# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>RabbitMQ for PCF</td>
<td>3</td>
</tr>
<tr>
<td>RabbitMQ® for PCF Release Notes</td>
<td>6</td>
</tr>
<tr>
<td>Unlocking the Power of On-Demand RabbitMQ for PCF</td>
<td>16</td>
</tr>
<tr>
<td>On-Demand Service Architecture</td>
<td>21</td>
</tr>
<tr>
<td>Deploying the RabbitMQ Pre-Provisioned Service</td>
<td>24</td>
</tr>
<tr>
<td>Installing and Configuring the On-Demand Service</td>
<td>28</td>
</tr>
<tr>
<td>Installing and Configuring the Pre-Provisioned Service</td>
<td>41</td>
</tr>
<tr>
<td>Smoke Tests</td>
<td>56</td>
</tr>
<tr>
<td>Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF</td>
<td>57</td>
</tr>
<tr>
<td>Setting Limits for On-Demand Service Instances</td>
<td>64</td>
</tr>
<tr>
<td>Controlling Access to Service Plans by Org</td>
<td>68</td>
</tr>
<tr>
<td>Isolating Clusters with the RabbitMQ for PCF Replicator</td>
<td>69</td>
</tr>
<tr>
<td>Setting Default Policies for the RabbitMQ Service</td>
<td>73</td>
</tr>
<tr>
<td>Using the RabbitMQ Management Dashboard</td>
<td>76</td>
</tr>
<tr>
<td>Clustering and Network Partitions</td>
<td>78</td>
</tr>
<tr>
<td>Upgrading RabbitMQ for PCF</td>
<td>80</td>
</tr>
<tr>
<td>Troubleshooting and FAQs for On-Demand RabbitMQ for PCF</td>
<td>82</td>
</tr>
<tr>
<td>Frequently Asked Questions for Pre-Provisioned RabbitMQ for PCF</td>
<td>97</td>
</tr>
<tr>
<td>Using On-Demand RabbitMQ for PCF</td>
<td>100</td>
</tr>
<tr>
<td>RabbitMQ Environment Variables</td>
<td>105</td>
</tr>
<tr>
<td>Troubleshooting Instances</td>
<td>107</td>
</tr>
</tbody>
</table>
RabbitMQ for PCF

Note: RabbitMQ for PCF v1.12 is no longer supported. The support period for v1.12 has expired. To stay up-to-date with the latest software and security updates, upgrade to a supported version.

About RabbitMQ for PCF

RabbitMQ for Pivotal Cloud Foundry (PCF) enables PCF app developers to provision and use the RabbitMQ message broker with a single command.

RabbitMQ for PCF v1.8 and later support two types of service, an on-demand service and a pre-provisioned service. This table summarizes the main differences between the two:

<table>
<thead>
<tr>
<th>Service Type</th>
<th>Available Since</th>
<th>VMs it Runs On</th>
<th>How VMs are Created</th>
<th>Metrics Name Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-Demand Service</td>
<td>v1.8</td>
<td>Dedicated VM that serves a single service instance.</td>
<td>PCF creates each VM on-demand when app developer creates service instance</td>
<td>p.rabbitmq (with a dot)</td>
</tr>
<tr>
<td>Pre-Provisioned Service</td>
<td>v1.2</td>
<td>Multi-tenant VMs shared by apps across PCF deployment</td>
<td>PCF creates all VMs when operator deploys or updates service</td>
<td>p-rabbitmq (with a dash)</td>
</tr>
</tbody>
</table>

This RabbitMQ for PCF documentation describes both service types. Documentation for RabbitMQ for PCF v1.7 and earlier only describes a pre-provisioned service.

What are On-Demand Instances

In RabbitMQ for PCF versions before v1.8.0, the RabbitMQ service instances correspond to a unique RabbitMQ vhost on the multi-tenant RabbitMQ cluster. RabbitMQ for PCF v1.8.0 introduced On-Demand Broker (ODB) support. That means that a new, single-tenant, cluster can be created and dedicated to a single app.

For more information, see Unlocking the Power of On-Demand RabbitMQ for PCF and On-Demand Service Architecture.

About RabbitMQ

RabbitMQ is a fast and dependable open-source message server, which supports a wide range of use cases including reliable integration, content-based routing and global data delivery, and high-volume monitoring and data ingestion.

Emerging as the de facto standard for cloud messaging, RabbitMQ is used for efficient communication between servers, apps and devices, and creates lasting value by enabling rapid development of modern decentralized app and data architectures that can scale with your business needs.

Product Snapshot

The following table provides version and version-support information about RabbitMQ for PCF.

<table>
<thead>
<tr>
<th>Element</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>v1.12.15</td>
</tr>
<tr>
<td>Release date</td>
<td>December 28, 2018</td>
</tr>
<tr>
<td>Software component versions</td>
<td>RabbitMQ OSS v3.6.16 and RabbitMQ OSS v3.7.9</td>
</tr>
<tr>
<td>Compatible Ops Manager versions</td>
<td>v2.1.x and v2.2.x</td>
</tr>
<tr>
<td>Compatible Pivotal Application Service versions</td>
<td>v2.1.x and v2.2.x</td>
</tr>
<tr>
<td>IaaS support</td>
<td>AWS, Azure, GCP, OpenStack, and vSphere</td>
</tr>
<tr>
<td>IPsec support</td>
<td>No</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
Features

On-Demand

- Create up to 5 different on-demand RabbitMQ plans which can be provisioned through the marketplace
- Choose whether a plan has one, three, five, or seven nodes
- Default resource sizes in plans to guide selection
- More control over which orgs and spaces have visibility of each configured plan
- Bind apps to an instance of the plan, providing unique credentials for each binding
- Management dashboard access to app developers
- Deployment into an availability zone specified by the plan
- Automated upgrades of RabbitMQ for major, minor, and patch releases (see release notes for downtime requirements)
- RabbitMQ Syslog forwarding configuration inherited from the pre-provisioned configuration
- RabbitMQ metrics are exposed on the Firehose
- Run smoke tests for on-demand plans on plan 1
- Errands are run on colocated VMs to decrease deployment times

For more information, see Unlocking the Power of On-Demand RabbitMQ for PCF.

Pre-Provisioned

- Provision an instance of the RabbitMQ service, which corresponds to a unique RabbitMQ vhost (virtual host)
- Bind apps to an instance of the plan, providing unique credentials for each binding
- Management dashboard access to PCF Operators and app developers
- Deployment across multiple availability zones, with nodes striped across the AZs automatically
- Enable SSL (Secure Sockets Layer) for the AMQP, MQTT, STOMP protocols
- HAProxy load balancer across all nodes to balance connections
- Plugin configuration can be easily changed at any time and the cluster redeployed and updated
- The cluster topology can be changed and easily scaled out
- Automated upgrades of RabbitMQ for major, minor, and patch releases (see Downtime When Upgrading for downtime requirements)
- Configure the end point for the RabbitMQ Syslog
- RabbitMQ and HAProxy metrics are exposed on the firehose
- Syslog forwarding on by default
- Errands are run on colocated VMs to decrease deployment times

Release Notes and Known Issues

Check the release notes for your release version for important information and known issues. To see release notes for another version, select the version from the dropdown list at the top of the page.

RabbitMQ for PCF and Other PCF Services

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

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

The following PCF services offer on-demand service plans:

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

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

For other services, like PCC and MySQL for PCF, only on-demand service plans are available.

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

<table>
<thead>
<tr>
<th>PCF service tile</th>
<th>Standalone product related to the service</th>
<th>Versions supporting on demand</th>
<th>Versions supporting pre-provisioned</th>
</tr>
</thead>
<tbody>
<tr>
<td>RabbitMQ for PCF</td>
<td>Pivotal RabbitMQ</td>
<td>v1.8 and later</td>
<td>All versions</td>
</tr>
<tr>
<td>Redis for PCF</td>
<td>Redis</td>
<td>v1.8 and later</td>
<td>All versions</td>
</tr>
<tr>
<td>MySQL for PCF</td>
<td>MySQL</td>
<td>v2.x</td>
<td>NA</td>
</tr>
<tr>
<td>PCC</td>
<td>Pivotal GemFire</td>
<td>All versions</td>
<td>NA</td>
</tr>
</tbody>
</table>

Please provide any bugs, feature requests, or questions to the [PCF Feedback list](mailto:PCFFeedback@pivotal.com).
RabbitMQ® for PCF Release Notes

Upgrade to the Latest Version

Pivotal recommends that you upgrade to the latest version of your current minor line, then upgrade to the latest available version of the new minor line. For example, if you use an older v1.11.x version, upgrade to the latest v1.11.x version before upgrading to the latest v1.12.x version.

For product versions and upgrade paths, see the Product Compatibility Matrix.

Note: Pivotal no longer recommends waiting for Spring Cloud Services (SCS) v3.0, where the dependency on pre-provisioned RabbitMQ for PCF will be removed, before upgrading from RabbitMQ for PCF v1.12 to v1.13. This is due to the risk of going out of support on your RabbitMQ for PCF tile, Pivotal Application Service (PAS), and other components.

WARNING: Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1 or on PCF v2.2.6 or later.

Use RabbitMQ v3.7 On-Demand Plans

RabbitMQ v3.7 plans are available. Ensure that all apps use the RabbitMQ v3.7 on-demand plans instead of the RabbitMQ v3.6 on-demand plans.

For more information, see the following:

• About Upgrading On-Demand Instances from RabbitMQ v3.6 to v3.7
• The ANN RabbitMQ 3.6.x support timeline post on the RabbitMQ users forum

v1.12.15

Release Date: December 28, 2018

Features

New features and changes in this release:

• Requires stemcell 3586.65
• Scripts to drop and restore AMQP(S) traffic. For more information, see Drop and Restore AMQP(S) Traffic to a RabbitMQ Instance

Fixed Issues

This release fixes the following issue:

• In RabbitMQ for PCF v1.12.14 and v1.12.13, you cannot see RabbitMQ metrics on the Loggregator Firehose.

Known Issues

This release has the following issues:

• [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
• If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
• If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.
• There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.
Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.9
- Erlang v20.3.8.15
- HAPerxy v1.6.13

v1.12.14

Release Date: December 20, 2018

Features

- Requires stemcell 3586.60
- Set swap to 1 GB for RabbitMQ nodes
- Supports the Log Cache cf CLI plugin that enables developers to access logs for an on-demand service instance using the command `cf tail`.

For more information about this feature, see Access RabbitMQ Metrics for On-Demand Service Instances.

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- If you upgrade to RabbitMQ for PCF to this version, you cannot see RabbitMQ metrics on the Loggregator Firehose. This issue does not affect new installations.

- There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.9
- Erlang v20.3.8.15
- HAPerxy v1.6.13
v1.12.13

Release Date: November 22, 2018

Security Fixes

This release includes the following security fix:

- Critical [CVE-2018-15759]: On Demand Services SDK Timing Attack Vulnerability

Features

- Adds mutual TLS between service metrics and the loggregator system
- Removes credentials from the dashboard URL in the pre-provisioned service

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see [CVE-2018-1279]: RabbitMQ cluster compromise due to deterministically generated cookie.
  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- If you upgrade to RabbitMQ for PCF to this version, you cannot see RabbitMQ metrics on the Loggregator Firehose. This issue does not affect new installations.

- There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager. Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.8
- Erlang v20.3.8.10
- HAPerxy v1.6.13

v1.12.12

Release Date: November 1, 2018

Features

- Requires stemcell 3468.78
- Smoke tests are now more resilient when using external load balancers.

Known Issues
[Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.

- If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
- If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.8
- Erlang v20.3.8.9
- HAProxy v1.6.13

v1.12.10

Release Date: October 5, 2018

Features

- Requires stemcell 3468.73
- RabbitMQ 3.7 updated to 3.7.8
- Erlang updated to v20.3.8.9

Known Issues

[Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.

- If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
- If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.8
- Erlang v20.3.8.9
v1.12.9

Release Date: September 13, 2018

Features

- Requires stemcell 3468.69
- Smoke tests now wait up to 5 minutes for a test app to deploy.

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- There is an issue with upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.2.5 and earlier. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1, or on PCF v2.2.6 or later.
- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.
- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.
- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.7
- Erlang v20.3.8.6
- HAPerxy v1.6.13

v1.12.8

Release Date: August 20, 2018

Features

- Requires stemcell 3468.64

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- There is an issue with upgrading from v1.12.x to v1.13.x on PCF v2.2. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1.
You cannot upgrade from RabbitMQ for PCF v.12 to RabbitMQ for PCF v.13 on PCF v2.2.0-2.2.5. You must upgrade on PCF v2.1 or on PCF v2.2.6 or later. Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue. Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment. When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager. Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.7
- Erlang v20.3.8.1
- HAProxy v1.6.13

v1.12.7

Release Date: July 13, 2018

Features

New features and changes in this release:

- Fixes bug in the smoke tests when CredHub is enabled

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

There is an issue with upgrading from v1.12.x to v1.13.x on PCF v2.2. Pivotal recommends upgrading from RabbitMQ for PCF v1.12.x to v1.13.x on PCF v2.1.

You cannot upgrade from RabbitMQ for PCF v1.2 to RabbitMQ for PCF v1.13 on PCF v2.2.0-2.2.5. You must upgrade on PCF v2.1 or on PCF v2.2.6 or later.

Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager. Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.7
- Erlang v20.3.8.1
- HAProxy v1.6.13

v1.12.6
Features

New features and changes in this release:

- Requires stemcell 3468.51.
- Updates OSS RabbitMQ Server and Erlang packages.

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  
- If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  
- If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.
  
Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.16
- OSS RabbitMQ v3.7.6
- Erlang v20.3.8.1
- HAProxy v1.6.13

V1.12.5

Release Date: June 5, 2018

Features

New features and changes in this release:

- Requires stemcell 3468.46.
- cf-cli golang version reduced to v1.9.5 from v1.10 because of an issue parsing x509 certificates. For more information, see the Golang issue crypto/x509: CAnotAuthorizedForExtKeyUsage is premature.

Fixed Issues

This release fixes the following issue:

- Smoke tests sometimes failed because of low time-out values.

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.

If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.

Pivotal recommends leaving the default run rule set to On.

Packages

- OSS RabbitMQ v3.6.15
- OSS RabbitMQ v3.7.4
- Erlang v19.3.6.4
- HAProxy v1.6.13

v1.12.4

Release Date: May 16, 2018

Features

New features and changes in this release:

- Requires stemcell 3468.42.
- Timestamps added to log entries that did not have them.
- Golang version reduced to v1.9.5 from v1.10 because of an issue parsing x509 certificates. For more information, see the Golang issue crypto/x509: CANotAuthorizedForExtKeyUsage is premature.

Known Issues

- [Security Issue] The method for generating the Erlang Cookie is not secure. For more information about this security issue, see CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie.
  
  If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see Security Issue with the Tile Generated Erlang Cookie.
  
  If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- Smoke tests might fail because time-out values in the smoke tests are too low. For more information about troubleshooting smoke tests, see Smoke Tests.

- Cluster scaling or changing the Erlang Cookie value require cluster downtime, and might result in failed deployments. For more information, see Cluster Scaling Known Issue and Changing the Erlang Cookie Value Known Issue.

- Changing networks and/or IP addresses for the RabbitMQ Server job results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

- When errand run rules are set to When Changed, Ops Manager may not run the errands when the tile has relevant changes. For more information, see Managing Errands in Ops Manager.
  
  Pivotal recommends leaving the default run rule set to On.
v1.12.3

Release Date: April 19, 2018

Features

New features and changes in this release:

On-Demand

- The following errands are colocated with their respective broker to decrease errand run time and VM footprint:
  - Register / Deregister On Demand Service Broker
  - On Demand Instance Smoke Tests
  - Upgrade All Service Instances
  - Delete All Service Instances

  For more information about errands, see [Errands](#).

- [Beta] Service Sharing allows RabbitMQ deployments to be shared across Cloud Foundry spaces and orgs.
  For more information about sharing service instances, see [Cloud Foundry Documentation](#).

- [Beta] Support for secure credentials (CredHub). Credentials are stored in a central repository and are restricted to components that actually need them.

  For more information about CredHub, see [Cloud Foundry Documentation](#).

Pre-Provisioned

- The following errands are colocated with their respective broker to decrease errand run time and VM footprint:
  - Broker Registrar / Deregistrar
  - Smoke Tests errands

  For more information about errands, see [Errands](#).

Known Issues

- **[Security Issue]** The method for generating the Erlang Cookie is not secure. For more information about this security issue, see [CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie](#).

  - If you rely on the tile to generate the cookie by leaving the Erlang cookie field blank, see [Security Issue with the Tile Generated Erlang Cookie](#).
  - If you have set the Erlang Cookie explicitly in the tile configuration, you are not affected by this issue.

- Cluster scaling or changing the [Erlang Cookie](#) value require cluster downtime, and might result in failed deployments. For more information, see [Cluster Scaling Known Issue](#) and [Changing the Erlang Cookie Value Known Issue](#).

- Changing networks and/or IP addresses for the [RabbitMQ Server](#) job results in a failed deployment. For more information, see [Changing Network or IP Addresses Results in a Failed Deployment](#).

- When errand run rules are set to **When Changed**, Ops Manager may not run the errands when the tile has relevant changes. For more information, see [Managing Errands in Ops Manager](#). Pivotal recommends leaving the default run rule set to **On**.

Packages

- OSS RabbitMQ v3.6.15
- OSS RabbitMQ v3.7.4
- Erlang v19.3.6.4
- HAProxy v1.6.13

View Release Notes for Another Version

To view the release notes for another product version, select the version from the drop-down list at the top of this page.
Unlocking the Power of On-Demand RabbitMQ for PCF

Introduction

RabbitMQ for Pivotal Cloud Foundry (PCF) responds to the demands of PCF operators to offer a RabbitMQ on-demand cluster for their application teams, in addition to the existing single-node on-demand plan. The on-demand cluster plan is aimed at workloads that require the same resilience requirements as the Pre-Provisioned offering, but also require their workloads be isolated.

The platform operations team can now configure a RabbitMQ for PCF cluster to meet their business requirements and empower app development teams to self-serve their own RabbitMQ cluster.

RabbitMQ for PCF also provides smoke tests for the on-demand plans so that operations teams can validate the app developer workflow for on-demand services. See Dedicated Instance Smoke Test Process.

Platform operators can now offer their app developers three types of RabbitMQ for PCF service plans:

- **Pre-provisioned**—For light to moderate messaging needs, this service is fully operated and managed by platform operators as a service.
- **On-demand single node**—For application teams requiring greater isolation than provided by the pre-provisioned approach. App development teams can have full access to their own message broker to adapt the runtime parameters to their requirements. For more information on these parameters, see Parameters and Policies in the RabbitMQ documentation.
- **On-demand cluster**—For an increased level of message resilience and cluster availability, as well as the benefits of workload isolation mentioned above.

This topic explains how to benefit from the two on-demand plans above.

For information about the pre-provisioned plan, see Deploying the RabbitMQ Pre-Provisioned Service. For information on using pre-provisioned plans to isolate workloads, see Creating Isolation with the Tile Replicator.

On-Demand Single Node Plan Using RabbitMQ 3.7

This plan is designed to be simple to configure, deploy, and use. It gives application teams fast access to the power of the leading open source message broker backed by BOSH to meet all but the most demanding high availability app messaging requirements.

This plan can suit high-performance workloads requiring messaging resilience and asynchronous messaging replication. RabbitMQ copies messages to disk for resilience and allows asynchronous messaging replication through the RabbitMQ Federation plug-in.

This plan offers:

- Fast access to an isolated instance of RabbitMQ scoped for the application teams
- Org and Space Administrator access to the RabbitMQ Management UI so application teams can have full control over the node
- Updates and upgrades initiated and controlled by the operator to keep the instance up-to-date with the latest security patches and bug fixes
- Message resilience provided through RabbitMQ exchange, queue Federation, and Shovel plugins.

To Be Deprecated: On-Demand Single Node Plan Using RabbitMQ 3.6

This plan provides the same benefits as described above, but uses RabbitMQ 3.6 for backwards compatibility.

RabbitMQ 3.6 is reaching end of support. Consider moving to RabbitMQ 3.7 as soon as possible. For more information about support for RabbitMQ 3.6, see the RabbitMQ 3.6.x support time line announcement.

If you are already using RabbitMQ for PCF, you can choose to provide service plans with RabbitMQ 3.6 for backwards compatibility. However, you should also enable RabbitMQ 3.7 plans to allow developers to migrate to the new version.

For new deployments, Pivotal recommends disabling all RabbitMQ 3.6 plans and using RabbitMQ 3.7 to avoid the need to upgrade in the near future.

On-Demand Cluster Plan Using RabbitMQ 3.7
Like the single node plan, this plan is designed to be simple to configure, deploy and use. It gives application teams fast access to the power of the leading Open Source message broker backed by BOSH to meet all but the most demanding high availability app messaging requirements.

This plan can suit high performance workloads requiring messaging resilience (copied to disk) and asynchronous messaging replication through the RabbitMQ Federation plugin. With this plan, however, you also scale out RabbitMQ for PCF to multiple nodes.

This plan offers:
- Fast access to an isolated, clustered instance of RabbitMQ scoped to the application team Orgs and Spaces
- Administrator access to the RabbitMQ Management UI to give application teams full control over the cluster
- Updates and upgrades initiated and controlled by the operator to keep the instance up-to-date with the latest security patches and bug fixes.
- Message resilience provided by mirroring queues across RabbitMQ nodes, and the option to use the Federation and Shovel plugins.

To Be Deprecated: On-Demand Cluster Plan Using RabbitMQ 3.6

This plan provides the same benefits as described above, but uses RabbitMQ 3.6 for backwards compatibility.

RabbitMQ 3.6 is reaching end of support. Consider moving to RabbitMQ 3.7 as soon as possible. For more information about support for RabbitMQ 3.6, see the RabbitMQ 3.6.x support time line announcement.

If you are already using RabbitMQ for PCF, you can choose to provide service plans with RabbitMQ 3.6 for backwards compatibility. However, you should also enable RabbitMQ 3.7 plans to allow developers to migrate to the new version.

For new deployments, Pivotal recommends disabling all RabbitMQ 3.6 plans and using RabbitMQ 3.7 to avoid the need to upgrade in the near future.

General Principles of the Cluster Plan

The following are some general principles to be aware of when configuring the cluster plan:

Designed for Consistency

RabbitMQ clustering is not primarily a solution for increased availability. Instead, it is designed for consistency and partition tolerance, as described in the CAP theorem. RabbitMQ clustering provides increased message consistency through queue mirroring. This means that messages accessed in one queue are exactly the same as in another queue. For more information, see Consistency or Availability Tradeoff.

Other options can be used for availability requirements, such as the use of federation between exchanges or queues.

For a detailed description of distributed RabbitMQ brokers, see the RabbitMQ documentation.

Number of Nodes

Every node in the on-demand cluster maintains a complete database of all metadata, and all changes to the metadata are confirmed by every node in the cluster. Therefore, going beyond seven nodes can have a significant negative impact on performance. For optimum resilience and performance, Pivotal recommends three nodes for most workloads.

Network Latency

RabbitMQ clusters are only recommended for deployment in low latency networks, which normally means that it is not advisable to deploy these clusters across availability zones (AZs). The stability and performance of the RabbitMQ cluster is heavily influenced by the workload on the nodes, replication choices, and network latency.

For this reason, Pivotal recommends that you deploy RabbitMQ clusters into a single Ops Manager AZ. However, where different AZs are in the same data center, with reliable low latency links, spanning AZs can be used.

For cloud IaaS deployments, Pivotal does not recommend that deployments span regions. For example, in Amazon Web Services (AWS) terms, deploying a RabbitMQ cluster across AZs within a region should provide high enough network performance to prevent impacting cluster stability. However, deploying across AWS regions is likely to lead to cluster instability. For more information, see the AWS documentation.
Consistency or Availability Tradeoff

In a distributed messaging system, a tradeoff must be made between availability or consistency when a network partition event occurs and one or more nodes are not able to communicate with each other. The cluster plan lets operators decide how they want the RabbitMQ cluster to react in the event of a network partition.

Pivotal recommends keeping the default cluster partition option of `pause_minority` because this satisfies most use cases. Choosing the `pause_minority` partition-handling strategy favors message consistency over availability. For more information about the options for handling partitions, see the RabbitMQ documentation. For a detailed description of the options available in RabbitMQ for PCF, see Clustering and Network Partitions.

Here is an example of how `pause_minority` works. If you create a RabbitMQ cluster with three nodes and one node becomes unable to communicate with the other two, this node is in the minority. The node that is in the minority is paused, and the other two nodes continue serving traffic. If each of the nodes loses connectivity with the other two, then the entire cluster is paused to preserve data since no majority can be established. The cluster heals when two or more nodes are able to communicate with each other.

RabbitMQ Queue Availability

It is important to be aware that message queue availability is different from cluster availability. So, having cluster availability does not mean that all of the messages within the queue are also available.

By default, queues within a RabbitMQ cluster are located on a single node—the node on which they were first declared. However, queues can be configured to mirror across multiple nodes, so that any message published to the queue is replicated to all mirrors. Enabling mirroring can have a negative impact on queue performance because messages must be copied to all mirrors before being acknowledged.

Each mirrored queue consists of one master and one or more mirrors, with the oldest mirror being promoted to the new master if the old master disappears for any reason. Consumers are connected to the master regardless of which node they connect to, and mirrors drop messages that have been acknowledged at the master. Queue mirroring enhances queue availability, but does not distribute load across nodes because each of the participating nodes must still do all the work.

App developers must decide if they want to use queue mirroring and determine the policy they want to apply to their queues. These choices have significant impact on the availability of their queues. For more information, see the RabbitMQ documentation.

Unlike the pre-provisioned plan, the cluster plan does not ship with a default load balancer. Therefore, developers must configure their app to use the array of hosts provided in `VCAP_SERVICES`. If developers enable queue mirroring, they must also ensure their apps have re-try logic and reconnection logic that iterates over the range of hosts provided. Most common RabbitMQ clients have this logic built into them. For more information, see the Spring AMQP documentation.

Because the cluster plan is designed to enable application teams to self-serve, not having a load balancer in front of the RabbitMQ cluster has these benefits:

- Manage resources better, as fewer VMs are needed.
- Help with troubleshooting. Client IP is now the IP of the source container and not the HAProx.
- Reduce the number of hops between apps and broker. This helps with latency.
- Determine queue placement. This makes sense for larger scale deployments.
- Empower application teams to manage their cluster in the best way for their app.
- Require re-try logic in an app if it needs HA access to a queue. Thus, all nodes can route to a queue if it is available.

Managing On-Demand Resources Through Plans

In configuring each plan, there are a number of operational controls that platform operations teams can use to manage the resources consumed by on-demand RabbitMQ:

- **Control Access**—Operators can choose the app development orgs and spaces for which the plans are available and visible. Each plan can be enabled or disabled, and service access and visibility can either be global, or enabled per org and space through the command line.

  For example, you may decide to enable the single node on-demand plan across all application teams to meet their demand to isolate their workload. You may then choose to offer the on-demand cluster plan only to a subset of application teams who require the extra resources.

- **Set Quotas**—You can set a global quota for all on-demand instances that takes precedence over each plan quota. This lets you guard against the risk of over-committing resources, but allows the flexibility of over-committing each plan, so you can meet the fluctuating demands of your app developers.

- **Control Resource Consumption**—Each plan offers more fine-grained control over individual plan resource consumption. At the highest level, you can use the plan quota to control the number of instances that can be deployed within a foundation. For each plan, you can also configure the number of
nodes that constitute a cluster (3, 5, or 7), the instance type, and persistent disk storage size to best suit your requirements.

- **Monitor**—You can monitor the number of instances that have been deployed against the quota you have set so that you can plan future resource requirements.

## Customizing Plan Options

The RabbitMQ for PCF on-demand plans expose a number of configuration options. In most cases, the default configurations meet most app demands. However, it is important for an operations team to consider the options to ensure that they provide the best service to their app developers. This section explains these options.

### Configuration Options

#### Single Node and Cluster Plans
- Enable/Disable plan
- Determine which orgs and spaces can see and access the plan
- Set Service Instance Quota
- Select AZ placement (where applicable)
- Set RabbitMQ instance size (CPU and Memory)
- Set persistent disk size (Persisted Message Store) for the RabbitMQ instance. Ensure the size of the persistent disk is at least twice as large as the instance memory.

#### Cluster Plan Only
- Set number of nodes to 3, 5, or 7
- Determine network partition behavior. See [Consistency or Availability Tradeoff](#) above.

**Note:** A load balancer, such as HAProxy, is not deployed with on-demand cluster plans.

### Things That Are Preconfigured

The following are preconfigured for both the single node and the cluster plans:

- **RabbitMQ VM Type**—When installing on PCF v2.0 or later, each RabbitMQ node is configured to have the following properties:
  - CPUs: 2
  - RAM: 8 GB
  - Ephemeral disk: 16 GB

  You can change these settings in the Service Plan Configuration page. Changing these settings affects all nodes.

- **Persistent Disk Type**—When installing on PCF v2.0 or later, each RabbitMQ node is configured to have 30 GB of persistent disk space.

  You can change this setting in the Service Plan Configuration page. Pivotal recommends you set this value to be twice the amount of RAM of the selected RabbitMQ VM Type.

- **Metrics**—Emitted to the Loggregator Firehose for all on-demand instances. The polling interval is set in the Ops Manager, in the Metrics polling interval field, in the Pre-Provisioned RabbitMQ tab of the RabbitMQ for PCF tile. Due to the impact of some of the cluster settings detailed below, Pivotal strongly recommends that you monitor the exposed metrics and configure alarms as recommended in [Monitoring and KPIs for On-Demand RabbitMQ for PCF](#). See also [Monitoring On-Demand RabbitMQ Clusters](#) below.

- **Logs**—RabbitMQ on-demand instance logs are forwarded using the same configuration as contained in the Syslog tab of the RabbitMQ for PCF tile.

- **Disk free space limit**—The disk free space limit is set to 150% of RAM of the instance type you select. For example, if you select an instance type with 10 GB of RAM, the disk free space limit is set to 15 GB. A cluster-wide alarm is triggered if the amount of free disk space drops below this, and all publishers are blocked. Instances must be configured to have persistent disks that are at least twice the size of instance RAM. For more information, see the [RabbitMQ documentation](#).

- **Memory threshold for triggering flow control**—Threshold at which flow control is triggered is set to 40% of the instance RAM. This means that when
the alarm is triggered, all connections publishing messages are blocked cluster-wide until the alarm is cleared.

For example, if you select an instance type with 10 GB of RAM, when more than 4 GB of memory is used, all publishing connections are blocked. For more information, see Memory Alarms in the RabbitMQ documentation.

- **Memory paging threshold**—This is the level at which RabbitMQ tries to free up memory by instructing queues to page their contents out to disk. This is done to try to avoid reaching the high watermark and blocking publishers. This threshold is set to 50% of the configured high watermark, which is 20% of configured memory. For more information on memory calculation, see Changes to Memory Allocation when Upgrading.

  For example, if you select an instance type with 10 GB of RAM, when more than 2 GB of memory is used, all queues start writing all queue contents to disk. For more information, see the RabbitMQ documentation.

Monitoring On-Demand RabbitMQ Clusters

- It is important to monitor and compare the number of instances that have been deployed against the quota you set via the metric exposed on the Firehose.

- Each instance is pre-configured to emit metrics to the Firehose and can be identified by the deployment tag, which has the service instance ID. It is important to monitor these metrics as recommended in Monitoring and KPIs for On-Demand RabbitMQ for PCF.
On-Demand Service Architecture

This topic describes the architecture for on-demand RabbitMQ® for Pivotal Cloud Foundry (PCF).

For information about architecture of the older, pre-provisioned service, see Deploying the RabbitMQ® Service.

Service Network Requirement

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

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

PCF v2.0 and Earlier

In PCF v2.0 and earlier, cloud operators pre-provision service instances from Ops Manager. For each service, Ops Manager allocates and recovers static IP addresses from a pre-defined block of addresses.

To enable on-demand services in PCF v2.0 and earlier, operators must create a service networks in BOSH Director and select the Service Network checkbox. Operators then can select the service network to host on-demand service instances when they configure the tile for that service.

PCF v2.1 and Later

PCF v2.1 and later include dynamic networking. In PCF v2.1 and later, operators can use dynamic networking with asynchronous service provisioning to define dynamically-provisioned service networks. For more information, see Default Network and Service Network.

In PCF 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 (PCF) 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.

<table>
<thead>
<tr>
<th>Key Components</th>
<th>Their Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>Creates and updates service instances as instructed by ODB.</td>
</tr>
<tr>
<td>BOSH Agent</td>
<td>Includes an agent on every VM that it deploys. The agent listens for instructions from</td>
</tr>
<tr>
<td></td>
<td>the BOSH Director and carries out those instructions. The agent receives job</td>
</tr>
<tr>
<td></td>
<td>specifications from the BOSH Director and uses them to assign a role, or job, to the</td>
</tr>
<tr>
<td></td>
<td>VM.</td>
</tr>
<tr>
<td>BOSH UAA</td>
<td>Issues OAuth2 tokens for clients to use when they act on behalf of BOSH users.</td>
</tr>
<tr>
<td>PAS</td>
<td>Contains the apps that are consuming services</td>
</tr>
<tr>
<td>ODB</td>
<td>Instructs BOSH to create and update services, and connects to services to create</td>
</tr>
<tr>
<td></td>
<td>bindings.</td>
</tr>
<tr>
<td>Deployed service</td>
<td>Runs the given data service. For example, the deployed Redis for PCF service instance</td>
</tr>
<tr>
<td>instance</td>
<td>runs the Redis for PCF data service.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>This component…</th>
<th>Must communicate with…</th>
<th>Default TCP Port</th>
<th>Communication direction(s)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODB</td>
<td>• BOSH Director</td>
<td>25555</td>
<td>One-way</td>
<td>The default ports are not configurable.</td>
</tr>
<tr>
<td></td>
<td>• BOSH UAA</td>
<td>8443</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODB</td>
<td>Deployed service</td>
<td>15672 (RabbitMQ</td>
<td>One-way</td>
<td>This connection is for administrative tasks. Avoid opening general</td>
</tr>
<tr>
<td>instances</td>
<td>instances</td>
<td>Management UI)</td>
<td></td>
<td>use, app-specific ports for this connection.</td>
</tr>
<tr>
<td>ODB</td>
<td>PAS (or Elastic)</td>
<td>8443</td>
<td>One-way</td>
<td>The default port is not configurable.</td>
</tr>
</tbody>
</table>

© Copyright Pivotal Software Inc, 2013-2019
<table>
<thead>
<tr>
<th><strong>Errand VMs</strong></th>
<th><strong>BOSH Agent</strong></th>
<th><strong>Deployed apps on PAS (or Elastic Runtime)</strong></th>
<th><strong>PAS (or Elastic Runtime)</strong></th>
<th><strong>Deployed apps on PAS</strong></th>
</tr>
</thead>
</table>
| - PAS (or Elastic Runtime)  
- ODB  
- Deployed Service Instances | BOSH Director | Deployed service instances | ODB | Runtime CredHub |
| - 8443  
- 8080  
- 15672 (RabbitMQ Management UI)  
- 5671-2 (AMQP/AMQPS) | 4222 | - 15672 (RabbitMQ Management UI)  
- 5671-2 (AMQP/AMQPS)  
- 61613-4 (STOMP/STOMPS)  
- 1883, 8883 (MQTT/MQTTS) | 8080 | 8844 |
| One-way | Two-way | One-way | One-way | One-way |
| The default port is not configurable. | 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. | This connection is for general use, app-specific tasks. Avoid opening administrative ports for this connection. | This port may be different for individual services. This port may also be configurable by the operator if allowed by the tile developer. | This port is needed if secure service instance credentials are enabled. For information, see [Beta On Demand - Secure Service Instance Credentials with Runtime CredHub](#). |
Deploying the RabbitMQ Pre-Provisioned Service

Default Deployment

Deploying RabbitMQ for Pivotal Cloud Foundry (PCF) through Ops Manager deploys a RabbitMQ cluster of 3 nodes by default.

The deployment includes a single load balancer haproxy which spreads connections on all of the default ports, for all of the shipped plugins across all of the machines within the cluster.

The deployment occurs in a single availability zone (AZ).

The default configuration is for testing purposes only and Pivotal recommends that customers have a minimum of 3 RabbitMQ nodes and 2 HAProxy nodes.

Considerations for this deployment

- Provides high availability for the RabbitMQ cluster
- Queues must be configured to be high availability as they are placed on one node by default
- Customers should decide which partition behavior is best suited to their use case. For two nodes ‘automatic’ is preferred.
- HAProxy is a single point of failure (SPOF)
- The entire deployment is in a single AZ, which does not protect against external failures from failures in hardware, networking, etc.

Recommended Deployment
Pivotal recommends that RabbitMQ for PCF is deployed across at least two AZs. Scale RabbitMQ server nodes to an odd number that is greater than or equal to three.

Only use replication of queues where required as it can have a big impact on system performance.

The HAProxy job instance count should also be increased to match the number of AZs to ensure there is a HAProxy located in each AZ. This removes the HAProxy SPOF and provides further redundancy.

The diagram above shows that you can now suffer the failure of a single HAProxy and single RabbitMQ node and still keep your cluster online and applications connected.

Upgrading to this deployment from a single AZ deployment

It is not possible to upgrade to this setup from the default deployment across a single AZ.

This is because the AZ setup cannot be changed after the tile has been deployed for the first time. This is to protect against data loss when moving jobs between AZs.

Upgrading to this deployment from a multi AZ deployment

If you have deployed the tile across two AZs, but with a single HAProxy instance, you can migrate to this setup by deploying an additional HAProxy instance through Ops Manager. New or re-bound applications to the RabbitMQ service see the IPs of both HAProxies immediately. Existing bound applications will continue to work, but only using the previously deployed HAProxy IP Address. They can be re-bound as required at your discretion.

Considerations for this deployment

- Requires IaaS configuration for AZs ahead of deploying the RabbitMQ tile
- Application developers are handed the IPs of each deployed HAProxy in their environment variables
- Queues must be configured to be high availability as they are placed on one node by default
Customers should decide on which partition behavior is best suited to their use case. For three or more nodes ‘pause_minority’ is preferred.

Advanced Deployment

This deployment builds upon the above recommended deployment and so follows the same upgrade paths.

This allows you to replace the use of HAPrxy with your own external load balancer.

You might choose to do this to remove any knowledge of the topology of the RabbitMQ setup from application developers.

Advantages

- Application developers do not need to handle multiple IPs for the HAPrxy jobs in their applications

Disadvantages

- The load balancer needs to be configured with the IPs of the RabbitMQ Nodes. These are only be known after the deployment is finished. The IPs should remain the same during subsequent deployments but there is a risk they might change.

Upgrading to this deployment from the recommended deployment

It is possible to first deploy with multiple HAPrxy jobs, as per the recommended deployment, and later use your own external load balancer.

This can be achieved without downtime to your applications. Follow these steps to do so:

1. Configure your external load balancer to point to the RabbitMQ Node IPs.

2. Configure the DNS name or IP address for the external load balancer (ELB) on the RabbitMQ tile in Ops Manager.

3. Deploy the changes. Any new instances of the RabbitMQ service or any re-bound connections will use the DNS name or IP address of the ELB in their VCAP_SERVICES. Any existing instances will continue to use the HAPrxy IP addresses in their VCAP_SERVICES.

4. Phase the re-binding of existing applications to update their environment variables.

5. After all applications are updated, reduce the instance count of the HAPrxy job in Ops Manager to 1.

6. Deploy the changes.

This approach works as any existing bound applications have their VCAP_SERVICES information cached in Cloud Controller and are only updated by a re-
Downgrading from this deployment to the recommended deployment

If you are currently using an external load balancer, then you can move back to using HAProxies instead.

You can achieve this by following the above steps in reverse order and re-instating the HAProy job.

Resource requirements

The following table shows the default resource and IP requirements for installing the tile:

<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>RabbitMQ</td>
<td>RabbitMQ node</td>
<td>3</td>
<td>2</td>
<td>8192</td>
<td>16384</td>
<td>30720</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>HAPoxy for RabbitMQ</td>
<td>1</td>
<td>1</td>
<td>2048</td>
<td>4096</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>RabbitMQ service broker</td>
<td>1</td>
<td>1</td>
<td>2048</td>
<td>4096</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RabbitMQ</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>RabbitMQ</td>
<td>Broker Deregistrar</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>RabbitMQ</td>
<td>Smoke Tests</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>RabbitMQ</td>
<td>RabbitMQ on-demand broker</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>8192</td>
<td>1024</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>Register On-Demand Service Broker</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>RabbitMQ</td>
<td>Deregister On-Demand Service Broker</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>RabbitMQ</td>
<td>Delete All Service Instances</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>RabbitMQ</td>
<td>Upgrade All Service Instances</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>2048</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes:

- The number of RabbitMQ nodes can be increased if required.
- Changing the number of RabbitMQ nodes when the erlang cookie is not defined will restart the cluster. Check here for more information.
Installing and Configuring the On-Demand Service

This topic provides instructions to Pivotal Cloud Foundry (PCF) operators about how to install, configure, and deploy the RabbitMQ for PCF tile to provide on-demand service.

The RabbitMQ open source product provides additional documentation. For more information about getting started with RabbitMQ and ensuring production readiness, see the Production Checklist in the RabbitMQ Documentation.

Note: For instructions on how to install, configure, and deploy the RabbitMQ for PCF tile as a pre-provisioned service, see Installing and Configuring the Pre-Provisioned Service.

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.

Prerequisites for Deploying the On-Demand Service

Before deploying RabbitMQ for PCF as an on-demand service, you must ensure that the required network rules are in place to allow various components to communicate.

See Required Networking Rules for On-Demand Services for details on the network connections that must be open to enable the on-demand service.

For information about the on-demand service architecture, see On-Demand Service Architecture.

Download and Install RabbitMQ for PCF

1. Download the product file from Pivotal Network.

2. Navigate to the Ops Manager Installation Dashboard and click Import a Product to upload the product file.

3. Under the Import a Product button, click + next to the version number of RabbitMQ for PCF. This adds the tile to your staging area.

4. Click the newly added RabbitMQ for PCF tile. This lets you begin configuring the tile. The installation is complete when you apply the changes from the configuration.

Configure On-Demand RabbitMQ for PCF

The configuration screen below appears when you click the RabbitMQ for PCF tile in Ops Manager. An orange circle beside a tab indicates that you must complete a configuration in the tab. A green checkmark indicates that the tab is preconfigured and you may optionally change its settings.
Which Settings Tabs to Configure for the On-Demand Service

Configure the following tabs for the on-demand service:

<table>
<thead>
<tr>
<th>RabbitMQ Settings Tab</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign AZs and Networks</td>
<td>Configure AZs and Networks</td>
</tr>
<tr>
<td>Pre-Provisioned RabbitMQ</td>
<td>Configure Admin Credentials and Metrics Polling Interval</td>
</tr>
<tr>
<td>Networking</td>
<td>Networking</td>
</tr>
</tbody>
</table>
Configure AZs and Networks

Follow the steps below to configure the AZs and networks.

1. Click **Assign AZs and Networks**.

   **Important:** You cannot change the regions or networks after you have clicked **Apply Changes** in the **Apply Changes from Your Configuration** below.

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place singleton jobs in</td>
<td>Select the region that you want for singleton VMs. PCF creates the RabbitMQ broker in this AZ.</td>
</tr>
<tr>
<td>Balance other jobs in</td>
<td>Select additional region. This selection does not affect the on-demand RabbitMQ for PCF service.</td>
</tr>
<tr>
<td>Network</td>
<td>Select a network for the RabbitMQ On-Demand Broker. This should be a separate network from the one you select for <strong>Service Network</strong>. For more information about the Default Network, see <a href="#">Default Network and Service Network</a>. Typically, you select the network used for the Pivotal Application Service (PAS) or Elastic Runtime components.</td>
</tr>
<tr>
<td>Service Network</td>
<td>Select a separate network that the on-demand service instances run on. A typical practice is to put all on-demand services on a single network, separate from the network that Pivotal Application Service (PAS) or Elastic Runtime and the On-Demand Broker run on. For information about the Service Network, see <a href="#">Default Network and Service Network</a>. This field is also required for the pre-provisioned service, though in that case, it doesn’t matter which network you select.</td>
</tr>
</tbody>
</table>

   **WARNING:** Changing the Network or Service Network after you have configured them, or changing their IP configurations, results in a failed deployment. For more information, see [Changing Network or IP Addresses Results in a Failed Deployment](#).

3. Click **Save**.

Configure Admin Credentials and Metrics Polling Interval

On the **Pre-Provisioned RabbitMQ** tab, you must configure two items only for the on-demand service:

- Specify admin credentials for using the RabbitMQ Management Dashboard.
- Enter a metrics polling interval.

Specify Admin User Credentials

In the **RabbitMQ admin user credentials** field of the Pre-Provisioned RabbitMQ tab, enter an admin username and password:
You can use a combination of upper or lowercase alphanumerics and supported special characters: `[]^_!"#$%&()*+,-./:;<=>?`.

This grants you full admin access to the RabbitMQ Management UI.

Enter a Metrics Polling Interval

In the Metrics polling interval field, set a metrics polling interval:

The default setting is 30 seconds for all deployed components. Pivotal recommends that you do not change this interval. In order to avoid overwhelming components, do not set this below 10 seconds.

Changing this setting affects all deployed instances.

For more information, see What are Metrics.

Configure Logging to Monitor RabbitMQ for PCF

Pivotal recommends that you configure logging to monitor the health of RabbitMQ for PCF. Follow Set Up Syslog Forwarding to configure logging.

Networking

To specify a static IP address for the On Demand Service Broker, do the following:

1. Select Networking.

2. Enter an IP address in the On Demand Service Broker Static IP field. This IP address is assigned to your On Demand Service Broker node. BOSH allocates an IP address if the field is left blank.

3. Click Save.

Configure Global Settings

Follow the steps below to configure global settings.

1. Click Global Settings for On-Demand Plans.
2. Configure the following:

- **Service instance quota** min: 0, max: 50 set the total number of on-demand service instances which can be deployed. For more information, see [Setting Limits for On-Demand Service Instances](#).

- **VM options**:

  **Allow outbound internet access (IaaS-dependent)**. This checkbox must be ticked to allow external log forwarding, sending backup artifacts to external destinations, and communicating 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.

  **(Beta) Shareable Instances**: Click Yes to enable the beta feature for sharing instances.

  Note: This is a beta feature based on an experimental feature in Cloud Foundry. Use the feature at your own risk in non-production environments. If you try this feature, please send your comments and feedback to [PCF Feedback List](#).
Sharing a service instance between spaces, allows apps in different spaces to share databases, messaging queues, and many other types of services. For more information, see [Sharing Service Instances (Beta)](#).

* (Beta) On Demand - Secure Service Instance Credentials with Runtime CredHub: For on-demand services instances, click **Yes** to secure credentials with CredHub.

3. Click **Save**.

### Configure the Service Plan

To enable the on-demand service, you must configure at least one on-demand plan.

- You can configure up to five on-demand plans: **On Demand Instance: Plan 1** – **On Demand Instance: Plan 5**.
- All on-demand plans can be configured to have 1, 3, 5, or 7 RabbitMQ nodes.
- If the on-demand service is not enabled, the on-demand broker is deployed alongside the RabbitMQ installation, but it is not available in the Marketplace.

**Note:** You must fully configure **On Demand Instance: Plan 1** even if you disable access to this plan (see **CF Service Access** in the table below).

1. Choose the on-demand service instance you want to configure:

   **On Demand Instance: Plan 1** (complete required fields even if you disable this plan):
Plan 1 Configuration

Please read the documentation before changing any of these settings, as improper use can lead to data loss.

CF Service Access*

Enable Service Access

Plan Name *

single-node

Plan Description *

Single node RabbitMQ dedicated instance

The plan description that will appear in the CF marketplace

Plan Features *

RabbitMQ

Plan Quota  (min: 0, max: 50) *

10

Number of Nodes  (min: 1, max: 7) *

1

Network Partition Behaviour *

pause_minority

AZ Placement *

- us-central1-a
- us-central1-b
- us-central1-c

RabbitMQ VM Type *

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

Persistent Disk Type *

2 GB

I acknowledge that I have configured the Persistent Disk Size to be at least 2x the amount of RAM of the selected VM type *

- Acknowledge

Save

On Demand Instance: Plan 2, 3, 4, and 5:

Plan 2 Configuration
Please read the documentation before changing any of these settings, as improper use can lead to data loss.

**Enable This Plan**
- Plan Disable
- Plan Enable

**CF Service Access**
- Enable Service Access

**Plan Name**
- cluster

**Plan Description**
- RabbitMQ dedicated cluster

The plan description that will appear in the CF marketplace

**Plan Features**
- RabbitMQ

**Plan Quota** (min: 0, max: 50)
- 10

**Number of Nodes** (min: 1, max: 7)
- 3

**Network Partition Behaviour**
- pause_minority

**AZ Placement**
- us-central1-a
- us-central1-b
- us-central1-c

**RabbitMQ VM Type**
- micro (cpu: 1, ram: 1 GB, disk: 8 GB)

**Persistent Disk Type**
- 2 GB

I acknowledge that I have configured the Persistent Disk Size to be at least 2x the amount of RAM of the selected VM type.
- Acknowledge

**Save**
2. Configure the fields as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable This Plan</td>
<td>(Plans 2 - 5 only) To enable, select Plan Enable.</td>
</tr>
<tr>
<td>CF Service Access</td>
<td>Enable or disable access to this plan, or leave access unchanged. If you enable Plan 1, the default setting for Plans 2 - 5 is Enable Service Access. If you change this default setting, the smoke tests fail. Therefore, if you enable Plan 1 and want to change this default, before doing so, set the On-Demand Instance Smoke Tests errand to Off. For more information, see Errands.</td>
</tr>
<tr>
<td>Plan Name</td>
<td>Accept the default or enter a name. This is the name that appears in the Marketplace.</td>
</tr>
<tr>
<td>Plan Description</td>
<td>Accept the default or enter a description. This description appears in the Marketplace.</td>
</tr>
<tr>
<td>Plan Features</td>
<td>Accept the default or enter a description. This description appears in Apps Manager.</td>
</tr>
<tr>
<td>Service Instance Quota</td>
<td>Enter the maximum number of on-demand service instances that can be available at one time. For more information, see Setting Limits for On-Demand Service Instances.</td>
</tr>
<tr>
<td>Number of Nodes</td>
<td>Enter 1, 3, 5 or 7. This setting only affects new service instances. Previously deployed service instances are not updated.</td>
</tr>
<tr>
<td>Network Partition Behaviour</td>
<td>Select pause_minority or autoheal. Pivotal recommends using pause minority. For more information, see Consistency or Availability Tradeoff.</td>
</tr>
<tr>
<td>RabbitMQ VM Type</td>
<td>Select a large VM type. The plan creates a service instance of this size. For more information, see Understanding RabbitMQ VM Types and Persistent Disk Size below.</td>
</tr>
<tr>
<td>Persistent Disk Type</td>
<td>This is where RabbitMQ pages messages to disk. Service instance deployments fail if this value is less than twice the volume of RAM of the selected RabbitMQ VM Type. For more information, see Understanding RabbitMQ VM Types and Persistent Disk Size below.</td>
</tr>
<tr>
<td>AZ Placement</td>
<td>This field is available after you complete the Assign AZs and Networks page.</td>
</tr>
<tr>
<td></td>
<td>- For a single-node plan, select one or more AZs.</td>
</tr>
<tr>
<td></td>
<td>- For a plan containing multiple nodes, select only one AZ. Pivotal recommends this for multi-node plans to minimize risks due to network latency and partitions. See Network Latency and Consistency or Availability Tradeoff for details.</td>
</tr>
</tbody>
</table>

3. Click Save.

Determine which AZs a Service Instance Uses

**Important:** If you change this configuration after you have selected AZs and deployed service instances, existing instances are not placed in the newly configured AZs when the Upgrade All Service Instances errand is run. This prevents re-creation of the VMs in different AZs, which can lead to data loss. All new service instances, however, will be created in the newly configured AZs.

To determine which AZs a service instance is placed in, do one of the following:

- Retrieve the service GUID using the `cf service SERVICE_INSTANCE --guid` command and then run the BOSH Instances command for the service-instance_GUID deployment.
- With syslog forwarding enabled, inspect the service broker logs when running the Upgrade All Service Instances errand. For each existing service
instance, the log message includes the service instance GUID and the AZs the service instance is running in.

Understanding RabbitMQ VM Types and Persistent Disk Size

The RabbitMQ VM Type and Persistent disk type are required fields on the service plan configuration pages. If you are installing on PCF v2.0 or later, these properties are pre-configured by default.

Pivotal recommends that the value of Persistent disk type be twice the amount of RAM of the selected RabbitMQ VM Type.

- You can change the RabbitMQ VM type and the size of the persistent disk that is attached to the RabbitMQ instances. For example, if you are running out of disk space you might want to increase the persistent disk size by changing the Persistent disk type field. If you make changes, ensure that the persistent disk size is still twice the size of the RAM of the RabbitMQ VM type.
- RabbitMQ raises alarms when disk space drops below the configured limit. Incorrect disk sizes might cause the deployed instance not to start. RabbitMQ declines to start if there is not enough space available according to the threshold.
- On-Demand instances are configured with a threshold set to the 150% of the memory (RAM) of the VM. Use the following table as a guide when selecting the size of the persistent disk.

The following table shows an example of possible RAM values, absolute minimal value below which RabbitMQ declines to start, and the disk size suggested for an average use case.

<table>
<thead>
<tr>
<th>RAM</th>
<th>Free disk alarm threshold (1.5xRAM)</th>
<th>Suggested disk size (2xRAM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 GB</td>
<td>15 GB</td>
<td>20 GB</td>
</tr>
<tr>
<td>16 GB</td>
<td>24 GB</td>
<td>32 GB</td>
</tr>
<tr>
<td>32 GB</td>
<td>48 GB</td>
<td>64 GB</td>
</tr>
</tbody>
</table>

Minimum resources required for each RabbitMQ VM:

- CPU: 2
- RAM: 1 GB
- Ephemeral disk: 2 GB
- Persistent disk: 4 GB

For more information, see the following:

- Memory and Disk Alarms
- Disk Alarms

For information on all preconfigured settings, see Things that are Preconfigured.

Verify the Stemcell

For Ops Manager v2.0:

1. Click Stemcell.
2. Verify and, if necessary, import a new stemcell version. For more information, see the information about importing the stemcell for your IaaS: AWS, Azure, GCP, or vSphere.

For Ops Manager v2.1:

1. Follow the procedure outlined in Importing and Managing Stemcells.

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. Click Apply Changes.

Errands

When deploying or updating RabbitMQ for PCF, Ops Manager can optionally run a series of Post-Deploy Errands. An example is the Smoke Tests errand, which checks the health of the RabbitMQ cluster after a deploy or upgrade.

You can decide whether to run errands by toggling them on or off before an update. This is a one-time setting on the installation dashboard:

![Pending Changes](pending_changes.png)

- **Important:** As of RabbitMQ for PCF v1.9.0, all post-deploy errands are on by default. Pivotal recommends keeping these defaults, because the smoke tests can encounter unexpected issues, and on-demand instances of RabbitMQ for PCF might fall behind if the Upgrade All Service Instances errand is not on by default.

However, if necessary, you can change these defaults by clicking Errands in the RabbitMQ for PCF Settings tab.

For more information on errand run rules, see Errand Run Rules.

RabbitMQ for PCF errands are colocated with their brokers to decrease errand run time and VM footprint. In earlier releases, a new VM was deployed for each errand. For more information about errands, see Errands.
### Post-Deploy Errands

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker Registrar</td>
<td>Makes the pre-provisioned RabbitMQ service plans available in the Marketplace</td>
</tr>
<tr>
<td>Smoke Tests</td>
<td>Checks that a pre-provisioned RabbitMQ service instance can be bound to a Cloud Foundry app, and that the app can publish and subscribe to a RabbitMQ cluster. See <a href="#">Pre-Provisioned Instance Smoke Tests</a>.</td>
</tr>
<tr>
<td>Register On-Demand Service Broker</td>
<td>Makes the on-demand RabbitMQ service plans available in the Marketplace. If you change the Service Plan Configuration, you must run this errand in order for the changes to be reflected in the Marketplace.</td>
</tr>
<tr>
<td>On-Demand Instance Smoke Tests</td>
<td>Checks that on-demand RabbitMQ service instances can be bound to a Cloud Foundry app, and that the app can publish and subscribe to a RabbitMQ cluster. See <a href="#">On-Demand Instance Smoke Tests</a> below.</td>
</tr>
<tr>
<td>Upgrade All Service Instances</td>
<td>On-Demand instances are updated and redeployed if there are changes to on-demand plan settings or the tile is upgraded. If this errand is set to Off or When Changed, updates to on-demand plan settings will not be applied to existing service instances. Pivotal strongly recommends that this errand is configured to always run.</td>
</tr>
</tbody>
</table>

### Pre-Delete Errands

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker Deregistrar</td>
<td>Removes the pre-provisioned RabbitMQ service from the Marketplace and deletes all associated service instances and bindings</td>
</tr>
<tr>
<td>Delete All Service Instances</td>
<td>Unbinds and deletes existing on-demand service instances. The duration of this errand depends on the number of deployed on-demand instances.</td>
</tr>
<tr>
<td>Deregister On-Demand Service Broker</td>
<td>Removes the on-demand RabbitMQ service from the Marketplace</td>
</tr>
</tbody>
</table>

### On-Demand Instance Smoke Tests

Smoke tests only run against Plan 1. For more information about the smoke tests process, see [Smoke Tests](#).

### Create an Admin User for a Service Instance

If you want to give app developers admin privileges to the RabbitMQ Management UI, you can create an admin user for a service instance and share the user credentials with app developers.

Both operators and app developers can use this procedure.

To create an admin user on a RabbitMQ instance do the following:

1. Run this command to create a service key:

   ```bash
   cf create-service-key SERVICE_INSTANCE SERVICE_KEY -c "{"tags": "administrator"}"
   ``

   where:

   - `SERVICE_INSTANCE` is the name you supplied when you ran `cf create-service`.
   - `SERVICE_KEY` is a name you choose to identify the service key.

   For example:

   ```bash
   $ cf create-service-key my-instance my-admin-key -c "{"tags": "administrator"}"
   Creating service key my-admin-key for service instance my-instance as user@example.com...
   OK
   ```

2. Run this command to get the admin user credentials:
cf service-key SERVICE_INSTANCE SERVICE_KEY  where the variables are the same as above.

This returns a Dashboard URL containing the admin credentials, which can be used to access the management UI. For example:

```
cf service-key my-instance my-admin-key
```

Getting key my-admin-key for service instance my-instance as user@example.com...

```
{  
  "dashboard_url": "https://my-instance.bosh-lite.com/#/login/admin-username/admin-password",  
  "username": "admin-username",  
  "password": "admin-password",  
...  
}
```

### RabbitMQ Server Settings That Cannot Be Disabled

The following plugins are enabled by default and cannot be disabled:

- rabbitmq_management
- rabbitmq_federation
- rabbitmq_federation_management
- rabbitmq_shovel
- rabbitmq_shovel_management
Installing and Configuring the Pre-Provisioned Service

This topic provides instructions to Pivotal Cloud Foundry (PCF) operators about how to install, configure, and deploy the RabbitMQ for PCF tile to provide a pre-provisioned service.

The RabbitMQ open source product provides additional documentation. For more information about getting started with RabbitMQ and ensuring production readiness, see the Production Checklist in the RabbitMQ Documentation.

Note: For instructions about how to install, configure, and deploy the RabbitMQ for PCF tile as an on-demand service, see Installing and Configuring RabbitMQ for PCF as an On-Demand Service.

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

1. Download the product file from Pivotal Network.

2. Navigate to the Ops Manager Installation Dashboard and click Import a Product to upload the product file.

3. Under the Import a Product button, click + next to the version number of RabbitMQ for PCF. This adds the tile to your staging area.

4. Click the newly added RabbitMQ for PCF tile. This lets you begin configuring the tile. The installation is complete when you apply the changes from the configuration.

Configure Pre-Provisioned RabbitMQ for PCF

The configuration screen below appears when you click the RabbitMQ for PCF tile in Ops Manager. An orange circle beside a tab indicates that you must complete a configuration in the tab. A green checkmark indicates that the tab is preconfigured and you may optionally change its settings.
Which Settings Tabs to Configure for the Pre-Provisioned Service

Configure the following tabs for the pre-provisioned service:

<table>
<thead>
<tr>
<th>RabbitMQ Settings Tab</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign AZs and Networks</td>
<td>Assign AZs and Networks</td>
</tr>
<tr>
<td>Pre-Provisioned RabbitMQ</td>
<td>RabbitMQ</td>
</tr>
<tr>
<td>Networking</td>
<td>Networking</td>
</tr>
</tbody>
</table>
Assign AZs and Networks

Follow the steps below to configure the AZs and networks.

1. In the Settings screen, click Assign AZs and Networks.

   **Important:** You cannot change the regions or networks after you have clicked Apply Changes in the final step below.

2. Configure the fields on the Assign AZs and Networks as follows. All fields are required, though some do not apply to the pre-provisioned service.

<table>
<thead>
<tr>
<th>Required Fields</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place singleton jobs in</td>
<td>Select a region. This selection only affects the on-demand service.</td>
</tr>
<tr>
<td>Balance other jobs in</td>
<td>Select additional region(s). This selection only affects the pre-provisioned service.</td>
</tr>
<tr>
<td>Network</td>
<td>Select a network for the RabbitMQ Broker. This should be a separate network from the one you select for Service Network. This network is represented by the Default Network, described in Default Network and Service Network. Typically, you select the network used for the Pivotal Application Service (PAS) or Elastic Runtime components.</td>
</tr>
<tr>
<td>Service Network</td>
<td>Select a network. This selection only affects the on-demand service.</td>
</tr>
</tbody>
</table>

   **WARNING:** Changing the Network after you have configured it, or changing the IP configuration, results in a failed deployment. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

3. Click Save.

RabbitMQ

To configure the following sections in the Pre-Provisioned RabbitMQ tab, in the Settings screen, click Pre-Provisioned RabbitMQ.

RabbitMQ Admin User Credentials

In the RabbitMQ admin user credentials field of the RabbitMQ pane, enter an admin username and password:

You can use a combination of upper or lowercase alphanumerics and supported special characters: `!@#$%^&*()-_=+`.

**Note:** You cannot use back quote ` or single quote `.

This grants you full admin access to the RabbitMQ Management UI.
Plugins

Choose which plugins you want to enable in this section of the Pre-Provisioned RabbitMQ tab.

You must leave the rabbitmq_management plugin enabled for this product to work.

For more information about RabbitMQ plugins, see the RabbitMQ documentation.

HAProxy Ports

Enter the ports HAProxy should load balance to the RabbitMQ nodes in this section of the Pre-Provisioned RabbitMQ tab:

- All the default ports of all the available plugins are load-balanced by default. However, if you install extra protocol plugins, or provide a custom configuration that changes the ports RabbitMQ listens on, then you must update the list of load-balanced ports.
- You must leave the management plugin listening on port 15672 and load balance that port.
- If you change the topology of your RabbitMQ cluster, the HAProxy is automatically reconfigured during the deployment.

SSL

(Optional) Provide SSL certificates and keys for use by the RabbitMQ cluster in this section of the Pre-Provisioned RabbitMQ tab:
SSL is simultaneously provided for AMQPS, STOMP and MQTT. No other plugins are automatically configured for use with SSL.

- If you provide SSL keys and certificates, non-SSL support is disabled. If you previously deployed this service without SSL support and have apps connected to the service, these apps lose their connections and must reconnect using SSL.
- SSL settings are applied equally across all VMs in the cluster.
- You can provide more than one CA certificate.

For more information about SSL support, see the RabbitMQ documentation.

Erlang Cookie

(Optional) Provide an Erlang cookie to be used by the cluster in this section of the Pre-Provisioned RabbitMQ tab. This is useful if you want to connect directly to the RabbitMQ cluster, for example with `rabbitmqctl`, or to connect other VMs running Erlang.

**Note:** Leaving this field blank is a security risk. See Security Issue with the Tile Generated Erlang Cookie below.

Known Issues with Erlang Cookie

There are three known issues associated with the Erlang cookie:

- Security Issue with the Tile Generated Erlang Cookie
- Cluster Scaling Known Issue
- Changing the Erlang Cookie Value Known Issue
Security Issue with the Tile Generated Erlang Cookie

If you leave the Erlang cookie field blank, the tile generates the cookie in a way that can be reverse-engineered. For more information about this security issue, see [CVE-2018-1279: RabbitMQ cluster compromise due to deterministically generated cookie](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-1279).

To avoid this issue, set the Erlang cookie to a secure password value. This requires cluster downtime, see Changing the Erlang Cookie Value Known Issue below.

Cluster Scaling Known Issue

If you have not set the Erlang cookie and you want to scale out your cluster size without downtime, follow these steps:

1. Follow the steps in the link below, up to and including the section Log in to the BOSH Director: Advanced Troubleshooting with the BOSH CLI.
2. Run the following command:
   
   ```bash
   bosh ssh rabbitmq-server/0
   ```
3. Run the following commands:
   
   ```bash
   sudo -i
   echo $(cat /var/vcap/store/rabbitmq/.erlang.cookie)
   ```
4. Paste the value returned from the last command into the Erlang cookie field in the Pre-Provisioned RabbitMQ tab. This field is shown above.
5. To increase the size of your cluster, navigate to the Resource Config tab and, in the first row, raise the value for the number of Instances of the RabbitMQ node.
6. Return to the Ops Manager Installation Dashboard, and click Apply Changes.

   **Note:** BOSH tells you that the cookie has changed—this is because the default value in the manifest is empty, which results in an auto-generated cookie. However, the value of the cookie on the server remains the same, so the known issue below does not apply.

Changing the Erlang Cookie Value Known Issue

Changing the Erlang cookie value requires cluster downtime. Pivotal strongly recommends that you do not change anything else during this time, because it is possible for the configuration to be inconsistently applied during this process.

The deployment might fail after this process. If so, redeploying fixes the issue.

RabbitMQ Configuration

(Optional) Provide a full rabbitmq.config file by pasting its contents in the RabbitMQ configuration field in the Pre-Provisioned RabbitMQ tab. This file is then provided to all the nodes in the cluster.

The input in this field must be Base64 encoded.

For example, suppose you want to configure the rates_mode of the rabbitmq_management stats below:

```json
|
|
---

[rabbitmq_management,

  [rates_mode, detailed]
]
```
1. Encode the file into Base64:

```
WwogIHtyYWJiaXRtcV9tYW5hZ2VtZW50LCBbCiAgICB7cmF0ZXNfbW9kZSwgZGV0YWlsZWR9CiAgXX0KXS4K
```

2. Paste the above into the RabbitMQ configuration field:

You can see an example rabbitmq.config file here. For more information about the RabbitMQ configuration, see the RabbitMQ documentation.

**TLS Support**

Configure TLS in this section of the Pre-Provisioned RabbitMQ tab:

- TLS v1.0 is disabled by default, due to security issues.
- TLS v1.1 and v1.2 are enabled by default and can be turned on and off.

**External Load Balancer**

(Optional) Enter a DNS name or IP address of an external load balancer to be returned in the binding credentials (VCAP_SERVICES) to app developers. Enter this in this section of the Pre-Provisioned RabbitMQ tab:

If you configure an external load balancer, to avoid an unnecessary VM deployment, in the Resource Config tab set the HAProxy for RabbitMQ instance count to 0.

**Metrics Polling Interval**

The metrics polling interval is set in this section of the Pre-Provisioned RabbitMQ tab:

The default setting is 30 seconds for all deployed components. Pivotal recommends that you do not change this interval. In order to avoid overwhelming components, do not set this below 10 seconds.
Changing this setting affects all deployed instances.

Disk Free Alarm Limit

Choose how much disk space RabbitMQ attempts to keep free at any given time in this section of the Pre-Provisioned RabbitMQ tab:

RabbitMQ periodically checks if there is sufficient free space on disk. If there is not, RabbitMQ temporarily stops accepting new messages. This gives your apps time to consume existing messages, and thus free up some disk space. The RabbitMQ tile provides four options for this value:

- **50MB** is the minimum value. (Not Recommended)
  - Selecting **50MB** is not recommended and can cause data loss. For more information, see [Dangers of Setting This Value Too Low](#) and [When to Use the 50MB Value](#).

- **100% Memory** ensures that at the time when RabbitMQ checks the available disk, there must be enough space for RabbitMQ to page all memory-based messages out to disk.

- **150% Memory** is recommended. This is because it is possible that in between disk-space checks, RabbitMQ may:
  - Write persistent messages to disk (using up some disk space).
  - Accept more memory-based messages into various queues.
  - Page all memory-based messages to disk.

  In the above situations, RabbitMQ might require more free disk than it has memory.

- **200% Memory** is a conservative value used when the operator wants higher confidence that RabbitMQ never runs out of disk space.

For more information about disk alarms, see the [RabbitMQ documentation](#).

**Dangers of Setting This Value Too Low**

If the disk of a given RabbitMQ node completely fills while RabbitMQ is running, that node crashes. This can lead to data loss, and loss of availability.

RabbitMQ reserves the right to page any and all messages in memory (even transient messages) to disk at any time. You must set your disk free alarm limit high enough to ensure that RabbitMQ always has at least enough space to do this.

**Disadvantages of Setting This Value Too High**

If you set your Disk free alarm limit to a value larger than the size of your persistent disk, then RabbitMQ is not able to free up enough disk space to accept new messages. Ensure that you have a large enough disk to persist all the messages you intend to persist while also leaving enough space free to satisfy the Disk free alarm limit that you choose.

**When to Use the 50MB Value**

Pivotal does not recommend using this value in production. However, if you are experimenting with a development environment you might want to use a small disk to keep down costs, though this increases the possibility that RabbitMQ crashes and loses data.

**Networking**

To specify static IP addresses, do the following:

1. From the Ops Manager Installation Dashboard, click the RabbitMQ tile.
2. In the Settings tab, select Networking.
3. Configure the fields as follows:

- **Pre-Provisioned HAProxy Static IPs**: Enter a comma-delimited list of IP addresses grouped in the order of AZs configured. These IP addresses are assigned to your Pre-Provisioned HAproxy nodes.

- **Pre-Provisioned RabbitMQ Server Static IPs**: Enter a comma-delimited list of IP addresses grouped in the order of AZs configured. These IP addresses are assigned to your Pre-Provisioned RabbitMQ Server nodes.

- **Pre-Provisioned Service Broker Static IP**: Enter an IP address. This address must be in the network range of the network name specified in the **Network** drop-down list in the **Assign AZs and Networks** tab and must not be in the **Service Network**. This IP address is assigned to your Pre-Provisioned Service Broker node.

**Note**: If any of the above fields are left blank, BOSH allocates an IP address.

### Static IP Allocations

Please read the documentation before changing any of these settings, as improper use can lead to data loss.

- Pre-Provisioned HAProxy Static IPs
- Pre-Provisioned RabbitMQ Server Static IPs
- Pre-Provisioned Service Broker Static IP

4. Click **Save**.

### Syslog

To enable monitoring for RabbitMQ for PCF, operators forward the syslog by designating an external syslog endpoint for RabbitMQ component log messages. This endpoint serves as the input to a monitoring platform such as Datadog, Papertrail, or SumoLogic.

To specify the destination for RabbitMQ for PCF log messages, do the following:

1. From the Ops Manager Installation Dashboard, click the RabbitMQ tile.

2. In the RabbitMQ tile, click the **Settings** tab.
3. Click **Syslog**.

4. Configure the fields on the Syslog pane as follows:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syslog address</td>
<td>IP or DNS address of the syslog server</td>
</tr>
<tr>
<td>Syslog port</td>
<td>Port of the syslog server</td>
</tr>
<tr>
<td>Transport protocol</td>
<td>Transport protocol of the syslog server. One of udp, tcp, relp.</td>
</tr>
<tr>
<td>Format for logs</td>
<td>Format for logs. Pivotal recommends <a href="https://tools.ietf.org/html/rfc5424">RFC 5424</a>, but Legacy Format can be used for compatibility reasons.</td>
</tr>
<tr>
<td>Enable TLS</td>
<td>Enable TLS to the syslog server.</td>
</tr>
<tr>
<td>Permitted Peer</td>
<td>If there are several peer servers that can respond to remote syslog connections, then you may provide a wildcard in the domain, such as *.example.com.</td>
</tr>
<tr>
<td>Custom CA Certificate</td>
<td>If the server certificate is not signed by a known authority, for example, an internal syslog server, provide the CA certificate of the log management service endpoint.</td>
</tr>
</tbody>
</table>
5. Click Save.

6. Return to the Ops Manager Installation Dashboard and click Apply Changes to redeploy with the changes.

Global Settings

Follow the steps below to enable the shareable instances beta feature. Sharing a service instance between spaces, allows apps in different spaces to share databases, messaging queues, and many other types of services. For more information, see Sharing Service Instances (Experimental).

1. Click Global Settings.

2. Select (Beta) Shareable Instances to enable the beta feature for sharing instances.

   App Developers can have the ability to share their Cloud Foundry Service Instances across Orgs and Spaces

3. Click Save.

Errands

(Optional) In the Errands tab, choose the defaults for when errands run.

Errands can be thought of as tasks. For example, when deploying or updating RabbitMQ for PCF, Ops Manager can optionally run a series of post-deploy errands. An example is the Smoke Tests errand, which checks the health of the RabbitMQ cluster after a deploy or upgrade.

You can decide whether to run post-deploy errands by toggling them on or off before you click Apply Changes to update a configuration in the Ops Manager Installation Dashboard:

This is a one-time action before an update. You can change the above defaults in the Errands tab, as well as the defaults for pre-delete errands.

Important: In RabbitMQ for PCF v1.9.0 and later, all post-deploy errands are on by default. Pivotal recommends keeping these defaults, because...
For more information on errand run rules, see Errand Run Rules.

Post-Deploy Errands

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Broker Registrar</strong></td>
<td>Makes the pre-provisioned RabbitMQ service plans available in the Marketplace</td>
</tr>
<tr>
<td><strong>Smoke Tests</strong></td>
<td>Checks that a pre-provisioned RabbitMQ service instance can be bound to a Cloud Foundry app, and that the app can publish and subscribe to a RabbitMQ cluster. See Pre-Provisioned Instance Smoke Tests below.</td>
</tr>
<tr>
<td><strong>Register On-Demand Service Broker</strong></td>
<td>Makes the on-demand RabbitMQ service plans available in the Marketplace. If you change the Service Plan Configuration, you must run this errand in order for the changes to be reflected in the Marketplace.</td>
</tr>
<tr>
<td><strong>On-Demand Instance Smoke Tests</strong></td>
<td>Checks that on-demand RabbitMQ service instances can be bound to a Cloud Foundry app, and that the app can publish and subscribe to a RabbitMQ cluster. See On-Demand Instance Smoke Tests.</td>
</tr>
<tr>
<td><strong>Upgrade All Service Instances</strong></td>
<td>On-demand instances are updated and redeployed if there are changes to the Dedicated Instance settings or the tile is upgraded. If this errand is set to Off or When Changed, updates to Dedicated Instance settings will not be applied to existing service instances. Pivotal strongly recommends that this errand is configured to always run.</td>
</tr>
</tbody>
</table>

Pre-Provisioned Instance Smoke Tests

Smoke tests run as a post-deployment errand. For more information about the smoke tests process, see Smoke Tests.

Pre-Delete Errands

Pre-delete errands run after an operator chooses to delete a product in the Ops Manager Installation Dashboard, but before Ops Manager finishes deleting the product.

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Broker Deregistrar</strong></td>
<td>Removes the pre-provisioned RabbitMQ service from the Marketplace and deletes all associated service instances and bindings</td>
</tr>
<tr>
<td><strong>Delete All Service Instances</strong></td>
<td>Unbinds and deletes existing dedicated service instances. The duration of this errand depends on the number of deployed on-demand instances.</td>
</tr>
<tr>
<td><strong>Deregister On-Demand Service Broker</strong></td>
<td>Removes the on-demand RabbitMQ service from the Marketplace</td>
</tr>
</tbody>
</table>

Stemcell

For Ops Manager v2.0:

1. Click Stemcell.

2. Verify and, if necessary, import a new stemcell version. For more information, see the information about importing the stemcell for your IaaS: AWS, Azure, GCP, or vSphere.

For Ops Manager v2.1:

1. Follow the procedure outlined in Importing and Managing Stemcells.

Apply Configuration and Complete the Installation

© Copyright Pivotal Software Inc, 2013-2019
Return to the Ops Manager Installation Dashboard and click **Apply Changes** to complete the installation of RabbitMQ for PCF.

Other Configuration Topics

Connecting to a Highly Available RabbitMQ Cluster

The RabbitMQ tile, allows for a highly available cluster through multiple HAProxy nodes. The **hostnames**, **uris**, and **hosts** properties have been added and should be used in preference over the equivalent singular properties. The singular properties are maintained for backwards compatibility and always contain the first value from the equivalent plural property. The singular properties will eventually be deprecated.

For example, with two HAProxy jobs deployed, the following properties will be present:

```
"hostname": "10.0.0.41",
"hostnames": [
  "10.0.0.41",
  "10.0.0.51"
]
```

**Port to protocol mappings**

- 15672 = Management dashboard
- 5672 = RabbitMQ
- 5671 = RabbitMQ SSL
- 1883 = MQTT
- 8883 = MQTT SSL
- 61613 = STOMP
- 61614 = STOMP SSL
- 15674 = Web STOMP
- 4567 = RabbitMQ Service Broker
- 3457 - 3459 = CF Loggregator

**Security Groups**

To enable access to the RabbitMQ tile service, you must ensure your security group allows access to the HAProxy and RabbitMQ Service Broker VMs configured in your deployment. You can obtain the IP addresses for these from the Ops Manager **Status** page for the RabbitMQ tile. Ensure the following ports are enabled for those VMs:

- 15672
- 5672
- 5671
- 1883
- 8883
- 61613
- 61614
- 15674
- 4567
- 3457 - 3459

The following is a template for configuring your Cloud Foundry security groups:

```
[{
  "protocol": "tcp",
  "destination": "<haproxy-node-IP-addresses>",
  "ports": "5671,5672,1883,8883,61613,61614,15672,15674"},
{
  "protocol": "tcp",
  "destination": "<service-broker-node-IP-addresses>",
  "ports": "4567"}
]`
Application Security Groups

To allow this service to have network access, you must create Application Security Groups (ASGs).

>Note: The service is unusable without Application Security Groups.

Application Container Network Connections

Application containers that use instances of the RabbitMQ 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>HAProxy IPs</td>
<td>5672</td>
<td>tcp</td>
<td>Application containers using AMQP</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>5671</td>
<td>tcp</td>
<td>Application containers using AMQP over SSL</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>1883</td>
<td>tcp</td>
<td>Application containers using MQTT</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>8883</td>
<td>tcp</td>
<td>Application containers using MQTT over SSL</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>61613</td>
<td>tcp</td>
<td>Application containers using STOMP</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>61614</td>
<td>tcp</td>
<td>Application containers using STOMP over SSL</td>
</tr>
<tr>
<td>HAProxy IPs</td>
<td>61613</td>
<td>tcp</td>
<td>Application containers using Web STOMP</td>
</tr>
</tbody>
</table>

Create an ASG named `rabbitmq-app-containers` with the above configuration and bind it to either:

- The appropriate space
- The `default-running` ASG set if you want to provide access to all started apps. Then restart your apps.

If you are using an external load balancer, or have more than one IP address for HAProxy, you must also create egress rules for these. For example:

```
[
  {
    "ports": ["5671-5672"],
    "protocol": "tcp",
    "destination": "10.10.10.32"
  }
]
```

Assigned IPs

RabbitMQ for PCF does not support changing the IP addresses which have been assigned to the RabbitMQ deployments. For example, you cannot change the subnet into which the RabbitMQ cluster was originally provisioned. Doing so causes the deployment to fail. For more information, see Changing Network or IP Addresses Results in a Failed Deployment.

Preserving Dynamically Assigned IPs

You cannot switch from dynamically assigned IP addresses to a different set of static IP addresses. However, you can configure Ops Manager so the current set of dynamically assigned IP addresses always continue to be used. This might be useful when upgrading.

To do this, follow these steps:

1. Go to the Status page in the RabbitMQ tile.
2. Take note of the IP addresses for the RabbitMQ Server and HAProxy for RabbitMQ jobs, in the order nodes appear in the UI.
3. Go to the Settings page, and click Networking.
4. Enter the IP addresses you got from the Status page as a comma-separated list.
5. Click **Save**.

### RabbitMQ Server Settings that Cannot be Overwritten

In all cases:

- `rabbit halt_on_upgrade_failure false`
- `rabbitmq_mqtt subscription_ttl 1800000`
- `log_levels [{connection,info}]`
- `halt_on_upgrade_failure false`
- `(rabbit, [{collect_statistics_interval, 60000}])`
- `(rabbitmq_management, [{rates_mode, none}])`

When SSL is enabled:

- `rabbit tcp_listeners []`
- `rabbit ssl_listeners [5671]`
- `rabbitmq_management listener [{port,15672},{ssl,false}]`
- `rabbitmq_mqtt ssl_listeners [8883]`
- `rabbitmq_stomp ssl_listeners [61614]`
Smoke Tests

RabbitMQ for PCF runs a set of smoke tests during installation to confirm system health.

Smoke Test Steps

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

1. Targets the org and creates a space to run the tests.
2. Deploys an instance of the CF RabbitMQ Example App to this space
3. Creates a RabbitMQ service instance and binds it to the CF RabbitMQ Example App
4. Checks that the CF RabbitMQ Example App can write to and read from the RabbitMQ service instance
5. Cleans up all deployed application and all its service bindings. Finally, the cf space is deleted.

Troubleshooting

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

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.

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 org without specifying a space. To enable global default ASGs, use `cf bind-running-security-group`.

Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF

This topic explains how to monitor the health of the pre-provisioned version of the RabbitMQ for Pivotal Cloud Foundry (PCF) service using the logs, metrics, and Key Performance Indicators (KPIs) generated by RabbitMQ for PCF component VMs.

Pre-provisioned RabbitMQ for PCF components generate many of the same metrics as the on-demand RabbitMQ service components. See Logging and Metrics for general information about logging and metrics in PCF.

Setting up Syslog Forwarding

 Operators can enable log forwarding by configuring an external syslog endpoint for RabbitMQ component log messages. For instructions on setting up syslog forwarding, see Syslog.

If syslog forwarding is enabled, log entries with timestamps are available locally in /var/log/messages. Logs are available under /var/vcap/sys/log/ whether syslog forwarding is enabled or not.

Logging Formats

With pre-provisioned RabbitMQ for PCF logging configured, three types of component generate logs: the RabbitMQ message server nodes, the service broker, and HAProxy. If you have multiple server or HAProxy nodes, you can identify logs from individual nodes by their index, which corresponds to the index of the RabbitMQ VM instances displayed in Ops Manager:

- The logs for RabbitMQ server nodes follow the format [job=rabbitmq-server-partition-GUID index=X]
- The logs for HAProxy nodes follow the format [job=rabbitmq-haproxy-partition-GUID index=X]
- The logs for the RabbitMQ service broker follow the format [job=rabbitmq-broker-partition-GUID index=X]

RabbitMQ and HAProxy servers log at the info level and capture errors, warnings, and informational messages.

For users familiar with documentation for previous versions of the tile, the tag we used to call the app_name is now called the program_name.

The generic log format is as follows:

PRI TIMESTAMP IP_ADDRESS PROGRAM_NAME [job=NAME index=JOB_INDEX id=JOB_ID] MESSAGE

The raw logs look similar to the following:

<7>2017-06-28T16:06:10.733560+00:00 10.244.16.133 vcap.agent [job=rabbitmq-server-partition-GUID index=0] 2017/06/28 16:06:10 CEF:0CloudFoundry:BOSH1:agent_apishell1:duer=dior <86>2017-06-28T16:16:36.646569+00:00 10.244.16.133 usermod [job=rabbitmq-server-partition-GUID index=0] new user: name=bosh_ly0d2rbjr, UID=1001, GID=1003, home=/var/vcap/bosh

Logs sent to external logging tools such as Papertrail may be presented in a different format.

The following table describes the logging tags used in this template:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>This is a value which in future will be used to describe the severity of the log message and which facility it came from.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>This is the timestamp of when the log is forwarded, for example, [2016-08-24T05:14:15.000003]. The timestamp value is typically slightly after when the log message was generated.</td>
</tr>
<tr>
<td>IP_ADDRESS</td>
<td>The internal IP address of server on which the log message originated</td>
</tr>
<tr>
<td>PROGRAM_NAME</td>
<td>Process name of the program generated the message. Same as program_name before v1.9.0. For more information about program name, see RabbitMQ Program Names below.</td>
</tr>
<tr>
<td>NAME</td>
<td>The BOSH instance group name (for example, rabbitmq_server)</td>
</tr>
<tr>
<td>JOB_INDEX</td>
<td>BOSH job index. Used to distinguish between multiple instances of the same job.</td>
</tr>
<tr>
<td>JOB_ID</td>
<td>BOSH VM GUID. This is distinct from the CID displayed in the Ops Manager Status tab, which corresponds to the VM ID assigned by the infrastructure provider.</td>
</tr>
</tbody>
</table>
RabbitMQ Program Names

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rabbitmq_server_cluster_check</td>
<td>Checks that the RabbitMQ cluster is healthy. Runs after every deploy.</td>
</tr>
<tr>
<td>rabbitmq_server_node_check</td>
<td>Checks that the RabbitMQ node is healthy. Runs after every deploy.</td>
</tr>
<tr>
<td>rabbitmq_route_registrar_stderr</td>
<td>Registers the route for the management API with the Gorouter in your Pivotal Application Service (PAS) or Elastic Runtime deployment.</td>
</tr>
<tr>
<td>rabbitmq_route_registrar_stdout</td>
<td>Registers the route for the management API with the Gorouter in your PAS or Elastic Runtime deployment.</td>
</tr>
<tr>
<td>rabbitmq_server</td>
<td>The Erlang VM and RabbitMQ apps. Logs may span multiple lines.</td>
</tr>
<tr>
<td>rabbitmq_server_drain</td>
<td>Shuts down the Erlang VM and RabbitMQ apps. Runs as part of the BOSH lifecycle.</td>
</tr>
<tr>
<td>rabbitmq_server_http_api_access</td>
<td>Access to the RabbitMQ management UI.</td>
</tr>
<tr>
<td>rabbitmq_server_init</td>
<td>Starts the Erlang VM and RabbitMQ.</td>
</tr>
<tr>
<td>rabbitmq_server_post_deploy_stderr</td>
<td>Runs the node check and cluster check. Runs after every deploy.</td>
</tr>
<tr>
<td>rabbitmq_server_post_deploy_stdout</td>
<td>Runs the node check and cluster check. Runs after every deploy.</td>
</tr>
<tr>
<td>rabbitmq_server_pre_start</td>
<td>Runs before the rabbitmq-server job is started.</td>
</tr>
<tr>
<td>rabbitmq_server_sasl</td>
<td>Supervisor, progress, and crash reporting for the Erlang VM and RabbitMQ apps.</td>
</tr>
<tr>
<td>rabbitmq_server_shutdown_stderr</td>
<td>Stops the RabbitMQ app and Erlang VM.</td>
</tr>
<tr>
<td>rabbitmq_server_shutdown_stdout</td>
<td>Stops the RabbitMQ app and Erlang VM.</td>
</tr>
<tr>
<td>rabbitmq_server_startup_stderr</td>
<td>Starts the RabbitMQ app and Erlang VM, then configures users and permissions.</td>
</tr>
<tr>
<td>rabbitmq_server_startup_stdout</td>
<td>Starts the RabbitMQ app and Erlang VM, then configures users and permissions.</td>
</tr>
<tr>
<td>rabbitmq_server_upgrade</td>
<td>Shuts down Erlang VM and RabbitMQ app if required during an upgrade.</td>
</tr>
</tbody>
</table>

Metrics

Metrics are regularly-generated log messages that report measured component states. The metrics polling interval defaults to 30 seconds. The **metrics polling interval** is a configuration option on the RabbitMQ tile (Settings > RabbitMQ). The interval setting applies to all components deployed by the tile.

Metrics are long, single lines of text that follow the format:

```
origin: "p-rabbitmq" eventType: ValueMetric timestamp:1441188462382091652 deployment:"cf-rabbitmq" job:"cf-rabbitmq-node" index:"0" ip:"10.244.3.46" valueMetric: < name:"/p-rabbitmq/rabbitmq/system/memory" value:1024 unit:"MB">
```

Partition Indicator

A new metric has been introduced to help to identify network partitions. Essentially it exposes how many nodes each node knows. When a node is in partition the only node that it recognizes is itself and that is a good indication that that node might be in a partition.

An example of that metrics is:

```
origin: "p-rabbitmq" eventType: ValueMetric timestamp:1441188462382091652 deployment:"cf-rabbitmq" job:"cf-rabbitmq-node" index:"0" ip:"10.244.3.46" valueMetric: < name:"/p-rabbitmq/rabbitmq/erlang/reachable_nodes" value:3 unit:"count">
```

Monitors can be created to emit alerts in case a cluster seems to be in a partition. A metrics is emitted for each node in the cluster. For example: in a three-node cluster a monitor can expect to have a total of 9 (nine) since each node is expected to emit 3 (2 reachable nodes and itself). Otherwise, an alert can be sent to the team.

Recovering from a network partition

Please refer to the official RabbitMQ guide to understand how to recover from a network partition: [https://www.rabbitmq.com/partitions.html](https://www.rabbitmq.com/partitions.html)
Key Performance Indicators

Key Performance Indicators (KPIs) for RabbitMQ for PCF are metrics that operators find most useful for monitoring their RabbitMQ 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 RabbitMQ 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, create new, installation-specific monitoring metrics, thresholds, and alerts based on learning from their own installations.

For a list of all RabbitMQ for PCF raw component metrics, see Component Metrics Reference.

Component Heartbeats

Key RabbitMQ for PCF components periodically emit heartbeat metrics: the RabbitMQ server nodes, HAProxy nodes, and the Service Broker. The heartbeats are Boolean metrics, where 1 means the system is available, and 0 or the absence of a heartbeat metric means the service is not responding and should be investigated.

Service Broker Heartbeat

<table>
<thead>
<tr>
<th>p-rabbitmq.service_broker.heartbeat</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>RabbitMQ Service Broker is alive poll, which indicates if the component is available and able to respond to requests.</td>
</tr>
<tr>
<td>Use: If the Service Broker does not emit heartbeats, this indicates that it is offline. The Service Broker is required to create, update, and delete service instances, which are critical for dependent tiles such as Spring Cloud Services and Spring Cloud Data Flow.</td>
</tr>
<tr>
<td>Origin: Doppler/Firehose</td>
</tr>
<tr>
<td>Type: boolean</td>
</tr>
<tr>
<td>Frequency: 30 s (default), 10 s (configurable minimum)</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: N/A</td>
</tr>
<tr>
<td>Red critical: &lt; 1</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>Check the RabbitMQ Service Broker logs for errors. You can find this VM by targeting your RabbitMQ deployment with BOSH and running the following command:</td>
</tr>
<tr>
<td>bosh -d service-instance_GUID vms</td>
</tr>
</tbody>
</table>

HAProxy Heartbeat

<table>
<thead>
<tr>
<th>p-rabbitmq.haproxy.heartbeat</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>RabbitMQ HAProxy is alive poll, which indicates if the component is available and able to respond to requests.</td>
</tr>
<tr>
<td>Use: If the HAProxy does not emit heartbeats, this indicates that it is offline. To be functional, service instances require HAProxy.</td>
</tr>
<tr>
<td>Origin: Doppler/Firehose</td>
</tr>
<tr>
<td>Type: boolean</td>
</tr>
<tr>
<td>Frequency: 30 s (default), 10 s (configurable minimum)</td>
</tr>
<tr>
<td><strong>Recommended measurement</strong></td>
</tr>
<tr>
<td>Average over last 5 minutes</td>
</tr>
<tr>
<td><strong>Recommended alert thresholds</strong></td>
</tr>
<tr>
<td>Yellow warning: N/A</td>
</tr>
<tr>
<td>Red critical: &lt; 1</td>
</tr>
<tr>
<td><strong>Recommended response</strong></td>
</tr>
<tr>
<td>Check the RabbitMQ HAProxy logs for errors. You can find the VM by targeting your RabbitMQ deployment with</td>
</tr>
</tbody>
</table>
Recommended response

BOSH and running the following command, which lists `HAProxy_GUID`:

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

Server Heartbeat

```
p-rabbitmq.rabbitmq.heartbeat
```

<table>
<thead>
<tr>
<th>Description</th>
<th>RabbitMQ Server <em>is alive</em> poll, which indicates if the component is available and able to respond to requests. Use: If the server does not emit heartbeats, this indicates that it is offline. To be functional, service instances require RabbitMQ Server. Origin: Doppler/Firehose Type: boolean Frequency: 30 s (default), 10 s (configurable minimum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended measurement</td>
<td>Average over last 5 minutes</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
<td>Yellow warning: N/A Red critical: &lt; 1</td>
</tr>
</tbody>
</table>
| Recommended response | Check the RabbitMQ Server logs for errors. You can find the VM by targeting your RabbitMQ deployment with BOSH and running one of the following commands, which lists `rabbitmq`:

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

RabbitMQ Server KPIs

The following KPIs from the RabbitMQ server component:

File Descriptors

```
p-rabbitmq.rabbitmq.system.file_descriptors
```

<table>
<thead>
<tr>
<th>Description</th>
<th>File descriptors consumed. Use: If the number of file descriptors consumed becomes too large, the VM may lose the ability to perform disk IO, which can cause data loss. Note: This assumes non-persistent messages are handled by retries or some other logic by the producers. Origin: Doppler/Firehose Type: count Frequency: 30 s (default), 10 s (configurable minimum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended measurement</td>
<td>Average over last 10 minutes</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
<td>Yellow warning: &gt; 250000 Red critical: &gt; 280000</td>
</tr>
</tbody>
</table>
| Recommended response | The default `ulimit` for RabbitMQ for PCF is 300000. If this metric is met or exceeded for an extended period of time, consider one of the following actions:

- Scaling the rabbit nodes in the tile `Resource Config` pane.
- Reduce the load on the server |

Erlang Processes

```
p-rabbitmq.rabbitmq.erlang.erlang_processes
```

Erlang processes consumed by RabbitMQ, which runs on an Erlang VM.
### BOSH System Health Metrics

The BOSH layer that underlies PCF generates healthmonitor metrics for all VMs in the deployment. As of PCF 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 (PAS) Release Notes.

All BOSH-deployed components generate the system health metrics below. These component metrics are from RabbitMQ for PCF components, and serve as KPIs for the RabbitMQ for PCF service.

#### RAM

<table>
<thead>
<tr>
<th>Description</th>
<th>Use: RAM being consumed by the <code>p-rabbitmq</code> VM.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use: RabbitMQ is considered to be in a good state when it has little or no messages. In other words, “an empty rabbit is a happy rabbit.” Alerting on this metric can indicate that there are too few consumers or apps that read messages from the queue. Healthmonitor reports when RabbitMQ uses more than 40% of its RAM for the past ten minutes.</td>
</tr>
<tr>
<td>Origin</td>
<td>BOSH HM</td>
</tr>
<tr>
<td>Type</td>
<td>percent</td>
</tr>
<tr>
<td>Frequency</td>
<td>30 s (default), 10 s (configurable minimum)</td>
</tr>
<tr>
<td>Recommended measurement</td>
<td>Average over last 10 minutes</td>
</tr>
<tr>
<td>Recommended alert thresholds</td>
<td>Yellow warning: &gt; 40</td>
</tr>
<tr>
<td></td>
<td>Red critical: &gt; 50</td>
</tr>
<tr>
<td>Recommended response</td>
<td>Add more consumers to drain the queue as fast as possible.</td>
</tr>
</tbody>
</table>

#### CPU

<table>
<thead>
<tr>
<th>Description</th>
<th>Use: CPU being consumed by the <code>p-rabbitmq</code> VM.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use: A node that experiences context switching or high CPU usage will become unresponsive. This also affects the ability of the node to report metrics. Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</td>
</tr>
<tr>
<td>Origin</td>
<td>BOSH HM</td>
</tr>
<tr>
<td>Type</td>
<td>percent</td>
</tr>
<tr>
<td>Frequency</td>
<td>30 s (default), 10 s (configurable minimum)</td>
</tr>
<tr>
<td>Recommended measurement</td>
<td>Average over last 10 minutes</td>
</tr>
</tbody>
</table>
Recommended alert thresholds

<table>
<thead>
<tr>
<th>Yellow warning</th>
<th>Red critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 60</td>
<td>&gt; 75</td>
</tr>
</tbody>
</table>

Recommended response

Remember that "an empty rabbit is a happy rabbit". Add more consumers to drain the queue as fast as possible.

Ephemeral Disk

<table>
<thead>
<tr>
<th>system.disk.percent</th>
</tr>
</thead>
</table>

Description

Ephemeral Disk being consumed by the p-rabbitmq VM.

Use: If system disk fills up, there are too few consumers.

Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.

Origin: BOSH HM
Type: percent
Frequency: 30 s (default), 10 s (configurable minimum)

Recommended measurement

Average over last 10 minutes

Recommended alert thresholds

<table>
<thead>
<tr>
<th>Yellow warning</th>
<th>Red critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 60</td>
<td>&gt; 75</td>
</tr>
</tbody>
</table>

Recommended response

Remember that "an empty rabbit is a happy rabbit". Add more consumers to drain the queue as fast as possible.

Persistent Disk

<table>
<thead>
<tr>
<th>persistent.disk.percent</th>
</tr>
</thead>
</table>

Description

Persistent Disk being consumed by the p-rabbitmq VM.

Use: If system disk fills up, there are too few consumers.

Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.

Origin: BOSH HM
Type: percent
Frequency: 30 s (default), 10 s (configurable minimum)

Recommended measurement

Average over last 10 minutes

Recommended alert thresholds

<table>
<thead>
<tr>
<th>Yellow warning</th>
<th>Red critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 60</td>
<td>&gt; 75</td>
</tr>
</tbody>
</table>

Recommended response

Remember that "an empty rabbit is a happy rabbit". Add more consumers to drain the queue as fast as possible.

Component Metric Reference

RabbitMQ for PCF component VMs emit the following raw metrics. The full name of the metric follows the format: /p-rabbitmq/COMPONENT/METRIC-NAME

RabbitMQ Server Metrics

RabbitMQ for PCF message server components emit the following metrics.

<table>
<thead>
<tr>
<th>Full Name</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/p-rabbitmq.rabbitmq.heartbeat</td>
<td>boolean</td>
<td>Indicates whether the RabbitMQ server is available and able to respond to requests</td>
</tr>
<tr>
<td>/p-rabbitmq/rabbitmq/erlang/erlang_processes</td>
<td>count</td>
<td>The number of Erlang processes</td>
</tr>
<tr>
<td>/p-rabbitmq/rabbitmq/system/memory</td>
<td>MB</td>
<td>The memory in MB used by the node</td>
</tr>
<tr>
<td>Name Space</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>/p-rabbitmq/haproxy/heartbeat</td>
<td>boolean</td>
<td>Indicates whether the RabbitMQ HAPer is available and able to respond to requests</td>
</tr>
<tr>
<td>/p-rabbitmq/haproxy/health/connections</td>
<td>count</td>
<td>The total number of concurrent front-end connections to the server</td>
</tr>
<tr>
<td>/p-rabbitmq/haproxy/backend/qsize/amqp</td>
<td>size</td>
<td>The total size of the AMQP queue on the server</td>
</tr>
<tr>
<td>/p-rabbitmq/haproxy/backend/retries/amqp</td>
<td>count</td>
<td>The number of AMQP retries to the server</td>
</tr>
<tr>
<td>/p-rabbitmq/haproxy/backend/ctime/amqp</td>
<td>time</td>
<td>The total time to establish the TCP AMQP connection to the server</td>
</tr>
</tbody>
</table>
Setting Limits for On-Demand Service Instances

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

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

- Global
- Plan
- Org
- Space

After you set quotas, you can:

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

Create Global-level Quotas

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

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

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

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.[1]

### 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.[1]

### 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
```
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:

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>on-demand-broker/SERVICE-NAME/quota_remaining</td>
<td>Quota remaining for all instances across all plans</td>
</tr>
<tr>
<td>on-demand-broker/SERVICE-NAME/PLAN-NAME/quota_remaining</td>
<td>Quota remaining for a specific plan</td>
</tr>
<tr>
<td>on-demand-broker/SERVICE-NAME/total_instances</td>
<td>Total instances created across all plans</td>
</tr>
<tr>
<td>on-demand-broker/SERVICE-NAME/PLAN-NAME/total_instances</td>
<td>Total instances created for a specific plan</td>
</tr>
</tbody>
</table>

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

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.

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.

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

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

\[
\text{plan quota} \times \text{cost of selected resources}
\]

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:

\[
(15 \times \text{cost of micro VM type}) + (15 \times \text{cost of 20 GB persistent disk}) = \text{max cost per plan}
\]

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:

\[
(\text{plan1 quota} \times \text{plan1 resource cost}) + (\text{plan2 quota} \times \text{plan2 resource cost}) = \text{max cost for all plans}
\]

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:

\[
(\text{plan1 total_instances} \times \text{plan1 resource cost}) + (\text{plan2 total_instances} \times \text{plan2 resource cost}) = \text{current cost for all plans}
\]
Controlling Access to Service Plans by Org

If you want to restrict access to a service plan to a specific org, follow the instructions below.

You can also limit the number of service instances by setting quotas—for instructions, see Setting Limits for On-Demand Instances.

Change Access to Service Plans

⚠️ **Note:** If the plan you are restricting is currently enabled for all orgs, you must first Disable Service Access for the plan, then grant access to the plan to specific orgs. Use the CF Service Access field to disable access in the service plan configuration.

To restrict access to a plan for a specific org, run this command:

```bash
cf enable-service-access p.rabbitmq -p PLAN_NAME -o ORG_NAME
```

For example:

```bash
$ cf enable-service-access p.rabbitmq -p my-cluster-plan -o my-dev-org
```

For more information about the above command, see Access Control.
Isolating Clusters with the RabbitMQ for PCF Replicator

Overview
RabbitMQ for PCF Replicator is a tool that allows you to install multiple RabbitMQ for Pivotal Cloud Foundry (PCF) tiles in a single Ops Manager environment. This lets you run multiple pre-provisioned RabbitMQ clusters that are isolated from each other.

For example, you may want to isolate the cluster serving Spring Cloud Services (SCS) from the cluster serving apps in the Marketplace. Or you may want to give a certain team their own dedicated, pre-provisioned cluster that you manage for them. For information on how to accomplish these scenarios, see Common Use Cases.

Common Use Cases
The image below illustrates how to isolate SCS on RabbitMQ for PCF from clustered and single node service instances dedicated to different teams. In this use case:

- The unreplicated RabbitMQ for PCF tile deploys two types of services:
  - A pre-provisioned service is used as a backing service for SCS
  - An on-demand service is used to create three completely isolated single node service instances
- Two replica tiles are used to create two dedicated pre-provisioned clusters, with each one dedicated to a specific team.

These scenarios are explained below in Running SCS on a Dedicated RabbitMQ Cluster and Providing a Pre-Provisioned Dedicated Cluster.
Running SCS on a Dedicated RabbitMQ Cluster

Replica RabbitMQ tiles cannot be used to provide a backing service for Spring Cloud Services (SCS) because SCS expects that the service is called p-rabbitmq. Therefore, if you want to isolate the RabbitMQ cluster that is used by SCS from other tenants, you can reserve the unreplicated RabbitMQ for PCF tile for SCS, as shown in the diagram below. You can then add replica RabbitMQ clusters for use by apps in the Marketplace.

Pivotal recommends that you use the unreplicated RabbitMQ for PCF tile solely for SCS to avoid contention between apps using SCS, and apps using RabbitMQ for PCF in the Marketplace.

To reserve the unreplicated tile for SCS, turn off the Broker Registrar errand to prevent the broker from being exposed in the Marketplace. For more information, see Errands.

To offer RabbitMQ as a cloud messaging service in the Marketplace, create one or several replicas, install them in Ops Manager, and either allow the broker registrar errand to run, or register the service manually using CF.

Providing a Pre-Provisioned Dedicated Cluster

To reserve a RabbitMQ cluster for use by a specific team, disable the Broker Registrar errand in Ops Manager. This prevents service registration in the Marketplace. For more information, see Errands.

After you deploy the tile, manually expose the service broker to your desired orgs and spaces. For instructions, see Register a Broker.

Using Replicas While Offering the On-Demand RabbitMQ Service

The On-Demand service is not offered in replica tiles, since the purpose of the replicator is to create additional pre-provisioned clusters. If you wish to offer on-demand service plans, use the unreplicated RabbitMQ for PCF tile as shown in the diagram above.

Blue-Green Upgrades (Advanced)

In order to do blue-green style upgrades to minimize downtime, you can stand up a new cluster and migrate data and users over to the new cluster over a period of time. Speak with your Platform Architect about how to enable this workflow.

Generating Replica Tiles

This topic describes how to install the replicator and generate replica tiles of RabbitMQ for PCF.

Prerequisites

- RabbitMQ for PCF v1.8.x or v1.9.x
- 2.5 GB of free disk space

Download the Replicator

The RabbitMQ for PCF Replicator is currently available from Pivotal Network. Search for and download this archive, and then run the enclosed binary.

Generate Replica Tiles

The following is the syntax for generating a replica tile:

```bash
./rabbitmq-replicator-darwin
--name YOUR_DESIRED_TILE_NAME
--output DESIRED_FILE_NAME.pivotal
```

The following are the parameters expected in the above syntax:
### Naming Conventions in Original and Replica Tiles

The table below shows the naming conventions for various components related to the original RabbitMQ for PCF tile and to the replica tile. For the purposes of this example, assume that when you generate a replica tile as shown above, in the `name` field you provide the string `finance`. Then the attributes for the original and replica tiles are as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Name with Original Tile</th>
<th>Name with Replica Tile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker name</td>
<td>p-rabbitmq</td>
<td>p-rabbitmq-finance</td>
</tr>
<tr>
<td>Broker URL</td>
<td>pivotal-rabbitmq-broker.YOUR_CF_DOMAIN</td>
<td>pivotal-rabbitmq-broker-finance.YOUR_CF_DOMAIN</td>
</tr>
<tr>
<td>Service name</td>
<td>p-rabbitmq</td>
<td>p-rabbitmq-finance</td>
</tr>
<tr>
<td>URL for the RabbitMQ Management UI Dashboard</td>
<td>pivotal-rabbitmq.YOUR_CF_DOMAIN</td>
<td>pivotal-rabbitmq-finance.YOUR_CF_DOMAIN</td>
</tr>
<tr>
<td>Tile display name in Ops Manager</td>
<td>RabbitMQ</td>
<td>RabbitMQ (finance)</td>
</tr>
<tr>
<td>Tile name used internally by Ops Manager</td>
<td>p-rabbitmq</td>
<td>p-rabbitmq-finance</td>
</tr>
<tr>
<td>Metrics/Logging Origin</td>
<td>p-rabbitmq</td>
<td>p-rabbitmq-finance</td>
</tr>
</tbody>
</table>

### Installing Replica Tiles

After you have generated a replica tile, you can upload it to Ops Manager as you would any other tile. After you have uploaded it, follow the instructions for Installing and Configuring RabbitMQ for PCF as a Pre-Provisioned Service. The On-Demand service is not offered on replica tiles.

### Limiting Access to Replica Tiles to Specific Orgs

When you replicate RabbitMQ for PCF, the replica tile has the Broker Registrar errand set to On by default. This field appears in the Errands tab in the tile:

With any tile, if the Broker Registrar errand is set to On, it runs automatically when you finish installing the tile and causes the tile to be available to all CF orgs.

If you want to limit access to the tile to a specific org, follow these steps:
1. Set the broker registrar errand to **Off**, and apply your changes.

2. Manually register the tile with a specific CF org using the following command. See the above table for **BROKER_NAME**, **BROKER_USERNAME** and **BROKER_URL**
   
   ```
   cf create-service-broker BROKER_NAME BROKER_USERNAME BROKER_PASSWORD BROKER_URL
   ```

3. To give access to the org, use the following command and repeat for each additional org:
   
   ```
   cf enable-service-access SERVICE_NAME -o ORG_NAME
   ```

---

### Upgrading Replica Tiles

You can upgrade replica tiles like regular tiles with one important difference. You must generate a replica of the newer version of the RabbitMQ for PCF tile, using the replicator, and give the new replica the same **name** as the existing replica. This is shown in the example workflow below.

---

#### Example of an In-Place Upgrade of a Replica

Suppose you used the replicator to generate a replica of v1 of the RabbitMQ for PCF tile, with the **name** `trading-team`, and you installed it in Ops Manager. Here is the sample replicator command you used for the initial installation:

```
./rabbitmq-replicator-darwin
--name trading-team
--path ./download/p-rabbitmq-v1.pivotal
--output ./output/p-rabbitmq-v1-trading-team.pivotal
```

To upgrade to v2, follow these steps:

1. Download the new RabbitMQ for PCF v2.

2. Run the replicator command, using the path to the new RabbitMQ for PCF v2 tile, and supply the same **name** `trading-team`, as shown below.

   ```
   ./rabbitmq-replicator-darwin
   --name trading-team
   --path ./download/p-rabbitmq-v2.pivotal
   --output ./output/p-rabbitmq-v2-trading-team.pivotal
   ```

3. After you have the replica tile `p-rabbitmq-v2-trading-team.pivotal`, upload it to Ops Manager. This upgrades the v1 replica tile in place.

You can then proceed with upgrading the cluster.

If you want to do a blue-green style upgrade, see **Blue-Green Upgrades**.

---

### Limitations

- The On-Demand service is not offered on replica tiles.
Setting Default Policies for the RabbitMQ Service

Understanding a RabbitMQ Policy

You can set a default queue and an exchange policy in the RabbitMQ for Pivotal Cloud Foundry (PCF) tile to be applied to the RabbitMQ cluster. After you deploy the tile, Pivotal recommends that you use the RabbitMQ Management Interface to make configuration changes.

For more information about RabbitMQ policies, see the RabbitMQ documentation.

Rules for Policies Set in the Tile

The following rules apply to policies set through the RabbitMQ for PCF tile:

- A new policy, or an update to a policy, only applies to new instances (vhosts). Existing instances are not affected by the policy.
- The policy can only be deleted manually from the RabbitMQ nodes.
- Policies can be added dynamically using the RabbitMQ Management Interface.
- It is not possible to use pattern matching with policies. Policies will be applied to all queues and exchanges.

For granular policy settings, Pivotal recommends using the RabbitMQ Management UI. Set a priority number lower than 50, the default.

An Example Policy: Mirror on Two Nodes

Here is an example policy that ensures messages are mirrored on two nodes:

```
{
  "ha-mode": "exactly",
  "ha-params": 2,
  "ha-sync-mode": "automatic"
}
```

Operators should consider some of the performance implications of making queues and exchanges highly available. For more information about highly available queues, see the RabbitMQ documentation.

Best Practice for Syncing Queues

When a queue syncs all its messages, they are loaded into memory. When queues are syncing, they can use as much memory as the total size of all messages. This applies to both nodes—the node where the queue leader runs (from node) and the node where the queue follower runs (to node), but only applies to newly created queue followers.

This behavior is especially relevant when any change affects the deployment, for example: stemcell updates, deployment configuration changes, and network changes. Verify that you have enough memory and disk available to support all messages.

For example:

There are 5 GB of messages in a mirrored queue that is set to automatic sync. When this queue needs to sync, the node where the queue leader runs can use up to 5 GB of extra memory. The same applies to the node where the new queue follower is created.

Setting or Changing the Policy

To set the RabbitMQ policy, do the following:

1. From the Ops Manager Installation Dashboard, click the RabbitMQ for PCF tile and then click Pre-Provisioned RabbitMQ Policy.
2. Select Enable custom policy on new instances.

3. In the Policy for new instances field, paste the policy.
   The policy must be valid JSON and should meet valid RabbitMQ policy criteria.

4. In the Select the network partition behavior of the RabbitMQ cluster, choose the desired behavior: pause_minority or autoheal.

   For more information about these options and on RabbitMQ clusters and network partitions, see the RabbitMQ documentation.

   For production purposes, Pivotal recommends that customers have at least three RabbitMQ server nodes and two HAProxies spread across low latency availability zones.

   Note: No policy validation occurs during the deployment, and errors can cause the deployment to fail or policies to be applied incorrectly.

Viewing Policies in the RabbitMQ Management Dashboard

You can view RabbitMQ policies in the RabbitMQ Management Dashboard, shown below. The example policy entered in the RabbitMQ for PCF tile above is applied to all queues and given a Priority of 50. This allows you to override it by defining another policy with a higher priority.

In the Queues section shown below, you can see that any new queues created have the policy automatically applied.
Note: Developers can obtain the URL of the policy from VCAP SERVICES for app developers.
Using the RabbitMQ Management Dashboard

RabbitMQ Management Dashboard

Admin User

To gain access to the management dashboard as the \textit{admin} user, visit \url{http://pivotal-rabbitmq.SYS-DOMAIN}. To retrieve your system domain, navigate to your Pivotal Application Service (PAS) or Elastic Runtime tile and locate the System Domain field of the Domains section.

The username and password is the username and password you provided in the RabbitMQ configuration in OpsManager, which is also shown in the Credentials tab.

Application Developer

Users of Cloud Foundry who create instances via the Apps Manager or the cf CLI also get access to the Management UI. This is done using credentials that provide access only to their particular \textit{vhost}.

The appropriate URL is accessible via the Manage button within the Apps Manager.
Or it is also injected into the `VCAP_SERVICES` environment variable provided to apps running on Cloud Foundry. This can also be found via the CLI using `cf env <your app name>`.

Logging

A TCP Syslog endpoint can be configured in OpsManager. Logs are currently only forwarded for the RabbitMQ cluster.

RabbitMQ CLI

If you wish to run commands such as `rabbitmqctl`, then you have two options:

SSH into one of the machines running the rabbitmq-server. IPs can be found from the Status tab and access credentials from the Credentials tab within the RabbitMQ component of the installer. From there you need to bring RabbitMQ and Erlang into your environment and from there you can use `rabbitmqctl`:

```bash-4.1#
export PATH=$PATH:/var/vcap/packages/rabbitmq-server/bin
export PATH=$PATH:/var/vcap/packages/erlang/bin
rabbitmqctl cluster_status
```

Cluster status of node rabbit@node0 ...

```json
[{nodes,[[disc,rabbit@node0,rabbit@node1,rabbit@node2,rabbit@node3]]},
 {running_nodes,[rabbit@node3,rabbit@node2,rabbit@node1,rabbit@node0]},
 {partitions,[]}
...done.
```

Alternatively, install RabbitMQ and Erlang on a machine of your choice. Be sure to match versions of both to the cluster: the Management UI shows both the version of RabbitMQ and Erlang.

Then set your `~/.erlang.cookie` to match the cookie used in the cluster (you may have supplied this as part of the installation; see above).

You will need to set up your `/etc/hosts` file to match the RabbitMQ nodes.
Clustering and Network Partitions

Clustering in RabbitMQ for PCF

In RabbitMQ for PCF, the RabbitMQ® broker is always deployed as a cluster of one or more virtual machines (nodes). A RabbitMQ broker is a logical grouping of one or several Erlang nodes, each running the RabbitMQ application and sharing users, virtual hosts, queues, exchanges, bindings, and runtime parameters.

What is Replicated between nodes in a RabbitMQ cluster?

All data/state required for the operation of a RabbitMQ broker is replicated across all nodes. An exception to this are message queues, which by default reside on one node, though they are visible and reachable from all nodes. This means that the RabbitMQ cluster may be available and serving requests, while an individual queue residing on a single node is offline.

Replicating message queues across nodes is an expensive operation and should only be done to the extent needed by the application. To understand more about replicating queues across nodes in a cluster, see the documentation on high availability.

Automatic Network Partition Behaviors in RabbitMQ Clusters

The RabbitMQ® tile uses the `pause_minority` option for handling cluster partitions by default. This ensures data integrity by pausing the partition of the cluster in the minority, and resumes it with the data from the majority partition. You must maintain more than two nodes. If there is a partition when you only have two nodes, both nodes immediately pause.

You can also choose the `autoheal` option in the Pre-Provisioned RabbitMQ Policy tab. In this mode, if a partition occurs, RabbitMQ automatically decides on a winning partition, and restarts all nodes that are not in the winning partition. This option allows you to continue to receive connections to both parts of partitions.

Detecting a Network Partition

When a network partition occurs, a log message is written to the RabbitMQ node log:

```
---ERROR REPORT----- 15-Oct-2012::18:02:30 ===
Mnesia(rabbit@da3be74c053640fe92c6a39e2d7a5e46): ** ERROR ** mnesia_event got {inconsistent_database, running_partitioned_network, rabbit@21b6557b73f343201277dbf290ae8b79}
```

You can also run the `rabbitmqctl cluster_status` command on any of the RabbitMQ nodes to see the network partition. To run `rabbitmqctl cluster_status`, do the following:

1. $ sudo su -
2. $ cd /var/vcap/packages
3. $ export ERL_DIR=$PWD/erlang/bin/
4. $ cd rabbitmq-server/bin/
5. $./rabbitmqctl cluster_status

```
[
  partitions,
  [(rabbit@da3be74c053640fe92c6a39e2d7a5e46, rabbit@21b6557b73f343201277dbf290ae8b79)]
]
```

Recovering

Because the RabbitMQ tile uses the `pause_minority` option, minority nodes recover automatically after the partition is resolved. After a node recovers, it resumes accessing the queue along with data from the queues on the other nodes. However, if your queues use `ha-mode: all`, they only synchronize fully
after consuming all the messages created while the node was down. This is similar to how messages synchronize when you create a new queue.

Manually Synchronizing after a Partition

After a network partition, a queue on a minority node synchronizes after consuming all the messages created while it was down. You can also run the `sync_queue` command to synchronize a queue manually. To run `sync_queue`, do the following on each node:

1. `$ sudo su -`
2. `$ cd /var/vcap/packages`
3. `$ export ERL_DIR=$PWD/erlang/bin/`
4. `$ cd rabbitmq-server/bin/`
5. `$ ./rabbitmqctl list_queues`
6. `$ ./rabbitmqctl sync_queue name`
Upgrading RabbitMQ for PCF

This product enables automated upgrades between versions of the product and is deployed through Ops Manager. In some cases, you might be required to take the cluster offline. When this is necessary, it is clearly noted in the release notes for that version.

The upgrade paths for each version are detailed at Pivotal Network - RabbitMQ for PCF page 2.

To upgrade on-demand instances from RabbitMQ v3.6 to v3.7, see About Upgrading On-Demand Instances from RabbitMQ v3.6 to v3.7.

Downtime When Upgrading

A guide for downtime during upgrade deployments is shown in the table below. In some cases, the cluster remains available during a tile upgrade, but individual queues on cluster nodes may be taken offline.

This is only a guide, so before upgrading, check the release notes for the version you are upgrading to.

The RabbitMQ cluster becomes unavailable only when upgrading between specific versions of Erlang or RabbitMQ. This is stated in the release notes for those versions.

<table>
<thead>
<tr>
<th>Upgrade Type</th>
<th>Will Downtime Be Required For This Upgrade / Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Tile Version</td>
<td>The RabbitMQ cluster is taken offline for the duration of the upgrade.</td>
</tr>
<tr>
<td>Minor Tile Version</td>
<td>The RabbitMQ cluster is taken offline for the duration of the upgrade.</td>
</tr>
<tr>
<td>Patch Tile Version</td>
<td>Normally these are rolling deployments with each node being updated in turn. In these cases the cluster remains available, but individual queues may be taken offline as each node is restarted. There are specific migration paths that require downtime, which are identified in the release notes for that version.</td>
</tr>
<tr>
<td>Stemcell-Only Patch Tile Version</td>
<td>Where the patch update is only a new stemcell version these are rolling deployments with each node being updated in turn. In these cases the cluster remains available, but individual queues may be taken offline as each node is restarted.</td>
</tr>
</tbody>
</table>

Notes on the Upgrade Process

Review the following before starting an upgrade of RabbitMQ for PCF:

- Upgrading to a newer version of the product does not cause any loss of data or configuration.
- It may take busy RabbitMQ nodes a long time to shut down during the upgrade and you must not interrupt this process.
- The benefit you get from stemcell rolling upgrades depends on how you have configured network partition handling and the Resource Config tab. An HAProxy instance count of 2 and a RabbitMQ node count of 3 are required for rolling stemcell upgrades. As of v1.7.7, these counts are the default. For more information, see Clustering and Network Partitions.
- The length of the downtime depends on whether there is a stemcell update to replace the operating system image or if the existing VM can just have the RabbitMQ software updated. Stemcell updates incur additional downtime while the IaaS creates the new VM.
- Ops Manager ensures the instances are updated with the new packages and any configuration changes are applied automatically.
- For issues with upgrading RabbitMQ for PCF, see Troubleshooting On-Demand RabbitMQ for PCF.

Before Upgrading RabbitMQ or Erlang

Ensure the cluster is healthy using the RabbitMQ Management UI. You cannot rely on the BOSH instances output because that reflects the state of Erlang VM, not RabbitMQ.

Upgrade RabbitMQ for PCF

To upgrade the product, follow these steps:

1. Download the latest version of the product from Pivotal Network.
2. Upload the new .pivotal file to Ops Manager.

3. Upload the stemcell associated with the update (if required).

4. Update any new mandatory configuration parameters (if required).

5. Click **Apply changes** in the Ops Manager Installation Dashboard. The rest of the process is automated.

---

### About Upgrading On-Demand Instances from RabbitMQ v3.6 to v3.7

Before RabbitMQ for PCF v1.12, on-demand service instances were deployed using RabbitMQ v3.6. RabbitMQ for PCF v1.12 provides on-demand service plans with RabbitMQ v3.6 and v3.7.

App developers can either start fresh with a v3.7 on-demand instance or they can migrate their RabbitMQ v3.6 instances to RabbitMQ v3.7 using blue-green app deployments without downtime.

To enable the migration from on-demand v3.6 instances to v3.7 instances, the on-demand v3.7 plans should be configured to mirror the existing v3.6 plans. For more information, see the blog [Blue-Green Application Deployments with RabbitMQ](#) and the video [Blue-Green Deployment of Applications leveraging RabbitMQ](#).

RabbitMQ v3.6 on-demand plans will be removed in an upcoming RabbitMQ for PCF tile release. Encourage all developers to move their apps away from these plans.

---

### Release Policy

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

For more information about the PCF release policy, see [Release Policy](#).
Troubleshooting and FAQs for On-Demand RabbitMQ for PCF

This topic provides operators with basic troubleshooting techniques and FAQs for on-demand RabbitMQ for Pivotal Cloud Foundry (PCF).

How to Retrieve a Service instance GUID

You need the GUID of your service instance to run some BOSH commands. To retrieve the GUID, run the command:

```
cf service SERVICE-INSTANCE-NAME --guid
```

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.

Troubleshooting Errors

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

Failed Install

1. Certificate issues: The on-demand broker (ODB) requires valid certificates. Ensure that your certificates are valid and generate new ones if necessary.

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 RabbitMQ for PCF 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 RabbitMQ for PCF 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: bd5-4684-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 RabbitMQ for PCF 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:
   a. Log into BOSH as an admin.
   b. 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 RabbitMQ for PCF 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:
  - 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 RabbitMQ 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 (PAS) 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 PAS tile and see the network it is assigned to. Make sure that these networks can access each other.

### Upgrade All Service Instances 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:
   a. Install the `cf nozzle` plugin. For instructions, see the [firehose plugin](https://github.com/CF-GE FIX) GitHub repository.
   b. To find logs from your service in the `cf nozzle` output, run the following:

```
cf nozzle -I 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.

Failed Deployment on Upgrade or after Apply Changes

If the deployment fails after editing the Assign AZs and Networks pane of the RabbitMQ for PCF tile, it might be due to a change to the IP addresses assigned to the RabbitMQ Server job. RabbitMQ for PCF requires that these IP addresses do not change once assigned. If you change them, the deployment fails. This includes changes made to your current installation or during an upgrade. To diagnose and solve this issue, see Changing Network or IP Addresses Results in a Failed Deployment.

Troubleshooting Components

Guidance on checking for and fixing issues in 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 RabbitMQ 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 RabbitMQ for PCF 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 RabbitMQ 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 (PAS) tile.
## Networking

Common issues with networking include:

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

### Validate Service Broker Connectivity to Service Instances

To validate connectivity, do the following:

1. To SSH into the RabbitMQ for PCF service broker, run the following command:

   ```bash
   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 Service Instance

Use `cf ssh` to access the app container, then try connecting to the RabbitMQ for PCF 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.

© Copyright Pivotal Software Inc, 2013-2019 87 1.12
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 RabbitMQ for PCF 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 and on-demand service instance BOSH deployments, and on 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`.
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 VM(s)

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:

<table>
<thead>
<tr>
<th>Log Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>broker.log</td>
<td>Requests to the on-demand broker and the actions the broker performs while orchestrating the request (e.g. generating a manifest)</td>
</tr>
</tbody>
</table>
and calling BOSH. Start here when troubleshooting.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>broker_ctl.log</td>
<td>Control script logs for starting and stopping the on-demand broker.</td>
</tr>
<tr>
<td>post-start.stderr.log</td>
<td>Errors that occur during post-start verification.</td>
</tr>
<tr>
<td>post-start.stdout.log</td>
<td>Post-start verification.</td>
</tr>
<tr>
<td>drain.stderr.log</td>
<td>Errors that occur while running the drain script.</td>
</tr>
</tbody>
</table>

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 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 RabbitMQ for PCF 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.
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:
   a. Running any pre-delete errands
   b. Deleting the BOSH deployment of the service instance
   c. Removing any ODB-managed secrets from Credhub
   d. 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 Instances 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:
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.

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

These details will also be available through the BOSH 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_30c3c1a7-b0be-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.
",
  "logs": {
    "blobstore_id": "d830c4bf-8086-4bc2-8c1d-54d3a36d88d"
  }
}
```

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

Errend '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:

```bash
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:
   a. Log in as an admin by running:
      ```
cf login
```
   b. Confirm that the Marketplace does not list RabbitMQ for PCF by running:
      ```
cf m
```
   c. Log in to BOSH as an admin by running:
      ```
bosh log-in
```
   d. Display your BOSH deployments to confirm that the output does not show RabbitMQ for PCF deployment by running:
      ```
bosh deployments
```
   e. Run the “delete-all-service-instances” errand to delete every instance of the service.
   f. Run the “deregister-broker” errand to delete the service broker.
   g. Delete the service broker BOSH deployment by running:
      ```
bosh delete-deployment BROKER-DEPLOYMENT-NAME
```
   h. 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 Service Instance Owner

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

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:

   a. `cf curl APP-URL` and record the app name under `entity.name`.
   b. `cf curl SPACE-URL` to obtain the space, using the `entity.space_url` from the above curl. Record the space name under `entity.name`.
   c. `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 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:

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

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

---

**Drop and Restore AMQP(S) Traffic to a RabbitMQ Instance**

While debugging a RabbitMQ instance, you can prevent apps from sending and receiving messages, for example, to decrease the server load. You can use `drop-amqp-traffic` and `restore-amqp-traffic` scripts, which run the necessary `iptables` commands to achieve that.

To stop and then restore traffic to a RabbitMQ instance, do the following:

1. To stop all AMQP(S) traffic to a RabbitMQ instance, run the following command:
   ```
   bosh -d service-instance_GUID ssh rabbitmq-server
   "echo y | sudo /var/vcap/packages/rabbitmq-admin/bin/drop-amqp-traffic"
   ```

2. After performing the troubleshooting steps, restore the traffic. To do this, run the following command:
   ```
   bosh -d service-instance_GUID ssh rabbitmq-server
   "echo y | sudo /var/vcap/packages/rabbitmq-admin/bin/restore-amqp-traffic"
   ```

Alternatively, you can run these scripts on individual nodes:

1. `bosh ssh` to a rabbitmq-server instance.
2. `sudo -s` to gain root privileges.
3. Execute `drop-amqp-traffic` to drop all AMQP(S) traffic to this instance, or `restore-amqp-traffic` to start accepting traffic again.

---

**Frequently Asked Questions**

---

© Copyright Pivotal Software Inc, 2013-2019
What should I check before deploying a new version of the tile?

Ensure that all nodes in the cluster are healthy via the RabbitMQ Management UI, or health metrics exposed via the firehose. You cannot rely solely on the BOSH instances output as that reflects the state of the Erlang VM used by RabbitMQ and not the RabbitMQ app.

What is the correct way to stop and start RabbitMQ in PCF?

Only BOSH commands should be used by the operator to interact with the RabbitMQ app.

For example:

```
  bosh stop rabbitmq-server and bosh start rabbitmq-server.
```

There are BOSH job lifecycle hooks which are only fired when rabbitmq-server is stopped through BOSH. You can also stop individual instances by running the stop command and specifying JOB [Index].

**Note:** Do not use `monit stop rabbitmq-server` as this does not call the drain scripts

What happens when I run bosh stop rabbitmq-server?

BOSH starts the shutdown sequence from the bootstrap instance.

We start by telling the RabbitMQ app to shutdown and then shutdown the Erlang VM within which it is running. If this succeeds, we run the following checks to ensure that the RabbitMQ app and Erlang VM have stopped:

1. If `[/var/vcap/sys/run/rabbitmq-server/pid]` exists, check that the PID inside this file does not point to a running Erlang VM process. Notice that we are tracking the Erlang PID and not the RabbitMQ PID.

2. Check that `rabbitmqctl` does not return an Erlang VM PID.

Once this completes on the bootstrap instance, BOSH will continue the same sequence on the next instance. All remaining rabbitmq-server instances will be stopped one by one.

What happens when bosh stop rabbitmq-server fails?

If the BOSH `stop` fails, you will likely get an error saying that the drain script failed with:

```
  result: 1 of 1 drain scripts failed. Failed Jobs: rabbitmq-server.
```

What do I do when bosh stop rabbitmq-server fails?

The drain script logs to `/var/vcap/sys/log/rabbitmq-server/drain.log`. If you have a remote syslog configured, this will appear as the `rmq_server_drain` program.

First, BOSH `ssh` into the failing rabbitmq-server instance and start the rabbitmq-server job by running `monit start rabbitmq-server`. You will not be able to start the job via BOSH `start` as this always runs the drain script first and will fail since the drain script is failing.

Once rabbitmq-server job is running (confirm this via `monit status` run `DEBUG=1 /var/vcap/jobs/rabbitmq-server/bin/drain`). This will tell you exactly why it’s failing.

How can I manually back up the state of the RabbitMQ cluster?

© Copyright Pivotal Software Inc, 2013-2019 96
It is possible to back up the state of a RabbitMQ cluster for both the on-demand and pre-provisioned services using the RabbitMQ Management API. Backups include vhosts, exchanges, queues and users.

Back up Manually
1. Log in to the RabbitMQ Management UI as the admin user you created.
2. Select export definitions from the main page.

Back up and Restore with a Script
Use the API to run scripts with code similar to the following:

- For the backup:
  
  ```bash
  curl -u "$USERNAME:$PASSWORD" "http://$RABBIT-ADDRESS:15672/api/definitions"
  -o "$BACKUP-FOLDER/rabbit-backup.json"
  ```

- For the restore:
  
  ```bash
  curl -u "$USERNAME:$PASSWORD" "http://$RABBIT-ADDRESS:15672/api/definitions"
  -X POST -H "Content-Type: application/json" -d
  "@$BACKUP-FOLDER/rabbit-backup.json"
  ```

What pre-upgrade checks should I do?
Before doing any upgrade of RabbitMQ, Pivotal recommends checking the following:

1. In Operations Manager check that the status of all of the instances is healthy.
2. Log into the RabbitMQ Management UI and check that no alarms have been triggered and that all nodes are healthy, that is, they should display as green.
3. Check that the system is not close to hitting either the memory or disk alarm. Do this by looking at what has been consumed by each node in the RabbitMQ Management UI.

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 ticket with Pivotal Support. Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To expedite troubleshooting, provide your service broker logs and your service instance logs. If your `cf service YOUR-SERVICE-INSTANCE` output includes a task-id, provide the BOSH task output.
Frequently Asked Questions for Pre-Provisioned RabbitMQ for PCF

This topic lists frequently asked questions that apply to the RabbitMQ for Pivotal Cloud Foundry (PCF) pre-provisioned service.

Frequently Asked Questions

What should I check before deploying a new version of the tile?

Ensure that all nodes in the cluster are healthy via the RabbitMQ Management UI, or health metrics exposed via the firehose. You cannot rely solely on the BOSH output as that reflects the state of the Erlang VM used by RabbitMQ and not the RabbitMQ application.

What is the correct way to stop and start RabbitMQ in PCF?

Only BOSH commands should be used by the operator to interact with the RabbitMQ app.

For example:

```
bosh stop rabbitmq-server and bosh start rabbitmq-server
```

There are BOSH job lifecycle hooks which are only fired when rabbitmq-server is stopped through BOSH. You can also stop individual instances by running the stop command and specifying

```
JOB [index]
```

**Note: Do not use**

```
man stop rabbitmq-server
```

as this does not call the drain scripts

What happens when I run bosh stop rabbitmq-server?

BOSH starts the shutdown sequence from the bootstrap instance.

We start by telling the RabbitMQ application to shutdown and then shutdown the Erlang VM within which it is running. If this succeeds, we run the following checks to ensure that the RabbitMQ application and Erlang VM have stopped:

1. If `/var/vcap/sys/run/rabbitmq-server/pid` exists, check that the PID inside this file does not point to a running Erlang VM process. Notice that we are tracking the Erlang PID and not the RabbitMQ PID.

2. Check that `rabbitmqctl` does not return an Erlang VM PID

Once this completes on the bootstrap instance, BOSH will continue the same sequence on the next instance. All remaining rabbitmq-server instances will be stopped one by one.

What happens when bosh stop rabbitmq-server fails?

If the BOSH `stop` fails, you will likely get an error saying that the drain script failed with:

```
result: 1 of 1 drain scripts failed. Failed Jobs: rabbitmq-server.
```

What do I do when bosh stop rabbitmq-server fails?

The drain script logs to `[/var/vcap/sys/log/rabbitmq-server/drain.log]`. If you have a remote syslog configured, this will appear as the `rabbitmq_server` program.
First, BOSH ssh into the failing rabbitmq-server instance and start the rabbitmq-server job by running `sudo monit start rabbitmq-server`. You will not be able to start the job via BOSH start as this always runs the drain script first and will fail since the drain script is failing.

Once rabbitmq-server job is running (confirm this via `sudo monit status`), run `DEBUG=1 /var/vcap/jobs/rabbitmq-server/bin/drain`. This will tell you exactly why it’s failing.

How can I manually back up the state of the RabbitMQ cluster?

It is possible to back up the state of a RabbitMQ cluster for both the on-demand and pre-provisioned services using the RabbitMQ Management API. Backups include vhosts, exchanges, queues and users.

**Back up manually**

1. Log in to the RabbitMQ Management UI as the admin user you created.
2. Select *export definitions* from the main page.

**Back up and restore with a script**

Use the API to run scripts with code similar to the following:

1. For the backup:
   ```
curl -u "USERNAME:PASSWORD" "http://$RABBIT_ADDRESS:15672/api/definitions" -o "$BACKUP_FOLDER/rabbit-backup.json"
   ```
2. For the restore:
   ```
curl -u "USERNAME:PASSWORD" "http://$RABBIT_ADDRESS:15672/api/definitions" -X POST -H "Content-Type: application/json" -d "@/$BACKUP_FOLDER/rabbit-backup.json"
   ```

What pre-upgrade checks should I do?

Before doing any upgrade of RabbitMQ, Pivotal recommends checking the following:

1. In Operations Manager check that the status of all of the instances is healthy.
2. Log into the RabbitMQ Management UI and check that no alarms have been triggered and that all nodes are healthy, that is, they should display as green.
3. Check that the system is not close to hitting either the memory or disk alarm. Do this by looking at what has been consumed by each node in the RabbitMQ Management UI.

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 ticket with Pivotal Support. Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`

To expedite troubleshooting, provide your service broker logs and your service instance logs. If your `cf service YOUR-SERVICE-INSTANCE` output includes a task-id, provide the BOSH task output.
Using On-Demand RabbitMQ for PCF

This topic provides instructions for developers using the on-demand RabbitMQ service for their Pivotal Cloud Foundry (PCF) apps. RabbitMQ enables messaging between cloud-based servers, apps and devices.

These procedures use the Cloud Foundry Command-Line Interface (cf CLI). You can also use Apps Manager to perform the same tasks using a graphical UI.

For general information, see Managing Service Instances with the cf CLI.

Prerequisites

To use on-demand RabbitMQ for PCF with your PCF apps, you need:

- A PCF installation with RabbitMQ for PCF installed and listed in the Marketplace.
- A Space Developer or Admin account on the PCF installation.
- A local machine with the following installed:
  - a browser
  - a shell
  - the Cloud Foundry Command-Line Interface (cf CLI)
  - the Linux `watch` command

  To log into the org and space containing your app.

Developer Guide

Entries in the VCAP_SERVICES Environment Variable

Apps running in Cloud Foundry gain access to the bound service instances via an environment variable credentials hash called VCAP_SERVICES. An example hash is shown below:

```json
{
  "p-rabbitmq": {
    "credentials": {
      "dashboard_url": "http://pivotal-rabbitmq.your.pcf.example.com/#/login/b5d0ad14-4352-48e8-8982-d5b1d257029f/tavk86pnnns1ddigadtbchurn",
      "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
      "password": "tavk86pnnns1ddigadtbchurn",
      "protocols": {
        "amqp": {
          "password": "tavk86pnnns1ddigadtbchurn",
          "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "uris": [
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnnns1ddigadtbchurn@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af",
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnnns1ddigadtbchurn@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af"
          ]
        }
      }
    }
  }
}
```

For more information about the environment variable VCAP_SERVICES, see RabbitMQ Environment Variables.

The Create-Bind Process

Because every app and service in PCF is scoped to a space, an app can only use a service if an instance of the service exists in the same space.

To use RabbitMQ in a PCF app:

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

**Confirm Service Availability**

For an app to use a service, 1) the service must be available in the Marketplace for its space and 2) 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 On-Demand RabbitMQ for PCF service is available in the Marketplace:
   a. Enter `cf marketplace`
   b. If the output lists `ondemand-rabbitmq` in the `service` column, on-demand RabbitMQ 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
[...]
ondemand-rabbitmq Solo RabbitMQ Service
[...]
```

2. To confirm that an On-Demand RabbitMQ for PCF instance is running in the space
   a. Enter `cf services`
   b. Any `ondemand-rabbitmq` listings in the `service` column are service instances of on-demand RabbitMQ in the space.

```
$ 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 ondemand-rabbitmq Solo create succeeded
```

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

**Create a Service Instance**

Unlike pre-provisioned services, on-demand services are created asynchronously, not immediately. The watch command shows you when your service instance is ready to bind and use.

To create an instance of the on-demand RabbitMQ for PCF service, run `cf create-service`:

1. Enter `cf create-service ondemand-rabbitmq Solo SERVICE_INSTANCE`
   Where `SERVICE_INSTANCE` is a name you choose to identify the service instance. This name will appear under `service [sic]` in output from `cf services`.

```
$ cf create-service ondemand-rabbitmq Solo my-instance
Creating service my-instance in org my-org / space my-space as user@example.com...
OK
```

2. Enter `watch cf services` and wait for the `last operation` for your instance to show as `create succeeded`.

```
$ watch 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 ondemand-rabbitmq Solo create succeeded
```

If you get an error, see [Troubleshooting Instances](#).
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`.

To bind an app to a RabbitMQ instance run:

```bash
$ cf bind-service
```

1. Enter `cf bind-service APP SERVICE_INSTANCE`
   
   Where `APP` is the app you want to use the RabbitMQ service instance and `SERVICE_INSTANCE` is the name you supplied when you ran `cf create-service`.

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

Binding service mydb 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

Use the RabbitMQ Service in Your App

To access the RabbitMQ service from your app:

1. Run `cf env APP_NAME` with the name of the app bound to the RabbitMQ for PCF instance.
2. In the output, note the connection strings listed in the `VCAP_SERVICES` object for the app.
3. In your app code, call the RabbitMQ service using the connection strings.

For how to code your app to use RabbitMQ messaging, see About Using Pivotal RabbitMQ > Client Documentation in the RabbitMQ documentation.

Updating a Service Instance

If you bind a new service or change the service bindings, you need to run `cf restart` to update the `VCAP_SERVICES` environment variable in the application container.

1. Enter `cf restart-app APP`
   
   Where `APP` is the app you want to use the updated service instance.

```bash
$ cf restart my-app
```

Pushing new version of an app automatically restages and restarts the app on any service instances it is bound to.

Unbind a Service Instance to Your App

To stop an app from using a service it no longer needs, unbind it from the service instance using `cf unbind-service`.

1. Enter `cf unbind-service APP SERVICE_INSTANCE`
   
   Where `APP` is the app you want to stop using the RabbitMQ service instance and `SERVICE_INSTANCE` is the name you supplied when you ran `cf create-service`.

```bash
$ cf unbind-service my-app my-instance
```

Unbinding app my-app from service my-instance in org my-org / space my-space as user@example.com...

OK

Delete a Service Instance

To delete a service instance, run `cf delete-service`.
1. Enter `cf delete-service SERVICE_INSTANCE`
Where `SERVICE_INSTANCE` is the name of the service to delete.

```
$ cf delete-service my-instance
Are you sure you want to delete the service my-instance? [y]
Deleting service my-service in org my-org / space my-space as user@example.com...
OK
```

2. Enter `watch cf service SERVICE_INSTANCE` and wait for a `Service instance not found` error indicating that the instance no longer exists.

You cannot delete a service instance that an app is bound to.

### Create an Admin User for a Service Instance

If you want to get admin privileges to the RabbitMQ Management UI, you can create an admin user for a service instance, and obtain user credentials that you can share with other app developers.

Both operators and app developers can use this procedure. For instructions, see [Create an Admin User for a Service Instance](#).

### Sharing Service Instances

In order to share service instances, the feature-flag `service_instance_sharing` must be enabled by your Operator. You can then follow the [documentation](#) to share your service instances across Cloud Foundry Organizations and Spaces.

### Federate Exchanges and Queues

You can federate exchanges and queues in RabbitMQ for PCF, as you would in any RabbitMQ deployment.

To federate exchanges and queues, do the following:

1. Create a service key by following the instructions in [Create an Admin User for a Service Instance](#).

   The output of the above procedure returns admin user credentials, along with other data.

2. In the output from the above step, look for the `uris` array. It will have this pattern:

   ```json
   { "uri": "amqp://USERNAME:PASSWORD@IP_ADDRESS/VHOST", "uris": [ "amqp://USERNAME:PASSWORD@IP_ADDRESS/VHOST" ] }
   ...
   }
   ``

   For example:

   ```json
   { "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97a01e1d7", "uris": [ "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97a01e1d7" ] }
   ``

3. Set up federation as you normally would, using the RabbitMQ Management UI or API, with the URIs found in the `uris` array you got from the step above.

   For instructions on federation, see the [RabbitMQ documentation](#).
Shovel Exchanges and Queues

You can shovel exchanges and queues in RabbitMQ for PCF, as you would in any RabbitMQ deployment.

To shovel exchanges and queues, do the following:

1. Create a service key by following the instructions in Create an Admin User for a Service Instance.

   The output of the above procedure returns admin user credentials, along with other data.

2. In the output from the above step, look for the uris array. It will have this pattern:

   ```json
   {
     "uri": "amqp://USERNAME:PASSWORD@IP_ADDRESS/VHOST",
     "uris": [
       "amqp://USERNAME:PASSWORD@IP_ADDRESS/VHOST"
     ]
   }
   ```

   For example:

   ```json
   {
     "uri": "amqp://b5d0ad14-4352-48c8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97a9f01cd7",
     "uris": [
       "amqp://b5d0ad14-4352-48c8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97a9f01cd7",
       "amqp://b5d0ad14-4352-48c8-8982-d5b1d257029f:passwordexample123456789@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97a9f01cd7"
     ]
   }
   ```

3. Set up shovel as you normally would, using the RabbitMQ Management UI or API, with the URIs found in the uris array you got from the step above.

   For shovel instructions, see the RabbitMQ documentation. At the moment, RabbitMQ for PCF only supports Dynamic Shovels.
RabbitMQ Environment Variables

This topic provides a reference for the environment variables that Pivotal Cloud Foundry (PCF) stores for RabbitMQ for PCF service instances. These variables include the credentials that apps use to access the service instances.

VCAP_SERVICES

Apps running in PCF gain access to the bound service instances through an environment variable credentials hash called `VCAP_SERVICES`. An example hash is show below:

```json
{
  "p-rabbitmq": {
    "label": "p-rabbitmq",
    "name": "my-rabbit-service-instance",
    "credentials": {
      "dashboard_url": "http://pivotal-rabbitmq.your.pcf.example.com/#/login/b5d0ad14-4352-48e8-8982-d5b1d257029f#tavk86pnnns1ddiqpsdtbchurn",
      "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
      "hostname": "10.0.0.41",
      "hostnames": ["10.0.0.41", "10.0.0.51"],
      "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
      "uris": ["amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
                "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7"],
      "http_api_url": "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
      "http_api_urlis": ["http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
                         "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:15672/api"],
      "protocols": {
        "amqp": {
          "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "password": "passwordexample123456789",
          "hostname": "10.0.0.41",
          "hostnames": ["10.0.0.41", "10.0.0.51"],
          "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
          "uris": ["amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
                    "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7"],
          "management": {
            "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
            "password": "passwordexample123456789",
            "hostname": "10.0.0.41",
            "hostnames": ["10.0.0.41", "10.0.0.51"],
            "uri": "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
            "uris": ["http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
                     "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:15672/api"]
        }},
        "mqtt": {"username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "password": "passwordexample123456789",
          "hostname": "10.0.0.41",
          "hostnames": ["10.0.0.41", "10.0.0.51"],
          "uri": "mqtt://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
          "uris": ["mqtt://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
                    "mqtt://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:15672/api"]},
        "stomp": {"username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "password": "passwordexample123456789",
          "hostname": "10.0.0.41",
          "hostnames": ["10.0.0.41", "10.0.0.51"],
          "uri": "stomp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
          "uris": ["stomp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.41:15672/api",
                    "stomp://b5d0ad14-4352-48e8-8982-d5b1d257029f:b5d0ad14-4352-48e8-8982-d5b1d257029f@10.0.0.51:15672/api"]}
      }
    }
  }
}
```

You can search for your service by its `name`, given when creating the service instance, or dynamically through the `tags` or `label` properties. The `credentials` property can be used as follows:

- The top level properties `uri`, `uris`, `vhost`, `username`, `password`, `hostname`, and `hostnames` provide access to the AMQP 0.9.1 protocol.
- A more flexible approach is provided by the `credentials.protocols` property, which has a key per enabled protocol. The possible keys are: `amqp`, `management`, `mqtt`, and `stomp`. If SSL is enabled, then the keys are: `amqp+ssl`, `management+ssl`, `mqtt+ssl`, and `stomp+ssl` respectively.
- The values associated with each of these keys gives access credentials specific to each protocol. In all cases, URIs are provided, along with the individual components.
Changing Enabled Plugins and Protocols

Note: Removing or adding plugins/protocols might cause apps bound with RabbitMQ to break.

If you adjust the plugins and protocols enabled for RabbitMQ, you might need to force all app's VCAP_SERVICES environment variable to be regenerated.

Adding and removing the following plugins require bound apps to be restaged:

- rabbitmq_management
- rabbitmq_stomp
- rabbitmq_mqtt
- rabbitmq_amqp1_0

In common with all services in PCF, the VCAP_SERVICES environment variable for an app is only modified when the app is bound to a service instance. Users need to cf unbind-service, cf bind-service, and cf restage their app in this scenario.

Note: Removing or adding plugins/protocols might cause apps bound with RabbitMQ to break.
Troubleshooting Instances

This topic provides basic instructions for app developers troubleshooting On-Demand RabbitMQ® for PCF.

Errors

You may see an error when using the Cloud Foundry Command-Line Interface (cf CLI) to perform basic operations on a RabbitMQ 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
```

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.
3. Run \texttt{cf service SERVICE-INSTANCE-NAME} to retrieve more information about a specific instance.

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

### Retrieve RabbitMQ Instance Credentials

If you want to access the Management Dashboard or the RabbitMQ server for troubleshooting, you can create a new service-key to retrieve RabbitMQ instance credentials. Pivotal recommends that you use this key for troubleshooting only, and that you delete the key after troubleshooting. To retrieve the credentials, do the following:

1. Create a service-key for your RabbitMQ instance using the command \texttt{cf create-service-key INSTANCE-NAME SERVICE-KEY-NAME}.

2. Retrieve the credentials using the command \texttt{cf service-key INSTANCE-NAME SERVICE-KEY-NAME}.

For example:

```
$ cf create-service-key my-rmq-instance my-key
Creating service key my-key for service instance my-rmq-instance as admin...
OK

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

### 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 \texttt{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.

### Delete RabbitMQ Instances

On-Demand Broker provides a BOSH command to delete all the On-Demand Broker deployed instances. To delete the instances, do the following procedure:

1. Run the following command to delete all instances of the On-Demand Broker:

   \texttt{bosh run-errand delete-sub-deployments}

   \textbf{WARNING:} This command deletes deployment instances serially. It is very destructive and cannot be undone.