

# RabbitMQ for PCF®

Version 1.11


## User's Guide

© Copyright Pivotal Software Inc, 2013-present

## Table of Contents

Table of Contents	2
RabbitMQ for PCF	3
RabbitMQ® for PCF Release Notes	6
Unlocking the Power of On-Demand RabbitMQ for PCF	13
On-Demand Service Architecture	17
Deploying the RabbitMQ Pre-Provisioned Service	20
Installing and Configuring the On-Demand Service	24
Installing and Configuring the Pre-Provisioned Service	35
Smoke Tests	49
Monitoring and KPIs for On-Demand RabbitMQ for PCF	50
Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF	59
Setting Limits for On-Demand Service Instances	66
Controlling Access to Service Plans by Org	72
Creating Isolation with the RabbitMQ for PCF Replicator	73
Setting Default Policies for the RabbitMQ Service	77
Managing the RabbitMQ® Service	80
Clustering and Network Partitions	82
Upgrading RabbitMQ for PCF	84
Troubleshooting and FAQs for On-Demand RabbitMQ for PCF	86
Frequently Asked Questions for Pre-Provisioned RabbitMQ for PCF	104
Using On-Demand RabbitMQ for PCF	107
RabbitMQ® Environment Variables	112
Troubleshooting Instances	114

## RabbitMQ for PCF

 **Note:** RabbitMQ for PCF v1.11 is no longer supported. The support period for v1.11 has expired. To stay up-to-date with the latest software and security updates, upgrade to RabbitMQ for PCF v1.12 or later.

## 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.

As of v 1.8, RabbitMQ for PCF supports two types of service, an *on-demand* service and a *pre-provisioned* service. This table summarizes the main differences between the two:

	Available Since	VMs it Runs On	How VMs are Created	Metrics Name Prefix
<b>On-Demand Service</b>	v1.8	Dedicated VM that serves a single service instance. See <a href="#">this topic</a> for details.	PCF creates each VM on-demand when app developer creates service instance	<code>p.rabbitmq</code> (with a dot)
<b>Pre-Provisioned Service</b>	v1.2	Multi-tenant VMs shared by apps across PCF deployment	PCF creates all VMs when operator deploys or updates service	<code>p-rabbitmq</code> (with a dash)

This RabbitMQ for PCF v1.8 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.

Element	Details
Version	v1.11.17
Release date	September 13, 2018
Software component version	RabbitMQ OSS v3.6.16
Compatible Ops Manager version(s)	v1.12.x and v2.0.x
Compatible Elastic Runtime version(s) *	v1.12.x
Compatible Pivotal Application Service version(s) *	v2.0.x
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	No

\* As of PCF v2.0, Elastic Runtime is renamed Pivotal Application Service (PAS). For more information, see [Pivotal Application Service \(PAS\) Highlights](#).

## Features

### On-Demand

- Create up to five different on-demand RabbitMQ plans which can be provisioned through the Marketplace
- Choose whether a plan has 1, 3, 5, or 7 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. For downtime requirements, see [RabbitMQ for PCF Release Notes](#).
- 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

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. For downtime requirements, see [Downtime When Upgrading](#).
- Configure the end point for the RabbitMQ Syslog
- RabbitMQ and HAProxy metrics are exposed on the firehose
- Syslog forwarding on by default

## 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, you install one tile for on-demand service plans and a different tile for pre-provisioned service plans.

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

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

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

## 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 v1.10.x version, upgrade to the latest v1.10.x version before upgrading to the latest v1.11.x version.

See the [Product Compatibility Matrix](#) for product versions and upgrade paths.

### v1.11.x

#### v1.11.17

**Release Date:** September 13, 2018

#### Features

- Requires [stemcell](#) 3445.67
- Smoke tests now wait up to 5 minutes for a test app to deploy.

#### Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v20.3.8.6
- HAProxy v1.6.13

#### v1.11.16

**Release Date:** August 17, 2018

#### Features

- Requires stemcell [3445.64](#)

#### Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v20.3.8.1
- HAProxy v1.6.13

## v1.11.15

**Release Date:** July 13, 2018

## Features

- Requires stemcell [3445.51](#)

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v20.3.8.1
- HAProxy v1.6.13

## v1.11.14

**Release Date:** June 5, 2018

## Features

- Requires stemcell [3445.48](#)
- 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: CANNOTAuthorizedForExtKeyUsage is premature](#).

## Fixed Issues

This release fixes the following issue:

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

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.13

**Release Date:** May 16, 2018

## Features

- Requires stemcell [3445.45](#)
- 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: CANNOTAuthorizedForExtKeyUsage is premature](#).

## Known Issues

- 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 requires 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
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.12

**Release Date:** March 28, 2018

## Features

- Requires stemcell [3445.30](#)
- Updates the following:
  - Java package to version 1.9
  - On-Demand Services SDK to [v0.20.0](#)
  - Loggregator release to [v102.1](#)
  - Routing release to [v0.174.0](#)



## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.8

**Release Date:** February 26, 2018

## Features

- [ **Security Fix** ] Requires stemcell [3445.28](#).

## Fixed Issues

- Upgrades no longer fail when SYSLOG is not configured.
- Special characters such as `[]^_!"#$%&()*+,-./\:;<=>`'`` no longer lead to issues logging into the Management UI. You can now use special characters, with the exception of ``` and `'`, in RabbitMQ usernames and passwords.
- Metrics are now emitted when using special characters in passwords.
- If you change the RabbitMQ admin username, the user no longer loses access to preexisting vhosts.

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.7

**Release Date:** January 29, 2018

## Features

- Update to RabbitMQ v3.6.15. For more information, see [RabbitMQ 3.6.15 Release Notes](#).

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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**.
- The RabbitMQ administrator password must be a combination of uppercase and lowercase alphanumerics. Special characters such as `[]^_!"#$%&()*+,-\/:;<=>'"` can lead to issues logging into the Management UI as that user.

## Packages

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

## v1.11.6 - Withdrawn

**Release Date:** January 24, 2018

**Withdrawal Date:** February 5, 2018

This release was withdrawn due to an issue upgrading when there are special characters in the RabbitMQ administrator credentials. Skip this version and upgrade to [v1.11.7](#) or later.

## Features

- [ **Security Fix** ] Requires stemcell [3445.24](#) to address Spectre vulnerabilities.

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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**.
- Certain special characters cannot be used with the RabbitMQ administrator password, and using them can lead to issues logging into the Management UI as that user.

## Packages

- OSS RabbitMQ v3.6.14
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.5

**Release Date:** January 19, 2018

## Features

- [ **Security Fix** ] Requires stemcell [3445.23](#) to address GNU C Library vulnerabilities.
- This release reverts implementation of the colocated errands feature for the RabbitMQ for PCF v1.11 line, due to incompatibility with some PCF v1.12

installations. Errands are no longer run colocated on existing VMs.

## Known Issues

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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**.
- Certain special characters cannot be used with the RabbitMQ administrator password, and using them can lead to issues logging into the Management UI as that user.

## Packages

- OSS RabbitMQ v3.6.14
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.3

**Release Date:** January 12, 2018

## Features

- [ **Security Fix** ] Requires stemcell [3445.22](#) to address the Meltdown security issue. For more information about Meltdown, see [Pivotal Vulnerability Report: Meltdown and Spectre Attacks](#).
- Errands now run colocated within existing instances
- Service metrics now uses a drain script to prevent monit timeout issues

## On-Demand

- `smoke-tests` errand is renamed to `on-demand-broker-smoke-tests`
- The CF timeouts for smoke tests have increased

## Known Issues

- When installing RabbitMQ for PCF v1.11.3 on Ops Manager v1.12, errands might fail with an error like the following:

```
Task 888513 | 23:16:53 | Preparing deployment: Preparing deployment (00:00:02)
L Error: Instance 'p-rabbitmq-06f95350e62d0375148d/smoke-tests/0' doesn't exist
Task 888513 | 23:16:55 | Error: Instance 'p-rabbitmq-06f95350e62d0375148d/smoke-tests/0' doesn't exist
```

If you experience this issue, disable smoke tests.

- Cluster scaling or changing the [Erlang Cookie](#) value requires 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**.
- Certain special characters cannot be used with the RabbitMQ administrator password, and using them can lead to issues logging into the Management UI as that user.

## Packages

- OSS RabbitMQ v3.6.14
- Erlang v19.3.6.4
- HAProxy v1.6.13

## v1.11.2

**Release Date:** December 22, 2017

### Features

#### On-Demand

- Register / Deregister On Demand Service Broker, On Demand Instance Smoke Tests, Upgrade All Service Instances and Delete All Service Instances errands are now colocated with their respective broker to decrease errand run time and VM footprint.  
For more information about errands, see [Errands](#).

#### Pre-Provisioned

- Broker Registrar / Deregistrar, Smoke Tests errands are now colocated with their respective broker to decrease errand run time and VM footprint.  
For more information about errands, see [Errands](#).

### Known Issues

- 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**.
- Certain special characters cannot be used with the RabbitMQ administrator password, and using them can lead to issues logging into the Management UI as that user.

### Packages

- OSS RabbitMQ v3.6.14
- 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

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.

### On-Demand Cluster Plan

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.

## 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 queues 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 HAProxy.
- 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. In PCF v2.0 and earlier, on-demand PCF services require that you host them on a network that runs separately from this network.

Cloud operators pre-provision service instances from Ops Manager. Then, 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 Ops Manager 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.

## Default Network and Service Network

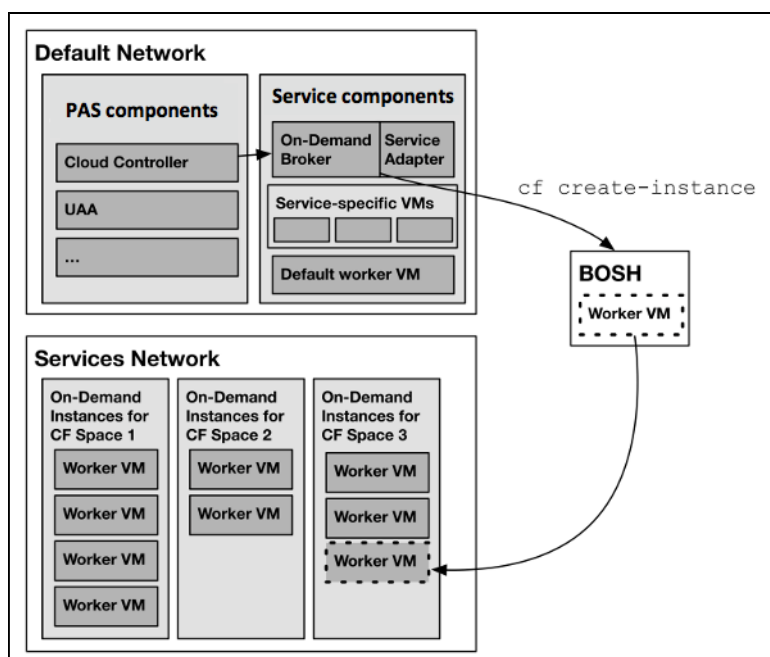
On-demand PCF services rely on the BOSH 2.0 ability to dynamically deploy VMs in a dedicated network. The on-demand service broker uses this capability to create single-tenant service instances in a dedicated service network.

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 application hogs resources on a shared cluster. Single-tenant services can also support regulatory compliance where sensitive data must be compartmentalized across separate machines.

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

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



## Required Networking Rules for On-Demand Services

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

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

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

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

This component...	Must communicate with...	Default TCP Port	Communication direction(s)	Notes
ODB	<ul style="list-style-type: none"> <li>BOSH Director</li> <li>BOSH UAA</li> </ul>	<ul style="list-style-type: none"> <li>25555</li> <li>8443</li> </ul>	One-way	The default ports are not configurable.
ODB	Deployed service instances	15672 (RabbitMQ Management UI)	One-way	This connection is for administrative tasks. Avoid opening general use, app-specific ports for this connection.
ODB	PAS (or Elastic Runtime)	8443	One-way	The default port is not configurable.
Errand VMs	<ul style="list-style-type: none"> <li>PAS (or Elastic Runtime)</li> <li>ODB</li> <li>Deployed Service Instances</li> </ul>	<ul style="list-style-type: none"> <li>8443</li> <li>8080</li> <li>15672 (RabbitMQ Management UI)</li> <li>5671-2 (AMQP/AMQPS)</li> </ul>	One-way	The default port is not configurable.
BOSH Agent	BOSH Director	4222	Two-way	The BOSH Agent runs on every VM in the system, including the BOSH Director VM. The BOSH Agent initiates the connection with the BOSH Director. The default port is not configurable.
Deployed apps on PAS (or Elastic Runtime)	Deployed service instances	<ul style="list-style-type: none"> <li>15672 (RabbitMQ Management UI)</li> <li>5671-2 (AMQP/AMQPS)</li> <li>61613-4 (STOMP/STOMPS)</li> <li>1883, 8883 (MQTT/MQTTS)</li> </ul>	One-way	This connection is for general use, app-specific tasks. Avoid opening administrative ports for this connection.
PAS (or Elastic Runtime)	ODB	8080	One-way	This port may be different for individual services. This port may also be configurable by the operator if allowed by the tile

| | | | developer. |

## Deploying the RabbitMQ Pre-Provisioned Service

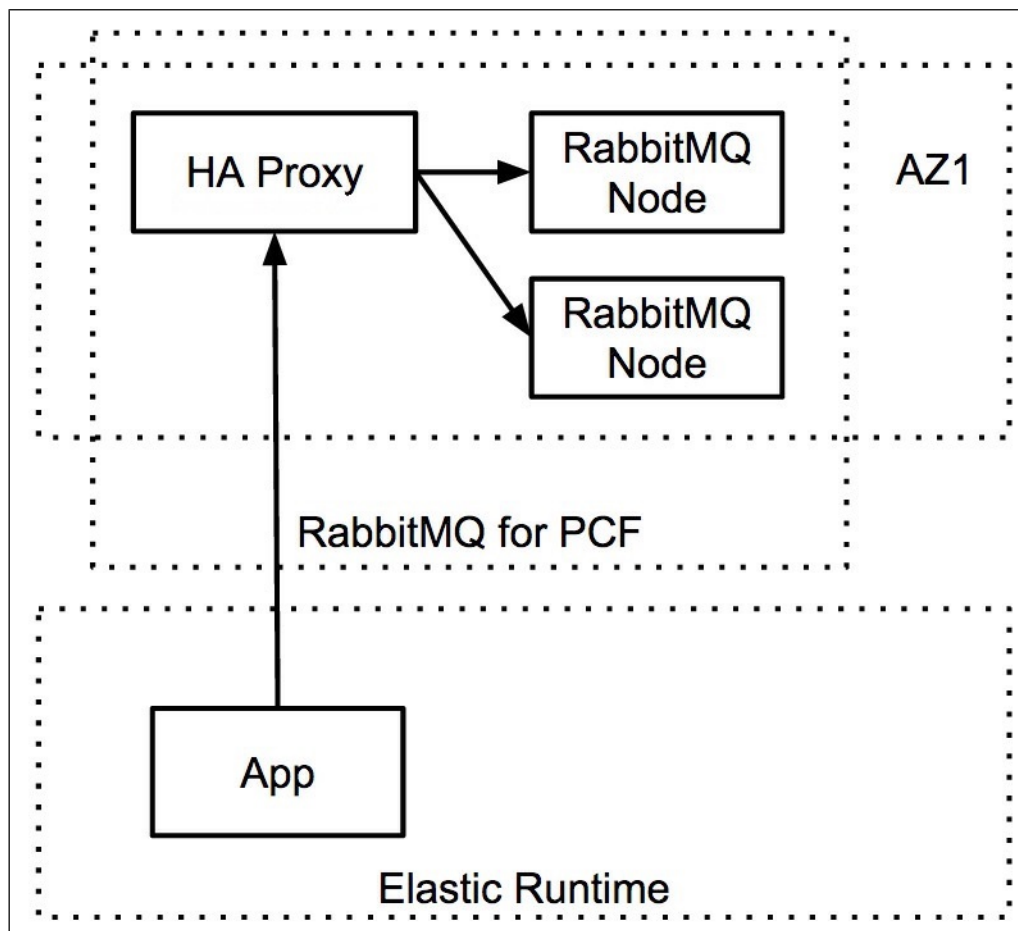
### Default Deployment

Deploying RabbitMQ for [Pivotal Cloud Foundry \(PCF\)](#) through Ops Manager will deploy 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 will occur in a single availability zone (AZ).

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



### Considerations for this deployment

- Provides HA for the RabbitMQ cluster
- Queues must be judiciously configured to be HA as they are placed on one node by default
- Customers should decide on which partition behaviour 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.

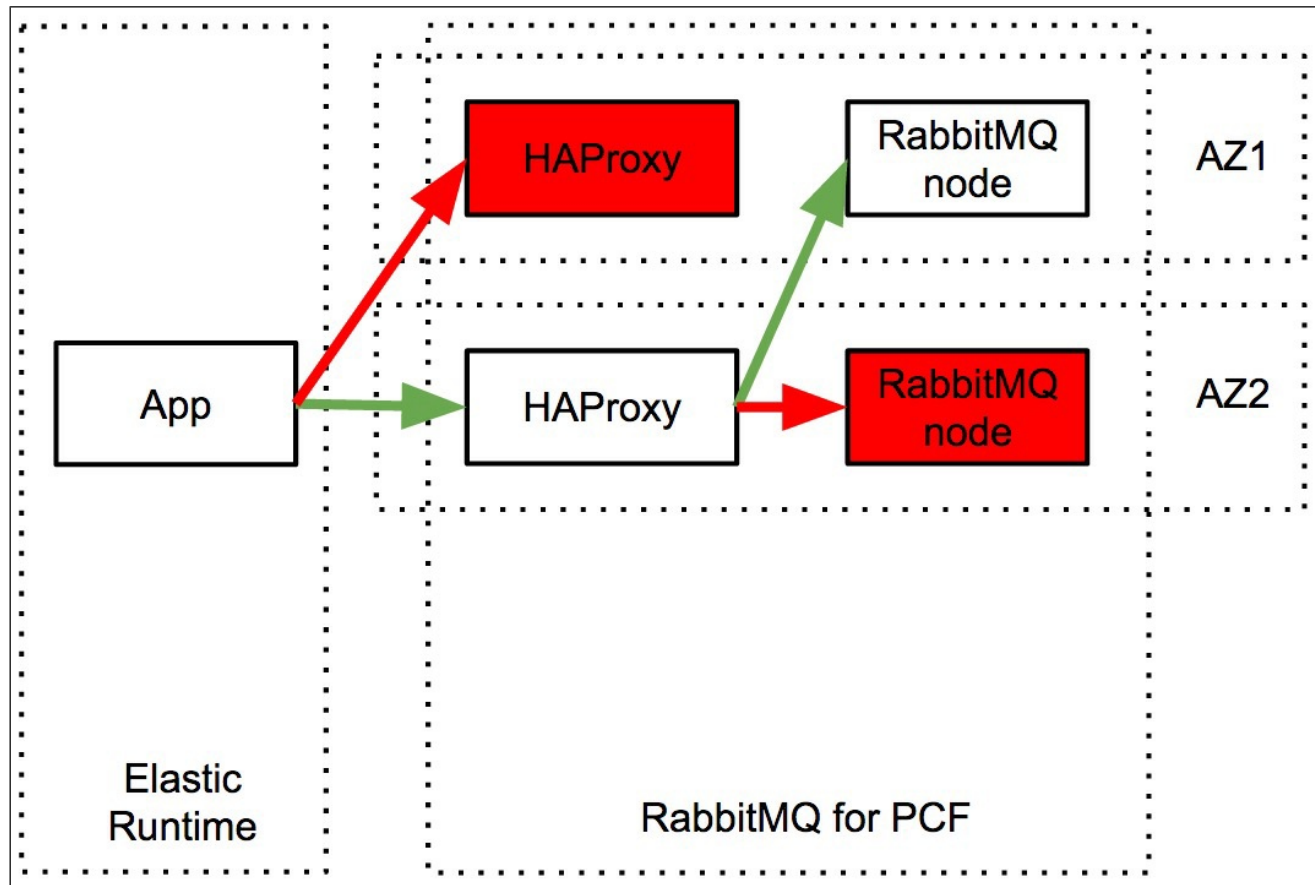
### Recommended Deployment

We recommend that RabbitMQ is deployed across at least two availability zones.

RabbitMQ server nodes should be scaled to an odd number and should be greater than 3.

Replication of queues should only be used 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.



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

It is also recommend that customers chooses an odd number of RabbitMQ server nodes of three or more.

## 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 once the tile has being 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 as follows:

1. Deploy an additional HAProxy instance through OpsManager
2. New or re-bound applications to the RabbitMQ service will see the IPs of both HAProxys immediately
3. 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

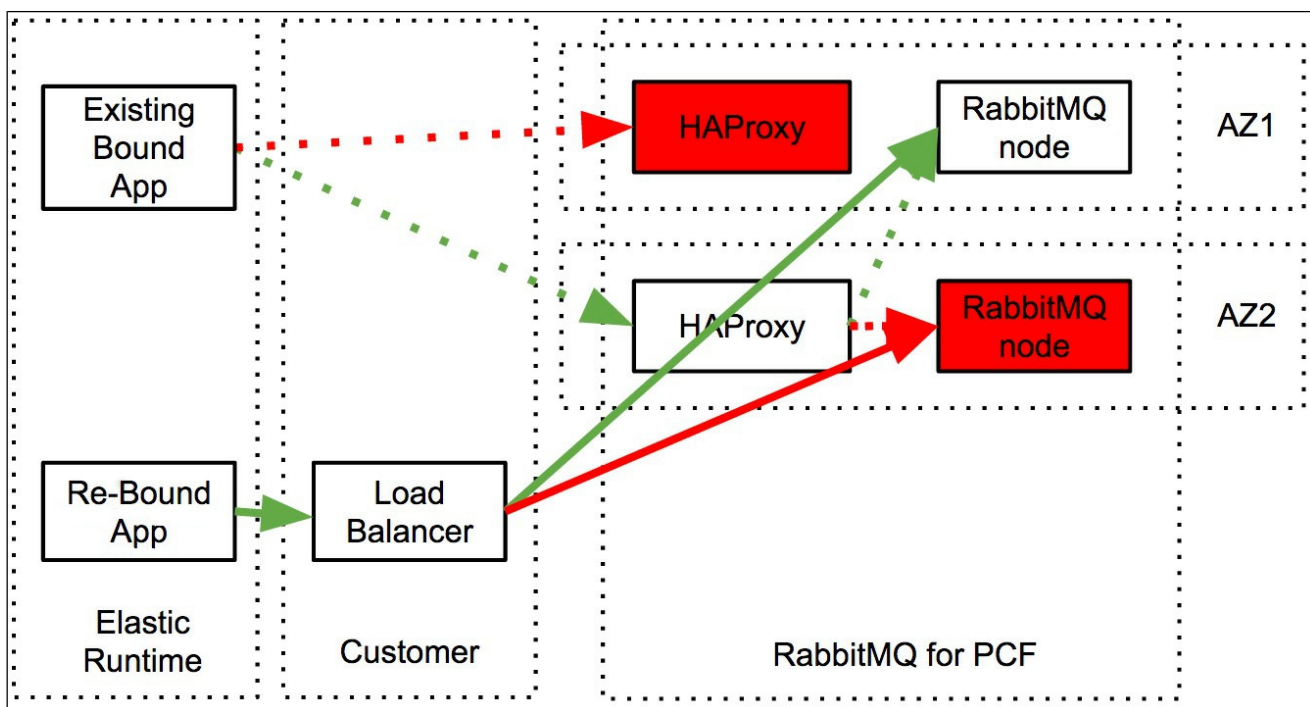
- Requires IaaS configuration for availability zones ahead of deploying the RabbitMQ tile
- Application developers will be handed the IPs of each deployed HAProxy in their environment variables
- Queues must be judiciously configured to be HA as they are placed on one node by default
- Customers should decide on which partition behaviour is best suited to their use case. For 3 or more nodes 'pause\_minority' is preferred

## Advanced Deployment

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

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

You may 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 HAProxy jobs in their applications

### Disadvantages

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

## Upgrading to this deployment from the recommended deployment

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

This can be achieved without downtime to your applications.

This can be achieved as follows:

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 OpsManager
3. Deploy the changes

4. 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`
5. Any existing instances will continue to use the HAProxy IP addresses in their `VCAP_SERVICES`
6. Phase the re-binding of existing applications to have their environment variables updated
7. Once all applications are updated
8. Reduce the instance count of the `HAProxy` job in OpsManager to 1
9. Deploy the changes

This approach works as any existing bound applications have their `VCAP_SERVICES` information cached in the cloud controller and are only updated by a re-bind request.

## Downgrading from this deployment to the recommended deployment

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

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

## Resource requirements

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

Product	Resource	Instances	CPU	Ram	Ephemeral	Persistent	Static IP	Dynamic IP
RabbitMQ	RabbitMQ node	3	2	8192	16384	30720	1	0
RabbitMQ	HAProxy for RabbitMQ	1	1	2048	4096	0	1	0
RabbitMQ	RabbitMQ service broker	1	1	2048	4096	0	1	0
RabbitMQ	Broker Registrar	1	1	1024	2048	0	0	1
RabbitMQ	Broker Deregistrar	1	1	1024	2048	0	0	1
RabbitMQ	Smoke Tests	1	1	1024	2048	0	0	1
RabbitMQ	RabbitMQ on-demand broker	1	1	1024	8192	1024	0	1
RabbitMQ	Register On-Demand Service Broker	1	1	1024	2048	0	0	1
RabbitMQ	Deregister On-Demand Service Broker	1	1	1024	2048	0	0	1
RabbitMQ	Delete All Service Instances	1	1	1024	2048	0	0	1
RabbitMQ	Upgrade All Service Instances	1	1	1024	2048	0	0	1


### Notes:

- The number of `RabbitMQ Node` 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.



The screenshot shows the PCF Ops Manager interface for RabbitMQ. The top navigation bar includes a 'P' logo and the text 'PCF Ops Manager'. Below this is a breadcrumb trail: '< Installation Dashboard' followed by 'RabbitMQ'. A secondary navigation bar contains tabs for 'Settings', 'Status', 'Credentials', and 'Logs'. The 'Settings' tab is active, showing a list of settings on the left and a detailed configuration area on the right.

**Settings List (Left Sidebar):**

- Assign AZs and Networks (highlighted)
- Pre-Provisioned RabbitMQ
- Pre-Provisioned RabbitMQ Policy
- Global Settings for On-Demand Plans
- On Demand Instance: Plan 1
- On Demand Instance: Plan 2
- On Demand Instance: Plan 3
- On Demand Instance: Plan 4
- On Demand Instance: Plan 5
- Networking
- Syslog
- Errands
- Resource Config
- Stemcell

**AZ and Network Assignments (Right Panel):**

Place singleton jobs in

- ☒ us-central1-f
- ☐ us-central1-a
- ☐ us-central1-c

Balance other jobs in

- ☒ us-central1-f
- ☒ us-central1-a
- ☒ us-central1-c

Network

lightdeepink-pas-subnet

Service Network

lightdeepink-services-subnet

Save

## Which Settings Tabs to Configure for the On-Demand Service

Configure the following tabs for the on-demand service:

RabbitMQ Settings Tab	Instructions
Assign AZs and Networks	<a href="#">Configure AZs and Networks</a>
Pre-Provisioned RabbitMQ	<a href="#">Configure Admin Credentials and Metrics Polling Interval</a>
Syslog	<a href="#">Set up Syslog Forwarding</a>
Global Settings for On-Demand Plans	<a href="#">Configure Global Settings</a>
On Demand Instance: Plan 1	<a href="#">Configure the Service Plan</a>
On Demand Instance: Plan 2	<a href="#">Configure the Service Plan</a>
On Demand Instance: Plan 3	<a href="#">Configure the Service Plan</a>
On Demand Instance: Plan 4	<a href="#">Configure the Service Plan</a>
On Demand Instance: Plan 5	<a href="#">Configure the Service Plan</a>
Errands	<a href="#">Errands</a>
Stemcell	

## 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:

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

**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:

RabbitMQ admin user credentials \*

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.

 **Note:** To rotate your administrator credentials, enter a new username and password, save your options, and redeploy by returning to the Ops Manager Installation Dashboard and clicking **Apply Changes**.

## Enter a Metrics Polling Interval

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

Metrics polling interval (min: 10) \*

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.

## Configure Global Settings


Follow the steps below to configure global settings.

1. Click **On-Demand Instance: Global Settings** and 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.

2. 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:

Field	Instructions
Enable This Plan	(Plans 2 - 5 only) To enable, select <b>Plan Enable</b> .
CF Service Access	<p>Enable or disable access to this plan, or leave access unchanged.</p> <p>If you enable Plan 1, the default setting for Plans 2 - 5 is <b>Enable Service Access</b>. 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 <b>On-Demand Instance Smoke Tests</b> errand to <b>Off</b>. For more information, see <a href="#">Errands</a>.</p> <ul style="list-style-type: none"> <li>◦ <b>Enable Service Access</b>—Gives access to all orgs, and displays the service plan to all developers in the Marketplace.</li> <li>◦ <b>Disable Service Access</b>—Disables access to all orgs, and hides the service plan to all developers in the Marketplace. This setting cannot be selected at a later time in the UI.</li> <li>◦ <b>Leave Service Access Unchanged</b>—Keeps any existing access settings, and only displays the service plan in the Marketplace to members of orgs that have access to the plan. You can change the access settings later using the cf CLI. For instructions, see <a href="#">Controlling Access to Service Plans by Org</a>.</li> </ul>
Plan Name	Accept the default or enter a name. This is the name that appears in the Marketplace.
Plan Description	Accept the default or enter a description. This description appears in the Marketplace.
Plan Features	Accept the default or enter a description. This description appears in Apps Manager.
Service Instance Quota	Enter the maximum number of on-demand service instances that can be available at one time. For more information, see <a href="#">Setting Limits for On-Demand Service Instances</a> .
Number of Nodes	Enter 1, 3, 5 or 7. This setting only affects new service instances. Previously deployed service instances are <b>not</b> updated.
Network Partition Behaviour	Select <code>pause_minority</code> or <code>autoheal</code> . Pivotal recommends using pause minority. For more information, see <a href="#">Consistency or Availability Tradeoff</a> .
RabbitMQ VM Type	Select a large VM type. The plan creates a service instance of this size. For more information, see <a href="#">Understanding RabbitMQ VM Types and Persistent Disk Size</a> below.
Persistent Disk Type	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 <a href="#">Understanding RabbitMQ VM Types and Persistent Disk Size</a> below.
AZ Placement	<p>This field is available after you complete the <a href="#">Assign AZs and Networks</a> page.</p> <ul style="list-style-type: none"> <li>◦ For a single-node plan, select one or more AZs.</li> <li>◦ 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 <a href="#">Network Latency</a> and <a href="#">Consistency or Availability Tradeoff</a> for details.</li> </ul> <p>If you change this selection after deployment, existing instances are not affected by the change. See <a href="#">Determine which AZs a Service Instance Uses</a>.</p>

## 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.

RabbitMQ VM Type\*

medium (cpu: 2, ram: 4 GB, disk: 8 GB) ▴ ▾

Persistent disk type\*

10 GB ▴ ▾

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.

RAM	Free disk alarm threshold (1.5xRAM)	Suggested disk size (2xRAM)
10 GB	15 GB	20 GB
16 GB	24 GB	32 GB
32 GB	48 GB	64 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

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](#) [↗](#).

## Apply Changes from Your Configuration

1. Return to the Ops Manager Installation Dashboard.
2. Click **Apply Changes**.

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.

Revert

Pending Changes

▼INSTALL RabbitMQ

Broker Registrar

Default

Smoke Tests

Default

Register On-Demand Service Broker

Default

On-Demand Instance Smoke Tests

Default

Upgrade All Service Instances

Default

Apply changes

Changelog

Errand	Description
Broker Registrar	Makes the pre-provisioned RabbitMQ service plans available in the Marketplace
Smoke Tests	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> .
Register On-Demand Service	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.



Broker	
On-Demand Instance Smoke Tests	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.
Upgrade All Service Instances	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 <b>Off</b> or <b>When Changed</b> , updates to on-demand plan settings will <b>not</b> be applied to existing service instances. <b>Pivotal strongly recommends that this errand is configured to always run.</b>

## Pre-Delete Errands

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

## 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:

```
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:

```
$ 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_mqtt`
- `rabbitmq_stomp`
- `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.

**PCF Ops Manager**

[Installation Dashboard](#)

## RabbitMQ

Settings | Status | Credentials | Logs

✓ Assign AZs and Networks

✓ Pre-Provisioned RabbitMQ

✓ Pre-Provisioned RabbitMQ Policy

✓ Global Settings for On-Demand Plans

✓ On Demand Instance: Plan 1

✓ On Demand Instance: Plan 2

✓ On Demand Instance: Plan 3

✓ On Demand Instance: Plan 4

✓ On Demand Instance: Plan 5

✓ Networking

✓ Syslog

✓ Errands

✓ Resource Config

✓ Stemcell

### AZ and Network Assignments

Place singleton jobs in

☒ us-central1-f

☐ us-central1-a

☐ us-central1-c

Balance other jobs in

☒ us-central1-f

☒ us-central1-a

☒ us-central1-c

Network

lightdeepink-pas-subnet

Service Network

lightdeepink-services-subnet

Save

## Which Settings Tabs to Configure for the Pre-Provisioned Service

Configure the following tabs for the pre-provisioned service:

RabbitMQ Settings Tab	Instructions
Assign AZs and Networks	<a href="#">Assign AZs and Networks</a>
Pre-Provisioned RabbitMQ	<a href="#">RabbitMQ</a>
Syslog	<a href="#">Syslog</a> <a href="#">↗</a>
Errands	<a href="#">Errands</a>
Stemcell	<a href="#">Stemcell</a>

## 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.

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

**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

Enter an admin username and password for RabbitMQ. This grants you full admin access to the RabbitMQ Management UI.

Enter the desired credentials in this section of the Pre-Provisioned RabbitMQ tab:

RabbitMQ admin user credentials \*

**Note:** To rotate your administrator credentials, enter a new username and password, save your options, and redeploy by returning to the Ops Manager Installation Dashboard and clicking **Apply Changes**.

## 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.

## RabbitMQ plugins \*

- ☐ rabbitmq\_amqp1\_0
- ☐ rabbitmq\_auth\_backend\_ldap
- ☐ rabbitmq\_auth\_mechanism\_ssl
- ☐ rabbitmq\_consistent\_hash\_exchange
- ☐ rabbitmq\_federation
- ☐ rabbitmq\_federation\_management
- ☒ rabbitmq\_management
- ☐ rabbitmq\_management\_visualiser
- ☐ rabbitmq\_mqtt
- ☐ rabbitmq\_shovel
- ☐ rabbitmq\_shovel\_management
- ☐ rabbitmq\_stomp
- ☐ rabbitmq\_tracing
- ☐ rabbitmq\_web\_stomp
- ☐ rabbitmq\_web\_stomp\_examples
- ☐ rabbitmq\_event\_exchange
- ☐ rabbitmq\_jms\_topic\_exchange

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:

### RabbitMQ cluster HAProxy ports \*

15672, 5672, 5671, 1883, 8883, 61613, 61614

- 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:

RabbitMQ server certificate

Certificate PEM

Private Key PEM

Generate RSA Certificate

RabbitMQ server CA certificate

☐ Enable 'verify\_peer' SSL certificate verification (default is 'verify\_none')
   
☐ Require peer certificate validation
   
 SSL certificate verification depth (min: 1, max: 32) \*
 

5

- 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.

Erlang cookie used by RabbitMQ nodes and rabbitmqctl


### Known Issues with Erlang Cookie

There are two known issues associated with the Erlang cookie when you do the following:

- [Cluster Scaling](#)
- [Changing the Erlang Cookie Value](#)


### 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:  

```
bosh ssh rabbitmq-server/0
```
3. Run the following commands:  

```
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 `rabbitmq.config` file is then provided to all the nodes in the cluster.

RabbitMQ configuration

RabbitMQ config file contents, can be blank

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:

```
[
  {rabbitmq_management, [
    {rates_mode, detailed}
  ]}
].
```

1. Encode the file into Base64:

```
WwogIHtyYWJiaXRlcV9tYW5hZ2VtZW50LCBbCiAgICB7cmF0ZXNfbW9kZSwgZGV0YWlscWR9CiAgXX0KXS4K
```

2. Paste the above into the **RabbitMQ configuration** field:



RabbitMQ configuration

WwogIHtyYWJiaXRtcV9tYW5hZ2VtZW50LCBbCiAgICB7cmF0ZXNfbW9kZSwgZGV0YWlsZWR9CiAgXX0KXS4K

RabbitMQ config file contents, can be blank

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:

RabbitMQ TLS Versions \*

☐ TLS v1.0 (required for JDK 6.0)
   
☒ TLS v1.1
   
☒ TLS v1.2

- 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 application developers. Enter this in this section of the **Pre-Provisioned RabbitMQ** tab:

External load balancer DNS name

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:

Metrics polling interval (min: 10) \*

30

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:

## Disk free alarm limit\*

- ☐ 50MB (legacy)
- ☒ 100% Memory (default)
- ☐ 150% Memory (recommended)
- ☐ 200% Memory (conservative)

Select the disk free limit after which RabbitMQ will raise an alarm. Value can be absolute or relative to the vm memory.

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](#) below.
- **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.

## 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.


**PCF Ops Manager**

[Installation Dashboard](#)

## RabbitMQ

Settings

Status

Credentials

Logs

✓ Assign AZs and Networks

✓ Pre-Provisioned RabbitMQ

✓ Pre-Provisioned RabbitMQ Policy

✓ Global Settings for On-Demand Plans

✓ On Demand Instance: Plan 1

✓ On Demand Instance: Plan 2

✓ On Demand Instance: Plan 3

✓ On Demand Instance: Plan 4

✓ On Demand Instance: Plan 5

✓ Networking

✓ **Syslog**

✓ Errands

✓ Resource Config

✓ Stemcell

### Forwards syslog messages to a syslog server

Do you want to configure syslog forwarding?\*

☐ No
 ☒ Yes

Syslog address \*

Syslog port \*

Transport protocol \*

TCP

Format for logs\*

RFC 5424

☐ Enable TLS

Permitted Peer

Custom CA Certificate

3. Click **Syslog**.

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

Option	Description
Syslog address	IP or DNS address of the syslog server
Syslog port	Port of the syslog server
Transport protocol	Transport protocol of the syslog server. One of <code>udp</code> , <code>tcp</code> , <code>relp</code> .
Format for logs	Format for logs. Pivotal recommends <code>RFC 5424</code> , but <code>Legacy Format</code> can be used for compatibility reasons.
Enable TLS	Enable TLS to the syslog server.
Permitted Peer	If there are several peer servers that can respond to remote syslog connections, then you may provide a wildcard in the domain, such as <code>*.example.com</code> .
Custom CA Certificate	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.

- Click **Save**.
- Return to the Ops Manager Installation Dashboard and click **Apply Changes** to redeploy with the changes.

## 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:

### Pending Changes

Revert

▼ INSTALL RabbitMQ

Broker Registrar

Smoke Tests

Register On-Demand Service Broker

On-Demand Instance Smoke Tests

Upgrade All Service Instances

Default

Default

Default

Default

Default

Apply changes

Changelog

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 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.

For more information on errand run rules, see [Errand Run Rules](#).

## Post-Deploy Errands

Errand	Description
Broker Registrar	Makes the pre-provisioned RabbitMQ service plans available in the Marketplace
Smoke Tests	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> below.
Register On-Demand Service Broker	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.

On-Demand Instance Smoke Tests	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> .
Upgrade All Service Instances	On-demand instances are updated and redeployed if there are changes to the <b>Dedicated Instance</b> settings or the tile is upgraded. If this errand is set to <b>Off</b> or <b>When Changed</b> , updates to <b>Dedicated Instance</b> settings will <i>not</i> be applied to existing service instances. Pivotal strongly recommends that this errand is configured to always run.

## 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.

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

## Stemcell

1. Click **Stemcell**.
2. If prompted, import a new stemcell version. For more information, see the information about importing the stemcell for your IaaS: [AWS](#), [Azure](#), [GCP](#), or [vSphere](#).

## Apply Configuration and Complete the Installation

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:

Destination	Ports	Protocol	Reason
HAProxy IPs	5672	tcp	Application containers using AMQP
HAProxy IPs	5671	tcp	Application containers using AMQP over SSL
HAProxy IPs	1883	tcp	Application containers using MQTT
HAProxy IPs	8883	tcp	Application containers using MQTT over SSL
HAProxy IPs	61613	tcp	Application containers using STOMP
HAProxy IPs	61614	tcp	Application containers using STOMP over SSL
HAProxy IPs	61613	tcp	Application containers using Web STOMP

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": "10.10.10.10/32"
  }
]
```

## 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.

HAProxy Static IPs

10.0.48.51,10.0.32.7

RabbitMQ Server Static IPs

10.0.48.25,10.0.48.58,10.0.32.5

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 `system` 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.

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

### 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.

## Monitoring and KPIs for On-Demand RabbitMQ for PCF

This topic explains how to monitor the health of the on-demand 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.

On-demand RabbitMQ for PCF components generate many of the same metrics as the pre-provisioned RabbitMQ service components. For information about metrics for the pre-provisioned service, see [Monitoring and KPIs for Pre-Provisioned RabbitMQ for PCF](#).

 **Note:** On-demand service metrics are prefixed with `p.rabbitmq` (with a dot), to distinguish them from the pre-provisioned service metrics.

See [Logging and Metrics](#) for general information about logging and metrics in PCF.


## Set up Syslog Forwarding

As of RabbitMQ for PCF v1.9.0, syslog forwarding is preconfigured and set to on by default. Pivotal recommends that you keep the default setting because it is good operational practice. However, you can opt out by selecting **No** for **Do you want to configure syslog?** in the Ops Manager **Settings** tab.

To enable monitoring for RabbitMQ for PCF, operators designate an external syslog endpoint for RabbitMQ component log messages. The 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.


**PCF Ops Manager**

[Installation Dashboard](#)

## RabbitMQ

Settings
Status
Credentials
Logs

Assign AZs and Networks

Pre-Provisioned RabbitMQ

Pre-Provisioned RabbitMQ Policy

Global Settings for On-Demand Plans

On Demand Instance: Plan 1

On Demand Instance: Plan 2

On Demand Instance: Plan 3

On Demand Instance: Plan 4

On Demand Instance: Plan 5

Networking

**Syslog**

Errands

Resource Config

Stemcell

### Forwards syslog messages to a syslog server

Do you want to configure syslog forwarding?\*

☐ No
☒ Yes

Syslog address \*

Syslog port \*

Transport protocol \*

TCP

Format for logs \*

RFC 5424

☐ Enable TLS

Permitted Peer

Custom CA Certificate

3. Click **Syslog**.

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

Option	Description
Syslog address	IP or DNS address of the syslog server
Syslog port	Port of the syslog server
Transport protocol	Transport protocol of the syslog server. One of <code>udp</code> , <code>tcp</code> , <code>relp</code> .
Format for logs	Format for logs. Pivotal recommends <code>RFC 5424</code> , but <code>Legacy Format</code> can be used for compatibility reasons.
Enable TLS	Enable TLS to the syslog server.
Permitted Peer	If there are several peer servers that can respond to remote syslog connections, then you may provide a wildcard in the domain, such as <code>*.example.com</code> .
Custom CA Certificate	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.

5. Click **Save**.

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

## Logging Format

With on-demand RabbitMQ for PCF logging configured, two types of components generate logs: the server nodes and the service broker:

- The logs for RabbitMQ server nodes follow the format `[job=rabbitmq-server-partition-GUID index=0]`
- The logs for the RabbitMQ service broker follow the format `[job=rabbitmq-broker-partition-GUID index=0]`

The RabbitMQ VMs 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=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] 2017/06/28 16:06:10 CEF:0|CloudFoundry|BOSH|1|agent_apilsshl|lduser=dir
<86>2017-06-28T16:06:16.704572+00:00 10.244.16.133 useradd [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] new group: name=bosh_ly0d2rbjr, GID=1003
<86>2017-06-28T16:06:16.704663+00:00 10.244.16.133 useradd [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] new user: name=bosh_ly0d2rbjr, UID=1001, GID=1003, home=/var/vcap/bosh_
<86>2017-06-28T16:06:16.736932+00:00 10.244.16.133 usermod [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] add 'bosh_ly0d2rbjr' to group 'admin'
<86>2017-06-28T16:06:16.736964+00:00 10.244.16.133 usermod [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] add 'bosh_ly0d2rbjr' to group 'vcap'
```

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:

Tag	Description
PRI	This is a value which in future will be used to describe the severity of the log message and which facility it came from.
TIMESTAMP	This is the timestamp of when the log is forwarded, for example, <code>2016-08-24T05:14:15.000003Z</code> . The timestamp value is typically slightly after when the log message was generated.
IP_ADDRESS	The internal IP address of server on which the log message originated
PROGRAM_NAME	Process name of the program the generated the message. Same as <code>app_name</code> before v1.9.0. For more information about program name, see <a href="#">RabbitMQ Program Names</a> below.
NAME	The BOSH instance group name (for example, <code>rabbitmq_server</code> )
JOB_INDEX	BOSH job index. Used to distinguish between multiple instances of the same job.
JOB_ID	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.
MESSAGE	The log message that appears

## RabbitMQ Program Names

Program Name	Description
rabbitmq_server_cluster_check	Checks that the RabbitMQ cluster is healthy. Runs after every deploy.
rabbitmq_server_node_check	Checks that the RabbitMQ node is healthy. Runs after every deploy.
rabbitmq_route_registrar_stderr	Registers the route for the management API with the Gorouter in your Pivotal Application Service (PAS) or Elastic Runtime deployment.
rabbitmq_route_registrar_stdout	Registers the route for the management API with the Gorouter in your PAS or Elastic Runtime deployment.
rabbitmq_server	The Erlang VM and RabbitMQ apps. <i>Logs may span multiple lines</i>
rabbitmq_server_drain	Shuts down the Erlang VM and RabbitMQ apps. Runs as part of the BOSH lifecycle.
rabbitmq_server_http_api_access	Access to the RabbitMQ management UI.
rabbitmq_server_init	Starts the Erlang VM and RabbitMQ.
rabbitmq_server_post_deploy_stderr	Runs the node check and cluster check. Runs after every deploy.
rabbitmq_server_post_deploy_stdout	Runs the node check and cluster check. Runs after every deploy.

Program Name	Description
rabbitmq_server_pre_start	Runs before the rabbitmq-server job is started.
rabbitmq_server_sasl	Supervisor, progress, and crash reporting for the Erlang VM and RabbitMQ apps.
rabbitmq_server_shutdown_stderr	Stops the RabbitMQ app and Erlang VM.
rabbitmq_server_shutdown_stdout	Stops the RabbitMQ app and Erlang VM.
rabbitmq_server_startup_stderr	Starts the RabbitMQ app and Erlang VM, then configures users and permissions.
rabbitmq_server_startup_stdout	Starts the RabbitMQ app and Erlang VM, then configures users and permissions.
rabbitmq_server_upgrade	Shuts down Erlang VM and RabbitMQ app if required during an upgrade.

## What Are 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/m
```

The following sections describe the metrics used as Key Performance Indicators and other useful metrics for monitoring the RabbitMQ for PCF on-demand service.

## 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 and 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

p.rabbitmq/service_broker/heartbeat	
Description	<p>RabbitMQ Service Broker <b>is alive</b> poll, which indicates if the component is available and able to respond to requests.</p> <p><b>Use:</b> 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.</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> boolean  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	<p>Yellow warning: N/A</p> <p>Red critical: &lt; 1</p>

Recommended response	Check the RabbitMQ Service Broker logs for errors. You can find this VM by targeting your RabbitMQ deployment with BOSH and running the command. <code>bosh -d service-instance_GUID vms</code>
----------------------	---


## Server Heartbeat

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

## RabbitMQ Server KPIs

The following KPIs from the RabbitMQ server component:

### File Descriptors

p.rabbitmq/rabbitmq/system/file_descriptors	
Description	<p>File descriptors consumed.</p> <p><b>Use:</b> If the number of file descriptors consumed becomes too large, the VM might lose the ability to perform disk IO, which can cause data loss.</p> <div>  <b>Note:</b> This assumes non-persistent messages are handled by retries or some other logic by the producers.         </div> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> count  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	<p><b>Yellow warning:</b> &gt; 50000  <b>Red critical:</b> &gt; 55000</p>
Recommended response	<p>The default <code>ulimit</code> for RabbitMQ for PCF v1.6 and later is 60000. If this metric is met or exceeded for an extended period of time, consider one of the following actions:</p> <ul style="list-style-type: none"> <li>Scaling the rabbit nodes in the tile <b>Resource Config</b> pane.</li> <li>Increasing the <code>ulimit</code></li> </ul>

### Erlang Processes

p.rabbitmq/rabbitmq/system/erlang_processes	
	<a href="#">Erlang</a> processes consumed by RabbitMQ, which runs on an Erlang VM.

<b>Description</b>	<p><b>Use:</b> This is the key indicator of the processing capability of a node.</p> <p><b>Origin:</b> Doppler/Firehose</p> <p><b>Type:</b> count</p> <p><b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes
<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> &gt; 900000</p> <p><b>Red critical:</b> &gt; 950000</p>
<b>Recommended response</b>	<p>The default Erlang process limit in RabbitMQ for PCF v1.6 and later is 1,048,816. If this metric meets or exceeds the recommended thresholds for extended periods of time, consider scaling the RabbitMQ nodes in the tile <b>Resource Config</b> pane.</p>

## 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

system.mem.percent	
<b>Description</b>	<p>RAM being consumed by the <code>p.rabbitmq</code> VM.</p> <p><b>Use:</b> RabbitMQ is considered to be in a good state when it has few 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.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its RAM for the past ten minutes.</p> <p><b>Origin:</b> JMX Bridge or BOSH HM</p> <p><b>Type:</b> percent</p> <p><b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes
<b>Recommended alert thresholds</b>	<p><b>Yellow warning:</b> &gt; 40</p> <p><b>Red critical:</b> &gt; 50</p>
<b>Recommended response</b>	Add more consumers to drain the queue as fast as possible.

### CPU

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

Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
Recommended response	Remember that “an empty rabbit is a happy rabbit”. Add more consumers to drain the queue as fast as possible.

## Ephemeral Disk

system.disk.percent	
Description	<p>Ephemeral Disk being consumed by the <code>p.rabbitmq</code> VM.</p> <p><b>Use:</b> If system disk fills up, there are too few consumers.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</p> <p><b>Origin:</b> JMX Bridge or BOSH HM  <b>Type:</b> percent  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
Recommended response	Remember that “an empty rabbit is a happy rabbit”. Add more consumers to drain the queue as fast as possible.

## Persistent Disk

persistent.disk.percent	
Description	<p>Persistent Disk being consumed by the <code>p.rabbitmq</code> VM.</p> <p><b>Use:</b> If system disk fills up, there are too few consumers.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</p> <p><b>Origin:</b> JMX Bridge or BOSH HM  <b>Type:</b> percent  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
Recommended response	Remember that “an empty rabbit is a happy rabbit”. Add more consumers to drain the queue as fast as possible.

## Determine If There Is a Network Partition

You can use the `reachable_nodes` metric to help to identify network partitions. This metric shows how many nodes in the cluster each individual node is aware of. A good indication that a node might be in a partition is when it is aware of only itself.

Here is an example of this metrics:

```
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/rea
```

You can create monitors to emit alerts in case a cluster seems to be in a partition. In a healthy cluster that is not undergoing upgrades, each node's `reachable_nodes` count is equal to the number of nodes in the cluster.

To monitor for network partition, Pivotal recommends alerting when one of the nodes starts reporting a `reachable_nodes` count that is less than the size of the cluster.



During rolling upgrades, nodes lose contact with other nodes. Therefore, only alert if a lowered `reachable_nodes` count persists longer than the expected upgrade time.

## Recover from a Network Partition

For information about how to recover from a network partition, see the [RabbitMQ documentation](#).

## 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.

Full Name	Unit	Description
<code>/p.rabbitmq/rabbitmq/heartbeat</code>	boolean	Indicates whether the RabbitMQ server is available and able to respond to requests
<code>/p.rabbitmq/rabbitmq/erlang/erlang_processes</code>	count	The number of Erlang processes
<code>/p.rabbitmq/rabbitmq/erlang/reachable_nodes</code>	count	The number of nodes the current node can reach
<code>/p.rabbitmq/rabbitmq/system/memory</code>	MB	The memory in MB used by the node
<code>/p.rabbitmq/rabbitmq/system/disk_free_alarm</code>	boolean	Indicates if the disk free alarm went off
<code>/p.rabbitmq/rabbitmq/system/mem_alarm</code>	boolean	Indicates if the memory alarm went off
<code>/p.rabbitmq/rabbitmq/connections/count</code>	count	The total number of connections to the node
<code>/p.rabbitmq/rabbitmq/consumers/count</code>	count	The total number of consumers registered in the node
<code>/p.rabbitmq/rabbitmq/messages/delivered</code>	count	The total number of messages with the status <code>deliver_get</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/delivered_no_ack</code>	count	The number of messages with the status <code>deliver_no_ack</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/delivered_rate</code>	rate	The rate at which messages are being delivered to consumers or clients on the node
<code>/p.rabbitmq/rabbitmq/messages/published</code>	count	The total number of messages with the status <code>publish</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/published_rate</code>	rate	The rate at which messages are being published by the node
<code>/p.rabbitmq/rabbitmq/messages/redelivered</code>	count	The total number of messages with the status <code>redeliver</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/redelivered_rate</code>	rate	The rate at which messages are getting the status <code>redeliver</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/get_no_ack</code>	count	The number of messages with the status <code>get_no_ack</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/get_no_ack_rate</code>	rate	The rate at which messages get the status <code>get_no_ack</code> on the node
<code>/p.rabbitmq/rabbitmq/messages/pending</code>	count	The number of messages with the status <code>messages_unacknowledged</code> on the node
<code>/p-rabbitmq/rabbitmq/messages/pending</code>	count	The number of messages with the status <code>messages_unacknowledged</code> on the node
<code>/p-rabbitmq/rabbitmq/messages/depth</code>	count	The number of messages with the status <code>messages_unacknowledged</code> or <code>messages_ready</code> on the node
<code>/p.rabbitmq/rabbitmq/system/file_descriptors</code>	count	The number of open file descriptors on the node
<code>/p.rabbitmq/rabbitmq/exchanges/count</code>	count	The total number of exchanges on the node
<code>/p.rabbitmq/rabbitmq/messages/available</code>	count	The total number of messages with the status <code>messages_ready</code> on the node
<code>/p.rabbitmq/rabbitmq/queues/count</code>	count	The number of queues on the node
<code>/p.rabbitmq/rabbitmq/channels/count</code>	count	The number of channels on the node
<code>/p-rabbitmq/rabbitmq/queues/VHOST-NAME/QUEUE-NAME/consumers</code>	count	The number of consumers per virtual host per queue

<code>/p-rabbitmq/rabbitmq/queues/VHOST-NAME/QUEUE-NAME/depth</code>	count	The number of messages with the status <code>messages_unacknowledged</code> or <code>messages_ready</code> per virtual host per queue
--	-------	---

## 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=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] 2017/06/28 16:06:10 CEF:0|CloudFoundry|BOSH|1|agent_apilsshl|1|duser=dir
<86>2017-06-28T16:06:16.704572+00:00 10.244.16.133 useradd [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] new group: name=bosh_ly0d2rbjr, GID=1003
<86>2017-06-28T16:06:16.704663+00:00 10.244.16.133 useradd [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] new user: name=bosh_ly0d2rbjr, UID=1001, GID=1003, home=/var/vcap/bosh
<86>2017-06-28T16:06:16.736932+00:00 10.244.16.133 usermod [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] add 'bosh_ly0d2rbjr' to group 'admin'
<86>2017-06-28T16:06:16.736964+00:00 10.244.16.133 usermod [job=rmq index=0 id=e37ecdca-5b10-4141-abd8-e1d777dfd8b5] add 'bosh_ly0d2rbjr' to group 'vcap'
```

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:

Tag	Description
PRI	This is a value which in future will be used to describe the severity of the log message and which facility it came from.
TIMESTAMP	This is the timestamp of when the log is forwarded, for example, <code>2016-08-24T05:14:15.000003Z</code> . The timestamp value is typically slightly after when the log message was generated.
IP_ADDRESS	The internal IP address of server on which the log message originated
PROGRAM_NAME	Process name of the program the generated the message. Same as <code>app_name</code> before v1.9.0. For more information about program name, see <a href="#">RabbitMQ Program Names</a> below.
NAME	The BOSH instance group name (for example, <code>rabbitmq_server</code> )
JOB_INDEX	BOSH job index. Used to distinguish between multiple instances of the same job.
JOB_ID	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.

MESSAGE The log message that appears

## RabbitMQ Program Names

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

## 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/m
```

## 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/rez
```

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>

## 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 and 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

p-rabbitmq.service_broker.heartbeat	
Description	<p>RabbitMQ Service Broker <code>is alive</code> poll, which indicates if the component is available and able to respond to requests.</p> <p><b>Use:</b> 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.</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> boolean  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	<p><b>Yellow warning:</b> N/A  <b>Red critical:</b> &lt; 1</p>
Recommended response	<p>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:</p> <pre>bosh -d service-instance_GUID vms</pre>

### HAProxy Heartbeat

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

Recommended response	BOSH and running the following command, which lists <code>HAProxy_GUID</code> : <code>bosh -d service-instance_GUID vms</code>
----------------------	---


## Server Heartbeat

p-rabbitmq.rabbitmq.heartbeat	
Description	<p>RabbitMQ Server <code>is alive</code> poll, which indicates if the component is available and able to respond to requests.</p> <p><b>Use:</b> If the server does not emit heartbeats, this indicates that it is offline. To be functional, service instances require RabbitMQ Server.</p> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> boolean  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 5 minutes
Recommended alert thresholds	<p><b>Yellow warning:</b> N/A  <b>Red critical:</b> &lt; 1</p>
Recommended response	<p>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 <code>rabbitmq</code> :</p> <p><code>bosh -d service-instance_GUID vms</code></p>

## RabbitMQ Server KPIs

The following KPIs from the RabbitMQ server component:

### File Descriptors

p-rabbitmq.rabbitmq.system.file_descriptors	
Description	<p>File descriptors consumed.</p> <p><b>Use:</b> 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.</p> <div>  <b>Note:</b> This assumes non-persistent messages are handled by retries or some other logic by the producers.         </div> <p><b>Origin:</b> Doppler/Firehose  <b>Type:</b> count  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	<p><b>Yellow warning:</b> &gt; 50000  <b>Red critical:</b> &gt; 55000</p>
Recommended response	<p>The default <code>ulimit</code> for RabbitMQ for PCF v1.6 and later is 60000. If this metric is met or exceeded for an extended period of time, consider one of the following actions:</p> <ul style="list-style-type: none"> <li>Scaling the rabbit nodes in the tile <b>Resource Config</b> pane.</li> <li>Increasing the <code>ulimit</code></li> </ul>

### Erlang Processes

p-rabbitmq.rabbitmq.system.erlang_processes	
	<a href="#">Erlang</a> processes consumed by RabbitMQ, which runs on an Erlang VM.

<b>Description</b>	<p>Use: This is the key indicator of the processing capability of a node.</p> <p>Origin: Doppler/Firehose</p> <p>Type: count</p> <p>Frequency: 30 s (default), 10 s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes
<b>Recommended alert thresholds</b>	<p>Yellow warning: &gt; 900000</p> <p>Red critical: &gt; 950000</p>
<b>Recommended response</b>	<p>The default Erlang process limit in RabbitMQ for PCF v1.6 and later is 1,048,816. If this metric meets or exceeds the recommended thresholds for extended periods of time, consider scaling the RabbitMQ nodes in the tile <b>Resource Config</b> pane.</p>

## 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

system.mem.percent	
<b>Description</b>	<p>RAM being consumed by the <code>p-rabbitmq</code> VM.</p> <p>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.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its RAM for the past ten minutes.</p> <p>Origin: JMX Bridge or BOSH HM</p> <p>Type: percent</p> <p>Frequency: 30 s (default), 10 s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes
<b>Recommended alert thresholds</b>	<p>Yellow warning: &gt; 40</p> <p>Red critical: &gt; 50</p>
<b>Recommended response</b>	Add more consumers to drain the queue as fast as possible.

### CPU

system.cpu.percent	
<b>Description</b>	<p>CPU being consumed by the <code>p-rabbitmq</code> VM.</p> <p>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.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</p> <p>Origin: JMX Bridge or BOSH HM</p> <p>Type: percent</p> <p>Frequency: 30 s (default), 10 s (configurable minimum)</p>
<b>Recommended measurement</b>	Average over last 10 minutes

Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
Recommended response	Remember that “an empty rabbit is a happy rabbit”. Add more consumers to drain the queue as fast as possible.

## Ephemeral Disk

system.disk.percent	
Description	<p>Ephemeral Disk being consumed by the <code>p-rabbitmq</code> VM.</p> <p><b>Use:</b> If system disk fills up, there are too few consumers.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</p> <p><b>Origin:</b> JMX Bridge or BOSH HM  <b>Type:</b> percent  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
Recommended response	Remember that “an empty rabbit is a happy rabbit”. Add more consumers to drain the queue as fast as possible.

## Persistent Disk

persistent.disk.percent	
Description	<p>Persistent Disk being consumed by the <code>p-rabbitmq</code> VM.</p> <p><b>Use:</b> If system disk fills up, there are too few consumers.</p> <p>Healthmonitor reports when RabbitMQ uses more than 40% of its CPU for the past ten minutes.</p> <p><b>Origin:</b> JMX Bridge or BOSH HM  <b>Type:</b> percent  <b>Frequency:</b> 30 s (default), 10 s (configurable minimum)</p>
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	Yellow warning: > 60 Red critical: > 75
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.

Full Name	Unit	Description
<code>/p-rabbitmq.rabbitmq.heartbeat</code>	boolean	Indicates whether the RabbitMQ server is available and able to respond to requests
<code>/p-rabbitmq.rabbitmq/erlang/erlang_processes</code>	count	The number of Erlang processes
	MB	The memory in MB used by the node



/n-rabbitmq/rabbitmq/system/memory /p-rabbitmq/rabbitmq/system/disk_free_alarm	boolean	Indicates if the disk free alarm went off
/p-rabbitmq/rabbitmq/system/mem_alarm	boolean	Indicates if the memory alarm went off
/p-rabbitmq/rabbitmq/connections/count	count	The total number of connections to the node
/p-rabbitmq/rabbitmq/consumers/count	count	The total number of consumers registered in the node
/p-rabbitmq/rabbitmq/messages/delivered	count	The total number of messages with the status <code>deliver_get</code> on the node
/p-rabbitmq/rabbitmq/messages/delivered_no_ack	count	The number of messages with the status <code>deliver_no_ack</code> on the node
/p-rabbitmq/rabbitmq/messages/delivered_rate	rate	The rate at which messages are being delivered to consumers or clients on the node
/p-rabbitmq/rabbitmq/messages/published	count	The total number of messages with the status <code>publish</code> on the node
/p-rabbitmq/rabbitmq/messages/published_rate	rate	The rate at which messages are being published by the node
/p-rabbitmq/rabbitmq/messages/redelivered	count	The total number of messages with the status <code>redeliver</code> on the node
/p-rabbitmq/rabbitmq/messages/redelivered_rate	rate	The rate at which messages are getting the status <code>redeliver</code> on the node
/p-rabbitmq/rabbitmq/messages/got_no_ack	count	The number of messages with the status <code>get_no_ack</code> on the node
/p-rabbitmq/rabbitmq/messages/get_no_ack_rate	rate	The rate at which messages get the status <code>get_no_ack</code> on the node
/p-rabbitmq/rabbitmq/messages/pending	count	The number of messages with the status <code>messages_unacknowledged</code> on the node
/p-rabbitmq/rabbitmq/messages/depth	count	The number of messages with the status <code>messages_unacknowledged</code> or <code>messages_ready</code> on the node
/p-rabbitmq/rabbitmq/system/file_descriptors	count	The number of open file descriptors on the node
/p-rabbitmq/rabbitmq/exchanges/count	count	The total number of exchanges on the node
/p-rabbitmq/rabbitmq/messages/available	count	The total number of messages with the status <code>messages_ready</code> on the node
/p-rabbitmq/rabbitmq/queues/count	count	The number of queues on the node
/p-rabbitmq/rabbitmq/channels/count	count	The number of channels on the node
/p-rabbitmq/rabbitmq/queues/VHOST-NAME/QUEUE-NAME/consumers	count	The number of consumers per virtual host per queue
/p-rabbitmq/rabbitmq/queues/VHOST-NAME/QUEUE-NAME/depth	count	The number of messages with the status <code>messages_unacknowledged</code> or <code>messages_ready</code> per virtual host per queue

## HAProxy Metrics

RabbitMQ for PCF HAProxy components emit the following metrics.

Name Space	Unit	Description
/p-rabbitmq.haproxy.heartbeat	boolean	Indicates whether the RabbitMQ HAProxy component is available and able to respond to requests
/p-rabbitmq/haproxy/health/connections	count	The total number of concurrent front-end connections to the server
/p-rabbitmq/haproxy/backend/qsizes/amqp	size	The total size of the AMQP queue on the server
/p-rabbitmq/haproxy/backend/retries/amqp	count	The number of AMQP retries to the server
/p-rabbitmq/haproxy/backend/ctime/amqp	time	The total time to establish the TCP AMQP connection to the server

## Setting Limits for On-Demand Service Instances

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

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

- [Global](#)
- [Plan](#)
- [Org](#)
- [Space](#)

After you set quotas, you can:

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


## Create Global-level Quotas

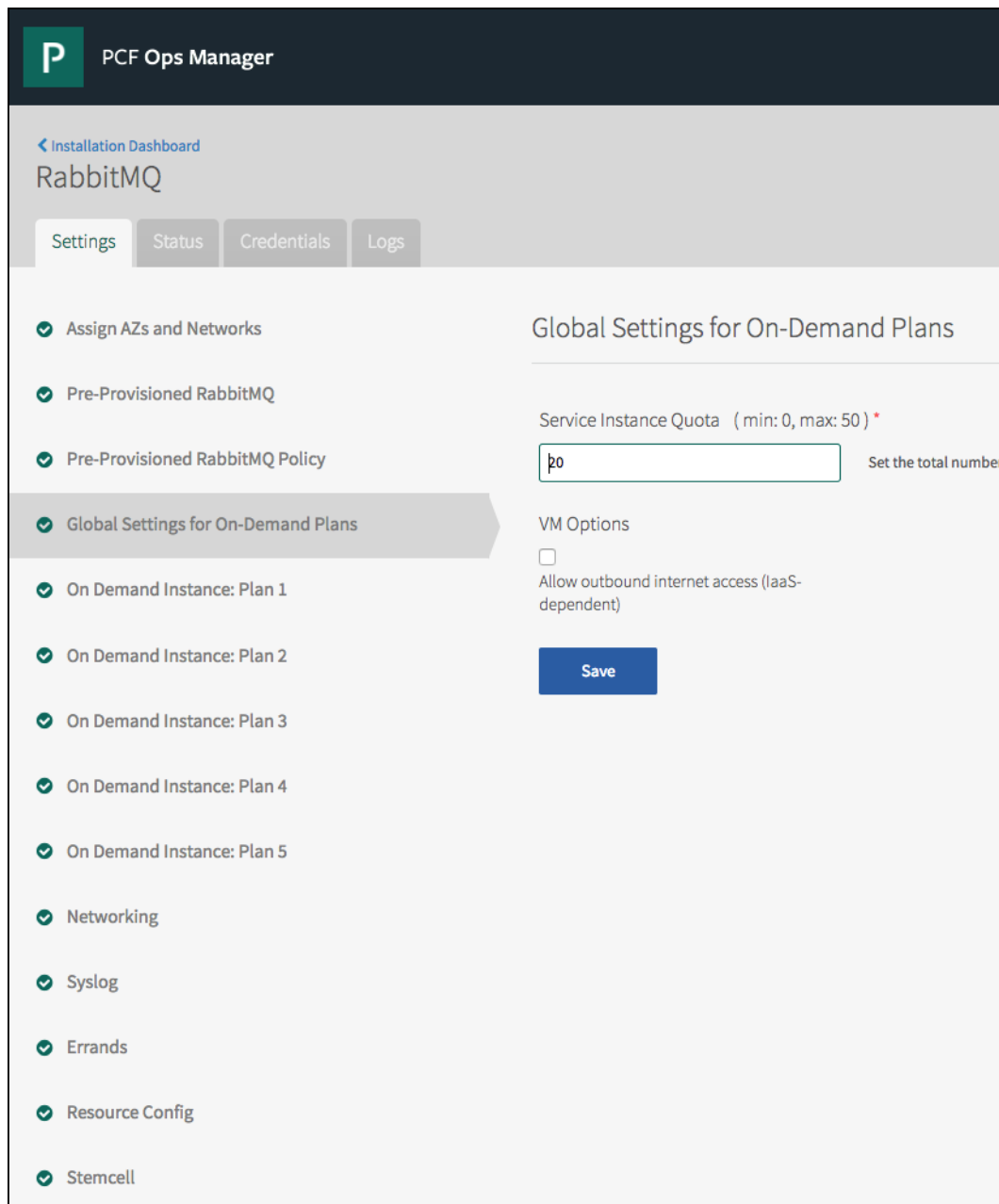
Each Pivotal Cloud Foundry (PCF) service has a separate service broker. A global quota at the service level sets the maximum number of service instances that can be created by a given service broker. If a service has more than one plan, then the number of service instances for all plans combined cannot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

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

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

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

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




## Create Plan-level Quotas

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

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

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

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

P

PCF Ops Manager

[Installation Dashboard](#)

## RabbitMQ

Settings

Status

Credentials

Logs

✓ Assign AZs and Networks

✓ Pre-Provisioned RabbitMQ

✓ Pre-Provisioned RabbitMQ Policy

✓ Global Settings for On-Demand Plans

✓ On Demand Instance: Plan 1

✓ On Demand Instance: Plan 2

✓ On Demand Instance: Plan 3

✓ On Demand Instance: Plan 4

✓ On Demand Instance: Plan 5

✓ Networking

✓ Syslog

✓ Errands

✓ Resource Config

✓ Stemcell

### Plan 3 Configuration

Please read the documentation before changing any of these settings, as

Enable This Plan\*

☐ Plan Disable
 ☒ Plan Enable

CF Service Access\*

Enable Service Access

Plan Name \*

plan-3

The plan name that will appear in the CF marketplace

Plan Description \*

plan-3 description

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-f
 ☒ us-central1-a
 ☐ us-central1-c

## Create and Set Org-level Quotas

An org-level quota applies to all PCF services and sets the maximum number of service instances an organization can create within PCF. For example, if you set your org-level quota to 100, developers can create up to 100 service instances in that org using any combination of PCF services.

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

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

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

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

where these variables are:

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

For example:

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

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

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

For example: `cf set-quota dev_org myquota`

For more information on managing org-level quotas, see [Creating and Modifying Quota Plans](#).

## Create and Set Space-level Quotas

A space-level service quota applies to all PCF services and sets the maximum number of service instances that can be created within a given space in PCF. For example, if you set your space-level quota to 100, developers can create up to 100 service instances in that space using any combination of PCF services.

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

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

1. Run the following command to create the quota:

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

where these variables are:

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

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

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

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

For example:

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

For more information on managing space-level quotas, see [Creating and Modifying Quota Plans](#).

## View Current Org and Space-level Quotas

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

```
cf org ORG-NAME
```

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


```
cf space SPACE-NAME
```

For more information on managing org and space-level quotas, see the [Creating and Modifying Quota Plans](#).

## Monitor Quota Use and Service Instance Count

Service-level and plan-level quota use, and total number of service instances, are available through the on-demand broker metrics emitted to Loggregator. These metrics are listed below:

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

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

## Calculate Resource Costs for On-Demand Plans

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 VM and Persistent Disk types selected in the plan configuration(s). The specific costs depend on your IaaS.


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

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

AZ Placement \*  
☒ europe-west1-b  
☐ europe-west1-c  
☐ europe-west1-d

VM Type\*

Persistent Disk Type\*

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

## Calculate Maximum Resource Cost Per On-Demand Plan

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

**plan quota x cost of selected resources**

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

**(15 x cost of micro.cpu VM type) + (15 x cost of 20 GB persistent disk)**

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

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

Here is an example:

**(plan1 quota x plan1 resource cost) + (plan2 quota x plan2 resource cost) = max cost for all plans**

## Calculate Actual Resource Cost of all On-Demand Plans

To calculate the current actual resource cost across all your on-demand plans:

1. Find the number of instances currently provisioned for each active plan by looking at the `total_instance` [metric](#) for that plan.
2. Multiply the `total_instance` count for each plan by that plan's resource costs. Record the costs for each plan.
3. Add up the costs noted in Step 2 to get your total current resource costs.

For example:


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

## 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:

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

For example:

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

For more information about the above command, see [Access Control](#).



## Creating Isolation with the RabbitMQ for PCF Replicator

### Overview

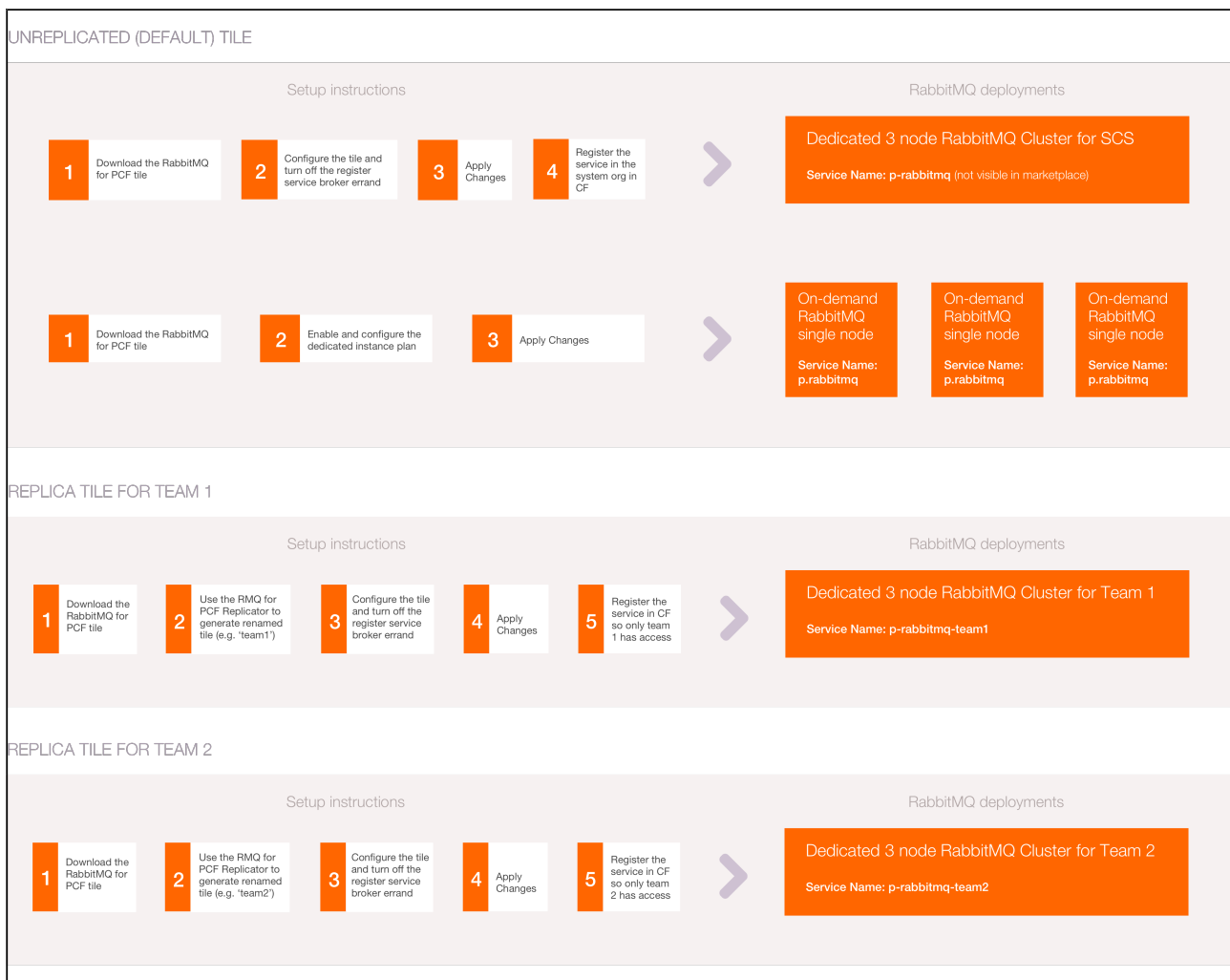
RabbitMQ for PCF Replicator is a tool that allows you to install multiple RabbitMQ for Pivotal Cloud Foundry (PCF) tiles in a single Ops Manager environment. This lets you run multiple pre-provisioned RabbitMQ clusters that are isolated from each other.

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, all you need to do is [disable the broker registrar errand in Ops Manager](#). This prevents service registration in the Marketplace. 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:

```
./rabbitmq-replicator-darwin\
--name YOUR_DESIRED_TILE_NAME\
--path PATH_TO_TILE\
--output DESIRED_FILE_NAME.pivotal
```

The following are the parameters expected in the above syntax:

Parameter	Description
<code>name</code>	The desired unique identifier for the replica tile, which is used to generate manifests, deployment names, and Marketplace name for the service. Only alphanumeric characters and <code>-</code> are accepted by the tool.
<code>path</code>	The location of the original, unreplicated, RabbitMQ for PCF source tile that you downloaded
<code>output</code>	The desired file name and path for the replica tile

## 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:

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

## 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_URL`, and `SERVICE_NAME`:

```
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. A policy configured in Ops Manager is only applied when a service instance (vhost) is created (using `cf create service`) and does not affect service instances (vhosts) that have already been deployed. 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 `priority number` applied through the Ops Manager configuration.

### 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**.

RabbitMQ Policy definition applied to new instances

☒ Enable custom policy on new instances

Policy for new instances

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

Select the network partition behavior of the RabbitMQ cluster\*

pause\_minority

Save

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.

Policies

▼ All policies

Filter:  ☐ Regex (?)

Name	Pattern	Apply to	Definition	Priority
<b>operator_set_policy</b>	.*	all	ha-mode: exactly ha-params: 2 ha-sync-mode: automatic	50


In the **Queues** section shown below, you can see that any new queues created have the policy automatically applied.

Queues

▼ All queues

Filter:  ☐ Regex (?)

Overview			Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
<b>test</b>	<span>D</span> operator_set_policy	<span>running</span>	0	0	0				

 **Note:** Developers can obtain the URL of the policy from `VCAP_SERVICES` for app developers.

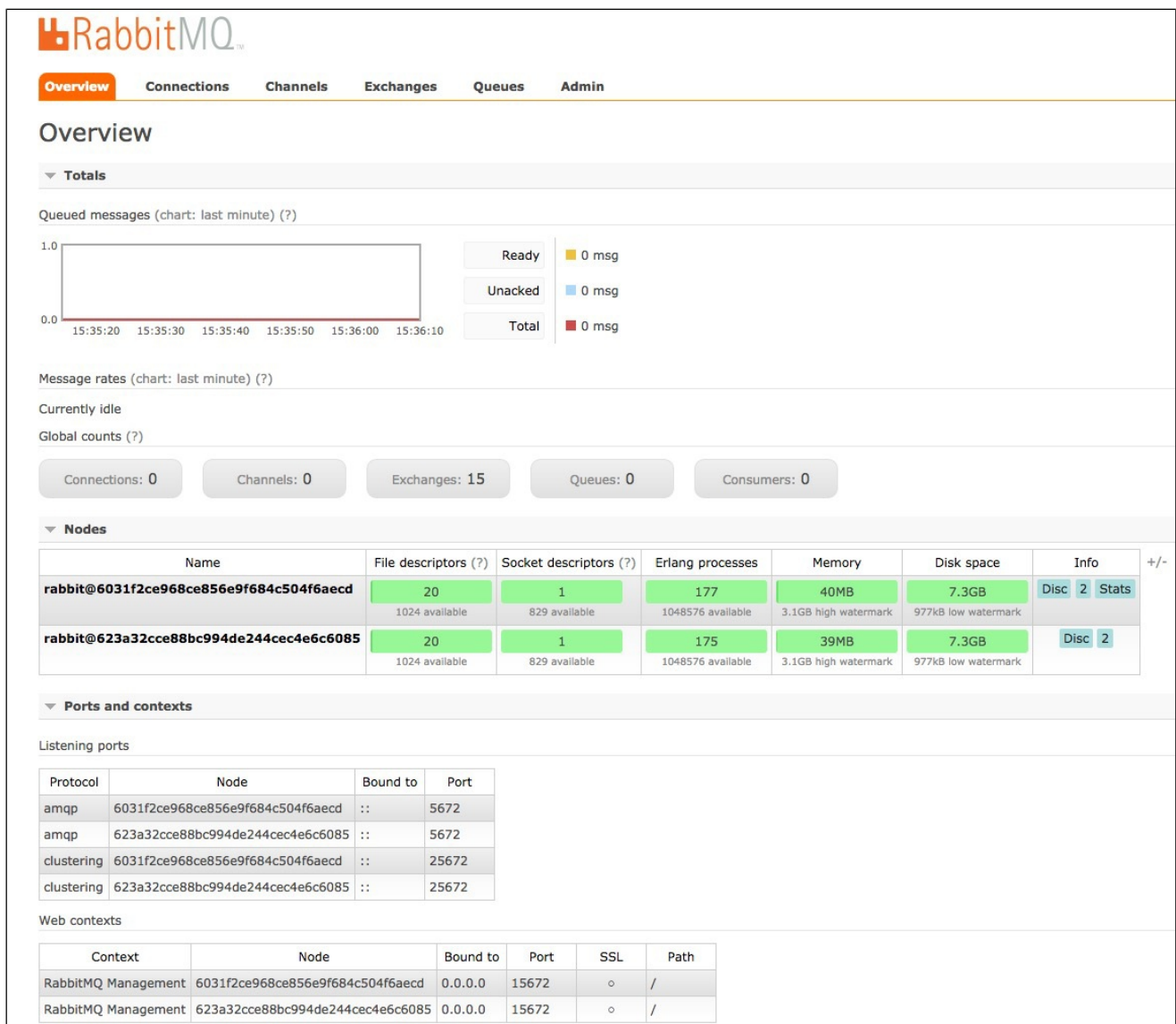
## Managing the RabbitMQ® Service

### RabbitMQ Management Dashboard

#### Admin User

To gain access to the management dashboard as the `admin` user, visit `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 **vhost**.

The appropriate URL is accessible via the Manage button within the Apps Manager.



SERVICES <span>Add Service</span>		
SERVICE INSTANCE	SERVICE PLAN	BOUND APPS
rabbit <a href="#">Manage</a>   <a href="#">Documentation</a>   <a href="#">Support</a>   <a href="#">Delete</a>	RabbitMQ for Pivotal CF Production	0

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>
```

```
➔ ~ cf env lab-rat
Getting env variables for app lab-rat in org development / space development as me...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "p-rabbitmq": [
      {
        "credentials": {
          "dashboard_url": "http://pivotal-rabbitmq.crystal.pepsi.cf-app.com/#/login/fd87c5d4-ca55-41f3-ba8a-cf0b0711f052/243bvlaa8c5m5dka1701b207ff",
```

## 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=$PATH:/var/vcap/packages/rabbitmq-server/bin
bash-4.1# export PATH=$PATH:/var/vcap/packages/erlang/bin
bash-4.1# rabbitmqctl cluster_status
Cluster status of node rabbit@node0 ...
[{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](#).

## 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.

Upgrade Type	Will Downtime Be Required For This Upgrade / Update
Major Tile Version	The RabbitMQ cluster is taken offline for the duration of the upgrade.
Minor Tile Version	The RabbitMQ cluster is taken offline for the duration of the upgrade.
Patch Tile Version	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.
Stemcell-Only Patch Tile Version	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.

## 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.
- The cluster becomes unavailable only when upgrading between specific versions of Erlang or RabbitMQ. This is clearly stated in the release notes for those versions.
- 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 to a Newer Version of RabbitMQ or Erlang

Review the following before starting an upgrade that includes a new version of RabbitMQ or Erlang:

- Ensure the cluster is healthy via the RabbitMQ Management UI. You cannot rely on the BOSH `instances` output, that reflects the state of Erlang VM, not RabbitMQ.
- Stop RabbitMQ when upgrading the RabbitMQ or Erlang VM version.

## 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.

## Release Policy

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

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

## Troubleshooting and FAQs for On-Demand RabbitMQ for PCF

In this topic:

- [About the BOSH CLI](#)
- [How to Retrieve a Service Instance GUID](#)
- [Troubleshooting Errors](#)
  - [Failed Install](#)
  - [Cannot Create or Delete Service Instances](#)
  - [Broker Request Timeouts](#)
  - [Cannot Bind to or Unbind from Service Instances](#)
  - [Cannot Connect to a Service Instance](#)
  - [Upgrade All Instances Fails](#)
  - [Missing Logs and Metrics](#)
  - [Failed Deployment on Upgrade or after Apply Changes](#)
- [Troubleshooting Components](#)
  - [BOSH problems](#)
  - [Configuration](#)
  - [Authentication](#)
  - [Networking](#)
  - [Quotas](#)
- [Techniques for Troubleshooting](#)
  - [Parse a Cloud Foundry \(CF\) Error Message](#)
  - [Access Broker and Instance Logs and VMs](#)
  - [Run Service Broker Errands to Manage Brokers and Instances](#)
  - [Select the BOSH Deployment for a Service Instance](#)
  - [Get Admin Credentials for a Service Instance](#)
  - [Reinstall a Tile](#)
  - [View Resource Saturation and Scaling](#)
  - [Identify Service Instance Owner](#)
  - [Monitor Quota Saturation and Service Instance Count](#)
- [Frequently Asked Questions](#)
  - [What should I check before deploying a new version of the tile?](#)
  - [What is the correct way to stop and start RabbitMQ in PCF?](#)
  - [What happens when I run bosh stop rabbitmq-server?](#)
  - [What happens when bosh stop rabbitmq-server fails?](#)
  - [What do I do when bosh stop rabbitmq-server fails?](#)
  - [How can I manually back up the state of the RabbitMQ cluster?](#)
  - [What pre-upgrade checks should I do?](#)
- [Knowledge Base \(Community\)](#)
- [File a Support Ticket](#)

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

## About the BOSH CLI

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

This topic provides examples of using each version of the BOSH CLI. Consult the table below to determine which version of the CLI is supported for your installation.

PCF Version	BOSH CLI Version
1.10	CLI v1

1.11	CLI v1 or CLI v2 (Pivotal recommends CLI v2)
1.12 and later	CLI v2

## 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 the following:

```
Instance provisioning failed: There was a problem completing your request. Please contact your operations team providing the following information: service: redis-acceptance, service-instance-g
```

1. If the BOSH error shows a problem with the deployment manifest:

- a. Download the manifest for the on-demand service instance by running:

```
bosh download manifest service-instance_SERVICE-INSTANCE-GUID MY-SERVICE.yml.
```

- b. Check the manifest for configuration errors.



**Note:** This error does not apply if you are using BOSH CLI v2. In that case, to troubleshoot possible problems with the manifest, open it in a text editor and inspect the manifest there.

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 one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh task TASK-ID</code>
1.11	<code>bosh2 task TASK-ID</code>
1.12 and later	<code>bosh task TASK-ID</code>

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/ser
```

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 one of these commands depending on your Ops Manager version:
    - 1.10 and earlier: `bosh tasks`
    - 1.11: `bosh2 tasks`
    - 1.12 and later: `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. Type `cf service MY-INSTANCE --guid`. This confirms that the the RabbitMQ for PCF service instance exists in BOSH and CF, and returns a GUID.
2. Using the GUID obtained above, run one of the following BOSH CLI commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh vms service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID vms</code>



1.12 and later v	BOSH Command
Ops Manager Version	id 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:

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 PCF Elastic Runtime tile is deployed to has network access to the service network. You can find the network definition for this service network in the Ops Manager 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 ERT tile and see the network it is assigned to. Make sure that these networks can access each other.

## Upgrade All 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
- b. Run `cf nozzle -f ValueMetric | grep --line-buffered "on-demand-broker/MY-SERVICE"` to find logs from your service in the `cf nozzle` output.

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

#### Missing BOSH Director UUID

 **Note:** This error does not occur if you are using BOSH CLI v2

If using the BOSH CLI v1, re-add the `director_uuid` to the manifest:

1. Run `bosh status --uuid` and record the `director_uuid` value from the output.
2. Edit the manifest and add the `director_uuid: DIRECTOR-UUID` from the last step at the top of the manifest.

For more, see [Deployment Identification](#) in the BOSH docs.

#### 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 Elastic Runtime tile.

## Networking

Common issues include:

1. Network latency when connecting to the RabbitMQ for PCF service instance to create or delete a binding.
  - Solution: Try again or improve network performance
2. Network firewall rules are blocking connections from the RabbitMQ for PCF service broker to the service instance.
  - Solution: 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.
3. Network firewall rules are blocking connections from the service network to the BOSH director network.
  - Solution: Ensure that service instances can access the Director so that the BOSH agents can report in.
4. Apps cannot access the service network.
  - Solution: Configure Cloud Foundry application security groups to allow runtime access to the service network.
5. Problems accessing BOSH's UAA or the BOSH director.
  - Solution: Follow network troubleshooting and check that the BOSH director is online.

### Validate Service Broker Connectivity to Service Instances

1. To validate you can `bosh2 ssh` onto the RabbitMQ for PCF service broker:
  - **With BOSH CLI v2:** Target the deployment, and reach the service instance.
  - **With BOSH CLI v1:** Download the broker manifest and target the deployment, then try to reach the service instance.
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 to 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.
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 one of the following commands:

Ops Manager Version	BOSH Command
1.10 or earlier	<code>bosh vms --vitals service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID vms --vitals</code>
1.12 and later	<code>bosh -d service-instance_GUID vms --vitals</code>

For additional information, run one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh instances --ps --vitals</code>
1.11	<code>bosh2 instances --ps --vitals</code>
1.12 and later	<code>bosh instances --ps --vitals</code>

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`.

```
$ cf service myservice

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

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

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

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

The `broker-request-guid` maps to the portion of the On-Demand Broker log containing the failed step. Access the broker log through your syslog aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

### Access Broker and Instance Logs and VMs

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

## Access Broker Logs and 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 one of the following commands, depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh deployments</code>
1.11	<code>bosh2 deployments</code>
1.12 and later	<code>bosh deployments</code>

2. For BOSH CLI v1 only:

- a. Run `bosh download manifest ODB-DEPLOYMENT-NAME odb.yml` to download the ODB manifest.
- b. Select the ODB deployment using `bosh deployment odb.yml`.

3. View VMs in the deployment using one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh instances</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME instances</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME instances</code>

4. SSH onto the VM by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh ssh service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID ssh</code>
1.12 and later	<code>bosh -d service-instance_GUID ssh</code>

5. Download the broker logs by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh logs service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID logs</code>
1.12 and later	<code>bosh -d service-instance_GUID logs</code>

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

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

## Access Service Instance Logs and VMs

1. To target an individual service instance deployment, retrieve the GUID of your service instance with the cf CLI command `cf service MY-SERVICE --guid`.
2. **For BOSH CLI v1 only:**

- a. Run `bosh status --uuid` to retrieve the BOSH Director GUID.

 **Note:** “GUID” and “UUID” mean the same thing.

- b. To download your BOSH manifest for the service, run `bosh download manifest service-instance_BOSH-DIRECTOR-GUID MANIFEST.yml` using the GUID you just obtained and a filename you want to save the manifest as.
- c. Edit the following line in the service instance manifest that you just saved, to include the current BOSH Director GUID:

```
director_uuid: BOSH-DIRECTOR-GUID
```

- d. Run `bosh deployment MANIFEST.yml` to select the deployment using the Director UUID.

3. View VMs in the deployment using one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh instances</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME instances</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME instances</code>

4. SSH onto a VM by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh ssh service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID ssh</code>
1.12 and later	<code>bosh -d service-instance_GUID ssh</code>

5. Download the instance logs by running one of the following commands:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh logs service-instance_GUID</code>
1.11	<code>bosh2 -d service-instance_GUID logs</code>
1.12 and later	<code>bosh -d service-instance_GUID logs</code>

## 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 errands:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command:

```
bosh deployment BOSH_MANIFEST.yml
```

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand ERRAND_NAME</code>

Ops Manager Version	BOSH Command
1.12 and later	<code>bosh -d DEPLOYMENT_NAME run-errand ERRAND_NAME</code>

Examples:

```
bosh run errand deregister-broker
```

```
bosh2 -d DEPLOYMENT-NAME run-errand deregister-broker
```

## Register Broker

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

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

The errand does the following:

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

To run the errand, do the following:

- For BOSH CLI v1 only:** Select the broker deployment by running this command:

```
bosh deployment BOSH_MANIFEST.yml
```

- Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand register-broker</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME run-errand register-broker</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME run-errand register-broker</code>

## 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, do the following:

- For BOSH CLI v1 only:** Select the broker deployment by running the command:

```
bosh deployment BROKER_MANIFEST.yml
```

- Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand deregister-broker</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME run-errand deregister-broker</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME run-errand deregister-broker</code>



## Upgrade All Service Instances

If you have made changes to the plan definition or uploaded a new tile into Ops Manager, you may want to upgrade all the RabbitMQ for PCF service instances to the latest software/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 serially:
  - Issues an upgrade command to the on-demand broker.
  - Re-generates 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. Run the errand by following either of the procedures below.

To run the errand, you can either select the errand through the Ops Manager UI and have it run when you click `Apply Changes`, or do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command:

```
bosh deployment BOSH_MANIFEST.yml
```

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand upgrade-all-service-instances</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME run-errand upgrade-all-service-instances</code>

## Delete All Service Instances

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

Orphan BOSH deployments don't correspond to a known service instance. While rare, orphan deployments can occur. Use the `orphan-deployments` errand to identify them.

The errand does the following:

- Unbinds all applications from the service instances.
- Deletes all service instances sequentially.
- Checks if any instances have been created while the errand was running.
- If newly-created instances are detected, the errand fails.

**⚠ 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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running the command:

```
bosh deployment BROKER_MANIFEST.yml
```

2. Run one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand delete-all-service-instances</code>
1.11	<code>bosh2 -d service-instance_GUID delete-deployment</code>
1.12 and later	<code>bosh -d service-instance_GUID delete-deployment</code>

## 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, do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running the command:  
`bosh deployment BROKER_MANIFEST.yml`
2. Run the errand using one of the following commands depending on your Ops Manager version:

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh run errand orphan-deployments</code>
1.11	<code>bosh2 -d DEPLOYMENT-NAME run-errand orphan-deployments</code>
1.12 and later	<code>bosh -d DEPLOYMENT-NAME run-errand orphan-deployments</code>

If orphan deployments exist, the errand script will:

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

For example:

```
[stdout]
[{"deployment_name":"service-instance_80e3c5a7-80be-49f0-8512-44840f3c4d1b"}]

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

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

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

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

If no orphan deployments exist, the errand script will:

- Exit with exit code 0
- Stdout will be an empty list of deployments

- Stderr will be `None`

```
[stdout]
[]

[stderr]
None

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

If the errand encounters an error during running it will:

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

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

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

Ops Manager Version	BOSH Command
1.10 and earlier	<code>bosh delete deployment service-instance_SERVICE-INSTANCE-GUID</code>
1.11	<code>bosh2 delete-deployment service-instance_SERVICE-INSTANCE-GUID</code>
1.12 and later	<code>bosh delete-deployment service-instance_SERVICE-INSTANCE-GUID</code>

## Select the BOSH Deployment for a Service Instance


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

1. Retrieve the GUID of your service instance with the command `cf service YOUR-SERVICE-INSTANCE --guid`.
2. To download your BOSH manifest for the service, run `bosh download manifest service-instance_SERVICE-INSTANCE-GUID myservice.yml` using the GUID you just obtained and a file name you want to use when saving the manifest.
3. Run `bosh deployment MY-SERVICE.yml` to select the deployment.

## Get Admin Credentials for a Service Instance

To retrieve the admin and read-only admin credentials for a service instance, perform the following steps:

1. [Identify the service deployment by GUID.](#)
2. [Log into BOSH](#).
3. [Download the manifest for the service instance](#) and add the GUID if using the BOSH CLI v1.

 Skip this step if you are using the BOSH CLI v2. You cannot download the manifest with the BOSH CLI v2. Open it in a text editor instead.

4. Look in the manifest for the `admin` and `roadmin` 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 with `cf login`.
- b. Use `cf m` to confirm that the Marketplace does not list RabbitMQ for PCF.
- c. Depending on which version of the BOSH CLI you are using, follow one of the steps below to log in to BOSH as an admin:
  - i. **For BOSH CLI v2:** Use `bosh2 log-in`.
  - ii. **For BOSH CLI v1:** Use `bosh login`.
- d. Depending on which version of the BOSH CLI you are using, follow one of the steps below to display your BOSH deployments to confirm that the output does not show a RabbitMQ for PCF deployment:
  - i. **For BOSH CLI v2:** Use `bosh2 deployments`.
  - ii. **For BOSH CLI v1:** Use `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. Depending on which version of the BOSH CLI you are using, follow one of the steps below:
  - i. **For BOSH CLI v2:** Use `bosh2 delete-deployment BROKER-DEPLOYMENT-NAME` to delete the service broker BOSH deployment.
  - ii. **For BOSH CLI v1:** Use `bosh delete deployment BROKER-DEPLOYMENT-NAME` to delete the service broker BOSH deployment.
- h. Reinstall the tile.

## View Resource Saturation and Scaling

### BOSH CLI v2: Viewing statistics

To view usage statistics for any service do the following:

1. **For BOSH CLI v1 only:** Select the broker deployment by running this command:

```
bosh deployment BOSH_MANIFEST.yml
```

2. Run the following commands depending on your Ops Manager version:


Ops Manager Version	BOSH Commands
v1.10 and earlier	Run the BOSH CLI v1 command <code>bosh vms --vitals</code> . To view process-level information, run <code>bosh instances --ps</code> .
v1.11	Run the BOSH CLI v2 command <code>bosh2 -d DEPLOYMENT-NAME vms --vitals</code> . To view process-level information, run <code>bosh2 -d DEPLOYMENT-NAME instances --ps</code>
v1.12 and later	Run the BOSH CLI v2 command <code>bosh -d DEPLOYMENT-NAME vms --vitals</code> . To view process-level information, run <code>bosh2 -d DEPLOYMENT-NAME instances --ps</code>

## 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:

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

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

## 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 `instances` 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 `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 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 `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?

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 support ticket [here](#). Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To help expedite troubleshooting, also provide your service broker logs, your service instance logs and BOSH task output, if your `cf service YOUR-SERVICE-INSTANCE` output includes a `task-id`.

## 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 `instances` 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 `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 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 `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?

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 support ticket [here](#). Be sure to provide the error message from `cf service YOUR-SERVICE-INSTANCE`.

To help expedite troubleshooting, also provide your service broker logs, your service instance logs and BOSH task output, if your `cf service YOUR-SERVICE-INSTANCE` output includes a `task-id`.



## 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:

```
{
  "p-rabbitmq": [{
    "credentials": {
      "dashboard_url": "http://pivotal-rabbitmq.your.pcf.example.com/#/login/b5d0ad14-4352-48e8-8982-d5b1d257029f/tavk86pnnns1ddiqpsdtbchurn",
      "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
      "password": "tavk86pnnns1ddiqpsdtbchurn",
      "protocols": {
        "amqp": {
          "password": "tavk86pnnns1ddiqpsdtbchurn",
          "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "uris": [
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnnns1ddiqpsdtbchurn@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af",
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnnns1ddiqpsdtbchurn@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af"
          ]
        }
      }
    }
  ]
}
```

You can read more details about the environment variable `VCAP_SERVICES` [here](#).

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

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

$ 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 `$ 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`.

```
$ 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` > `credentials` 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](#).

## Restage an App with a New Service Instance

If a service instance has been updated based on a new version of the service, you need to run `cf restage` to restage your app to use the new instance.

1. Enter `cf restage-app APP`

Where `APP` is the app you want to use the updated service instance.

```
$ cf restage-app my-app
```

When a new version of a service obsoletes an older version, the platform operator may ask you to update your instances of the service and restage any apps bound to the service instances.

Pushing new version of an app automatically restages 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`.

```
$ 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](#).

## 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:

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

For example:

```
{
  ...
  "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
  "uris": [
    "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
    "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7"
  ],
  ...
}
```

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:

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

For example:

```
{
  ...
  "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
  "uris": [
    "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
    "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7"
  ],
  ...
}
```

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](#).

## RabbitMQ® Environment Variables

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

### VCAP\_SERVICES

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


```
{
  "p-rabbitmq": [{
    "label": "p-rabbitmq",
    "name": "my-rabbit-service-instance",
    "plan": "standard",
    "tags": ["rabbitmq", "messaging", "message-queue", "amqp", "pivotal"],
    "credentials": {
      "dashboard_url": "http://pivotal-rabbitmq.your.pcf.example.com/#/login/b5d0ad14-4352-48e8-8982-d5b1d257029f/tavk86pnns1ddiqpsdtbchurn",
      "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
      "vhost": "62e5ab21-7b38-44ac-b139-6aa97af01cd7",
      "password": "#passwordexample123456789",
      "ssl": false,
      "hostname": "10.0.0.41",
      "hostnames": [
        "10.0.0.41",
        "10.0.0.51"],
      "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.41/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
      "uris": [
        "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.41/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
        "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.51/62e5ab21-7b38-44ac-b139-6aa97af01cd7"],
      "http_api_uri": "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.41:15672/api",
      "http_api_uris": [
        "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.41:15672/api",
        "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:tavk86pnns1ddiqpsdtbchurn@10.0.0.51:15672/api"],
      "protocols": {
        "amqp": {
          "password": "passwordexample123456789",
          "port": 5672,
          "ssl": false,
          "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
          "vhost": "62e5ab21-7b38-44ac-b139-6aa97af01cd7",
          "host": "10.0.0.41",
          "hosts": [
            "10.0.0.41",
            "10.0.0.51"],
          "uri": "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
          "uris": [
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7",
            "amqp://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.51:5672/62e5ab21-7b38-44ac-b139-6aa97af01cd7"]],
          "management": {
            "username": "b5d0ad14-4352-48e8-8982-d5b1d257029f",
            "password": "passwordexample123456789",
            "path": "/api",
            "port": 15672,
            "ssl": false,
            "host": "10.0.0.41",
            "hosts": [
              "10.0.0.41",
              "10.0.0.51"],
            "uri": "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:15672/api",
            "uris": [
              "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.41:15672/api",
              "http://b5d0ad14-4352-48e8-8982-d5b1d257029f:passwordexample123456789@10.0.0.51:15672/api"]}]]}]}
```

You can search for your service by its `name`, given when creating the service instance, or dynamically via 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 will be `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 may cause apps bound with RabbitMQ to break.

If you adjust the plugins and protocols enabled for RabbitMQ, you may need to force all app's `VCAP_SERVICES` environment variable to be regenerated. Adding and removing the following plugins require bound applications to be restaged:

- `rabbitmq_management`
- `rabbitmq_stomp`
- `rabbitmq_mqtt`
- `rabbitmq_amqp1_0`

In common with all services in [Pivotal Cloud Foundry](#) (PCF), the `VCAP_SERVICES` environment variable for an application is only modified when the application is bound to a service instance. Users will need to `cf unbind-service`, `cf bind-service` and `cf restage` their app in this scenario.

## 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

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

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

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

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

The `broker-request-guid` maps to the portion of the On-Demand Broker log containing the failed step. Access the broker log through your syslog aggregator, or access BOSH logs for the broker by typing `bosh logs broker 0`. If you have more than one broker instance, repeat this process for each instance.

### Retrieve Service Instance Information

1. Log into the space containing the instance or failed instance.

```
$ cf login
```

2. If you do not know the name of the service instance, run `cf services` to see a listing of all service instances in the space. The service instances are listed in the `name` column.

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

3. Run `cf service SERVICE-INSTANCE-NAME` to retrieve more information about a specific instance.
4. Run `cf service SERVICE-INSTANCE-NAME --guid` to retrieve the GUID of the instance, which is useful for debugging.

## 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 `cf create-service-key INSTANCE-NAME SERVICE-KEY-NAME`.
2. Retrieve the credentials using the command `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 `cf service YOUR-SERVICE-INSTANCE`.

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

## 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:

**WARNING:** This command deletes deployment instances serially. It is very destructive and cannot be undone.

```
bosh run-errand delete-sub-deployments
```