Redis for PCF®

Version 1.7

User's Guide

© Copyright Pivotal Software Inc, 2013-2018
Table of Contents

Table of Contents .......................................................... 2
Redis for PCF ..................................................................... 3
Overview of Redis for PCF v1.7 ............................................. 5
Release Notes ................................................................... 7
Redis for PCF 1.6 Architecture and Lifecycle ......................... 11
Redis for PCF Recommended Usage and Limitations ................ 15
Redis for PCF Security ...................................................... 17
Best Practices for Operating Redis for PCF ......................... 18
Installing and Upgrading Redis for PCF ............................... 19
Manual Backup and Restore of Redis for PCF ....................... 32
Monitoring Redis for PCF ................................................. 39
Troubleshooting Redis for PCF .......................................... 43
For App Developers .......................................................... 45
Using Redis for PCF ....................................................... 46
Sample Redis Configuration .............................................. 49
Redis for PCF

**Note: Redis for PCF v1.7 is no longer supported.** The support period for v1.7 has expired. To stay up-to-date with the latest software and security updates, upgrade to Redis for PCF v1.10 or later.

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

This documentation:

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

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

Product Snapshot

The following table provides version and version-support information about Redis for PCF:

<table>
<thead>
<tr>
<th>Element</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>v1.7.8</td>
</tr>
<tr>
<td>Release date</td>
<td>January 2, 2018</td>
</tr>
<tr>
<td>Software component version</td>
<td>Redis OSS v3.2.11</td>
</tr>
<tr>
<td>Compatible Ops Manager version(s)</td>
<td>v1.8.0, v1.9.0, and v1.10.0</td>
</tr>
<tr>
<td>Compatible Elastic Runtime version(s)</td>
<td>v1.8.0, v1.9.0, and v1.10.0</td>
</tr>
<tr>
<td>IaaS support</td>
<td>AWS, Azure, GCP, OpenStack, and vSphere</td>
</tr>
<tr>
<td>IPsec support</td>
<td>Yes</td>
</tr>
</tbody>
</table>

About Upgrading to the Latest Version

Consider the following compatibility information before upgrading Redis for PCF.

For more information, see the Product Compatibility Matrix.

<table>
<thead>
<tr>
<th>Ops Manager Version</th>
<th>Supported Upgrades from Imported Redis Installation</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.5.x, v1.6.x</td>
<td>v1.4.0 – v1.4.3</td>
<td>v1.4.4 – latest v1.4.x</td>
</tr>
<tr>
<td></td>
<td>v1.4.4 – latest v1.4.x</td>
<td>v1.4.4 – latest v1.4.x</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2018
For more information about… | See…
--- | ---
product compatibility | Product Version Matrix [🔗]
a particular version of Redis for PCF | Release Notes
how to upgrade Redis for PCF | Upgrading Redis for PCF
how to use Redis | Redis Documentation [🔗]

Feedback

Please provide any bugs, feature requests, or questions to the Pivotal Cloud Foundry Feedback list.
Overview of Redis for PCF v1.7

This topic describes the significant new features in Redis for PCF v1.7. This topic also presents a checklist that you can use to decide if Redis for PCF is ready to meet your business requirements.

Introduction

Redis for PCF v1.7 focuses on enhanced backup and restore capabilities. Operators can configure a schedule for backups, as well as several options for where to put the backup artifacts. The release also includes scripts that can be run by the operator to back up and restore dedicated-vm or shared-vm service instances. Redis for PCF 1.7 is configured as a datastore, if you wish to use Redis as a cache, please upgrade to Redis for PCF 1.8.

New in This Release

The following features are new in Redis for PCF v1.7

- **A choice of destinations for backup artifacts** — Redis for PCF v1.7 can send backup artifacts to AWS S3, SCP, Azure or Google Cloud Storage
- **Scripts for manual backup and restore** — Redis for PCF v1.7 includes scripts for backing up and restoring dedicated and shared VM instances.
- **Operator configured backup schedule** — The operator can configure a cron schedule for triggering backups.

Known Issues

- Upgrade from 1.6 sometimes fails at cf-redis-broker. Clicking apply changes again will successfully complete upgrade.
- Backup configuration is not preserved when upgrading from PCF Redis 1.5 or 1.6; if backing up to AWS S3, operators will need to re-enter S3 configuration.
- The service broker and all shared-vm service instances are unavailable while the manual restore script is being run.
- If the manual restore script fails, the Redis service instance that is being restored to is left in an unusable state.
- The manual restore script can time out on very large files, e.g. larger than 60GB.
- The Redis broker vm instance listens on port 12350 instead of port 80. Any vms that need to communicate to the broker (e.g. broker-registrar, cloud_controller) should have this port open. Please ensure you have the appropriate firewall settings.

Enterprise-Ready Checklist

Review the following table to determine if Redis for PCF v1.7 has the features needed to support your enterprise.

<table>
<thead>
<tr>
<th>Plans and Instances</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated and shared plans</td>
<td>Redis for PCF provides both dedicated VM and shared VM plans.</td>
</tr>
<tr>
<td>Customizable plans</td>
<td>For the dedicated VM plan, the operator can customize the VM and disk size.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Installation and Upgrades</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product upgrades</td>
<td>Redis for PCF can be upgraded from v1.5 and v1.6 tiles</td>
</tr>
<tr>
<td>Deployment Smoke Tests</td>
<td>Redis for PCF installation and upgrade runs a post deployment BOSH errand that validates basic Redis operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance and Backups</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational Monitoring and Logging</td>
<td>Redis for PCF v1.7 provides metrics for health monitoring and syslog redirection to external log ingestors.</td>
</tr>
<tr>
<td>Backup and Restore</td>
<td>Redis for PCF v1.7 provides automatic backups on a configurable schedule to a variety of destinations, as well as scripts for backup and restore of service instances.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scale and Availability</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>Redis for PCF has been tested with 60GB of data</td>
</tr>
<tr>
<td>Ability to Scale Up / Down</td>
<td>Operators can scale VMs up, but not down</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2018
Down
Rolling Deployments
Redis for PCF does not support rolling deployments because it is single node; the service is unavailable during upgrades.

AZ Support
Assigning multiple AZs to Redis jobs does not guarantee high availability.

Encryption

About Multiple AZs in Redis for PCF

More Information

About Multiple AZs in Redis for PCF v1.7
Redis for PCF v1.7 supports configuring multiple AZs. However, assigning multiple AZs to Redis jobs does not guarantee high availability.

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

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

<table>
<thead>
<tr>
<th>For more information about…</th>
<th>See…</th>
</tr>
</thead>
<tbody>
<tr>
<td>the v1.7 releases</td>
<td>Release Notes</td>
</tr>
</tbody>
</table>
Release Notes

Redis for PCF v1.7.8

January 2, 2018

Compatibility

This tile uses the following releases: Service Backup 18.1.3 and Service Metrics 1.5.8.

See the change to metrics configuration introduced in v1.7.2.

Features

No change

Bug Fixes

No change

Known Issues

No change

Redis for PCF v1.7.7

November 15, 2017

Compatibility

This release is compatible with stemcell line 3445.

See the change to metrics configuration introduced in v1.7.2.

Features

No change

Bug Fixes

No change

Known Issues

No change. Redis backups to AWS S3 are still limited to standard region.

Redis for PCF v1.7.6

October 27, 2017

Compatibility

This release is compatible with stemcell line 3468.

This release uses OS Redis 3.2.11.

See the change to metrics configuration introduced in v1.7.2.

Features

No change
Bug Fixes
No change

Known Issues
This stemcell line was not intended to be supported. Redis Backups to AWS S3 still limited to standard region.

Redis for PCF v1.7.5
April 27, 2017

Compatibility
Bump stemcell line to 3363.

See the change to metrics configuration introduced in v1.7.2.

Features
Implemented Redis 3.2.8

Adjusted smoketests to be more resilient to unresponsive factors external to Redis, which will reduce false failures.

Bug Fixes
No change

Known Issues
No change. Redis Backups to AWS S3 still limited to standard region.

Redis for PCF v1.7.4
April 5, 2017

Compatibility
No change. See the change to metrics configuration introduced in v1.7.2.

Features
No change

Bug Fixes
Fix an issue with process watcher being restarted every two/three hours.

Known Issues
No change. Redis Backups to AWS S3 still limited to standard region.

Redis for PCF v1.7.3
March 6, 2017

Compatibility
No change. See the change to metrics configuration introduced in v1.7.2.

Features
No change

Bug Fixes

Fix an issue with Redis being restarted every three hours. This issue was introduced in Redis for PCF v1.7.1 and also exists in 1.7.2.

Redis for PCF v1.7.2

Known Issues

No change. Redis Backups to AWS S3 still limited to standard region.

February 16, 2017

Compatibility

In order for Redis for PCF v1.7 to work with PCF v1.8 and later, the operator needs to be able to configure whether TLS is turned on or off for metrics communications. Redis for PCF v1.7.2 adds the ability for the operator to turn on/off TLS communications for metrics, via a checkbox on the metrics configuration page in Ops Manager. You should configure this checkbox for different versions of PCF as follow:

- **PCF v1.8**: This checkbox must be checked for metrics to be emitted to the Firehose.
- **PCF v1.9**: Metrics communications are secure by default but will work in a non-secure mode.
- **PCF v1.10 and later**: Metrics communications are secure by default so this checkbox must be unchecked for metrics to be emitted to the Firehose.

Features

No change

Bug Fixes

No change

Known Issues

In order to enable TLS communications for metrics, the addition of a consul agent to all instances was required. This means there are now new open ports required. Please make changes to any firewall rules to allow outbound traffic from all Redis vm instances on port 8301.

Redis Backups to AWS S3 still limited to standard region.

Redis for PCF v1.7.1

January 25, 2017

Compatibility

No change

Features

No change

Bug Fixes

Fix issue with migration of backup configuration when upgrading from Redis for PCF v1.6.

Fix issue with the service broker and shared vm instances being unavailable while running the restore script on a shared VM instance.

Fix issue with very large files being restored.

Resolved Security Issues

Upgrade to Golang v1.7.4
Known Issues

No change. Redis Backups to AWS S3 still limited to standard region.

Redis for PCF v1.7.0

December 22, 2016

Known Issues

Redis Backups to AWS S3 Limited to Standard Region

Backups are only sent to AWS S3 buckets that have been created in the US Standard region, "us-east-1."

Release notes here.
Redis for PCF 1.6 Architecture and Lifecycle

How Redis for PCF Configures Redis

Redis for PCF configures Redis in the following ways. These configurations cannot be changed.

- Redis is configured with a maxmemory-policy of no-eviction. This policy means that when the memory is full, the service does not evict any keys or perform any write operations until memory becomes available.
- Persistence is configured for both RDB and AOF. The default maximum number of connections, maxclients, is set at 10000 but this number is adjusted by Redis according to the number of file handles available.
- Replication and event notification are not configured.

A sample `redis.conf` from a Dedicated-VM plan instance can be viewed on Sample Redis Configuration.

Service Plans

Redis for PCF offers Dedicated-VM and Shared-VM plans.

Shared-VM Plan

- This plan deploys a Redis instance inside the service broker VM.
- You can disable this plan by setting the `Max instances limit` on the `Shared-VM plan` tab in OpsManager to 0.
- The maximum number of instances can be increased from the default 5 to a value of your choosing. If you increase the number of instances that can be run on this single VM, you should consider increasing the resources allocated to the VM. In particular RAM and CPU. You can overcommit to a certain extent, but may start to see performance degradations.
- You can also increase the maximum amount of RAM allocated to each Redis process (service instance) that is running on this VM
- If you decrease the service instance limit, any instances that are running where the count is now greater than the limit are not terminated. They are left to be removed naturally, until the total count drops below the new limit you cannot create any new instances. For example if you had a limit of 10 and all were used and reduced this to 8, the two instances will be left running until you terminate them yourself.

Dedicated-VM Plan

- This plan deploys the operator-configured number of dedicated Redis VMs alongside the service broker VM.
- These instances are pre-provisioned during the deployment of the tile from OpsManager into a pool. The VMs are provisioned and configured with a Redis process ready to be used when an instance of the Dedicated-VM plan is requested.
- A default deployment will provision 5 instances of the Dedicated-VM plan into the pool. This number can be increased on the `Resource Config` tab in Ops Manager, either in the initial deployment, or subsequently thereafter. The number of VMs cannot be decreased once deployed.
- When a user provisions an instance, it is marked as in use and taken out of the pool.
- When a user deprovisions an instance, the instance is cleansed of any data and configuration to restore it to a fresh state and placed back into the pool, ready to be used again.
- You can disable this plan by setting the number of instances of the `Dedicated node` job in Ops Manager to 0.

Redis for PCF Architecture

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

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.
Create service broker VM

Create requested number of dedicated-VM instances

Broker registers as available in marketplace

Run smoke tests

Service broker ready

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

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

If an instance is available, it is allocated to the requester's org and space

$ cf create-service p-redis dedicated-vm mydedicatedinstance

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

$ cf bind-service my- application mysharedinstance

Redis credentials removed from application's VCAPSERVICES environment variable

$ cf unbind-service my- application mysharedinstance

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

$ cf delete-service mysharedinstance

The service instance data is flushed and the service instance is returned to the deprovisioned state
Delete Redis
Apply changes

Service broker and all provisioned instances are deleted

Broker deregistrar errand runs cf purge-service

is returned to the prep provisioned pool for reuse
Redis for PCF Recommended Usage and Limitations

Recommended Use Cases

Redis for PCF can be used as a datastore or cache.
It is configured with maxmemory-policy = noeviction and RDB and AOF persistence.

Redis can be used in many different ways, including:

- Key/value store for strings and more complex data structures including Hashes, Lists, Sets, Sorted Sets
- Session cache - persistence enables preservation of state
- Full page cache - persistence enables preservation of state
- Database cache - cache queries
- Data ingestion - because Redis is in memory it can ingest data very quickly
- Message Queues - push, pop, and blocking queue commands.
- Leaderboards/Counting - increments and decrements of sets and sorted sets using ZRANGE, ZADD, ZREVRANGE, ZRANK, INCRBY, INCR
- Pub/Sub - published and subscribe operations - publish, subscribe, unsubscribe

Service Plan Recommended Usage and Limitations

Dedicated-VM Plan

- Each dedicated VM plan instances is deployed to its own VM and is suitable for production workloads.
- The number of Dedicated-VM plan instances available to developers is set by the operator. Configurations of up to 100 Dedicated-VM plan instances have been tested.
- No ability to change the Redis configuration. The CONFIG command is disabled.
- Cannot scale down the number of VMs on the plan once deployed.
- Cannot scale down the size of VMs on the plan once deployed (this protects against data loss).
- The default maximum number of connections, maxclients, is set at 10000 but this number is adjusted by Redis according to the number of file handles available.

Shared-VM Plan

- The Shared-VM plan does not manage ‘noisy neighbor’ problems so it is not recommended for production apps.
- 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. Please see Configuring Service Plans for more detail.
- It cannot be scaled beyond a single virtual machine.
- The following commands are disabled: CONFIG, MONITOR, SAVE, BGSAVE, SHUTDOWN, BGREWRITEAOF, SLAVEOF, DEBUG, and SYNC.
- Constraining CPU and/or disk usage is not supported.
- The default maximum number of connections, maxclients, is set at 10000 but this number is adjusted by Redis according to the number of file handles available.

Availability Using Multiple AZs

Redis for PCF 1.6 supports configuring multiple availability zones but this configuration does not provide high availability.
Downtime During Redeploys

Redeploying PCF Redis for configuration changes or upgrades will result in Redis being inaccessible to apps for a brief period of time.

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.
Redis for PCF Security

Security

Pivotal recommends that Redis for PCF is run in its own network.

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

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

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

**Best Practices**

Pivotal recommends that operators do the following:

- **Resource Allocation** — Work with app developers to anticipate memory requirements and to configure VM sizes. Redis for PCF is configured by default with small VMs. For information about configuring VM sizes, see [Configure Redis Service Plans](#).
- **Logs** — Configure a syslog output. Storing logs in an external service helps operators debug issues both current and historical.
- **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.

**About Creating Backups of Redis Instances**

You can back up Redis for PCF instances in two way:

- Configure automatic backups to be run for each instance, across both service plans. For information about setting up automatic backups, see [Configure Backups](#).
- Create manual backups of individual instances. For information about how to make manual backups of instances, see [Manual Backup and Restore of Redis for PCF](#).

**About Monitoring Redis for PCF**

**Redis Metrics**

Redis for PCF emits Redis metrics via the firehose. Details [here](#).

**Logging**

Syslog can be forwarded to an external log service.

The following example shows syslog message:

```bash
Nov 15 17:05:01 10.0.24.10 audispd: [job=dedicated-node index=4] node=7bfe8b1b-6cfd-4d33-b704-c9214ce6bb3e type=USER_ACCT msg=audit(1479229501.290:86): pid=6655 uid=0 auid=4294967295 ses=4294967295 msg="/usr/sbin/cron" ext="" user="" admin="" hostname="" addm="" terminal="" cron res="" success"
```

For information about how to set up syslog output, see [Configure Syslog Output](#).

**Smoke Tests**

Redis for PCF has smoke tests that are run as a post-install errand by Ops Manager. Information on what they do is [here](#). They can be run by the operator via

```
bosh run errand smoke-tests
```
Installing and Upgrading Redis for PCF

Installation Steps

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

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

After installing, be sure to:
- Monitor the health and performance of your Redis instances by setting up logging and tracking metrics.
- Configure automatic backups.

Configuring PCF Redis

Configure Redis Service Plans

Select the Redis tile from the Installation Dashboard to display the configuration page and allocate resources to Redis service plans.

Shared-VM Plan

1. Select the Shared-VM Plan tab.

2. Configure these fields:
   - Redis Instance Memory Limit—Maximum memory used by a shared-VM instance
   - Redis Service Instance Limit—Maximum number of shared-VM instances

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

3. Click Save.

4. If you do not want to use the on-demand service, you must make all of the on-demand service plans inactive. Click the tab for each on-demand plan, and select Plan Inactive. See the example in Step 4 of Removing On-Demand Service Plans above.
5. To change the allocation of resources for the Redis broker, click the **Resource Config** tab.

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

Memory Limits for Shared-VM Plans

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

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

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

Here are some examples for setting these limits:

<table>
<thead>
<tr>
<th>Redis Broker VM Total Memory</th>
<th>Redis Instance Memory Limit</th>
<th>Redis Service Instance Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 GB</td>
<td>512 MB</td>
<td>14</td>
</tr>
<tr>
<td>16 GB</td>
<td>256 MB</td>
<td>28</td>
</tr>
<tr>
<td>64 GB</td>
<td>512 MB</td>
<td>56</td>
</tr>
</tbody>
</table>

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

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

Dedicated-VM Plan

1. Select the **Resource Config** tab to change the allocation of resources for the Dedicated Node.

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

Pivotal recommends the persistent disk be set to 3.5x the amount of system RAM.

2. After you have finished configuring resource allocations, click \texttt{Save}.

### Configuring Resources for Dedicated-VM and Shared-VM Plans

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

- The Shared-VM plan is on the \texttt{Redis Broker} resource.
- The Dedicated-VM plan is on the \texttt{Dedicated Node} resource.

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Resource</th>
<th>Instances</th>
<th>CPU</th>
<th>Ram</th>
<th>Ephemeral</th>
<th>Persistent</th>
<th>Static IP</th>
<th>Dynamic IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>Redis Broker</td>
<td>1</td>
<td>2</td>
<td>3072</td>
<td>4096</td>
<td>9216</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Redis</td>
<td>Dedicated Node</td>
<td>5</td>
<td>2</td>
<td>1024</td>
<td>4096</td>
<td>4096</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Redis</td>
<td>Broker Registrar</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>2048</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Redis</td>
<td>Broker De-Registrar</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>2048</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Redis</td>
<td>Compilation</td>
<td>2</td>
<td>2</td>
<td>1024</td>
<td>4096</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Disable Shared and Dedicated VM Plans

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

1. Ensure at least one On-Demand plan is active.

2. Configure the following tabs:

   - **Shared-VM Plan**:
     a. Set \texttt{Redis Service Instance Limit} to 0.
     b. Click \texttt{Save}.

   - **Errands**:
     a. Set \texttt{Broker Registrar} to Off.
     b. Set \texttt{Smoke Tests} to Off.
     c. Set \texttt{Broker Deregistrar} to Off.
     d. Leave all four On-Demand errands On.
     e. Click \texttt{Save}.

   - **Resource Config**:
     a. Decrease \texttt{Redis Broker} Persistent disk type to the smallest size available.
     b. Decrease \texttt{Redis Broker} VM type to the smallest size available.
     c. Set \texttt{Dedicated Node} instances to 0.
     d. Click \texttt{Save}.

### Configuring Secure Communication

Redis for PCF v1.7.2 lets the operator turn on/off TLS communications for metrics, via a \texttt{Use non-secure communication for metrics} checkbox on the metrics configuration page in Ops Manager. Configure this checkbox for different versions of PCF as follows:

- **PCF v1.8**: Select this checkbox to send metrics to the Firehose.
- **PCF v1.9**: Clear this checkbox to send metrics to the Firehose securely, or select it to send metrics insecurely.
PCF v1.10 and later: Clear this checkbox to send metrics to the Firehose and avoid errors.

Setting this checkbox incorrectly can cause a Cannot generate manifest... unknown property "cf_etcd_client_cert" error.

Configure Syslog Output

Pivotal recommends that operators configure a syslog output.

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

2. Click Save.

Creating Backups of Redis Instances

You can configure backups to be run for all instances, across both service plans.

The key features are:
Runs on a configurable schedule

Every instance is backed up, across both service plans

The Redis broker statefile is backed up

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

You can configure AWS S3, SCP, Azure or Google Cloud Storage as your destination

Data from Redis is flushed to disk, before the backup is started by running a `BGSAVE` on each instance

Backups are labelled with timestamp, instance GUID and plan name

Configuration

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

Click on the tile in OpsManager, followed by the `Backups` link on the left hand menu.

S3 backup fields
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Key ID</td>
<td>The access key for your S3 account</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Secret Access Key</td>
<td>The Secret Key associated with your Access Key</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Endpoint URL</td>
<td>The endpoint of your S3 account, e.g. <a href="http://s3.amazonaws.com">http://s3.amazonaws.com</a></td>
<td>Optional, defaults to <a href="http://s3.amazonaws.com">http://s3.amazonaws.com</a> if not specified</td>
</tr>
<tr>
<td>Bucket Name</td>
<td>Name of the bucket you wish the files to be stored in.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Path</td>
<td>Path inside the bucket to save backups to.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Backup timeout</td>
<td>The amount of time, in seconds, that the backup process will wait for the BGSAVE command to complete on your instance, before transferring the RDB file to your configured destination</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Cron Schedule</td>
<td>Backups schedule in crontab format. For example, once daily at 2am is <code>0 2 * * *</code>. Also accepts a pre-defined schedule: any of <code>@yearly</code>, <code>@monthly</code>, <code>@weekly</code>, <code>@daily</code>, <code>@hourly</code>, or <code>@every &lt;time&gt;</code>, where <code>&lt;time&gt;</code> is any supported time string (e.g. <code>1h30m</code>). For more information, see the cron package documentation.</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

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

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

Notes:

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

SCP backup fields
# Configure blob store for Redis backups

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>The username to use for transferring backups to the scp server</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Private Key</td>
<td>The private ssh key of the user configured in Username</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Hostname</td>
<td>The hostname or IP address of the SCP server</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Destination Directory</td>
<td>The path in the scp server, where the backups will be transferred</td>
<td>Mandatory</td>
</tr>
<tr>
<td>SCP Port</td>
<td>The scp port of the scp server</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Cron Schedule</td>
<td>Backups schedule in crontab format. Refer to table for S3 backups for details</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Backup timeout</td>
<td>The amount of time, in seconds, that the backup process will wait for the BGSAVE command to complete on your instance, before transferring the RDB file to the scp server</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2018
PCF Redis uses service account credentials to upload backups to Google Cloud Storage. The service account should have Storage Admin permissions. Please refer to the documentation for details on how to set up a GCP service account.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project ID</td>
<td>GCP Project ID</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Bucket name</td>
<td>Name of the bucket you wish the files to be stored in.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Service account private key</td>
<td>The JSON Secret Key associated with your Service Account. See documentation for details on how to set up service account keys.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Cron Schedule</td>
<td>Backups schedule in crontab format. For example, once daily at 2am is &quot;2 * * * *&quot;. Also accepts a pre-defined schedule: any of @yearly, @monthly, @weekly, @daily, @hourly, or @every , where is any supported time string (e.g. 1h30m). For more information, see the cron package documentation.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Backup timeout</td>
<td>The amount of time, in seconds, that the backup process will wait for the BGSAVE command to complete on your instance, before transferring the RDB file to your configured destination</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
For each backup destination, the field **Backup timeout** causes backups to fail after a configured timeout. Redis’ BGSAVE will continue, but backups will not be uploaded to destinations if this timeout is reached.
Networks, Security, and Assigning AZs

Network Configuration

The following ports and ranges are used in this service:

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Direction and Network</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>8300-8301</td>
<td>tcp</td>
<td>Inbound to CloudFoundry network, outbound from service broker and service instance networks*</td>
<td>Communication between the CF consul_server and consul_agents on Redis deployment; used for metrics</td>
</tr>
<tr>
<td>4001</td>
<td>tcp</td>
<td>Inbound to CloudFoundry network, outbound from service broker and service instance networks*</td>
<td>Used by the Redis metron_agent to forward metrics to the CloudFoundry etcd server</td>
</tr>
<tr>
<td>8</td>
<td>tcp</td>
<td>Outbound from CloudFoundry to the cf-redis-broker service broker network</td>
<td>(Only if using a cf-redis-broker) Access to the cf-redis-broker from the cloud controllers.</td>
</tr>
<tr>
<td>6379</td>
<td>tcp</td>
<td>Outbound from CloudFoundry to any service instance networks</td>
<td>Access to all nodes from the Diego Cell and Diego Brain network(s)</td>
</tr>
<tr>
<td>32768-61000</td>
<td>tcp</td>
<td>Outbound from CloudFoundry to the cf-redis-broker service broker network</td>
<td>From the Diego Cell and Diego Brain network(s) to the service broker VM. This is only required for the shared service plan.</td>
</tr>
<tr>
<td>80 or 443</td>
<td>http or https</td>
<td>Outbound from any service instance networks</td>
<td>Access to the backup blobstore</td>
</tr>
</tbody>
</table>

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

Application Security Groups

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

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

Application Container Network Connections

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

<table>
<thead>
<tr>
<th>Destination</th>
<th>Ports</th>
<th>Protocol</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSIGNED_NETWORK</td>
<td>32768-61000</td>
<td>tcp</td>
<td>Enable application to access shared vm service instance</td>
</tr>
<tr>
<td>ASSIGNED_NETWORK</td>
<td>6379</td>
<td>tcp</td>
<td>Enable application to access dedicated vm service instance</td>
</tr>
</tbody>
</table>

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:

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

Assigning AZs

Assigning multiple AZs to Redis jobs will not guarantee high availability.

All of your Shared-VM instances will run on a single node in just one of the configured availability zones and are therefore not highly available.

Each Dedicated-VM instance could be assigned to any of the configured availability zones, however each instance still operates as a single node with no
Validating Installation

Smoke tests

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

Upgrading Redis for PCF

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

The upgrade paths are detailed [here](#) for each released version.

To upgrade the product:

- The Operator should download the latest version of the product from [Pivotal Network](#).
- Upload the new .pivotal file to Ops Manager.
- Upload the stemcell associated with the update *(if required)*
- Update any new mandatory configuration parameters *(if required)*
- Press “Apply changes” and the rest of the process is automated

During the upgrade deployment each Redis instance will experience a small period of downtime as each Redis instance is updated with the new software components. This downtime is because the Redis instances are single VMs operating in a non HA setup. The length of the downtime depends on whether there is a stemcell update to replace the operating system image or whether the existing VM can simply have the redis software updated. Stemcells updates incur additional downtime while the IaaS creates the new VM while 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. This is explicitly tested for during our build and test process for a new release of the product.

Release policy

When a new version of Redis is released we aim to release a new version of the product containing this soon after.

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

To uninstall Redis for PCF, click on the trashcan icon in the lower right hand corner of the PCF Redis tile in the PCF Ops Manager Installation dashboard. Confirm deletion of the product and click apply changes.
Manual Backup and Restore of Redis for PCF

Triggering a Manual Backup

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

- **Follow these steps** to log into your Ops Manager installation and target the Redis tile deployment.
- **Identify the VM which holds your instance by running `bosh vms`**.
  - For the `shared-vm` plan this will be the job name containing `cf-redis-broker`. Running `manual-backup` will back up all of the shared-vm instances and the broker state in the `statefile.json` file.
  - For the `dedicated-vm` plan this will be the job name containing `dedicated-node`. Running `manual-backup` will back up the Redis dump.rdb file for that dedicated-vm instance.
  - You can identify the exact node for your `dedicated-vm` service instance by comparing the IP Address from your application bindings.

An example output from `bosh vms`:

```plaintext
Deployment - p-redis-9adbfcff0493b5e0361
Director task 129
Task 129 done

<table>
<thead>
<tr>
<th>Job/Index</th>
<th>State</th>
<th>Resource Pool</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>cf-redis-broker-partition-default-az-gw/0</td>
<td>Running</td>
<td>cf-redis-broker-partition-default-az-gw/0</td>
<td>10.0.0.10</td>
</tr>
<tr>
<td>dedicated-node-partition-default-az-gw/0</td>
<td>Running</td>
<td>dedicated-node-partition-default-az-gw/0</td>
<td>10.0.0.10</td>
</tr>
</tbody>
</table>
```

- Target the manifest of your deployed Redis with `bosh deployment PATH-TO-MANIFEST.yml`. If you do not have this file, you can download it by running `bosh download manifest DEPLOYMENT-NAME`.
- `bosh ssh` into the node you wish to back up (or the `cf-redis-broker` node in a `shared-vm` plan).

Once you have connected to the node, a manual backup can be triggered with these steps:

1. Switch to root using `sudo -i`.
2. Run `/var/vcap/jobs/service-backup/bin/manual-backup`

Notes

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

Making Your Own Backups

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

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

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

Shared-VM Plan

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

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

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

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

For example, here are two instances:

```
root@663581fe-3428-4ef9-9083-9ac70c3a78b0:/var/vcap/store/cf-redis-broker#
find -type f | xargs ls -1
./redis-data/31245378-c92-446a-ad12-ab865b978b00/db/dump.rdb
./redis-data/31245378-c92-446a-ad12-ab865b978b00/redis.conf
./redis-data/31245378-c92-446a-ad12-ab865b978b00/redis-server.pid
./redis-data/62333bf9-f023-4566-b233-6666b68b8f4d/db/dump.rdb
./redis-data/62333bf9-f023-4566-b233-6666b68b8f4d/redis.conf
./redis-data/62333bf9-f023-4566-b233-6666b68b8f4d/redis-server.pid
./statefile.json
```

Dedicated-VM Plan

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

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

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

Restore Redis Instance from a Backup

To a Local System

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

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

To Pivotal Cloud Foundry

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

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

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

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

Redis for PCF version 1.7 and later provides a script to automatically restore data in a newly provisioned Redis instance.

**Preparation**

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

**Dedicated-VM Plan**

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

**Execution**

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

**Debugging**

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

The script logs at `/var/vcap/sys/log/redis-backups/redis-backups.log`.

If a step has failed, resolve the reason that caused it to fail and execute the failed step and every next step manually. You can retrieve through the binding to your service instance: `cf service-key {instance_name} {key_name}`

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

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

3. **DeleteExistingPersistenceFiles**
   
   Clean up existing Redis data files:
   ```
   rm -f /var/vcap/store/redis/appendonly.aof
   rm -f /var/vcap/store/redis/dump.rdb
   ```

4. **CopyBackupFileWithCorrectPermissions**
   
   Restore your Redis backup file to `/var/vcap/store/redis/dump.rdb` and correct the owner and permissions with `chown vcap:vcap /var/vcap/store/redis/dump.rdb && chmod 660 /var/vcap/store/redis/dump.rdb`

5. **SetAppendOnly**
   
   Edit the template Redis config file with `vim $(find /var/vcap/data/jobs -name redis.conf)` and make the following line changes:
• appendonly yes → appendonly no

6. StartAll
Run monit start all

7. WaitForStart
Wait for monit services to enter the running state, you can watch this with watch monit summary

8. RewriteAOF
Run /var/vcap/packages/redis/bin/redis-cli -a {instance_password} BGREWRITEAOF

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

10. StopAll
Run monit stop all

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

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

13. SetAppendOnly
Edit the template Redis config file with vim $(find /var/vcap/data/jobs/ -name redis.conf) and make the following line changes:
  • appendonly no → appendonly yes

14. StartAll
Run monit start all

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

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

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

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

1. StopAll
Run monit stop all
2. WaitForStop
Wait for monit services to enter the not monitored state, you can watch this with watch monit summary
3. SetConfigCommand
Edit the template Redis config file with vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf and comment out
the line:

```
* rename-command CONFIG "configalias" -> #rename-command CONFIG "configalias"
```

4. **SetRewriteCommand**

Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and comment out the line:

```
* rename-command BGREWRITEAOF "" -> #rename-command BGREWRITEAOF ""
```

5. **DeleteExistingPersistenceFiles**

Clean up existing Redis data files if they exist:

```
* rm -f /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/appendonly.aof
* rm -f /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb
```

6. **CopyBackupFileWithCorrectPermissions**

Restore your Redis backup file to `/var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/db/dump.rdb` and correct the owner and permissions with

```
```

7. **SetAppendOnly**

Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and make the following line changes:

```
* appendonly yes -> appendonly no
```

8. **StartAll**

Run `monit start all`

9. **WaitForStart**

Wait for monit services to enter the running state, you can watch this with `watch monit summary`

10. **RewriteAOF**

Run `/var/vcap/packages/redis/bin/redis-cli -a {instance_password} BGREWRITEAOF`

11. **RewriteAOF**

Run `watch '/var/vcap/packages/redis/bin/redis-cli -a {instance_password} INFO | grep aof_rewrite_in_progress' until aof_rewrite_in_progress 0`

12. **StopAll**

Run `monit stop all`

13. **WaitForStop**

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

14. **ChownToUserAndGroup**

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

15. **SetAppendOnly**

Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and make the following line changes:

```
* appendonly no -> appendonly yes
```

16. **SetConfigCommand**

Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and uncomment the line:

```
* rename-command CONFIG "configalias" -> rename-command CONFIG "configalias"
```

17. **SetRewriteCommand**

Edit the template Redis config file with `vim /var/vcap/store/cf-redis-broker/redis-data/{instance_guid}/redis.conf` and uncomment the line:

```
* rename-command BGREWRITEAOF "" -> rename-command BGREWRITEAOF ""
```

18. **StartAll**

Run `monit start all`
Recovering Redis Instances

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

Shared-VM Plan Caveats

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

Dedicated-VM Plan Caveats

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

Note

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

Recovery Procedure

After redeploying Redis, take the following steps.

Shared-VM Plan

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

Dedicated-VM Plan

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

The PCF Firehose emits Redis metrics.

Configuring Secure Communication

Redis for PCF v1.7.2 lets the operator turn on/off TLS communications for metrics, via a Use non-secure communication for metrics checkbox on the metrics configuration page in Ops Manager. Configure this checkbox for different versions of PCF as follows:

- **PCF v1.8**: Select this checkbox to send metrics to the Firehose.
- **PCF v1.9**: Clear this checkbox to send metrics to the Firehose securely, or select it to send metrics insecurely.
- **PCF v1.10 and later**: Clear this checkbox to send metrics to the Firehose and avoid errors.

Setting this checkbox incorrectly can cause a Cannot generate manifest... unknown property "cf_etcd_client_cert" error.

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).

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

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

Redis emits a number of metrics that can be used to monitor the health and performance of your Redis deployment.

### keyspace_hits

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

### keyspace_misses

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

### used_memory

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of bytes allocated by Redis. “/p-redis/info/memory/used_memory”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>Grows as the number of unsaved keys increases.</td>
</tr>
</tbody>
</table>

### maxmemory

<table>
<thead>
<tr>
<th>Description</th>
<th>Maximum number of bytes available in Redis. “/p-redis/info/memory/maxmemory”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>Indicates the max memory available in Redis.</td>
</tr>
</tbody>
</table>

### blocked_clients

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of connected clients pending on a blocking call. “/p-redis/info/clients/blocked_clients”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>Can be used as an indicator to detect deadlocks.</td>
</tr>
</tbody>
</table>

### connected_clients

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of clients connected to the Redis instance. “/p-redis/info/clients/connected_clients”</th>
</tr>
</thead>
</table>

### rdb_changes_since_last_save

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

<table>
<thead>
<tr>
<th>Description</th>
<th>Total number of commands processed by Redis. “/p-redis/info/stats/total_commands_processed”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>A crude indicator of activity. Can be used in conjunction with <code>uptime_in_seconds</code>.</td>
</tr>
</tbody>
</table>

### mem_fragmentation_ratio

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

### total_instances

<table>
<thead>
<tr>
<th>Description</th>
<th>Total number of <em>dedicated-vm</em> instances of Redis. “/p-redis/service-broker/dedicated_vm_plan/total_instances”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>Used in conjunction with <code>available_instances</code>, provides information about used instances.</td>
</tr>
</tbody>
</table>

### available_instances

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of available <em>dedicated-vm</em> instances of Redis. “/p-redis/service-broker/dedicated_vm_plan/available_instances”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>If zero, no more instances are available.</td>
</tr>
</tbody>
</table>

### total_instances

<table>
<thead>
<tr>
<th>Description</th>
<th>Total number of <em>shared-vm</em> instances of Redis. “/p-redis/service-broker/shared_vm_plan/total_instances”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>Used in conjunction with <code>available_instances</code>, provides information about used instances.</td>
</tr>
</tbody>
</table>

### available Instances

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of available <em>shared-vm</em> instances of Redis. “/p-redis/service-broker/shared_vm_plan/available_instances”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance</td>
<td>If zero, no more instances are available.</td>
</tr>
</tbody>
</table>

### Other Metrics

Redis also exposes the following metrics. For more information, see the [Redis documentation](https://redis.io/).

- `arch_bits`
- `uptime_in_seconds`
- `uptime_in_days`
- `hz`
- `lru_clock`
- `client_longest_output_list`
- `client_biggest_input_buf`
- `used_memory_rss`
- `used_memory_peak`
- `used_memory_lua`
- mem_fragmentation_ratio
- loading
- rdb_bgsave_in_progress
- rdb_last_save_time
- rdb_last_bgsave_time
- rdb_current_bgsave_time
- aof_rewrite_in_progress
- aof_rewrite_scheduled
- aof_last_rewrite_time
- aof_current_rewrite_time
- total_connections_received
- total_commands_processed
- instantaneous_ops_per_sec
- total_net_input_bytes
- total_net_output_bytes
- instantaneous_input_kbps
- instantaneous_output_kbps
- rejected_connections
- sync_full
- sync_partial_ok
- sync_partial_err
- expired_keys
- evicted_keys
- keyspace_hits
- keyspace_misses
- pubsub_channels
- pubsub_patterns
- latest_fork_usec
- migrate_cached_sockets
- connected_slaves
- master_repl_offset
- repl_backlog_active
- repl_backlog_size
- repl_backlog_first_byte_offset
- repl_backlog_histlen
- used_cpu_sys
- used_cpu_user
- used_cpu_sys_children
- used_cpu_user_children
- cluster_enabled
- rdb_last_bgsave_status
- aof_last_bgrewrite_status
- aof_last_write_status
Troubleshooting Redis for PCF

This topic lists troubleshooting information relevant to Redis for PCF.

Troubleshooting Guides for PCF

General troubleshooting guides for Pivotal Cloud Foundry:

- PCF 1.8
- PCF 1.7
- PCF 1.6

Knowledge Base Articles

Pivotal Knowledge Base articles specifically about Redis for PCF:

- Can’t redeploy PCF Redis if shared-vm persistent disk full
- Issue with upgrading tile
- Issue with deploy failing
- Redis Instance Alive after Successful De-provisioning
- PCF Redis dedicated instance fails to persist to disk
- Redis error when saving changes after a back to AWS S3: Error: Access Denied for bucket

Other Issues

| Error | Cannot generate manifest for product Redis: Error in {".properties.metrics_disable_etcd_tls.value ? .properties.null_string.value : ...cf.properties.cf_etcd_client_cert.cert_pem }: unknown property "cf_etcd_client_cert" (Product "Redis" / Job: nil) |
| Cause | You are using PCF 1.10 and did not untick the ‘Use non-secure communication for metrics’ checkbox on the metrics configuration page in Ops Manager. |
| Solution | Please untick the checkbox and redeploy. See here for more information. |

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

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

Useful Debugging Information

If you encounter an issue, here is a list of useful information to gather, especially before you perform any destructive operations such as `cf purge-service-offers` or `bosh delete deployment`.

- PCF Redis version
- Previous PCF Redis version if upgrading
- Ops Manager version, and previous if upgrading Ops Manager
- IaaS description
From OpsManager:

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

For all VMs, unless specified otherwise:

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

  Full `ps aux` - Has monit done its job?
  ```
  ps aux | grep redis-serve(r)```

  - Are Redis instances running?
  ```
  df -h```

  - disk usage
  ```
  free -m```

  - memory usage
  ```
  cf m
  ```

  ```
  tree /var/vcap/store/cf-redis-broker/redis-data [broker only]
  ```

  Copy of /var/vcap/store/cf-redis-broker/statefile.json [broker only]
For App Developers

Redis Configuration

Redis is configured with a maxmemory-policy of no-eviction. This policy means that the once memory is full, the service will not evict any keys and no write operations will be possible until memory becomes available. Persistence is configured for both RDB and AOF. The default maximum number of connections, maxclients, is set at 10000 but this number is adjusted by Redis according to the number of file handles available. Replication and event notification are not configured.

Service Plans

PCF Redis offers Dedicated VM and Shared VM plans. The memory allocated to the plans is determined by the operator at deploy time. For more information on the plans see the architecture and recommended usage pages.

Using Redis for PCF

Instructions for creating, binding to, and deleting an instance of the dedicated-VM or shared-VM plan are here.

Getting Started

Using PCF Redis with Spring


PCF Dev

PCF Dev is a small footprint version of PCF that’s small enough to run on a local developer machine. More info herehttps://pivotal.io/pcf-dev [9].

Redis Example App

Sample ruby code that uses PCF can be found here https://github.com/pivotal-cf/cf-redis-example-app [9].

Redis

To learn more about Redis itself, visit redis.io [9].
Using Redis for PCF

Redis for PCF can be used both via Pivotal Apps Manager and the CLI, both methods are outlined below. An example application has also been created to help application developers get started with Redis for PCF, and can be downloaded here.

See Redis for PCF Recommended Usage for recommendations regarding Redis for PCF service plans, and memory allocation.

Creating a Redis Service Instance

The following procedure describes how to create a Redis service instance in the Pivotal Cloud Foundry Elastic Runtime environment.

Available Plans

Before creating a Redis instance, it is worth being aware of the two available plans:

<table>
<thead>
<tr>
<th>Plan Name</th>
<th>Suitable for</th>
<th>Tenancy Model per Instance</th>
<th>Highly Available</th>
<th>Backup Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared-VM</td>
<td>Lighter workloads that do not require dedicated resources</td>
<td>Shared VM</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dedicated-VM</td>
<td>Increased workloads that require dedicated resources</td>
<td>Dedicated VM</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Using Pivotal Apps Manager

1. From within Pivotal Apps Manager, select Marketplace from the left navigation menu under Spaces. The Services Marketplace displays.
2. Select Redis from the displayed tiles and click to view the available plans.
3. Click on the appropriate Select this plan button to select the required Redis Service Plan.
4. In the Instance Name field, enter a name that will identify this specific Redis service instance.
5. From the Add to Space drop-down list, select the space where you or other users will deploy the applications that will bind to the service.
6. Click the Add button.

Using the CLI

1. Run `cf marketplace` to view the available service plans:

   ```
   cf marketplace
   Getting services from marketplace in org system / space apps-manager as admin...
   OK
   service plans description
   p-redis shared-vm, dedicated-vm Redis service to provide a key-value store
   TIP: Use `cf marketplace -s SERVICE` to view descriptions of individual plans of a given service.
   ```

2. Run `cf create service` to create the service plan. Include the service plan name as listed in the Services Markeplace and a descriptive name you want to use for the service:

   ```
   cf create-service p-redis SERVICE-PLAN-NAME SERVICE-INSTANCE-NAME
   ```

   For example:

   ```
   cf create-service p-redis shared-vm redis
   ```
Binding an Application to the Redis Service

The following procedures describe how to bind a Redis service instance to your Pivotal Cloud Foundry application. This can be done via the Pivotal Apps Manager or Using the Pivotal Cloud Foundry CLI.

Using Pivotal Apps Manager

1. Select the application that you wish to bind to the service. A page displays showing the already bound services and instances for this application.
2. Click Bind. A list of available services displays.
3. Click the Bind button for the Redis service you want to bind to this application.
4. Start or restage your app from the command line:

   \$ cf restage APPLICATION-NAME

Using the CLI

1. Run \texttt{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-redis-instance    p-redis   shared-vm

2. Run \texttt{cf bind-service} to bind the application to the service instance.

   \$ cf bind-service APPLICATION-NAME SERVICE-INSTANCE-NAME

   For example:

   \$ cf bind-service my-application redis
3. Run \texttt{cf restage} to restage your application.

   \$ cf restage APPLICATION-NAME

Entries in the VCAP_SERVICES Environment Variable

Applications running in Cloud Foundry gain access to the bound service instances via an environment variable credentials hash called VCAP_SERVICES. An example hash is show below:
```
{
  "p-redis": {
    "credentials": {
      "host": "10.0.0.11",
      "password": "<redacted>",
      "port": 6379
    },
    "label": "p-redis",
    "name": "redis",
    "plan": "dedicated-vm",
    "provider": null,
    "syslog_drain_url": null,
    "tags": [
      "pivotal",
      "redis"
    ],
    "volume_mounts": []
  }
}
```

You can search for your service by its `name`, given when creating the service instance, or dynamically via the `tags` or `label` properties.

Deleting a Redis Instance

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

Using Pivotal Apps Manager

1. Locate the row under Services that contains the service instance you want to delete and click **Delete**.

2. If you had applications that were bound to this service, you may need to restage or re-push your application for the application changes to take effect.

   $ cf restage APPLICATION-NAME

Using the CLI

1. Run `cf delete-service` and include the service instance name that you would like to delete. Enter `y` when prompted to confirm.

   $ cf delete-service SERVICE-INSTANCE-NAME

   For example:

   $ cf delete-service my-redis-instance

   Really delete the service my-redis-instance? [y] y

   Deleting service my-redis-instance in org system / space apps-manager as admin...

   OK

2. If you had applications that were bound to this service, you may need to restage or re-push your application for the application changes to take effect.

   $ cf restage APPLICATION-NAME
Sample Redis Configuration

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

daemonize yes
pidfile /var/vcap/sys/run/redis.pid
port 6379
tcp-backlog 511
timeout 0
tcp-keepalive 0
logfile /var/vcap/sys/log/redis/redis.log
syslog-enabled yes
syslog-ident redis-server
syslog-facility local0
databases 16
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
appendfilename dump.rdb
dir /var/vcap/store/redis
slave-sent-stale-data yes
slave-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-ping-slave-period 10
repl-timeout 60
repl-disable-tcp-nodelay no
slaves-priority 100
maxmemory-policy noeviction
appendonly yes
appendfilename appendonly.aof
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
has-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events **
hash-max-zipmap-entries 512
hash-max-zipmap-value 64
list-max-zipmap-entries 512
list-max-zipmap-value 64
set-max-intset-entries 512
zset-max-zipmap-entries 128
zset-max-zipmap-value 64
hll-sparse-max-bytes 3000
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
aof-rewrite-incremental-sync yes
rename-command CONFIG "A-B-Ab1A2za_._Aa1C2bAbB22a_a1Baa"
rename-command SAVE "SAVE"
rename-command BGSAVE "BGSAVE"
rename-command DEBUG **
rename-command SHUTDOWN **
rename-command SLAVEOF **
rename-command SYNC **
requirepass 1a1a2b3b-0cc-2222-444b-1e1e1e2222
maxmemory 1775550873