the GUID of your service instance with the following cf CLI command:

```
cf service MY-SERVICE --guid
```

2. To view VMs in the deployment, run the following command:

```
bosh -d DEPLOYMENT-NAME instances
```

3. To SSH into a VM, run the following command:

```
bosh -d service-instance_GUID ssh
```

4. To download the instance logs, run the following command:

```
bosh -d service-instance_GUID logs
```

## Run Service Broker Errands to Manage Brokers and Instances

From the BOSH CLI, you can run service broker errands that manage the service brokers and perform mass operations on the service instances that the brokers created. These service broker errands include:

- `register-broker` registers a broker with the Cloud Controller and lists it in the Marketplace.
- `deregister-broker` deregisters a broker with the Cloud Controller and removes it from the Marketplace.
- `upgrade-all-service-instances` upgrades existing instances of a service to its latest installed version.
- `delete-all-service-instances` deletes all instances of service.
- `orphan-deployments` detects “orphan” instances that are running on BOSH but not registered with the Cloud Controller.

To run an errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand ERRAND-NAME
```

For example:

```
bosh -d my-deployment run-errand deregister-broker
```

### Register Broker

The `register-broker` errand registers the broker with Cloud Foundry and enables access to plans in the service catalog. Run this errand whenever the broker is re-deployed with new catalog metadata to update the Cloud Foundry catalog.

Plans with disabled service access are not visible to non-admin Cloud Foundry users, including Org Managers and Space Managers. Admin Cloud Foundry users can see all plans including those with disabled service access.

The errand does the following:

- Registers the service broker with Cloud Controller.
- Enables service access for any plans that have the radio button set to `enabled` in the tile plan page.
- Disables service access for any plans that have the radio button set to `disabled` in the tile plan page.
- Does nothing for any for any plans that have the radio button set to `manual`.

To run the errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand register-broker
```

## Deregister Broker

This errand deregisters a broker from Cloud Foundry.

The errand does the following:

- Deletes the service broker from Cloud Controller
- Fails if there are any service instances, with or without bindings

Use the [Delete All Service Instances errand](#) to delete any existing service instances.

To run the errand, run the following command:

```
bosh -d DEPLOYMENT-NAME run-errand deregister-broker
```

## Upgrade All Service Instances

If you have made changes to the plan definition or uploaded a new tile into Ops Manager, you might want to upgrade all the Redis for Pivotal Platform service instances to the latest software or plan definition.

The `upgrade-all-service-instances` errand does the following:

- Collects all of the service instances the on-demand broker has registered
- For each instance the errand does the following serially
  - ◊ Issues an upgrade command to the on-demand broker
  - ◊ Regenerates the service instance manifest based on its latest configuration from the tile
  - ◊ Deploys the new manifest for the service instance
  - ◊ Waits for this operation to complete, then proceeds to the next instance
- Adds to a retry list any instances that have ongoing BOSH tasks at the time of upgrade
- Retries any instances in the retry list until all are upgraded

If any instance fails to upgrade, the errand fails immediately. This prevents systemic problems from spreading to the rest of your service instances.

To run the errand, do one of the following:

- Select the errand through the Ops Manager UI and have it run when you click **Apply Changes**.
- Run the following command.

```
bosh -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances
```

## Delete All Service Instances

This errand uses the Cloud Controller API to delete all instances of your broker's service offering in every Cloud Foundry org and space. It only deletes instances the Cloud Controller knows about. It does not delete orphan BOSH deployments.



**Note:** Orphan BOSH deployments do not correspond to a known service instance. While rare, orphan deployments can occur. Use the `orphan-deployments` errand to identify them.

The `delete-all-service-instances` errand does the following:

1. Unbinds all apps from the service instances.
2. Deletes all service instances sequentially. Each service instance deletion includes:
  1. Running any pre-delete errands
  2. Deleting the BOSH deployment of the service instance
  3. Removing any ODB-managed secrets from BOSH CredHub
  4. Checking for instance deletion failure, which results in the errand failing immediately
3. Determines whether any instances have been created while the errand was running. If new instances are detected, the errand returns an error. In this case, Pivotal recommends running the errand again.



**Warning:** Use extreme caution when running this errand. You should only use it when you want to totally destroy all of the on-demand service instances in an environment.

To run the errand, run the following command:

```
bosh -d service-instance_GUID delete-deployment
```

## Detect Orphaned Service Instances

A service instance is defined as “orphaned” when the BOSH deployment for the instance is still running, but the service is no longer registered in Cloud Foundry.

The `orphan-deployments` errand collates a list of service deployments that have no matching service instances in Cloud Foundry and return the list to the operator. It is then up to the operator to remove the orphaned BOSH deployments.

To run the errand, run the following command:

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

**If orphan deployments exist**—The errand script does the following:

- Exit with exit code 10
- Output a list of deployment names under a `[stdout]` header
- Provide a detailed error message under a `[stderr]` header

For example:

```
[stdout]
[{"deployment\_name": "service-instance\_80e3c5a7-80be-49f0-8512-44840f3c4d1b"}]
```



```
[stderr]
Orphan BOSH deployments detected with no corresponding service instance in Cloud Foundry. Before deleting any deployment it is recommended to verify the service instance no longer exists in Cloud Foundry and any data is safe to delete.

Errand 'orphan-deployments' completed with error (exit code 10)
```

These details will also be available through the BOSH `/tasks/` API endpoint for use in scripting:

```
$ curl 'https://bosh-user:bosh-password@bosh-url:25555/tasks/task-id/output?type=result' | jq .
{
  "exit_code": 10,
  "stdout": "[{"deployment_name":"service-instance_80e3c5a7-80be-49f0-8512-44840f3c4d1b"}]\n",
  "stderr": "Orphan BOSH deployments detected with no corresponding service instance in Cloud Foundry. Before deleting any deployment it is recommended to verify the service instance no longer exists in Cloud Foundry and any data is safe to delete.\n",
  "logs": {
    "blobstore_id": "d830c4bf-8086-4bc2-8c1d-54d3a3c6d88d"
  }
}
```

**If no orphan deployments exist**—The errand script does the following:

- Exit with exit code 0
- Stdout will be an empty list of deployments
- Stderr will be `None`

```
[stdout]
[]

[stderr]
None

Errand 'orphan-deployments' completed successfully (exit code 0)
```

**If the errand encounters an error during running**—The errand script does the following:

- Exit with exit 1
- Stdout will be empty
- Any error messages will be under stderr

To clean up orphaned instances, run the following command on each instance:



**WARNING:** Running this command may leave IaaS resources in an unusable state.

```
bosh delete-deployment service-instance_SERVICE-INSTANCE-GUID
```

%>

## Get Admin Credentials for a Service Instance

1. [Identify the service deployment by GUID.](#)
2. [Log in to BOSH.](#)
3. Open the manifest in a text editor.
4. Look in the manifest for the credentials.

## Reinstall a Tile

To reinstall a tile in the same environment where it was previously uninstalled:

1. Ensure that the previous tile was correctly uninstalled as follows:
  1. Log in as an admin by running:

```
cf login
```

2. Confirm that the Marketplace does not list Redis for PCF by running:

```
cf m
```

3. Log in to BOSH as an admin by running:

```
bosh log-in
```

4. Display your BOSH deployments to confirm that the output does not show Redis for PCF deployment by running:

```
bosh deployments
```

5. Run the [“delete-all-service-instances”](#) errand to delete every instance of the service.
6. Run the [“deregister-broker”](#) errand to delete the service broker.
7. Delete the service broker BOSH deployment by running:

```
bosh delete-deployment BROKER-DEPLOYMENT-NAME
```

8. Reinstall the tile.

## View Resource Saturation and Scaling

To view usage statistics for any service, do the following:

1. Run the following command:

```
bosh -d DEPLOYMENT-NAME vms --vitals
```

2. To view process-level information, run:

```
bosh -d DEPLOYMENT-NAME instances --ps
```

## Identify a Service Instance Owner

If you want to identify which apps are using a specific service instance from the BOSH deployments name, do the following:

1. Take the deployment name and strip the `service-instance_` leaving you with the GUID.
2. Log in to CF as an admin.
3. Obtain a list of all service bindings by running the following:

```
cf curl /v2/service_instances/GUID/service_bindings
```

4. The output from the above curl gives you a list of `resources`, with each item referencing a service binding, which contains the `APP-URL`. To find the name, org, and space for the app, run the following:
  1. `cf curl APP-URL` and record the app name under `entity.name`.
  2. `cf curl SPACE-URL` to obtain the space, using the `entity.space_url` from the above curl. Record the space name under `entity.name`.
  3. `cf curl ORGANIZATION-URL` to obtain the org, using the `entity.organization_url` from the above curl. Record the organization name under `entity.name`.



**Note:** When running `cf curl` ensure that you query all pages, because the responses are limited to a certain number of bindings per page. The default is 50. To find the next page curl the value under `next_url`.

## Monitor the Quota Saturation and Service Instance Count

Quota saturation and total number of service instances are available through ODB metrics emitted to Loggregator. The metric names are shown below:

Metric Name	Description
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/quota_remaining</code>	global quota remaining for all instances across all plans
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/quota_remaining</code>	quota remaining for a particular plan
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/total_instances</code>	total instances created across all plans
<code>on-demand-broker/SERVICE-NAME-MARKETPLACE/PLAN-NAME/total_instances</code>	total instances created for a given plan



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

## Pivotal Support Articles

The following are [Pivotal Support](#) articles about Redis for PCF:

- [Creating an Empty Services Network when using on-demand Service Tiles for Non-On-Demand Usage Only](#)

- [Pivotal Cloud Foundry Redis full disk scaling issue](#)
- [Pivotal Cloud Foundry Redis tile upgrade issue](#)
- [Pivotal Cloud Foundry Redis Deploy Fails to Complete](#)
- [Pivotal Cloud Foundry 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 'pcf-redos-backup-sgp-intra-test'](#)
- [For service settings on Redis Tile, the VM options checkbox needs to be checked for GCP Environment](#)
- [Removing dedicated-vm Service Instances on CF when already deleted from BOSH](#)
- [Migrating from dedicated-vm service plans to on-demand service plans](#)

[Create a pull request or raise an issue on the source for this page in GitHub](#)

# Application Developer Guide

## Introduction for App Developers



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

This section introduces 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, or Shared-VM plan, see [Using Redis for PCF](#).

## Redis for PCF Services

Redis for PCF v2.0 and later offers On-Demand 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.
- **Shared-VM Service**—Provides support for a number of Redis instances running in a single VM. It is designed for testing and development purposes only, **do not use the Shared-VM service in production environments**. The Shared-VM instances are pre-provisioned by the operator with a fixed number of instances and memory size. App developers can then use one of these pre-provisioned instances.



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

For more information about the plans, see the [On-Demand Service Offering](#) and the [Shared-VM Service Offering](#).

## Getting Started

### Using Redis for PCF with Spring

Spring Cloud Spring Service Connectors connect to Redis for PCF. See the [Redis](#) section in the Spring Cloud Spring Service Connector documentation.

Spring Cloud Cloud Foundry connectors automatically connect to Redis for PCF. See the [Redis](#) section in the Spring Cloud Cloud Foundry Connector documentation.

To view an example Spring app using Redis as a cache with failover, see the [Redis Reference Architectures](#) repository.

### Using Redis for PCF with Steeltoe

Steeltoe Cloud Connectors can connect to Redis for PCF. See the [Steeltoe Cloud Connectors](#) documentation.

To view examples of Steeltoe apps using Redis as a cache with failover, see the [Example Steeltoe App](#) repository in GitHub.



**Warning:** The Steeltoe connector for Redis requires Redis for PCF to support Lua scripting. Check whether the language you are using requires Lua scripting. If it does, contact your operator. By default, Lua scripting is disabled for Redis for PCF, but a PCF operator can change the setting to enable it by selecting the **Lua Scripting** checkbox in each service plan's **On-Demand Plan** configuration pane.

### 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, see <https://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](https://redis.io).

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Quickstart Guide for App Developers



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

Page last updated:

This topic provides some sample apps in various languages to demonstrate how to get Redis for Pivotal Cloud Foundry (PCF) up and running quickly. It also highlights the critical components of the apps that allow them to connect to a Redis instance. Credentials to connect to a Redis for PCF instance are passed to the apps as environment variables under `VCAP_SERVICES`.

Additionally, this topic includes advice for setting up Spring Sessions with Redis for PCF.

## Feedback

If you have feedback about this page, or you want more information (other quickstart guides, sample use cases), please send a message to [Pivotal Cloud Foundry Feedback](#).

## Quickstart Apps

All apps using Redis for PCF must parse and read the Redis for PCF instance credentials from the environment. The credentials are available to the app once a Redis for PCF instance is bound to it and are viewable by typing `$cf env {app_name}`.

Prerequisites for these examples include access to a Marketplace with `p-redis` or `p.redis`.

For reference, `p.redis` refers to the Redis service that provides On-Demand instances and `p-redis` refers to the Redis service that provides Shared-VM instances. Any Redis for PCF service and plan works with the following examples. Available plans and instance types can be viewed with in the Marketplace.

## Quickstart Java App

This is a basic Java app with the capability to get and set keys in Redis and view configuration information. Prerequisites include [Maven](#).

Here we use a cache-small plan of the `p.redis` service, but any `p-redis` or `p.redis` instance works.

```
$ git clone git@github.com:colrich/RedisForPCF-Java-Example.git java_redis_app
$ cd java_redis_app
$ mvn package
$ cf create-service p.redis cache-small redis_instance
$ cf push redis_example_app -p target/RedisExample-0.0.1-SNAPSHOT.jar
$ cf bind-service redis_example_app redis_instance
$ cf restage redis_example_app
```

You can then visit the app in your browser window. The app has three entry points:

- `/` — Gets info about a bound Redis instance
- `/set` — Sets a given key to a given value. E.g., `{APP_URL}/set?kn=somekeyname&kv=valuetoaset`
- `/get` — Gets the value stored at a given key. E.g., `{APP_URL}/get?kn=somekeyname`

In the [application code](#), the snippet where `VCAP_SERVICES` is read and parsed is here:

```
@RequestMapping("/")
public RedisInstanceInfo getInfo() {
    LOG.log(Level.WARNING, "Getting Redis Instance Info in Spring controller...");
```

```

// first we need to get the value of VCAP_SERVICES, the environment variable
// where connection info is stored
String vcap = System.getenv("VCAP_SERVICES");
LOG.log(Level.WARNING, "VCAP_SERVICES content: " + vcap);

// now we parse the json in VCAP_SERVICES
LOG.log(Level.WARNING, "Using GSON to parse the json...");
JsonElement root = new JsonParser().parse(vcap);
JsonObject redis = null;
if (root != null) {
    if (root.getAsJsonObject().has("p.redis")) {
        redis = root.getAsJsonObject().get("p.redis").getAsJsonArray().get(0).getAsJsonObject();
        LOG.log(Level.WARNING, "instance name: " + redis.get("name").getString());
    }
    else if (root.getAsJsonObject().has("p-redis")) {
        redis = root.getAsJsonObject().get("p-redis").getAsJsonArray().get(0).getAsJsonObject();
        LOG.log(Level.WARNING, "instance name: " + redis.get("name").getString());
    }
    else {
        LOG.log(Level.SEVERE, "ERROR: no redis instance found in VCAP_SERVICES");
    }
}

// then we pull out the credentials block and produce the output
if (redis != null) {
    JsonObject creds = redis.get("credentials").getAsJsonObject();
    RedisInstanceInfo info = new RedisInstanceInfo();
    info.setHost(creds.get("host").getString());
    info.setPort(creds.get("port").getAsInt());
    info.setPassword(creds.get("password").getString());

    // the object will be json serialized automatically by Spring web - we just need to return it
    return info;
}
else return new RedisInstanceInfo();
}

```

## Quickstart Node App

This is a basic node app with the capability to get and set keys in Redis and view configuration information. Prerequisites are the `cf cli` and access to a Marketplace with `p-redis` or `p.redis`.

Here we use a `cache-small` plan for the `p.redis` service, but any `p-redis` or `p.redis` instance works.

```

$ git clone git@github.com:colrich/RedisForPCF-Node-Example.git node_redis_app
$ cd node_redis_app
$ cf create-service p.redis cache-small redis_instance
$ cf push redis_example_app
$ cf bind-service redis_example_app redis_instance
$ cf restage redis_example_app

```

You can then visit the app in your browser window. The app has three entry points:



- “/” — Gets info about bound redis instance
- “/set” — Sets a given key to a given value. E.g., {APP\_URL}/set?kn=somekeyname&kv=valuetoaset
- “/get” — Gets the value stored at a given key. E.g., {APP\_URL}/get?kn=somekeyname

In the [application code](#), the snippet where VCAP\_SERVICES is read and parsed is here:

```
// parses the VCAP_SERVICES env var and looks for redis service instances
function getVcapServices() {
  var vcstr = process.env.VCAP_SERVICES;
  if (vcstr != null && vcstr.length > 0 && vcstr != '{}') {
    console.log("found VCAP_SERVICES: " + vcstr)

    var vcap = JSON.parse(vcstr);
    if (vcap != null) {
      if (vcap.hasOwnProperty("p.redis")) {
        console.log("found redis instance: " + vcap["p.redis"][0].name);
        return vcap["p.redis"][0]
      }
      else if (vcap.hasOwnProperty("p-redis")) {
        console.log("found redis instance: " + vcap["p-redis"][0].name);
        return vcap["p-redis"][0]
      }
      else {
        console.log("ERROR: no redis service bound!")
      }
    }
    else {
      console.log("ERROR: no redis service bound!")
    }
  }
  else {
    console.log("ERROR: VCAP_SERVICES does not contain a redis block")
  }
  return null
}

// pulls the necessary connection info out of the parsed VCAP_SERVICES block for
// the redis connection
function getRedisInfo(vcap) {
  var info = {}
  info["host"] = vcap["credentials"]["host"]
  info["port"] = vcap["credentials"]["port"]
  info["password"] = vcap["credentials"]["password"]
  return info
}

// set the port to listen on; for apps in PCF it's important to listen on $PORT (usual
// ly 8000)
app.set('port', (process.env.PORT || 8080))

// this method looks in VCAP_SERVICES for a redis service instance and outputs the
// host / port / password info to the response
app.get('/', function(request, response) {
  console.log("Getting Redis connection info from the environment...")
})
```

```

var vcap = getVcapServices()
if (vcap != null) {
  var info = getRedisInfo(vcap)
  console.log("connection info: " + info.host + " / " + info.port + " / " + info.password)
  response.send("connection info: " + info.host + " / " + info.port + " / " + info.password)
}
else {
  console.log("ERROR: VCAP_SERVICES does not contain a redis block or no redis bound")
  response.send("ERROR: VCAP_SERVICES does not contain a redis block or no redis bound")
}
})

```

## Quickstart Ruby App

This is a basic ruby app with the capability to get and set keys in Redis and view configuration information. Here we use an instance of the `dedicated_vm` service, but any `p-redis` or `p.redis` instance works.

```

$ git clone git@github.com:pivotal-cf/cf-redis-example-app.git ruby_redis_app
$ cd ruby_redis_app
$ cf create-service p-redis dedicated-vm redis_instance
$ cf push redis_example_app --no-start
$ cf bind-service redis_example_app redis_instance
$ cf start redis_example_app"

```

You can then get, set, and delete keys:

```

$ export APP=redis-example-app.my-cloud-foundry.com
$ curl -X PUT $APP/foo -d 'data=bar'
success
$ curl -X GET $APP/foo
bar
$ curl -X DELETE $APP/foo
success

```

In the [application code](#), the method where `VCAP_SERVICES` is read is here:

```

def redis_credentials
  service_name = ENV['service_name'] || "redis"

  if ENV['VCAP_SERVICES']
    all_pivotal_redis_credentials = CF::App::Credentials.find_all_by_all_service_tags(
      ['redis', 'pivotal'])
    if all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
      all_pivotal_redis_credentials && all_pivotal_redis_credentials.first
    else
      redis_service_credentials = CF::App::Credentials.find_by_service_name(service_name)
      redis_service_credentials
    end
  end
end

```

The method where VCAP\_SERVICES is parsed is here:

```
def redis_client
  @client ||= Redis.new(
    host: redis_credentials.fetch('host'),
    port: redis_credentials.fetch('port'),
    password: redis_credentials.fetch('password'),
    timeout: 30
  )
end
```

## Spring Session with Redis for PCF

One common use case of Redis for PCF is management of a user's session information with [Spring Session](#). Spring Session provides an API and implementations with which to manage sessions.

This topic describes how to use Redis for PCF as the backend with Spring Session to manage user session information.

This documentation is adopted from the [Spring Session docs](#) and extends to include instructions for use with Redis for PCF. The document is also adopted from this [Spring Session - Spring Boot guide](#).

## Setting Up Spring Session

### Updating Dependencies

To use Spring Session, update your dependencies to include spring-session-data-redis. The below example is for Maven.

`pom.xml`

```
<dependencies>
  <!-- ... -->
  <dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
    <version>1.3.1.RELEASE</version>
    <type>pom</type>
  </dependency>
  <dependency>
    <groupId>biz.paluch.redis</groupId>
    <artifactId>lettuce</artifactId>
    <version>3.5.0.Final</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.3.4.RELEASE</version>
  </dependency>
</dependencies>
```

### Spring Java Configuration

After adding the required dependencies, we can create our Spring configuration.

The Spring configuration is responsible for creating a Servlet Filter that replaces the `HttpSession` implementation with an implementation backed by Spring Session. Add the following Spring Configuration:

```
@EnableRedisHttpSession (1)
public class Config {

    @Bean
    public LettuceConnectionFactory connectionFactory() {
        return new LettuceConnectionFactory(); (2)
    }
}
```

- 1 The `@EnableRedisHttpSession` annotation creates a Spring Bean with the name of `springSessionRepositoryFilter` that implements `Filter`. The filter is what is in charge of replacing the `HttpSession` implementation to be backed by Spring Session. In this instance Spring Session is backed by Redis.
- 2 We create a `RedisConnectionFactory` that connects Spring Session to the Redis Server. We configure the connection to connect to localhost on the default port (6379) For more information on configuring Spring Data Redis, refer to the [reference documentation](#).

## Java Servlet Container Initialization

Our Spring Configuration created a Spring Bean named `springSessionRepositoryFilter` that implements `Filter`. The `springSessionRepositoryFilter` bean is responsible for replacing the `HttpSession` with a custom implementation that is backed by Spring Session.

In order for our `Filter` to do its magic:

- Spring needs to load our `Config` class.
- We need to ensure that our Servlet Container (i.e. Tomcat) uses our `springSessionRepositoryFilter` for every request.

Fortunately, Spring Session provides a utility class named `AbstractHttpSessionApplicationInitializer`, which helps us confirm that these two requirements are met.

The example below shows how to extend `AbstractHttpSessionApplicationInitializer`:

`src/main/java/sample/Initializer.java`

```
public class Initializer extends AbstractHttpSessionApplicationInitializer { (1)

    public Initializer() {
        super(Config.class); (2)
    }
}
```

The name of our class (Initializer) does not matter. What is important is that we extend `AbstractHttpSessionApplicationInitializer`. Doing this achieves the following:

- It ensures that the Spring Bean by the name `springSessionRepositoryFilter` is registered with our Servlet Container for every request.

- It provides a mechanism to easily ensure that Spring loads our `Config`.

## Configuring Redis for PCF as a Backend

At this stage, Spring Session is now configured to use a Redis instance. To use a Redis for PCF instance, create a `session-replication` tag for it.

```
$ cf update-service INSTANCE_NAME -t session-replication
```

## Other Considerations

The `RedisHttpSessionConfiguration` tries to use the Redis CONFIG command. The CONFIG command is not available due to security recommendations.

This feature can be disabled by exposing `ConfigureRedisAction.NO_OP` as a bean:

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

However, disabling the configuration means that Redis cannot send namespace notifications. This functionality is critical for apps that require `SessionDestroyedEvent` to be fired to clean up resources, such as for WebSocket apps to ensure open WebSockets are closed when the `HttpSession` expires.

If you want a workaround for this use case, send email to [redis-feedback](mailto:redis-feedback).

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Using Redis for PCF



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### 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 [On-Demand Service Offering](#) and the [Shared-VM Service Offering](#).



**Breaking Change:** If you use dedicated VMs, then migrate to the on-demand service plan or back up data before upgrading to Redis for PCF v2.0 to prevent data loss in Redis deployments used as persistent storage.

If you are upgrading from Redis for PCF v1.14.3 or earlier, you must run the `upgrade-all-service-instances` errand after upgrading.

## 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 two Redis services are listed differently in the Marketplace, ensure that 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:
  1. Enter `cf marketplace`.
  2. 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, 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       Redis service to provide pre-provisioned instances configured as a datastore, running on a shared 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:

1. Enter `cf services`.
2. 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 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

#### Shared-VM Service

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



**Note:** Shared-VM services are designed for testing and development purposes. Shared-VMs should not be used in production environments

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

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

where:

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

```
$ cf create-service p-redis shared-vm my-instance
Creating service my-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   0             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.

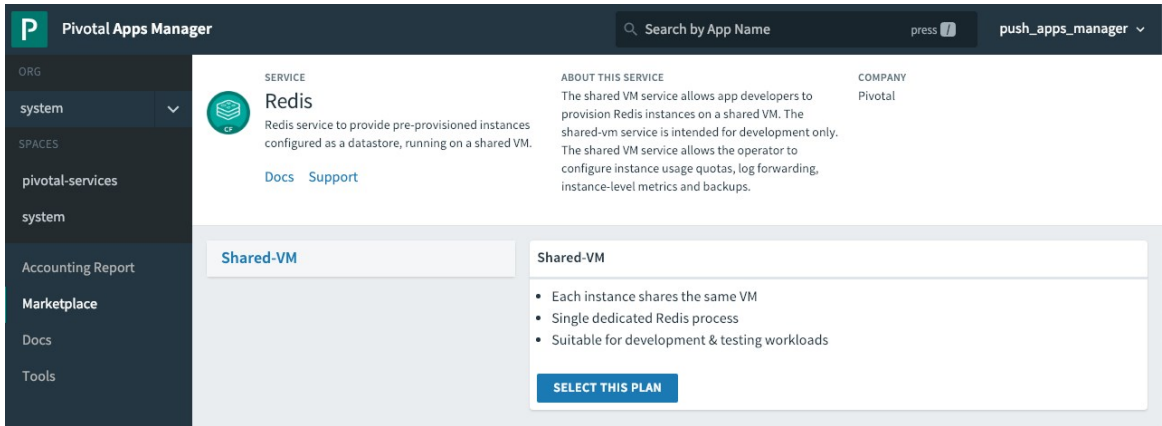
The screenshot shows the Pivotal Apps Manager interface. The top navigation bar includes the Pivotal logo, the text 'Pivotal Apps Manager', a search bar for 'Search by App Name', and a user profile dropdown for 'push\_apps\_manager'. The left sidebar contains a navigation menu with 'ORG' (system), 'SPACES' (pivotal-services, system), 'Accounting Report', 'Marketplace' (selected), 'Docs', and 'Tools'. The main content area is titled 'Marketplace' and contains a search bar and a list of services:

- Redis**: Redis service to provide pre-provisioned instances configured as a datastore, running on a shared VM.
- Redis On-Demand**: Redis service to provide on-demand dedicated instances configured as a cache.
- User Provided Service**: Add an external service to your apps.

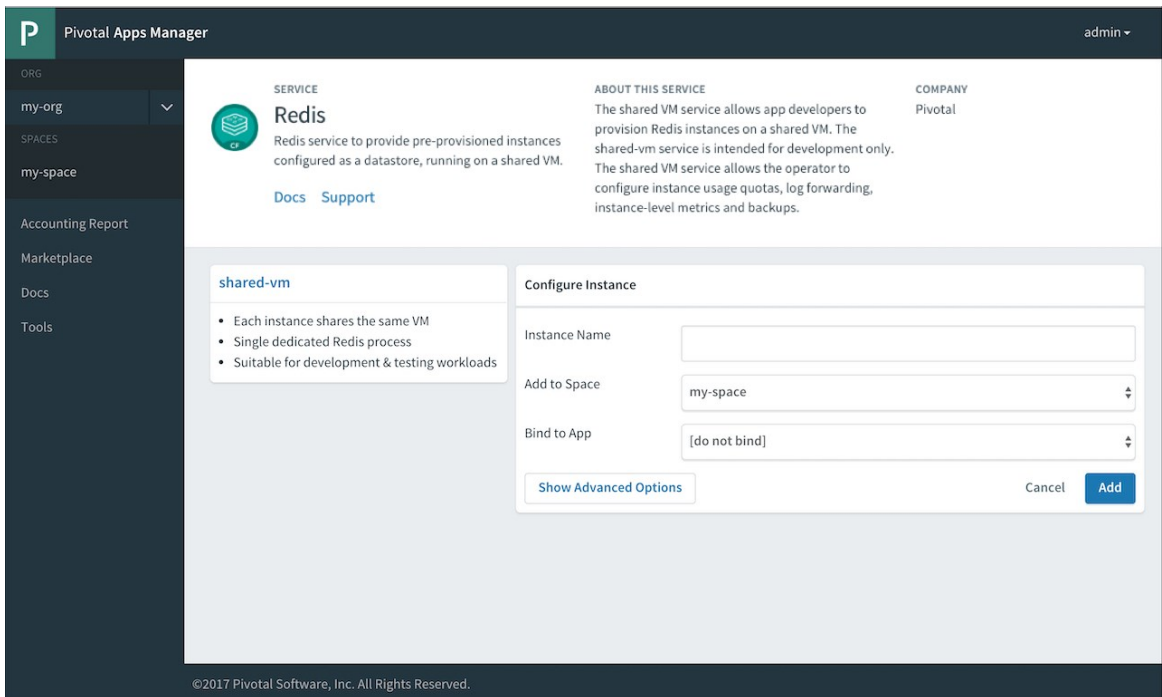


## Shared-VM Service

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



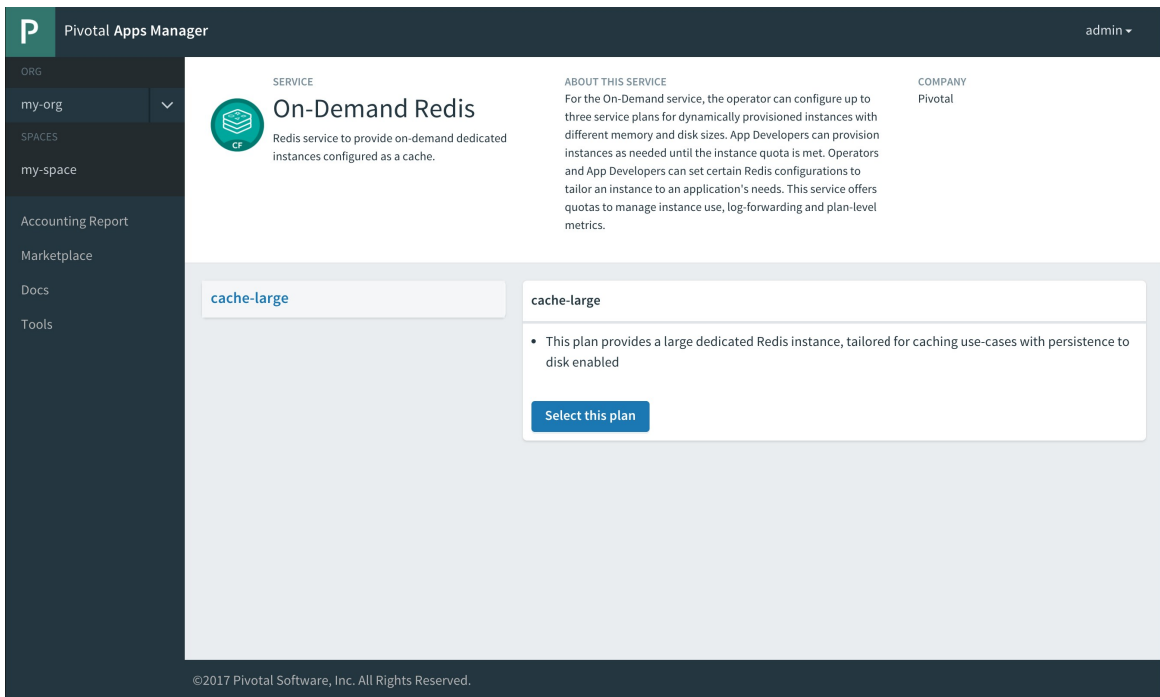
2. Click on the appropriate **Select this plan** button to select the required **Redis Service Plan**.



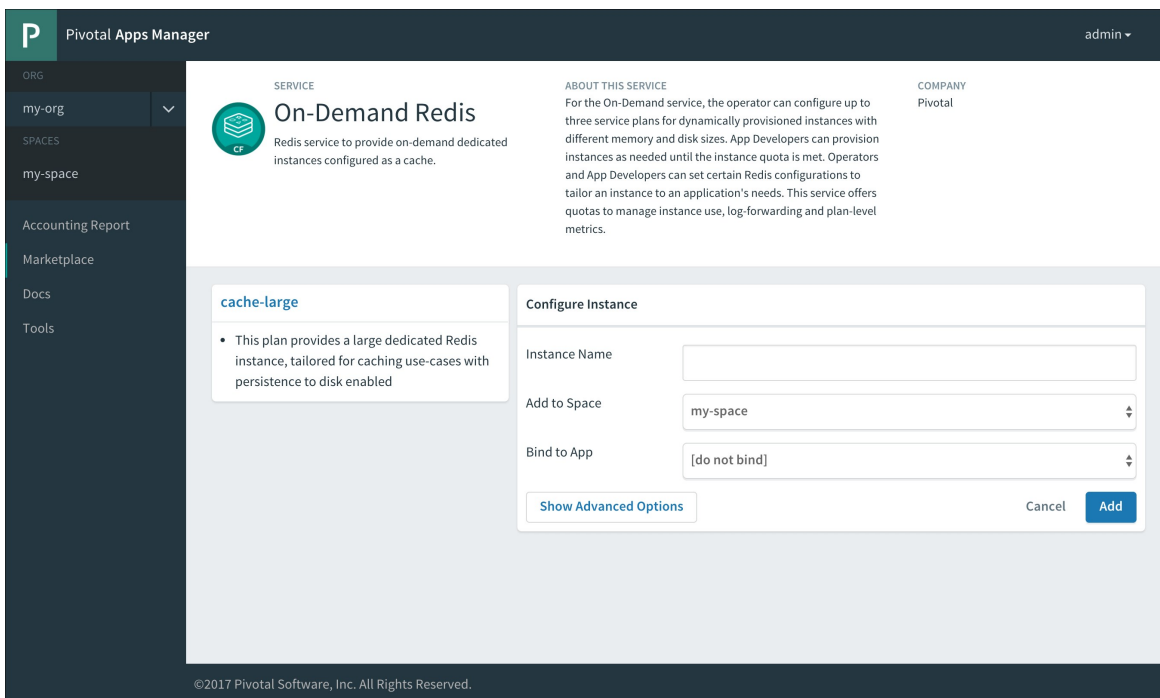
3. In the **Instance Name** field, enter a name that will identify this specific Redis service instance.
4. From the **Add to Space** dropdown, select the space where you or other users will deploy the apps that will bind to the service.
5. Click the **Add** button.

## On-Demand Service

1. Select **Redis On-Demand** 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** dropdown, 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
----------	---------	---------	-------------

<code>maxmemory-policy</code>	<code>allkeys-lru</code>	<code>allkeys-lru, noeviction, volatile-lru, allkeys-random, volatile-ttl, volatile-lfu, allkeys-lfu</code>	Sets the behavior Redis follows when <code>maxmemory</code> is reached
<code>notify-keyspace-events</code>	""	Set a combination of the following characters (e.g., <code>E1g</code> ): K, E, g, \$, l, s, h, z, x, e, A	Sets the keyspace notifications for events that affect the Redis data set
<code>slowlog-log-slower-than</code>	10000	0-20000	Sets the threshold execution time (seconds). Commands that exceed this execution time are added to the slowlog.
<code>slowlog-max-len</code>	128	1-2024	Sets the length (count) of the slowlog queue.

## Customize an On-Demand Instance with the cf CLI



**Note:** Arbitrary parameters are only supported for on-demand service instances. Shared-VM plans do not support the use of CLI commands with arbitrary parameters to configure service instances.

You can customize an instance in two ways:

- While creating the instance, run:  

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

```
cf update-service NAME -c '{"PROPERTY":"SETTING"}'
```

For both scenarios, the `-c` flag requires a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file.

```
$ cf update-service my-instance -c '{"maxmemory-policy":"noeviction"}'
```

You can pass through multiple arbitrary parameters:

```
$ cf update-service my-instance -c '{"maxmemory-policy":"noeviction", "notify-keyspace-events":"E1"}'
```

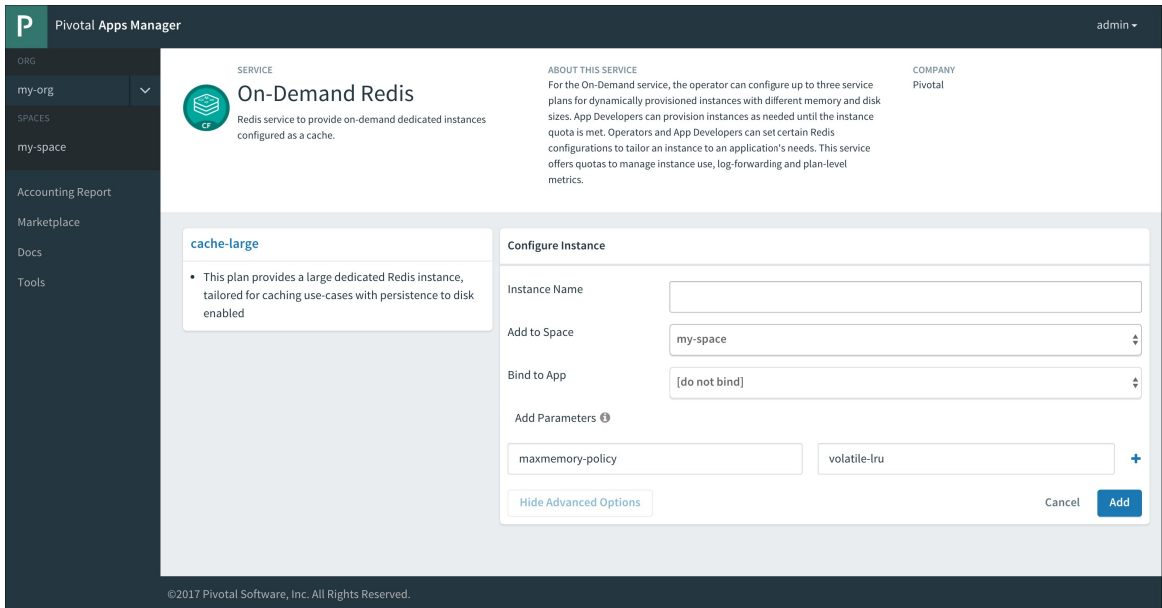
If the update is not successful, an error is displayed with a description of what went wrong. Here is an example where a hyphen is added to the `noeviction` setting.

```
$ cf update-service my-instance -c '{"maxmemory-policy":"no-eviction", "notify-keyspace-events":"E1"}'
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
```

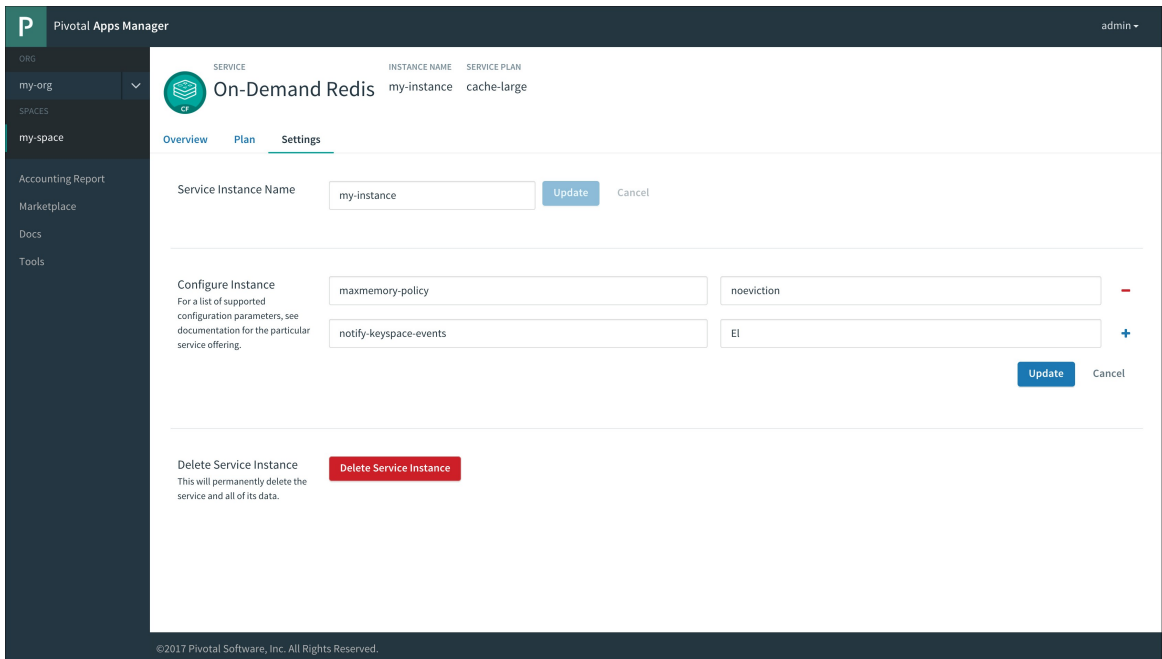
## Customize an On-Demand Instance with the Apps Manager

You can customize an instance in two ways:

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



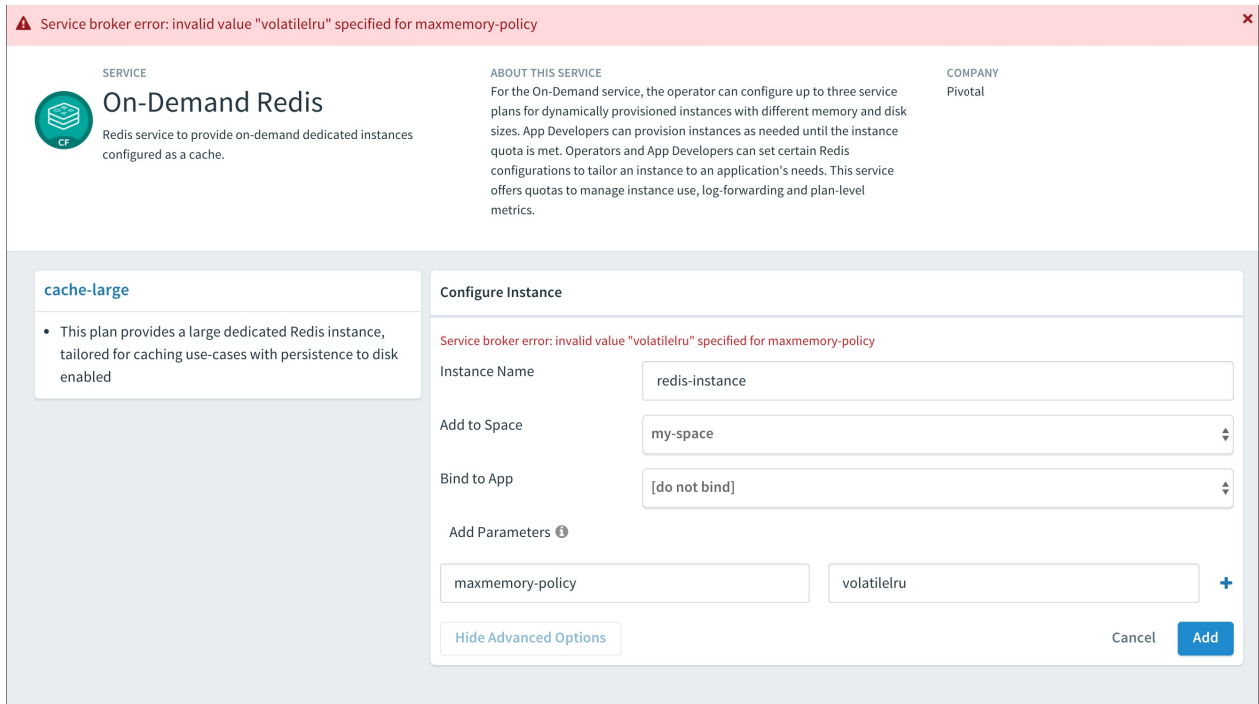
- After creating the instance, navigate to the instance Settings page.



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 displayed with a description of what went wrong. Here is an example where we forgot the hyphen in the `volatile-lru` setting.



## Retrieve the Password for a Redis Service Instance

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, where the user is `admin`:

```
$ 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", # IP or BOSH DNS hostname for ODB instances
  "password": "admin-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

Environment variables are how Cloud Foundry communicates with a deployed app about its

environment. To access the environment variables, bind your app to an instance and run `cf env APP_NAME` from the cf cli.

To access the Redis service from your app:

1. Run `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": "",
      "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.

## Manage Key Eviction for Shared-VM Instances

Shared-VM plans provision Redis instances with a max-memory policy set to `no-eviction`.

It is up to the app developer to manage eviction of keys. The following are a few options for doing this:

- After setting keys, use `EXPIRE` to set key expiry, or use `SETEX` to set key value and expiry at the same time.
- Explicitly delete keys after the app is done using them.
- Add a lua script to delete keys after a specified time period.

## Access Redis Metrics for On-Demand Service Instances

To access metrics for Redis for Pivotal PCF service instances, you can use Loggregator's Log Cache feature with the Log Cache CLI plugin. Log Cache is enabled by default in Pivotal Application Service v2.2 and later.

To access metrics for on-demand service instances, do the following:

1. To install the cf CLI plugin, run the following command:

```
cf install-plugin -r CF-Community "log-cache"
```

2. To access metrics for a service instance, run the following command:

```
cf tail SERVICE-INSTANCE-NAME
```

Where `SERVICE-INSTANCE-NAME` is the name of your service instance.

For example:

```
$ cf tail my-instance
Retrieving logs for service my-instance in org system / space pivotal-services
as admin...
2018-07-03T09:54:14.84+0100 [my-instance] GAUGE info.clients.blocked_clients:0
.000000 metric info.clients.client_biggest_input_buf:0.000000 metric ...
```

For more information about the metrics output, see [Redis KPIs](#).

For more information about how to enable Log Cache and about the `cf tail` command, see [Enable Log Cache](#).

## Sharing a Redis Instance with Another Space

Sharing a service instance allows apps in different spaces to access the same Redis instance. Tile operators must enable this behavior and a cf admin must turn it on. For more information about this feature, see [Sharing Service Instances](#) in the Cloud Foundry documentation.

### Share a Redis Service Instance

To share an instance, run the following command:

```
cf v3-share-service REDIS-SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```

Where:

- `REDIS-SERVICE-INSTANCE` is the name of the Redis instance.
- `OTHER-SPACE` is the name of the other space you want to share this instance with.
- `OTHER-ORG` is the name of another org you want to share this instance with (optional).

### Unshare a Redis Service Instance



**WARNING:** Redis only has one password and password rotation does not occur on unshare. After unsharing a service, any bound apps continue to have access to the Redis instance until the apps are restaged.

To unshare an instance, run the following command:

```
cf v3-unshare-service REDIS-SERVICE-INSTANCE -s OTHER-SPACE [-o OTHER-ORG]
```



Where:

- `REDIS-SERVICE-INSTANCE` is the name of the Redis instance.
- `OTHER-SPACE` is the name of the other space you want to share this instance with.
- `OTHER-ORG` is the name of another org you want to share this instance with (optional).

## Delete a Redis Instance

When you delete a Redis service instance, all apps that are bound to that service are automatically unbound and any data in the service instance is cleared.

### Delete a Redis 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

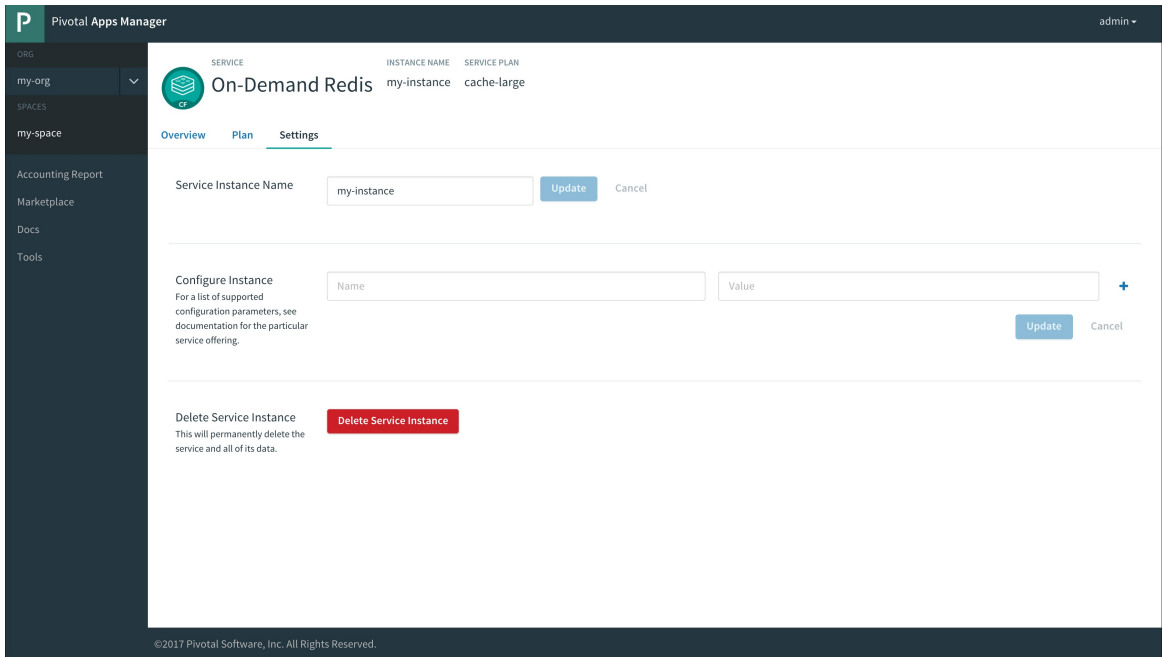
Really delete the service my-redis-instance?> y
Deleting service my-redis-instance in org system / space apps-manager as admin.
..
OK
```

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**.
-



- 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
```

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Troubleshooting Instances

**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

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

## Debugging Using the CF CLI

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

To view the...	Command
API endpoint, org, and space	<code>cf target</code>
Service offerings available in the targeted org and space	<code>cf marketplace</code>
Apps deployed to the targeted org and space	<code>cf apps</code>
Service instances deployed to the targeted org and space	<code>cf services</code>

GUID for a given service instance	<code>cf service SERVICE-INSTANCE --guid</code>
Service instance or application logs	<code>cf tail SERVICE-INSTANCE/APP</code>

## 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 more information on a failed BOSH task, use the

`bosh 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     cache-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.
5. **Redis for PCF v1.14 and later:** Run `cf tail SERVICE-INSTANCE-NAME/APP-NAME` to retrieve the service instance of the app logs if the Log Cache CLI plugin is enabled. See [Log Cache CLI plugin](#).

## Retrieve the Password for a Redis Service Instance

If you want to access the Redis server for troubleshooting, you can find a Redis service instance password by creating a new service key.



**Note:** Pivotal recommends that you use this key for troubleshooting only, and that you delete the key after troubleshooting by using the command `cf delete-service-key SERVICE-INSTANCE KEY-NAME`.

For instructions on how to retrieve the password, see [\[Retrieve the Password for a Redis Service Instance\]\(./using.html#call\)](#). `### Error: Failed to Set Credentials in Credential Store`

If you encounter:

```
error: failed to set credentials in credential store:
The request includes an unrecognized parameter 'mode'.
Please update or remove this parameter and retry your request.
```

after upgrading from PCF 2.3 to 2.4, it is likely because one or more installed on-demand brokers were not restarted during the upgrade. To clear the cached server version and enable your on-

demand brokers to communicate with CredHub v2, do the procedure in [Run BOSH Restart on Your On-Demand Service Brokers](#).

## Error Messages from the Redis Client

Certain errors are returned to the Redis client instead of being recorded in the logs. The Redis protocol represents errors as simple strings beginning with a `-` character.

This section helps to troubleshoot the following errors:

- [Maximum Number of Clients Reached](#)
- [Maxmemory Limit Reached](#)
- [Error When Running the Save Command](#)

### Maximum Number of Clients Reached

#### Symptom

You receive the following error:

```
-ERR max number of clients reached
```

#### Explanation

This is usually caused by apps opening multiple client connections to Redis.

#### Solution

Share or pool Redis connections within an app. Redis for PCF configures Redis to accept 10000 client connections. This can be confirmed by running the `INFO` command using the Redis CLI.

### Maxmemory Limit Reached

#### Symptom

You receive the following error:

```
-OOM command not allowed when used memory > 'maxmemory'.
```

#### Explanation

This occurs when the Redis server has reached its `maxmemory` limit.

#### Solution

Consider changing your `maxmemory-policy`. You can update this using the `cf update-service` parameters. For how to do this, see [Customize an On-Demand Service Instance](#).

## Error When Running the Save Command

### Symptom

You receive the following error message when running `redis-cli SAVE` or issuing the save command using another Redis client:

```
-ERR
```

### Explanation

This might occur when the Redis server's disk is full.

### Solution

A more informative message might be logged in the syslog. For more information, see [Syslog Errors](#).

## Unknown Command Error

### Symptom

You receive the following error message when running `redis-cli COMMAND` or issuing a command using another Redis client:

```
-ERR unknown command
```

### Explanation

For security reasons, certain commands such as `CONFIG`, `SAVE`, and `BGSAVE` are not available by default.

### Solution

Talk to your operator about the availability of the command.

## 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 expedite troubleshooting, if possible, 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.

[Create a pull request or raise an issue on the source for this page in GitHub](#)

## Sample Redis Configuration



**Warning:** Redis for PCF v2.0 is no longer supported because it has reached the End of General Support (EOGS) phase as defined by the [Support Lifecycle Policy](#). To stay up to date with the latest software and security updates, upgrade to a supported version.

### Page last updated:

The following is the default `redis.conf` file from an on-demand plan instance:

```
daemonize yes
pidfile /var/vcap/sys/run/redis.pid
port 6379
requirepass 1a1a2bb0-0ccc-222a-444b-1e1e1e1e2222

# Logging
logfile /var/vcap/sys/log/redis/redis.log
syslog-enabled yes
syslog-ident redis-server
syslog-facility local0

# Persistence
dbfilename dump.rdb
dir /var/vcap/store/redis
appendonly no
appendfilename appendonly.aof
save 900 1
save 300 10
save 60 10000

# Arbitrary Parameters
maxmemory-policy allkeys-lru
slowlog-log-slower-than 10000
slowlog-max-len 128
notify-keyspace-events ""

# Plan Properties:
timeout 3600s
tcp-keepalive 60
maxclients 10000
rename-command EVAL "EVAL"
rename-command EVALSHA "EVALSHA"

# Command Masking
rename-command CONFIG "A-B-Ab1AZec_-AaC1A2bAbB22a_a1Baa"
rename-command SAVE "SAVE"
rename-command BGSAVE "BGSAVE"
rename-command DEBUG ""
rename-command SHUTDOWN ""
rename-command SLAVEOF ""
rename-command SYNC ""
maxmemory 1775550873
```

[Create a pull request or raise an issue on the source for this page in GitHub](#)