

Table of Contents

Table of Contents	1
Pivotal Cloud Foundry Documentation	6
Getting Started with Pivotal Cloud Foundry	7
Installing Pivotal Cloud Foundry on AWS	8
Deploying the CloudFormation Template for PCF on AWS	9
Launching an Ops Manager Director Instance on AWS	14
Configuring Ops Manager Director for AWS	19
Deploying Elastic Runtime on AWS	26
Deleting an AWS Installation from the Console	41
Manually Configuring AWS for PCF	45
Configuring Elastic Runtime for AWS	60
Configuring Ops Manager Director for AWS	73
Installing Pivotal Cloud Foundry on OpenStack	80
Provisioning the OpenStack Infrastructure	82
Configuring Ops Manager Director after Deploying PCF on OpenStack	89
Installing Elastic Runtime after Deploying PCF on OpenStack	98
Installing Pivotal Cloud Foundry on vSphere and vCloud Air	106
Understanding Availability Zones in VMware Installations	110
Provisioning a Virtual Disk in vSphere	112
Deploying Operations Manager to vSphere	114
Deploying Operations Manager to vCloud Air and vCloud	117
Configuring Ops Manager Director for VMware vSphere	126
Configuring Ops Manager Director for vCloud Air and vCloud	132
Configuring Elastic Runtime for vSphere and vCloud	137
Using Ops Manager Resurrector on VMware vSphere	152
Configuring PCF SSL Termination for vSphere Deployments	154
Identifying Elastic Runtime Jobs Using vCenter	156
Troubleshooting Ops Manager for VMware vSphere	158
Using Operations Manager	161
Prerequisites to Deploying Operations Manager and Elastic Runtime	163
Preparing Your Firewall for Deploying PCF	166
Using Your Own Load Balancer	168
Configuring Network Isolation Options in Ops Manager	172
Using the Cisco Nexus 1000v Switch with Ops Manager	182
Understanding the Ops Manager Interface	184
Adding and Deleting Products	185
Starting and Stopping PCF VMs	187
Creating New Elastic Runtime User Accounts	190
Logging into the Apps Manager	191
Controlling Apps Manager User Activity with Environment Variables	192
Configuring Your App Autoscaling Instance	193
Managing Scheduled Scaling in the App Autoscaling Service	197
Backing Up Pivotal Cloud Foundry	201
Restoring Pivotal Cloud Foundry from Backup	208
Upgrading Operations Manager	215

Upgrading Elastic Runtime and Other PCF Products	219
Monitoring VMs in PCF	222
PCF Troubleshooting Guide	224
Advanced Troubleshooting with the BOSH CLI	231
PCF Security Overview and Policy	237
Cloud Foundry Concepts	239
Cloud Foundry Overview	240
Cloud Foundry Components	241
Diego Architecture	243
Diego Components	245
How Diego Allocates Work	249
How Applications Are Staged	251
Zero Downtime Deployment and Scaling in CF	252
Orgs, Spaces, Roles, and Permissions	256
Understanding Cloud Foundry Security	258
Stacks	263
Glossary	265
Operator's Guide	266
Isolating a PCF Deployment with a Private Network	267
Understanding the Elastic Runtime Network Architecture	270
Load Balancer	270
Router	270
Changing the Quota Plan of an Organization with cf CLI	271
Understanding the Effects of Single Components on a PCF Upgrade	274
Configuring Single Sign-On	276
Configuring LDAP	278
Switching Application Domains	280
Diego-SSH Overview	283
Diego-SSH Proxy	285
Providing a Certificate for your SSL Termination Point	288
Monitoring Instance Usage with the Accounting Report	290
Enabling IPv6 for Hosted Applications	291
Using Diego in Pivotal Cloud Foundry	292
Deploying Diego for Windows	294
Administering and Operating Cloud Foundry	296
Adding Buildpacks to Cloud Foundry	297
Managing Domains and Routes	299
Creating and Managing Users with the cf CLI	300
Creating and Managing Users with the UAA CLI (UAAC)	302
Getting Started with the Notifications Service	306
Application Security Groups	308
Feature Flags	310
Page not Found	311
Page not Found	312
Using the Apps Manager	313
Getting Started with the Apps Manager	314
Understanding Apps Manager Permissions	315

Managing Spaces Using the Apps Manager	316
Managing User Accounts in Spaces Using the Apps Manager	318
Adding Existing LDAP Users to Your PCF Deployment	320
Managing User Permissions Using the Apps Manager	322
Developer Guide	324
Considerations for Designing and Running an Application in the Cloud	325
Deploy an Application	327
Migrating Apps to Diego	331
Deploying a Large Application	333
Creating Domains and Routes	335
Domains and Shared Domains	340
Deploying with Application Manifests	341
Cloud Foundry Environment Variables	350
Accessing Apps with Diego-SSH	357
Using Blue-Green Deployment to Reduce Downtime and Risk	358
Troubleshooting Application Deployment and Health	361
Identifying the API Endpoint for your Elastic Runtime Instance	366
Installing the cf Command Line Interface	367
Getting Started with the cf CLI	368
Using the cf CLI with an HTTP Proxy Server	374
Using cf CLI Plugins	377
About Starting Applications	379
Scaling an Application Using cf scale	380
Services Overview	381
Managing Service Instances with the CLI	383
Managing Service Keys	386
User-Provided Service Instances	388
Configuring Play Framework Service Connections	391
Migrating a Database in Cloud Foundry	392
Using Log Management Services	394
Service-Specific Instructions for Streaming Application Logs	396
Streaming Application Logs to Splunk	401
Buildpacks	403
Buildpack Detection	405
Custom Buildpacks	406
Packaging Dependencies for Offline Buildpacks	409
Binary Buildpack	411
Supported Binary Dependencies	413
Java Buildpack	414
Getting Started Deploying Grails Apps	415
Getting Started Deploying Ratpack Apps	422
Getting Started Deploying Spring Apps	429
Tips for Java Developers	436
Configure Service Connections for Grails	442
Configure Service Connections for Play Framework	445
Configure Service Connections for Spring	446
Cloud Foundry Eclipse Plugin	453

Cloud Foundry Java Client Library	472
Build Tool Integration	474
Node.js Buildpack	478
Tips for Node.js Applications	479
Environment Variables Defined by the Node Buildpack	482
Configure Service Connections for Node.js	483
Ruby Buildpack	485
Getting Started Deploying Ruby Apps	486
Getting Started Deploying Ruby on Rails Apps	491
Deploy a Sample Ruby on Rails Application	493
Configure a Production Server for Ruby Apps	496
Configure Rake Tasks for Deployed Apps	497
Tips for Ruby Developers	498
Environment Variables Defined by the Ruby Buildpack	504
Configure Service Connections for Ruby	505
Services	508
Overview	509
Service Broker API v2.7	510
Managing Service Brokers	525
Access Control	528
Catalog Metadata	533
Binding Credentials	537
Dashboard Single Sign-On	539
Example Service Brokers	543
Application Log Streaming	544
Supporting Multiple Cloud Foundry Instances	545
Creating a Pivotal Cloud Foundry Product Tile	546
Deploying a PCF Product Using BOSH	547
Deploying Your Product as a Tile in Ops Manager	550
Product Template Reference	556
Understanding Lifecycle Errands	565
Decrypting and Encrypting Installation Files	567
Overview of the Loggregator System	569
Using Loggregator	569
Loggregator Components	569
Loggregator Guide for Cloud Foundry Operators	571
Application Logging in Cloud Foundry	574
Cloud Foundry System Metrics	577
Key Metrics for Monitoring a Pivotal Cloud Foundry Deployment	588
Deploying a Nozzle to the Loggregator Firehose	595
Cloud Foundry Data Sources	601
Installing the Loggregator Firehose Plugin for cf CLI	602
Using SSL with a Self-Signed Certificate in Pivotal Ops Metrics	603
Deploying Pivotal Ops Metrics	607
Using Pivotal Ops Metrics	613
Pivotal Cloud Foundry Release Notes and Known Issues	615
Pivotal Elastic Runtime v1.6.0.0 Release Notes	616

Pivotal Cloud Foundry Ops Manager v1.6.0 Release Notes	620
Pivotal Elastic Runtime v1.5 Known Issues	621
Pivotal Cloud Foundry Ops Manager v1.6.0 Known Issues	622

Pivotal Cloud Foundry Documentation

1. [Getting Started](#)
A quick guide to installing and getting started with [Pivotal Cloud Foundry](#) (PCF). If you are new to PCF, start here.
2. [Using Ops Manager](#)
A guide to using the Pivotal Cloud Foundry Operations Manager interface to manage your PCF PaaS.
3. [Elastic Runtime Concepts](#)
An explanation of the components in Pivotal Cloud Foundry Elastic Runtime and how they work.
4. [Operating Elastic Runtime](#)
A guide to running the Elastic Runtime component of PCF.
5. [Administering Elastic Runtime](#)
A guide to managing Elastic Runtime at the administrator level.
6. [Using the Apps Manager](#)
A guide to using the web-based Apps Manager application for managing users, organizations, spaces, and applications.
7. [Deploying Applications](#)
A guide for developers on deploying and troubleshooting applications running in Elastic Runtime (Cloud Foundry).
8. [Buildpacks](#)
A guide to using system buildpacks and extending your Elastic Runtime with custom buildpacks.
9. [Custom Services](#)
A guide to extending your Elastic Runtime with custom services.
10. [Packaging Pivotal Cloud Foundry](#)
A guide to packaging PCF products.
11. [Release Notes](#)
Release notes and known issues for Pivotal Operations Manager, Pivotal Elastic Runtime and Pivotal MySQL Developer.

Getting Started with Pivotal Cloud Foundry

Welcome to Pivotal Cloud Foundry!

The following IaaS-specific guides are intended to walk you through the process of getting your PCF deployment up and running.

If you experience a problem while following the steps below, check the [Known Issues](#), or refer to the [PCF Troubleshooting Guide](#).

Once you have completed the steps in this guide, explore the documentation on [docs.pivotal.io](#) to learn more about [Pivotal Cloud Foundry](#) (PCF) and the Pivotal product suite.

- [Installing Pivotal Cloud Foundry on AWS](#)
- [Installing Pivotal Cloud Foundry on OpenStack](#)
- [Installing Pivotal Cloud Foundry on vSphere and vCloud Air](#)

Installing Pivotal Cloud Foundry on AWS

This guide describes how to install [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS) using the PCF CloudFormation template.

The CloudFormation template for Pivotal Cloud Foundry describes the set of necessary AWS resources and properties. When you create an AWS stack using the PCF template, CloudFormation provisions all the infrastructure that you need to deploy PCF on AWS.

Pivotal strongly recommends using CloudFormation to install PCF on AWS, but see the [Manually Configuring AWS for PCF](#) topic if you cannot use CloudFormation for your installation.

 **Note:** If you are performing an upgrade from PCF 1.5 to 1.6, please review [Upgrading Ops Manager](#) for critical upgrade information.

Prerequisites

You must have the following in order to follow the procedure described here:

- An AWS account that can accommodate the [minimum resource requirements](#) for a PCF installation.
- The appropriate region selected within your AWS account. See the [Amazon documentation on regions and availability zones](#) for help selecting the correct region for your deployment.
- The [AWS CLI](#) installed on your machine, and configured with user credentials that have admin access to your AWS account.
- Sufficiently high instance limits (or no instance limits) on your AWS account. Installing Pivotal Cloud Foundry requires more than the default 20 concurrent instances.
- A key pair to use with your Pivotal Cloud Foundry deployment. [Create a key pair in AWS](#).
- A registered wildcard domain for your PCF installation. You will need this registered domain when configuring your SSL certificate and Cloud Controller.
- An SSL certificate for your PCF domain. This can be a self-signed cert, but Pivotal only recommends using a self-signed cert for testing and development. You should obtain a cert from your Certificate Authority for use in production.

Complete the following procedures to install PCF using CloudFormation:

1. [Deploying the CloudFormation Template for PCF on AWS](#)
2. [Launching an Ops Manager Director Instance on AWS](#)
3. [Configuring Ops Manager Director on AWS](#)
4. [Deploying Elastic Runtime on AWS](#)

Deploying the CloudFormation Template for PCF on AWS

This topic describes how to deploy the CloudFormation template for [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS).

An AWS CloudFormation template describes a set of AWS resources and properties. Follow the instructions below to use a CloudFormation template to create the infrastructure that you need to deploy PCF to AWS.

Note: Before following the procedure below, confirm that you have selected the correct region within your AWS account. All of the AWS resources for your deployment must exist within a single region. See the [Amazon documentation on regions and availability zones](#) for help selecting the correct region for your deployment.

Step 1: Download the PCF CloudFormation Template

1. Sign in to [Pivotal Network](#).
2. Select **Elastic Runtime**. From the **Releases** drop-down menu, select the release that you wish to install.
3. Download the **PCF Elastic Runtime and Ops Manager CloudFormation Template for AWS**.
4. Save the file as `pcf.json`.

Step 2: Upload an SSL Certificate to AWS

The [AWS CLI](#) must be installed on your machine and configured to a user account with admin access privileges on your AWS account.

1. Obtain or create an SSL server certificate. See the [AWS docs on SSL certificates](#).

Note: When you create a certificate signing request (CSR) in the “Create a Server Certificate” instructions, you must use your system wildcard domain (example: *.my-pcf-apps-domain.com) as the Common Name input.

2. Upload your SSL certificate to AWS. For more information, see the [AWS docs on uploading SSL certs via the CLI](#).

```
$ aws iam upload-server-certificate \
--server-certificate-name YOUR-CERTIFICATE \
--certificate-body file://YOUR-PUBLIC-KEY-CERT-FILE.pem \
--private-key file://YOUR-PRIVATE-KEY-FILE.pem \
```

For example:

```
$ aws iam upload-server-certificate \
--server-certificate-name myServerCertificate \
--certificate-body file:///my-certificate.pem \
--private-key file:///my-private-key.pem
```

Note: If you receive an upload error (MalformedCertificate), run the following command to convert your server certificate to the PEM format as required by the AWS Identity and Management (IAM) service: `$ openssl x509 -inform PEM -in my-certificate.pem`
Then try your upload again.

3. After successfully uploading the certificate to your AWS account, you will see output metadata for your certificate. For example:

```
{
"ServerCertificateMetadata": {
"ServerCertificateId": "ASCAI3HRYUTD55KNAF64",
"ServerCertificateName": "myServerCertificate",
"Expiration": "2016-10-18T18:41:59Z",
"Path": "/"}
```

```
"Arn": "arn:aws:iam::9240874958318:server-certificate/myServerCertificate",
"UploadDate": "2015-10-19T19:10:57.404Z"
}
```

4. Record the value of the `Arn` key to use when configuring your AWS resource stack. Alternatively, if you know the name of the certificate, you can run the following command to retrieve certificate metadata at a later point:

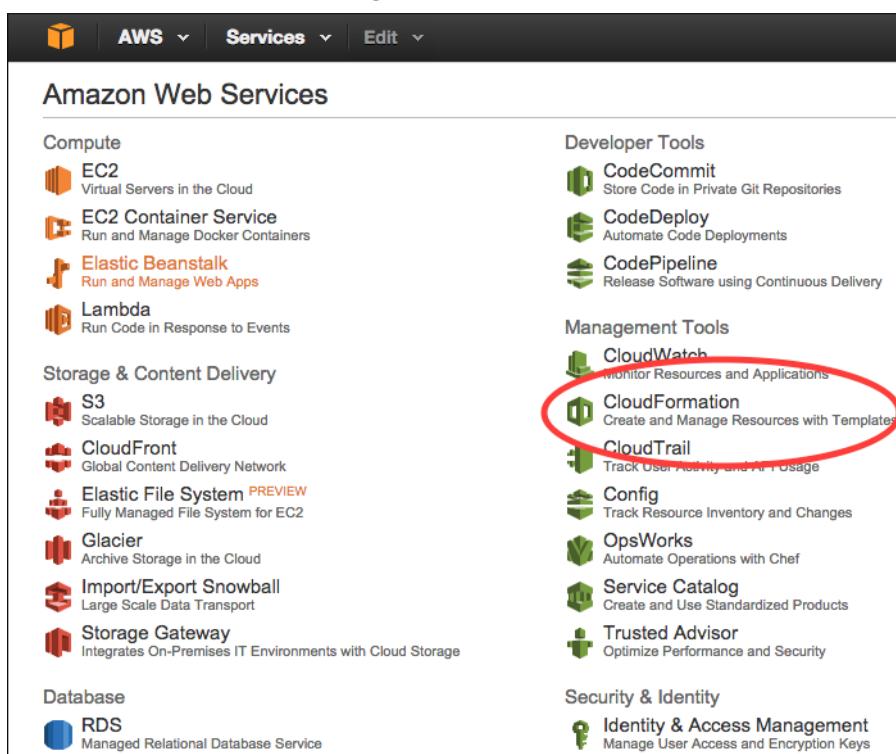
```
$ aws iam get-server-certificate --server-certificate-name YOUR-CERT-NAME
```

For example:

```
$ aws iam get-server-certificate --server-certificate-name myServerCertificate
```

Step 3: Create a Resource Stack Using the CloudFormation Template

1. Log in to the [AWS Console](#).
2. In the second column, under **Management Tools**, click **CloudFormation**.



3. Click **Create New Stack**.

Create a Stack

AWS Cloudformation allows you to quickly and easily deploy your infrastructure resources and applications on AWS. You can use one of the templates we provide to get started quickly with applications like WordPress or Drupal, one of the many sample templates or create your own template.

You do not currently have any stacks. Click the "Create New Stack" button below to create a new AWS Cloudformation Stack.

Create New Stack

4. Select **Upload a template to Amazon S3**.

Choose a template A template is a JSON-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

- Select a sample template
- Upload a template to Amazon S3
- Specify an Amazon S3 template URL

[Choose File](#) pcf.json

5. Click **Browse**. Browse to and select the `pcf.json`, the **Pivotal Cloud Foundry Elastic Runtime and Ops Manager CloudFormation Template for AWS** file that you downloaded. Click **Next**.

6. On the next screen, name the stack `pcf-stack`.

7. In the **Specify Parameters** page, complete the following fields:

Parameters	
01NATKeyPair	<input type="text" value="pcf_rc_install"/> Select the SSH keypair to use for the NAT vm
02NATInstanceType	<input type="text" value="t2.medium"/> Select the Instance Type to use for the NAT vm
03OpsManagerIngress	<input type="text" value="0.0.0.0/0"/> CIDR range allowed to connect to Ops Manager instance
04RdsDBName	<input type="text" value="bosh"/> BOSH database name
05RdsUsername	<input type="text" value="docs"/> BOSH database username
06RdsPassword	<input type="password" value="....."/> BOSH database password
07SSLCertificateARN	<input type="text" value="958318:server-certificate/myServerCertificate"/> ARN for pre-uploaded SSL certificate
08OpsManagerTemplate	<input type="text" value="https://s3.amazonaws.com/pcf-cloudformation/opsmanager-cf-template.yaml"/> S3 Location for OpsManager CloudFormation Template
09ElbPrefix	<input type="text"/>
Prefix for the name of the ELBs generated. NOTE: Leave empty to use default prefix of <code>AWS::StackName</code>	

- **01NATKeyPair:** Use the drop-down menu to select the name of your pre-existing AWS key pair. If you do not have a pre-existing key pair, [create one in AWS](#) and return to this step.
- **02NATInstanceType:** Do not change this value.
- **03OpsManagerIngress:** Do not change this value.
- **04RdsDBName:** Do not change this value.
- **05RdsUserName:** Enter a username for the RDS database.

Note: Do not enter the username `rdsadmin`. AWS reserves the `rdsadmin` user account for internal database instance management.

- **06RdsPassword:** Enter a password for the RDS database.
- **07SSLCertificateARN:** Enter your uploaded SSL Certificate ARN.
- **08OpsManagerTemplate:** The default template link provided here works. Otherwise you can enter your own S3 bucket location of the Ops Manager CloudFormation script.
- **09ElbPrefix:** Prefix for the generated names of the ELBs. Any string you specify in this field will be prefixed to `-pcf-elb` to form the name of your ELBs. Leave empty to use the default prefix of `AWS::StackName`.

8. Click **Next**.

9. On the **Options** page, leave the fields blank and click **Next**.

Options

Tags

You can specify tags (key-value pairs) for resources in your stack.
You can add up to 10 unique key-value pairs for each stack.

	Key (127 characters maximum)	Value (255 characters maximum)	
1	<input type="text"/>	<input type="text"/>	+

Advanced

You can set additional options for your stack, like notification options and a stack policy.

[Cancel](#) [Previous](#) [Next](#)

- On the **Review** page, select the **I acknowledge that this template might cause AWS CloudFormation to create IAM resources** checkbox and click **Create**.

i The following resource(s) require capabilities: [AWS::IAM::User, AWS::IAM::Policy, AWS::IAM::AccessKey]

This template might include Identity and Access Management (IAM) resources, which can include groups, IAM users, and IAM roles with certain permissions. Ensure that the template you are using is from a trusted source. [Learn more](#).

I acknowledge that this template might cause AWS CloudFormation to create IAM resources.

[Cancel](#) [Previous](#) [Create](#)

AWS runs the CloudFormation script and creates the infrastructure that you need to deploy PCF to AWS. This may take a few moments. You can click on the **Events** tab to view the progress of the setup.

When the installation process successfully completes, AWS displays **CREATE_COMPLETE** as the status of the stack.

Create Stack Actions ▾ Design template

Filter: Active ▾ By Name:

Showing 2 stacks

Stack Name	Created Time	Status	Description
<input type="checkbox"/> pcf-stack-OpsManStack-1FUM	2015-10-19 12:29:26 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS
<input checked="" type="checkbox"/> pcf-stack	2015-10-19 12:29:10 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS

Overview Outputs Resources Events Template Parameters Tags Stack Policy

Key	Value	Description
PcfElbDnsName	pcf-stack-pcf-elb-1271201696.us-west-2.elb.amazonaws.com	
PcfElbSshDnsName	pcf-stack-pcf-ssh-elb-874966686.us-west-2.elb.amazonaws.com	
PcfVpc	vpc-bafbadf	
PcfIamUserName	pcf-stack-OpsManStack-1FUMOC6ILTY7U-PcfIamUser-156K WIAF33IZ9	
PcfIamUserAccessKey	[REDACTED]	
PcfIamUserSecretAccessKey	[REDACTED]	
PcfOpsManagerS3Bucket	pcf-stack-opsmanstack-1fumo-pcfopsmanagers3bucket-z163 5iz8hquw	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-pcfelasticruntimes3buildpacksbucket-178ge6dg6n5 d9	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-pcfelasticruntimes3dropletsbucket-bqsohw18lmrx	
PcfElasticRuntimeS3PackagesBucket	pcf-stack-pcfelasticruntimes3packagesbucket-1bfsttjbu6itk	
PcfElasticRuntimeS3ResourcesBucket	pcf-stack-pcfelasticruntimes3resourcesbucket-1gddu6k64txac	
PcfVmsSecurityGroupId	sg-48f0992c	
PcfOpsManagerSecurityGroupId	sg-4ef0992a	

After completing this procedure, complete all of the steps in the following topics:

- [Launching an Ops Manager Director Instance on AWS](#)
- [Configuring Ops Manager Director for AWS](#)
- [Deploying Elastic Runtime on AWS](#)

Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation.](#)

Launching an Ops Manager Director Instance on AWS

This topic describes how to deploy Ops Manager Director after deploying the CloudFormation template for [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS).

Before beginning this procedure, ensure that you have successfully completed all of the steps in the [Deploying the CloudFormation Template for PCF on AWS](#) topic. After you complete this procedure, follow the instructions in the [Configuring Ops Manager Director on AWS CloudFormation](#) and [Configuring Elastic Runtime on AWS CloudFormation](#) topics.

Step 1: Open the Outputs Tab in AWS Stacks

1. In the EC2 dashboard of your [AWS Console](#), click on **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.

The screenshot shows the AWS CloudFormation Stacks Dashboard. At the top, there are tabs for 'Create Stack', 'Actions', and 'Design template'. Below that is a search bar with 'Filter: Active' and 'By Name:' dropdowns, and a status indicator 'Showing 2 stacks'. A table lists two stacks:

Stack Name	Created Time	Status	Description
<input type="checkbox"/> pcf-stack-OpsManStack-1FUM	2015-10-19 12:29:26 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS
<input checked="" type="checkbox"/> pcf-stack	2015-10-19 12:29:10 UTC-0700	CREATE_COMPLETE	Cloudformation template for configuring Pivotal Cloud Foundry on AWS

Below the stacks, the tabs 'Overview', 'Outputs' (which is selected), 'Resources', 'Events', 'Template', 'Parameters', 'Tags', and 'Stack Policy' are visible. The 'Outputs' tab displays a table of key-value pairs:

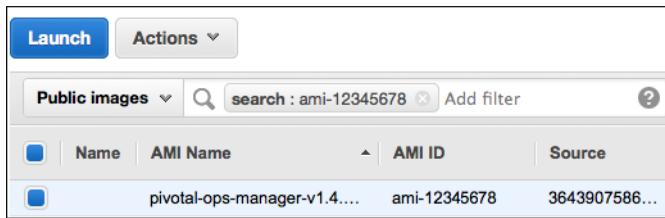
Key	Value	Description
PcfElbDnsName	pcf-stack-pcf-elb-1271201696.us-west-2.elb.amazonaws.com	
PcfElbSshDnsName	pcf-stack-pcf-ssh-elb-874966686.us-west-2.elb.amazonaws.com	
PcfVpc	vpc-bafbadff	
PcfIamUserName	pcf-stack-OpsManStack-1FUMOC6ILTY7U-PcfIamUser-156KWIAF33IZ9	
PcfIamUserAccessKey	[REDACTED]	
PcfIamUserSecretAccessKey	[REDACTED]	
PcfOpsManagerS3Bucket	pcf-stack-opsmanstack-1fumo-pcfopsmanagers3bucket-z1635lz8hquw	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-pcfelasticruntimes3buildpacksbucket-178ge6dg6n5d9	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-pcfelasticruntimes3dropletsbucket-bqsohw18lmrx	
PcfElasticRuntimeS3PackagesBucket	pcf-stack-pcfelasticruntimes3packagesbucket-1bfsttjbu6itk	
PcfElasticRuntimeS3ResourcesBucket	pcf-stack-pcfelasticruntimes3resourcesbucket-1gddu6k64txac	
PcfVmsSecurityGroupId	sg-48f0992c	
PcfOpsManagerSecurityGroupId	sg-4ef0992a	

In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

Step 2: Select a Pivotal Ops Manager AMI Instance

1. Log into the [Pivotal Network](#) and click on **Ops Manager**.

2. From the **Releases** dropdown, select the release you wish to install.
3. Select **Pivotal Cloud Foundry Ops Manager for AWS** to download the [OpsManagerx.x.xonAWSFulfillmentInstructions.pdf](#) file. This document lists AMI IDs for Pivotal Ops Manager for specific regions.
4. Log in to the [AWS Console](#). Navigate to the EC2 Dashboard.
5. In the left navigation panel, click **AMIs**.
6. Using the [OpsManagerx.x.xonAWSFulfillmentInstructions.pdf](#) document, enter the AMI ID for your AWS region in the Public images search field. This search locates the appropriate Pivotal Ops Manager AMI for your region within public images.



7. Select this AMI and click **Launch**.
8. Choose **m3.large** for your instance type.

Filter by: All instance types ▾ All generations ▾ Show/Hide Columns					
Currently selected: m3.large (6.5 ECUs, 2 vCPUs, 2.5 GHz, Intel Xeon E5-2670v2, 7.5 GiB memory, 1 x 32 GB instance storage)					
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
<input type="checkbox"/>	Micro instances	t1.micro <small>Free tier eligible</small>	1	0.613	EBS only
<input type="checkbox"/>	General purpose	m3.medium	1	3.75	1 x 4 (SSD)
<input checked="" type="checkbox"/>	General purpose	m3.large	2	7.5	1 x 32 (SSD)
<input type="checkbox"/>	General purpose	m3.xlarge	4	15	2 x 40 (SSD)

9. Click **Next: Configure Instance Details**.

Step 3: Configure Instance Details

1. Complete the **Config Instance Details** page with information from the [Outputs tab](#) in the AWS Stacks Dashboard:

Number of instances

Purchasing option Request Spot Instances

Network

Subnet
249 IP Addresses available

Auto-assign Public IP

IAM role

Shutdown behavior

Enable termination protection Protect against accidental termination

Monitoring Enable CloudWatch detailed monitoring
Additional charges apply.

EBS-optimized instance Launch as EBS-optimized instance
Additional charges apply.

Tenancy

Additional charges will apply for dedicated tenancy.

▼ Network interfaces

Device	Network Interface	Subnet	Primary IP	Secondary IP addresses
eth0	New network interf	subnet-[REDACTED]	Auto-assign	Add IP

► Advanced Details

- Select the **Network** that matches the value of **PcfVpc**.
 - Select the **Subnet** that matches the value of **PcfPublicSubnetId**.
- Set **Auto-assign Public IP** to **Enable**.
 - Click **Next: Add Storage**.
 - On the **Add Storage** page, adjust the **Size (GiB)** value. Pivotal recommends increasing this value to a minimum of 100 GB.
 - Click **Next: Tag Instance**.

Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS
Root	/dev/sda1	snap-[REDACTED]	100	General Purpose (SSD)	300 / 3000
Instance Store 0	/dev/sdb	N/A	N/A	N/A	N/A

- On the **Tag Instance** page, add a **Key** with **Value** .
- Click **Next: Configure Security Group**.

Key (127 characters maximum)	Value (255 characters maximum)
<input type="text" value="Name"/>	<input type="text" value="Ops Manager"/>

(Up to 10 tags maximum)

Step 4: Configure Security Group

- Select the **Select an existing security group** option.

2. Select the **Security Group ID** that matches the value of **PcfOpsManagerSecurityGroupId** located in the [Outputs](#) tab of the Stacks dashboard.

The screenshot shows the 'Assign a security group' step of a CloudFormation stack creation. At the top, there are two radio button options: 'Create a new security group' and 'Select an existing security group'. The second option is selected, and a dropdown menu labeled 'VPC security groups' is open, showing a list of existing security groups:

Security Group ID	Name	Description	Actions
sg-26716d42	default	default VPC security group	Copy to new
sg-3a716d5e	pcf-stack-PcfMysqlSecurityGroup-1K7ZNDGY2FU0H	PCF MySQL Security Group	Copy to new
sg-39716d5d	pcf-stack-PcfNatSecurityGroup-10T80HF3NLTD0	NAT Security Group	Copy to new
sg-04716d60	pcf-stack-PcfOpsManagerSecurityGroup-OEGO	Ops Manager Security Group	Copy to new
sg-06716d62	pcf-stack-PcfVmSecurityGroup-GYPF6VACFYMD	PCF VMs Security Group	Copy to new

Below the list, a section titled 'Inbound rules for sg-04716d60 (Selected security groups: sg-04716d60)' displays the rules for the selected security group:

Type	Protocol	Port Range	Source
SSH	TCP	22	0.0.0.0/0
SSH	TCP	22	10.0.0.0/16
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	6868	10.0.0.0/16
Custom TCP Rule	TCP	25555	10.0.0.0/16
HTTPS	TCP	443	0.0.0.0/0

At the bottom right of the screen, there are three buttons: 'Cancel', 'Previous', and a blue 'Review and Launch' button.

1. Click **Review and Launch**.

Step 5: Deploy Ops Manager

1. Review the instance launch details. Click **Launch**.

The screenshot shows the 'Step 7: Review Instance Launch' step of the CloudFormation stack creation. The page is divided into several sections:

- AMI Details:** Shows the selected AMI: 'pivotal-ops-manager-v1.4.2.0-RC1 - ami-a03126c8'. It also shows the root device type as 'ebs' and the virtualization type as 'paravirtual'.
- Instance Type:** Shows the selected instance type as 'm3.large' with 6.5 ECUs, 2 vCPUs, 7.5 GiB of memory, 1 x 32 GB of instance storage, EBS-Optimized available, and moderate network performance.
- Security Groups:** Shows the selected security group 'sg-04716d60' with the name 'pcf-stack-PcfOpsManagerSecurityGroup-OEGO' and description 'Ops Manager Security Group'. It also lists all selected security groups' inbound rules.
- Storage:** Shows the storage configuration.
- Tags:** Shows the tags configuration.

At the bottom right of the screen, there are three buttons: 'Cancel', 'Previous', and a blue 'Launch' button.

2. Use the first drop-down menu to select **Choose an existing key pair**. Use the second drop-down menu to select the name of your pre-existing AWS key pair.
3. Select the acknowledgement checkbox.
4. Click **Launch Instances**. If successful, you will see the [Launch Status Page](#).

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

pcf

I acknowledge that I have access to the selected private key file (pcf.pem), and that without this file, I won't be able to log into my instance.

5. Click **View Instances**. Or alternately, navigate to **Instances** from the left navigation panel of the EC2 Dashboard.
6. AWS deploys Ops Manager. This may take a few minutes. When complete, AWS displays an **Instance State** of `running` and a **Status Check** of `passed` when the Ops Manager deployment successfully completes.

<input type="button" value="Launch Instance"/>	<input type="button" value="Connect"/>	<input type="button" value="Actions ▾"/>
<input type="text"/> Filter by tags and attributes or search by keyword		
Instance ID	Instance Type	Instance State
i-00b9b5f	m3.xlarge	running

Record the **Public IP** of your Ops

Manager EC2 instance. You use this IP address when [Configuring Ops Manager Director on AWS CloudFormation](#).

Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Configuring Ops Manager Director for AWS

This topic describes how to configure the Ops Manager Director after deploying the CloudFormation template for [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS). Use this topic when [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Deploying the CloudFormation Template for PCF on AWS](#) and the [Launching an Ops Manager Director Instance on AWS CloudFormation](#) topics. After you complete this procedure, follow the instructions in the [Deploying Elastic Runtime on AWS CloudFormation](#) topic.

Step 1: Open the Outputs Tab in AWS Stacks

1. In the EC2 dashboard of your [AWS Console](#), click on **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.

Key	Value	Description
PcfElbDnsName	pcf-stack-pcf-elb-1271201696.us-west-2.elb.amazonaws.com	
PcfElbSshDnsName	pcf-stack-pcf-ssh-elb-874966686.us-west-2.elb.amazonaws.com	
PcfVpc	vpc-bafbadff	
PcfIamUserName	pcf-stack-OpsManStack-1FUMOC6ILTY7U-PcfIamUser-156K WIAF33IZ9	
PcfIamUserAccessKey	[REDACTED]	
PcfIamUserSecretAccessKey	[REDACTED]	
PcfOpsManagerS3Bucket	pcf-stack-opsmanstack-1fumo-pcfopsmanagers3bucket-z163 51z8hquw	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-pcfelasticruntimes3buildpacksbucket-178ge6dg6n5 d9	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-pcfelasticruntimes3dropletsbucket-bqsohw18lmrx	
PcfElasticRuntimeS3PackagesBucket	pcf-stack-pcfelasticruntimes3packagesbucket-1bsttjbu6itk	
PcfElasticRuntimeS3ResourcesBucket	pcf-stack-pcfelasticruntimes3resourcesbucket-1gddu6k64txac	
PcfVmsSecurityGroupId	sg-48f0992c	
PcfOpsManagerSecurityGroupId	sg-4ef0992a	

In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

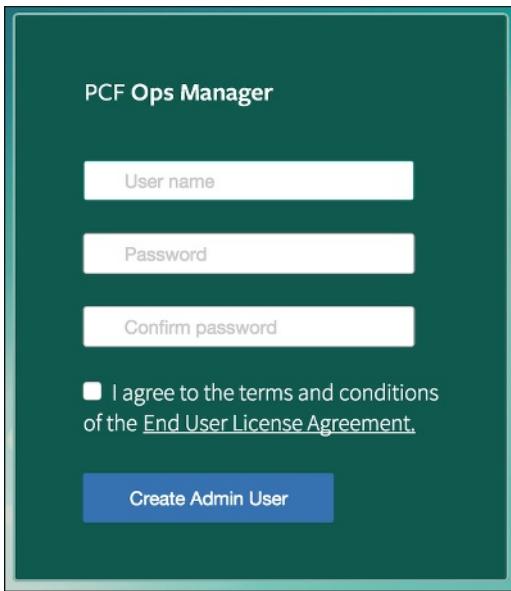
Step 2: Access Ops Manager

1. In the AWS Console, navigate to the EC2 Dashboard.

- Record the **Public IP** of your Ops Manager EC2 instance.

Launch Instance		Connect	Actions
<input type="text"/> Filter by tags and attributes or search by keyword			
Instance ID	Instance Type	Instance State	Status Checks
i-0000000f	m3.xlarge	running	2/2 checks passed
			ec2-52-173-3-213.amazonaws.com 52.173.3.213 pdf

- Open this IP address in a web browser. A login page appears.



The image shows a dark-themed login interface for PCF Ops Manager. It features three input fields: 'User name', 'Password', and 'Confirm password'. Below these is a checkbox labeled 'I agree to the terms and conditions of the [End User License Agreement](#)'. At the bottom is a blue button labeled 'Create Admin User'.

- Enter a **Username** and a **Password**.
- Read the **End User License Agreement**, and select the checkbox to accept.
- Click **Create Admin User** to create an Ops Manager administrator account.

Step 3: AWS Config Page

- Click the **Ops Manager Director** tile.



- Select **AWS Config** to open the **AWS Management Console Config** page.
- Complete the **AWS Management Console Config** page with information from the **Outputs** tab for your stack in the AWS Console:

Settings Status Credentials

- AWS Config
- Director Config
- Create Availability Zones
- Assign Availability Zones
- Create Networks
- Assign Networks
- Security
- Resource Config
- Experimental Features

AWS Management Console Config

Access Key ID*

AWS Secret Key*

VPC ID*

Security Group Name*

Key Pair Name*

SSH Private Key*

```
-----BEGIN RSA PRIVATE KEY-----
4rRd6H9V
tzg/ECxglWmtXtMQbSu/Tdf8y4FbmEsqt1MtYG4A2jgbZbfq4zMCFaJ4oDdp01QS0C+pNiY
MdjnN
Zp1rBnYisx2ZKhStLK69hAf+WTZcdZuCTTWLbbrVartvYUlthspCWJskf66M12+lOMP
-----END RSA PRIVATE KEY-----
```

Region*

Encrypt EBS Volumes

Save

- **Access Key ID:** Use the value of **PcfIamUserAccessKey**.
- **AWS Secret Key:** Use the value of **PcfIamUserSecretAccessKey**.
- **VPC ID:** Use the value of **PcfVpc**.
- **Security Group Name:** Open the AWS EC2 Dashboard and click **Security Groups**. Select the security group with the **Description** `PCF VMs Security Group`. Copy the **Group Name** into the Ops Manager **Security Group Name** field.

Note: You must use the `Group Name`, not the `Group ID`.

- **Key Pair Name:** Use the value of **PcfKeyPairName**.
- **SSH Private Key:** Open your pre-existing AWS key pair `.pem` file in a text editor. Copy the contents of the `.pem` file and paste into the **SSH Private Key** field.
- **Region:** Select the region where you deployed Ops Manager.
- **Encrypt EBS Volumes:**

4. Click **Save**.

Step 4: Director Config Page

1. Select **Director Config** to open the **Director Config** page.

AWS Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

VM Passwords

Resource Config

Director Config

NTP Servers (comma delimited)*
0.amazon.pool.ntp.org,1.amazon.pool.ntp

Metrics IP Address

Enable VM Resurrector Plugin

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*
 https://s3-us-west-1.amazonaws.com

Bucket Name*
 pcf-stack-pcfopsmanagers3bucket-1234adc

Access Key*
 ABCDEFGHIJKLMNOPQRSTUVWXYZ

Secret Key*

Database Location
 Internal
 External MySQL Database

Host*
 abcd1234.us-east-1.rds.amazonaws.com

Port*
 3306

Username*
 admin

Password*

Database*
 bosh

Max Threads

Save

- Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a

comma:

- 0.amazon.pool.ntp.org
- 1.amazon.pool.ntp.org
- 2.amazon.pool.ntp.org
- 3.amazon.pool.ntp.org

3. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.
4. For **Blobstore Location**, select the **S3 Compatible Blobstore** option.
5. For **S3 Endpoint**:
 - a. In a browser, reference the [Amazon Simple Storage Service \(Amazon S3\) table](#), and find the region for your AWS account.
 - b. Prepend `https://` to the **Endpoint** for your region, and copy it into the Ops Manager **S3 Endpoint** field. For example, in the **us-west-2** region, enter `https://s3-us-west-2.amazonaws.com` into the field.
6. Complete the **Director Config** page with information from the **Outputs** tab in the AWS Console:
 - **Bucket Name**: Use the value of **PcfOpsManagerS3Bucket**.
 - **Access Key ID**: Use the value of **PcfIamUserAccessKey**.
 - **AWS Secret Key**: Use the value of **PcfIamUserSecretAccessKey**.
 - **Database Location**: Select the **External MySQL Database** option.
 - **Host**: Use the value of **PcfRdsAddress**.
 - **Port**: Use the value of **PcfRdsPort**.
 - **Username**: Use the value of **PcfRdsUsername**.
 - **Password**: Use the value of **PcfRdsPassword**.
 - **Database**: Use the value of **PcfRdsDBName**.
 - **Max Threads**:
7. Click **Save**.

Step 5: Availability Zones Pages

1. Select **Create Availability Zones**.

Create Availability Zones

Amazon Availability Zone*

us-west-2a

The Amazon Availability Zone name (ex: 'us-east-1b')

Save

2. Enter the value of **PcfPrivateSubnetAvailabilityZone** from the **Outputs** tab in the AWS Console. Click **Save**.
3. Select **Assign Availability Zones**.
4. Select the Availability Zone that you created in **Create Availability Zones**.

AWS Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Assign Availability Zones

The Ops Manager Director is a single instance. Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.

Singleton Availability Zone

us-east-1a

Save

5. Click **Save**.

Step 6: Network Pages

1. Select **Create Networks**.

AWS Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Security

Resource Config

Experimental Features

Create Network

Warning: Pivotal recommends choosing the IPs you need correctly initially because if you change this value later on to reserve additional IPs, you may need to contact Pivotal Support to complete the procedure.

Name*

docsNetwork

A unique name for this network

VPC Subnet ID*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

Save

2. In the **Name** field, enter a name for your PCF Network.

3. In the **VPC Subnet ID** field, use the value of **PcfPrivateSubnetId** from the **Outputs** tab in the AWS Console.

4. Complete the **Create Networks** page with the following information:

- **Subnet (CIDR Range)**: Enter
- **Excluded IP Ranges**: Enter
- **DNS**: Enter
- **Gateway**: Enter

5. Click **Save**.

6. If the following ICMP error message appears, you can ignore the warning. Dismiss the warning, and move on to the next step.



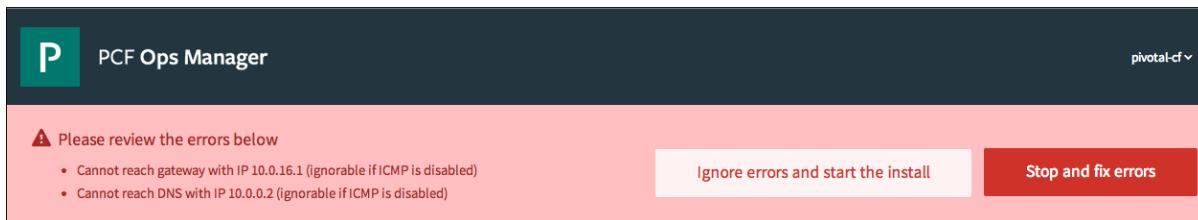
- Select **Assign Networks** and assign the Director to the network that you created in the **Create Networks** step.

A screenshot of the "Assign Networks" configuration page. On the left, there is a sidebar with several checked options: AWS Config, Director Config, Create Availability Zones, Assign Availability Zones, Create Networks, and Resource Config. The main area is titled "Assign Networks" and contains the instruction: "The Ops Manager director can be configured to have an IP on a network." Below this is a "Network" dropdown menu set to "PCFNetwork", and a blue "Save" button.

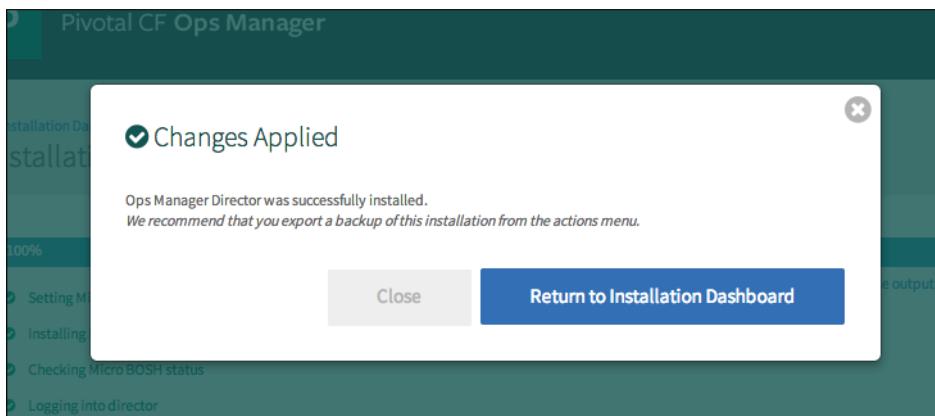
- Click **Save**.

Step 7: Complete the Ops Manager Director Installation

- Click the **Installation Dashboard** link to return to the Installation Dashboard.
- Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



- Ops Manager Director installs. This may take a few moments. The image shows the **Changes Applied** window that displays when the installation process successfully completes.



Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Deploying Elastic Runtime on AWS

This topic describes how to install and configure Elastic Runtime after deploying the CloudFormation template for [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS). Use this topic when [installing Pivotal Cloud Foundry on AWS](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Deploying the CloudFormation Template for PCF on AWS](#) and [Configuring Ops Manager Director after Deploying PCF on AWS using CloudFormation](#) topics.

Step 1: Open the Outputs Tab in AWS

1. In the EC2 dashboard of your [AWS Console](#), click on **CloudFormation**. The Stacks Dashboard displays.
2. Select the **pcf-stack** checkbox, then select the **Outputs** tab.

Key	Value	Description
PcfElbDnsName	pcf-stack-pcf-elb-1271201696.us-west-2.elb.amazonaws.com	
PcfElbSshDnsName	pcf-stack-pcf-ssh-elb-874966686.us-west-2.elb.amazonaws.com	
PcfVpc	vpc-bafbadff	
PcfIamUserName	pcf-stack-OpsManStack-1FUMOC6ILTY7U-PcfIamUser-156KWIAF33IZ9	
PcfIamUserAccessKey	[REDACTED]	
PcfIamUserSecretAccessKey	[REDACTED]	
PcfOpsManagerS3Bucket	pcf-stack-opsmanstack-1fumo-pcfopsmanagers3bucket-z1635lz8hquw	
PcfElasticRuntimeS3BuildpacksBucket	pcf-stack-pcfelasticruntimes3buildpacksbucket-178ge6dg6n5d9	
PcfElasticRuntimeS3DropletsBucket	pcf-stack-pcfelasticruntimes3dropletsbucket-bqsohw18lmrx	
PcfElasticRuntimeS3PackagesBucket	pcf-stack-pcfelasticruntimes3packagesbucket-1bfsttjbu6itk	
PcfElasticRuntimeS3ResourcesBucket	pcf-stack-pcfelasticruntimes3resourcesbucket-1gddu6k64txac	
PcfVmsSecurityGroupId	sg-48f0992c	
PcfOpsManagerSecurityGroupId	sg-4ef0992a	

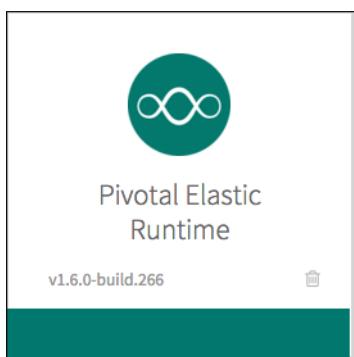
In the steps described below, use the information from the **Value** column of the **Outputs** tab to configure your PCF installation.

Step 2: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. If you have not downloaded Elastic Runtime, click the Pivotal Network link on the left to download the Elastic

Runtime .pivotal file. Click **Import a Product** to add the tile to Ops Manager. For more information, refer to the [Adding and Deleting Products](#) topic.

3. Click the **Elastic Runtime** tile in the Installation Dashboard.



Step 3: Assign Availability Zones

1. Select **Assign Availability Zones**.
2. Select the option button and checkbox to define the Availability Zone for singleton jobs and job balancing.

Assign Networks

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

Availability Zone Assignments

Place singleton jobs in us-west-2a

Balance other jobs in us-west-2a

Save

3. Click **Save**.

Step 4: Create System Databases

You must create the databases required by Elastic Runtime on the RDS instance provisioned by the CloudFormation script.

1. Add the AWS-provided key pair to your SSH profile so that you can access the Ops Manager VM:

```
ssh-add aws-keypair.pem
```

2. SSH into your Ops Manager VM using its IP address and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_IP
```

Use the **Public IP** address that you noted when [configuring Ops Manager](#), or find the IP address for the Ops Manager VM on the AWS EC2 console.

3. Run the following terminal command to log in to your RDS instance through the MySQL client, using values from your AWS dashboard **Outputs** tab to fill in the following output keys:

```
mysql --host=PcfRdsAddress --user=PcfRdsUsername --password=PcfRdsPassword
```

For example:

```
mysql --host=pp19dd336auydlw.cpdgtp8njpud.us-west-2.rds.amazonaws.com --user=docs --password=jks563!fjlk
```

- Run the following MySQL commands to create databases for each of the six Elastic Runtime components that require a database:

```
CREATE database uaa;
CREATE database ccdb;
CREATE database console;
CREATE database notifications;
CREATE database autoscale;
CREATE database app_usage_service;
```

- Type `exit` to quit the MySQL client and `exit` again to close your connection to the Ops Manager VM.

Step 5: Configure System Databases

- Select **System Database Config**.
- Select the **External Databases** option.

The screenshot shows the 'System Database Config' page with the 'External Databases' option selected. The configuration fields are as follows:

- Hostname DNS Name:** pp19dd336auydlw.cpdgtp8njpud.us-west-2.rds.amazonaws.com
- TCP Port:** 3306
- Username:** docs
- Password:** (redacted)

A blue 'Save' button is at the bottom of the form.

- Complete the **System Database Config** page with information from the corresponding **Outputs** tab in the AWS Console, according to the following table:

Ops Manager Field	Outputs Key
Hostname DNS Name	PcfRdsAddress
TCP Port	PcfRdsPort
Username	PcfRdsUsername
Password	PcfRdsPassword

4. Click **Save**.

5. (**Optional**) Select **MySQL Proxy Config**. Enter one or more IP addresses for the MySQL proxy instances configured on your external load balancer.

Use of a proxy permits transparent failover to other nodes within the cluster in the event of a node failure. When a node becomes unhealthy, the proxy re-routes all subsequent connections to a healthy node and all existing connections to the unhealthy node are closed. See [MySQL for PCF: Proxy](#) for more information.

The screenshot shows the Pivotal Elastic Runtime interface. At the top, there are tabs: Settings (which is selected), Status, Credentials, and Logs. On the left, a sidebar lists configuration sections: Assign Networks, Assign Availability Zones, System Database Config, File Storage Config, IPs and Ports, Security Config, MySQL Proxy Config (which is highlighted with a grey arrow pointing to it), and Cloud Controller. The main content area has a heading 'A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.' Below this, there are two input fields: 'MySQL Proxy IPs' and 'MySQL Service Hostname'. A placeholder text 'Enter the IP address(es) for the MySQL proxy instances configured on your external load balancer.' is shown next to the 'MySQL Proxy IPs' field. At the bottom right is a blue 'Save' button.

Step 6: Configure File Storage

1. In the Elastic Runtime tile, select **File Storage Config**.

2. Select the **External S3-Compatible Filestore** option and complete the following fields:

<input checked="" type="checkbox"/> Assign Networks <input checked="" type="checkbox"/> Assign Availability Zones <input checked="" type="checkbox"/> System Database Config <input checked="" type="checkbox"/> File Storage Config <input checked="" type="checkbox"/> IPs and Ports <input type="checkbox"/> Security Config <input checked="" type="checkbox"/> MySQL Proxy Config <input type="checkbox"/> Cloud Controller <input checked="" type="checkbox"/> System Logging <input checked="" type="checkbox"/> SSO Config <input checked="" type="checkbox"/> LDAP Config <input checked="" type="checkbox"/> SMTP Config <input checked="" type="checkbox"/> Diego <input checked="" type="checkbox"/> Experimental Features <input checked="" type="checkbox"/> Errands <input checked="" type="checkbox"/> Resource Config <input type="checkbox"/> Stemcell	<p>Configurations should only be made on the initial installation of Elastic Runtime (do NOT change anything on this section when upgrading). Once Elastic Runtime is installed, a manual data migration is required in order to change this configuration safely. Consult Pivotal Cloud Foundry documentation for details.</p> <p>Configure your Cloud Controller's filesystem*</p> <p><input type="radio"/> Internal</p> <p><input checked="" type="radio"/> External S3-Compatible Filestore</p> <p>URL Endpoint *</p> <p><input type="text" value="https://s3-us-west-2.amazonaws.com"/> <input type="button" value="Edit"/></p> <p>URL endpoint of your S3-compatible blobstore</p> <p>Access Key *</p> <p><input type="text" value="A*****"/></p> <p>Secret Key *</p> <p><input type="text" value="*****"/> <input type="button" value="Edit"/></p> <p>Buildpacks Bucket Name *</p> <p><input type="text" value="pcf-stack-pcfelasticruntimes3buildpacksbucket-178"/></p> <p>Droplets Bucket Name *</p> <p><input type="text" value="pcf-stack-pcfelasticruntimes3dropletsbucket-bqsoh"/></p> <p>Packages Bucket Name *</p> <p><input type="text" value="pcf-stack-pcfelasticruntimes3packagesbucket-1bstl"/></p> <p>Resources Bucket Name *</p> <p><input type="text" value="pcf-stack-pcfelasticruntimes3resourcesbucket-1gdd"/></p> <p style="text-align: center;"><input type="button" value="Save"/></p>
--	---

- For **URL Endpoint**:

- In a browser, open the [Amazon Simple Storage Service \(Amazon S3\) table](#).
- Prepend `https://` to the **Endpoint** for your region and copy it into the Ops Manager **URL Endpoint** field.
For example, in the **us-west-2** region, use `https://s3-us-west-2.amazonaws.com/`.

- Use the values in your AWS **Outputs** tab to complete the remaining fields as follows:

Ops Manager Field	Outputs Key
Buildpacks Bucket Name	PcfElasticRuntimeS3BuildpacksBucket
Droplets Bucket Name	PcfElasticRuntimeS3DropletsBucket
Packages Bucket Name	PcfElasticRuntimeS3PackagesBucket
Resources Bucket Name	PcfElasticRuntimeS3ResourcesBucket
Access Key ID	PcfiamUserAccessKey
AWS Secret Key	PcfiamUserSecretAccessKey

Note: If you are performing an upgrade from Elastic Runtime 1.5 to 1.6, do not modify the existing Bucket Name configuration or you may lose data. You must migrate your existing data first before changing the configuration. See the [Upgrading Ops Manager](#) topic for additional upgrade information.

- Click **Save**.

Step 7: Set IPs and Loggregator Port

1. Select IPs and Ports.

HAProxy is the default load balancer for SSL termination. Alternatively, you can use your own load balancer and forward traffic to the Cloud Foundry router IP.

<input checked="" type="checkbox"/> Assign Networks	Router IPs
<input checked="" type="checkbox"/> Assign Availability Zones	<input type="text"/>
<input checked="" type="checkbox"/> System Database Config	HAProxy IPs
<input checked="" type="checkbox"/> File Storage Config	<input type="text"/>
<input checked="" type="checkbox"/> IPs and Ports	Loggregator Port
<input type="checkbox"/> Security Config	<input type="text" value="4443"/>
<input checked="" type="checkbox"/> MySQL Proxy Config	
<input type="checkbox"/> Cloud Controller	
<input checked="" type="checkbox"/> System Logging	

Save

2. Leave the **Router IPs** and **HAProxy IPs** fields blank.

3. Set **Loggregator Port** to **4443**.

4. Click **Save**.

Step 8: Configure Security and SSL Certificates

1. Click on **Security Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configure security settings

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIICKTCAZICCCVjs8Jtc84cTANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJV
MTUwNTAzMTkyMjEyWjBZMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExiTafBgNV
BAMTCGJhY2tldCBvZiBhbmQgcmVhZG9ubHkgQ0EwHhcNMTIwMjAxMDAwMDAeMBIG
AoGBAKsRCylqUkSi5uHwW2B09isuv1X4zTa3daoSdy6fIEfGEF5RBKAAnPrd7Xxv
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQCSCWbiXc86fFy4mpAf6PmxQTn36d+bj8fUDo6NZQ7B0twTW5iQ
-----END RSA PRIVATE KEY-----
```

[Generate Self-Signed RSA Certificate](#)

Ignore SSL certificate verification

HAProxy SSL Ciphers

Router SSL Ciphers

Disable HTTP traffic to HAProxy

Enable cross-container traffic

Enable TLS on the Router

Save

- Provide an **SSL Termination Certificate** for your SSL Termination Point. This certificate must match the one that you uploaded to AWS earlier in the [Upload an SSL Certificate](#) section of the [Deploying the CloudFormation Template for PCF on AWS](#) topic.

Note: Pivotal does not recommend using a self-signed certificate for production deployments.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

- Configure **Ignore SSL certificate verification**. Select this option if you are using self-signed certificates or certificates generated from Ops Manager.
- Configure **HAProxy SSL Ciphers** and **Router SSL Ciphers**. Leave these fields blank unless you want to use a specific set of SSL ciphers for HAProxy or the Router.

5. Configure **Disable HTTP traffic to HAProxy**. If you select the **Disable HTTP traffic to HAProxy** checkbox, your deployment rejects all port 80 traffic to HAProxy. Additionally, this option sets the secure flag in the `VCAP_ID` cookie that the Router generates.

Note: If you enable this checkbox and your deployment is not using HAProxy, configure your external load balancer to reject port 80 traffic. If you do not do this, traffic to port 80 is forwarded to applications but loses session stickiness.

6. Configure **Enable cross container traffic**. By selecting this checkbox, you disable the restriction that prevents containers in the same DEA or Diego Cell from communicating with each other. Pivotal does not recommend selecting this checkbox in multi-tenant environments. You should select this option for microservices such as Pivotal Spring Cloud.
7. Configure **Enable TLS on the Router**. Selecting this enables [SSL termination](#) on the Router.
8. Click **Save**.

Step 9: Add CNAME Record for Your Custom Domain

In the [Use the AWS CLI to upload your SSL Cert](#) step, you uploaded an SSL certificate for your PCF wildcard domain to AWS. In this step you redirect all wildcard queries for your domain to the DNS name of your ELB.

Note: Do not point your wildcard domain at the numeric IP address for your ELB because this changes frequently.

1. Find the DNS hostname of your ELB. The **Output** tab of the CloudFormation page in the AWS dashboard lists this as the value for the key `PcfElbDnsName`.
2. Log in to the DNS registrar that hosts your domain (for example, Network Solutions, Godaddy, or Register.com).
3. Create a CNAME record with your DNS registrar that points `*.YOUR-DOMAIN.com` to the DNS hostname of your ELB.
4. Save changes within the web interface of your DNS registrar.
5. In the terminal, run the following `dig` command to confirm that you created your CNAME record successfully:

```
dig xyz.MY-DOMAIN.COM
```

You should see the CNAME record that you just created:

```
; ANSWER SECTION:  
xyz.MY-DOMAIN.COM. 1767 IN CNAME pcf-ert-frankfurt-pcf-elb-428333773.eu-central-1.elb.amazonaws.com.
```

Note: You **must** complete this step before proceeding to Cloud Controller configuration. A difficult-to-resolve problem can occur if the wildcard domain is improperly cached before the CNAME is registered.

Step 10: Configure Cloud Controller Page

1. Select **Cloud Controller** and enter your system and application domains.

<input checked="" type="checkbox"/> Assign Networks <input checked="" type="checkbox"/> Assign Availability Zones <input checked="" type="checkbox"/> System Database Config <input checked="" type="checkbox"/> File Storage Config <input checked="" type="checkbox"/> IPs and Ports <input checked="" type="checkbox"/> Security Config <input checked="" type="checkbox"/> MySQL Proxy Config Cloud Controller <input checked="" type="checkbox"/> System Logging <input checked="" type="checkbox"/> SSO Config <input checked="" type="checkbox"/> LDAP Config <input checked="" type="checkbox"/> SMTP Config <input checked="" type="checkbox"/> Diego <input checked="" type="checkbox"/> Experimental Features <input checked="" type="checkbox"/> Errands <input checked="" type="checkbox"/> Resource Config Stemcell	<p>Coordinates Pivotal CF Elastic Runtime application lifecycles</p> <p>System Domain *</p> <input type="text" value="system.example.com"/> <p>Apps Domain *</p> <input type="text" value="apps.example.com"/> <p>Cloud Controller DB Encryption Key</p> <input type="text" value="Secret"/> <p>Maximum File Upload Size (MB) (min: 1024, max: 2048) *</p> <input type="text" value="1024"/> <p><input type="checkbox"/> Disable Custom Buildpacks</p> <p>Default Quota App Memory (MB) (min: 10240, max: 102400) *</p> <input type="text" value="10240"/> <p>Default Quota Service Instances (min: 0, max: 1000) *</p> <input type="text" value="100"/> <p style="text-align: center;">Save</p>
---	---

- The **System Domain** defines your target when you push apps to Elastic Runtime.
- The **Apps Domain** defines where Elastic Runtime should serve your apps.

Note: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes. For example, name your system domain `system.EXAMPLE.COM` and your apps domain `apps.EXAMPLE.COM`.

2. Click **Save**.

Step 11: (Optional) Configure SMTP

Elastic Runtime uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Configuration for SMTP** page if you want to enable end-user self-registration.

1. Select **SMTP Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configuration for Simple Mail Transfer Protocol

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

Username

Password

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

2. Enter your reply-to and SMTP email information.
3. For **SMTP Authentication Mechanism**, select .
4. Click **Save**.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See [Creating and Managing Users with the cf CLI](#) for more information.

Step 12: Configure Diego

In the PCF 1.6 release, Diego replaces the [DEAs](#) and the [Health Manager](#). Diego is installed and enabled in PCF 1.6 by default.

For more information about Diego, see the [Diego Architecture](#) and [Diego Components](#) topics.

Note: Before you install PCF 1.6, you must uninstall any Diego Beta installations. For more information on upgrading to PCF 1.6, see [Upgrading Operations Manager](#).

Pivotal Elastic Runtime

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

Assign Networks Diego is a new replacement for DEAs.

Assign Availability Zones Use Diego by default instead of DEAs

System Database Config Allow SSH access to apps

File Storage Config

IPs and Ports Application Containers Subnet Pool *

Security Config

MySQL Proxy Config

Cloud Controller

System Logging

SSO Config

LDAP Config

SMTP Config

Diego

1. Select **Diego**.

2. Select the **Use Diego by default instead of DEAs** checkbox to ensure that all applications pushed to your PCF deployment use the Diego container management system. For new installations, this option is selected by default. If you are upgrading to PCF 1.6, you must select this option to automatically enable Diego for newly pushed applications.

Note: If you do not select this option, application developers must explicitly target Diego when pushing their applications.

3. (Optional) Select the **Allow SSH access to apps** checkbox to allow application developers to `ssh` directly into their application instances on Diego. See the [Diego-SSH Overview](#) topic for more details.

Step 13: (Optional) Enable Experimental Features

Use caution when enabling experimental features if you have other Pivotal Cloud Foundry service tiles installed in your Pivotal Cloud Foundry deployment. Not all of the services are guaranteed to work as expected with these features enabled.

This release does not contain any experimental features.

Step 14: Configure Errands Page

Errands are scripts that Ops Manager runs to automate tasks. By default, Ops Manager runs the post-install errands listed below when you deploy Elastic Runtime. However, you can prevent a specific post-install errand from running by deselecting its checkbox on the **Errands** page.

Note: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, deselecting the checkbox for the corresponding errand on a

subsequent deployment does not stop the app.

Errands

Errands are scripts that run at designated points during an installation.

Post Install

<input checked="" type="checkbox"/> Push Apps Manager	Pushes the Pivotal Apps Manager application to your Elastic Runtime installation
<input checked="" type="checkbox"/> Run Smoke Tests	Runs Smoke Tests against your Elastic Runtime installation
<input checked="" type="checkbox"/> Push App Usage Service	Monitors usage of an application pushed to your Elastic Runtime installation
<input checked="" type="checkbox"/> Notifications	Notifications release for PCF
<input checked="" type="checkbox"/> Notifications-UI	Notifications UI Component for PCF
<input checked="" type="checkbox"/> Deploy CF Autoscaling App	Deploys the CF Autoscaling App
<input checked="" type="checkbox"/> Register autoscaling service broker	Register autoscaling service broker

There are no pre-delete errands for this product.

[Save](#)

- **Push Apps Manager** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the cf CLI. After Apps Manager has been deployed, we recommend deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the Apps Manager, see [Getting Started with the Apps Manager](#).
- **Run Smoke Tests** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Push App Usage Service** deploys the Usage Service, a dashboard that provides resource usage data and analytics for your apps.
- **Notifications** deploys an API for sending email notifications to your PCF platform users.

Note: The Notifications app requires that you [configure SMTP](#) with a username and password, even if [SMTP Authentication Mechanism](#) is set to none.

- **Notifications with UI** deploys a dashboard for users to manage notification subscriptions.
 - **Deploy CF Autoscaling App** enables your deployment to automatically scale the number of instances of an app in response to changes in its usage load. To enable Autoscaling for an app, you must also bind the Autoscaling service to it. For more information, see the [Bind a Service Instance](#) section of the *Managing Service Instances with the CLI* topic.
- Note:** The Autoscaling app requires the Notifications app to send scaling action alerts by email.
- **Register Autoscaling Service Broker** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.

Step 15: Configure Router to Elastic Load Balancer

1. If you do not know it, find the name of your ELB by clicking **Load Balancers** in the AWS EC2 dashboard.

1. In the **Elastic Runtime** tile, click **Resource Config**.
2. In the **ELB Name** field of the **Router** row, enter the name of your load balancer. You may configure multiple load balancers by entering the names separated by commas.
3. In the **ELB Name** field of the **Diego Brain** row, enter the name of your SSH load balancer. You may configure multiple load balancers by entering the names separated by commas.
4. Click **Save**.

Resource Config

JOB	INSTANCES	EPHEMERAL DISK (MB)	PERSISTENT DISK (MB)	INSTANCE TYPE	ELB NAMES
NATS	1	2048	0	t2.micro	
consul	1	2048	1024	t2.micro	
etcd	1	2048	1024	t2.micro	
Diego etcd	1	2048	1024	t2.micro	
NFS Server	0	2048	102400	t2.micro	
Router	1	2048	0	t2.micro	pcf-stack-pcf-elb
MySQL Proxy	0	4096	0	t2.micro	
MySQL Server	0	30000	100000	r3.large	
Cloud Controller Database (Postgres)	0	2048	2048	t2.micro	
UAA Database (Postgres)	0	2048	8192	t2.micro	
Apps Manager Database (Postgres)	0	2048	1024	t2.micro	
Cloud Controller	1	20480	0	m3.large	
HAProxy	0	2048	0	t2.micro	
Health Manager	1	2048	0	t2.micro	
Clock Global	1	2048	0	t2.micro	
Cloud Controller Worker	1	2048	0	t2.micro	
Collector	0	2048	0	t2.micro	
UAA	1	2048	0	t2.micro	
Diego Brain	1	4096	1024	t2.small	
Diego Cell	3	131072	0	r3.xlarge	
DEA	0	32768	0	r3.xlarge	
Doppler Server	1	2048	0	t2.micro	
Loggregator Trafficcontroller	1	2048	0	t2.micro	
Push Apps Manager	1	1024	0	t2.micro	
Push App Usage Service	1	1024	0	t2.micro	
Run Smoke Tests	1	1024	0	t2.micro	
Notifications	1	1024	0	t2.micro	
Notifications-UI	1	1024	0	t2.micro	
Deploy CF Autoscaling App	1	1024	0	t2.micro	
Register autoscaling service broker	1	1024	0	t2.micro	
Destroy autoscaling service broker	1	1024	0	t2.micro	
Run Diego Acceptance Tests	1	1024	0	t2.micro	
Run CF Acceptance Tests	1	1024	0	t2.micro	
Run CF Acceptance Tests without internet	1	1024	0	t2.micro	
Compilation	4	20480	0	c4.large	

Save

Step 16: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

JOB	INSTANCES	EPHEMERAL DISK (MB)
NATS	1	2048
consul	1	2048
etcd	1	2048
Diego etcd	1	2048
NFS Server	0	2048
Router	1	2048
MySQL Proxy	0	4096
MySQL Server	0	30000
Cloud Controller Database (Postgres)	0	2048
UAA Database (Postgres)	0	2048
Apps Manager Database (Postgres)	0	2048
Cloud Controller	1	20480
HAProxy	0	2048

2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:

- **NFS Server**: Enter 0 in **Instances**.

3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:

- **MySQL Proxy**: Enter 0 in **Instances**.
- **MySQL Server**: Enter 0 in **Instances**.
- **Cloud Controller Database**: Enter 0 in **Instances**.
- **UAA Database**: Enter 0 in **Instances**.
- **Apps Manager Database**: Enter 0 in **Instances**.

4. If you are using an External Load Balancer instead of HAProxy, enter 0 in the **Instances** field for **HAProxy**.

Note: AWS deployments do not use HAProxy.

5. Click **Save**.

Step 17: Download Stemcell

This step is only required if your Ops Manager does not already have the Stemcell version required by Elastic Runtime.

1. Select **Stemcell**.
2. Log into the [Pivotal Network](#) and click on **Stemcells**.
3. Download the appropriate stemcell version targeted for your IaaS.
4. In Ops Manager, import the downloaded stemcell .tgz file.

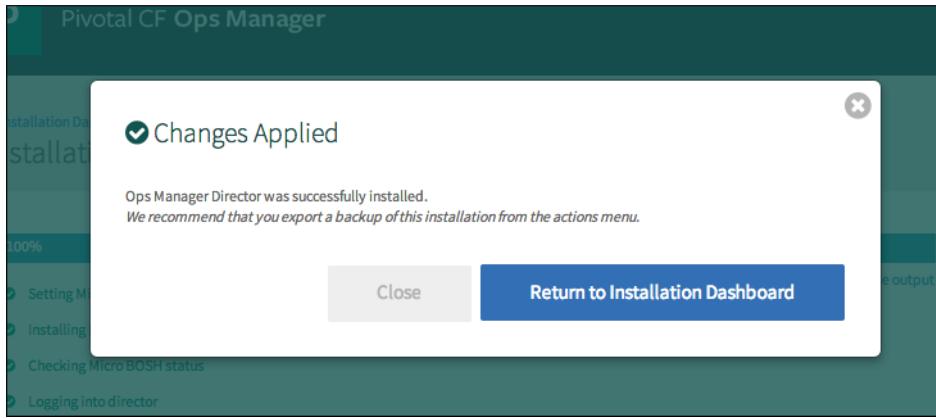
Step 18: Complete the Elastic Runtime Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.

2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



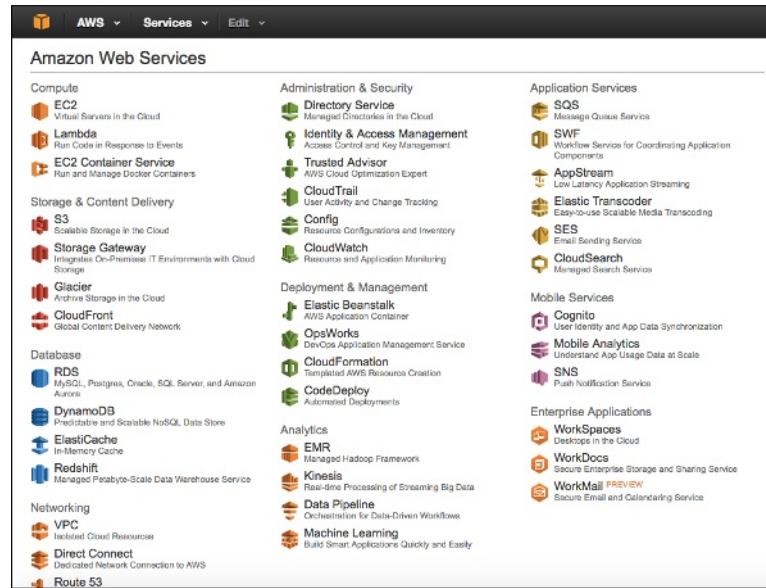
The install process generally requires a minimum of 90 minutes to complete. The image shows the Changes Applied window that displays when the installation process successfully completes.



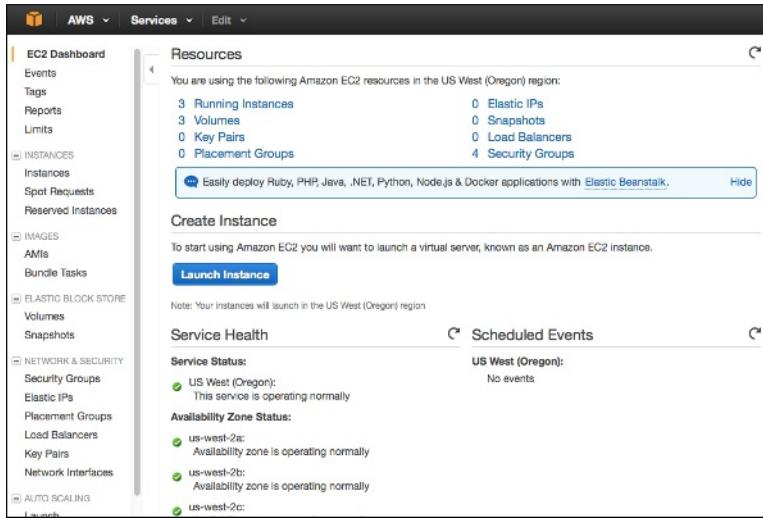
Return to [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Deleting an AWS Installation from the Console

When you deploy Pivotal Cloud Foundry to Amazon Web Services, you provision a set of resources. This topic describes how to delete the AWS resources associated with a PCF deployment. You can use the AWS console to remove an installation of all components, but retain the objects in your bucket for a future deployment.



1. Log into your AWS Console.
2. Navigate to your EC2 dashboard. Select **Instances** from the menu on the left side.

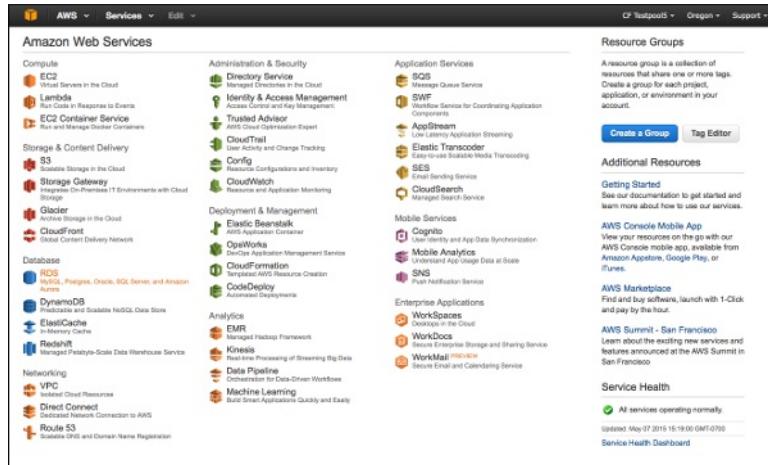


3. Terminate all your instances.

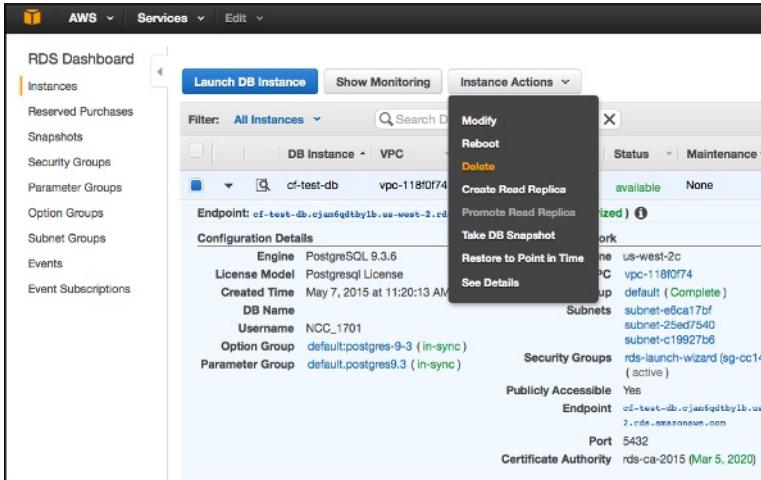
The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links like EC2 Dashboard, Events, Tags, Reports, Limits, Instances, Spot Requests, Reserved Instances, AMIs, Bundle Tasks, Volumes, Snapshots, Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, and Network Interfaces. The Instances section is selected. In the main area, three instances are listed: i-060963f0, i-0c0e64fa, and i-ba0e644c. All three instances are in the 'running' state. A context menu is open over the third instance (i-ba0e644c), with 'Terminate' highlighted.

4. Select **Load Balancers**. Delete all load balancers.

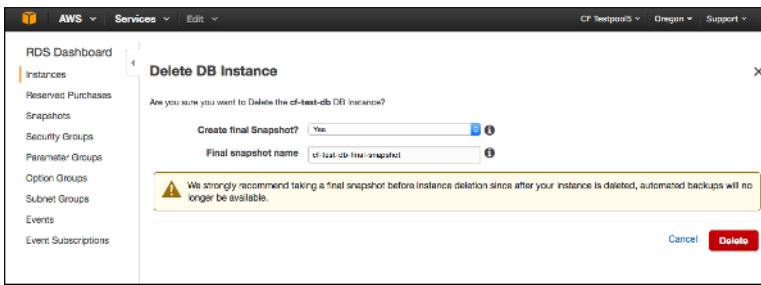
The screenshot shows the AWS Load Balancers page. The sidebar includes links for EC2 Dashboard, Events, Tags, Reports, Limits, Instances, AMIs, Volumes, Security Groups, Load Balancers, and Network Interfaces. The Load Balancers section is selected. One load balancer, named 'CF-Test', is listed. A context menu is open over the 'CF-Test' entry, with 'Delete' highlighted.

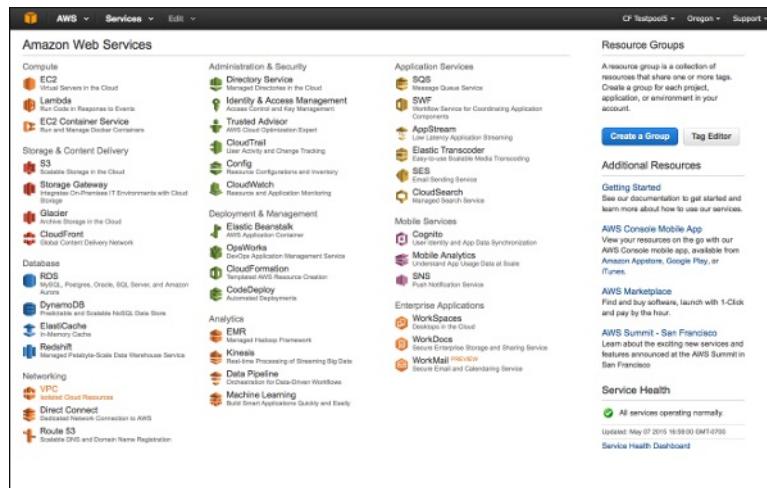


- From the AWS Console, select **RDS**.
 - Select **Instances** from the menu on the left side. Delete the RDS instances.



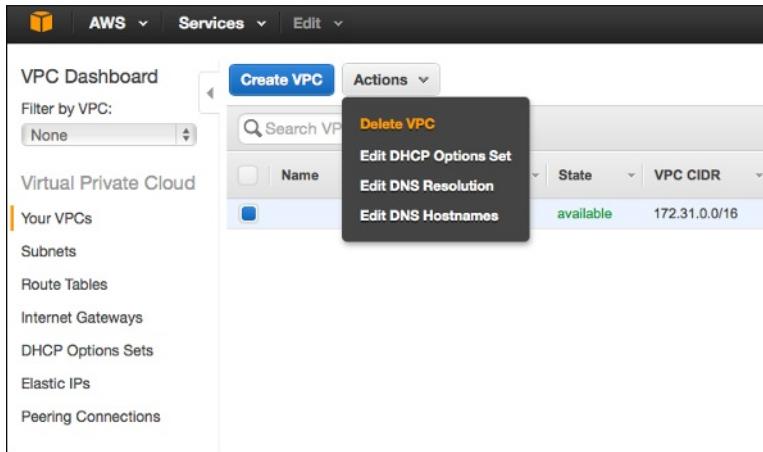
7. Select **Create final Snapshot** from the drop-down menu. Click **Delete**



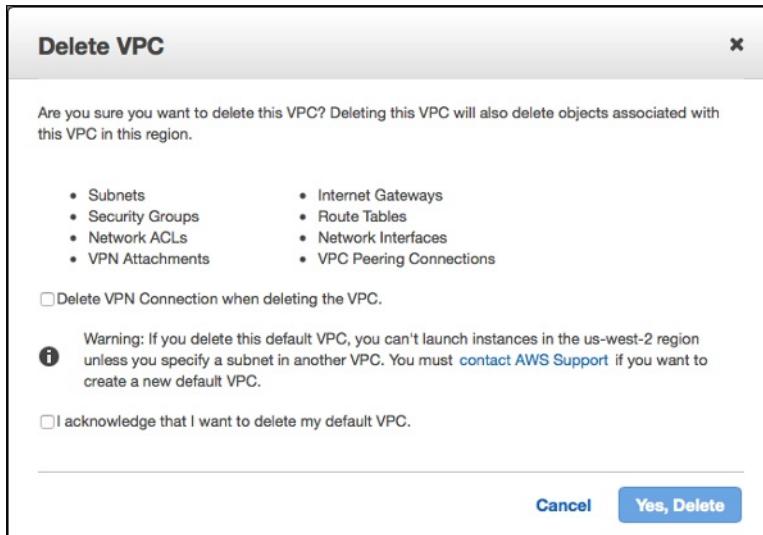


8. From the AWS Console, select **VPC**.

9. Select **Your VPCs** from the menu on the left. Delete the VPCs.



10. Check the box to acknowledge that you want to delete your default VPC. Click **Yes, Delete**.



Manually Configuring AWS for PCF

This topic describes how to manually configure the AWS components that you need to run [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS).

Pivotal strongly recommends installing Pivotal Cloud Foundry using AWS CloudFormation, rather than doing it manually as described here. The procedure using CloudFormation is described in [Installing Pivotal Cloud Foundry Using AWS CloudFormation](#).

Note: PCF does not support the eu-central-1 (Frankfurt) region because of limitations in Amazon's S3 signature API.

Note: PCF for AWS functionality currently does not support the Amazon EC2 Auto Recovery feature. Do not enable this feature for any of your instances.

Note: File a ticket with Amazon to ensure that your account can launch more than the default 20 instances. In the ticket, ask for a limit of **50 t2.micro** instances and **20 c4.large** instances in the region you are using. You can check the limits on your account by visiting the EC2 Dashboard on the AWS console and clicking **Limits** on the left navigation.

Overview of Amazon Objects to Create

The following list describes the objects that you need to create using the AWS console. Complete the steps in order. For more detailed information about a step, click the **Details** link.

Step 1: S3 Buckets for Ops Manager and Elastic Runtime

Dashboard: S3

OPS_MANAGER_S3_BUCKET

ELASTIC_RUNTIME_S3_BUCKET

Must be empty when you install or reinstall PCF.

Step 2: IAM User for PCF

Dashboard: Identity & Access Management

Name: pcf-user

[Policy document](#)

[Details](#)

Step 3: Key Pair

Dashboard: EC2

Name: pcf-user

Refer to the [AWS Documentation](#) to add the private key locally.

Step 4: VPC (Public and Private Subnets)

Dashboard: VPC

IP CIDR block: 10.0.0.0/16

Public subnet: 10.0.0.0/24, public-az1

Availability Zones: Same zone for both subnets

Private subnet: 10.0.16.0/20, pcf-private-az1

NAT Instance type: m1.small

Key Pair name: pcf

Enable DNS hostnames: Yes

Hardware tenancy: Default

[Details](#)

Step 5: Security Group for Ops Manager

Dashboard: EC2

Name: OpsManager
Source: My IP
HTTP, Port 80
HTTPS, Port 443
SSH, Port 22
BOSH Agent, Port 6868
BOSH Director, Port 25555
[Details](#)

Step 6: Security Group for PCF VMs

Dashboard: EC2

Name: pcfVMs
Source: Custom IP, 10.0.0.0/16
All Traffic (0 - 65535)

[Details](#)

Step 7: Security Group for the ELB

Dashboard: EC2

Name: PCF_ELB_SecurityGroup
Source: Anywhere, 0.0.0.0/0
Custom TCP rule, Port 4443
HTTP, Port 80
HTTPS, Port 443

[Details](#)

Step 8: Security Group for the Outbound NAT

Dashboard: EC2

Name: OutboundNAT
Source: Custom IP, 10.0.0.0/16
All Traffic (0 - 65535)

[Details](#)

Step 9: Security Group for MySQL

Dashboard: EC2

Name: MySQL
Source: Custom IP, 10.0.0.0/16
MySQL, Port 3306

[Details](#)

Step 10: Launch an Ops Manager AMI

Dashboard: EC2

Find the AMI ID in the *Ops Manager for AWS* PDF on [Pivotal Network](#).

[Details](#)

Step 11: Load Balancer

Dashboard: EC2

Name: pcf-aws-lb
LB Inside: pcf-vpc
Selected Subnet: public-az1
Security Group: PCF_ELB_SecurityGroup
Health Check: TCP Port 80
Instances: None

[Details](#)

Step 12: Wildcard DNS

Dashboard: N/A

CNAME Host: *
Points to: pcf-aws-lb URL
[Details](#)

Step 13: Secure the NAT Instance

Dashboard: EC2

Name: OutboundNAT

Assign the NAT instance to the OutboundNAT security group.

[Details](#)

Step 14: Subnets for RDS

Dashboard: VPC

rds-1 (region A)

rds-2 (region B)

[Details](#)

Step 15: RDS Subnet Group

Dashboard: RDS

Name: PCF_RDSGroup

[Details](#)

Step 16: MySQL Database using AWS RDS

Dashboard: VPC

Name: bosh

DB Instance Class: db.m3.large - 2 vCPU, 7.5 GiB RAM

Multi-AZ Deployment: Yes

Allocated Storage: 100 GB

DB Instance Identifier: pcf-bosh

VPC: pcf-vpc

Subnet Group: PCF_RDSGroup

Publicly Accessible: No

VPC Security Groups: MySQL

Database Name: bosh

[Details](#)

[Next Step: Configure Ops Manager Director for AWS](#)

Detailed Instructions

The following section contains detailed instructions for completing the steps listed above.

Create an IAM User for PCF

You must create an Amazon Identity and Access Management user (IAM) with the minimal permissions necessary to run and install PCF.

1. Log in to your AWS account.
2. Select **Identity & Access Management** to access the IAM Dashboard.
3. Select **Users>Create New Users**.
4. Enter a user name, such as `pcf-user`.

Create User

Enter User Names:

1.
2.
3.
4.
5.

Maximum 64 characters each

Generate an access key for each user

Users need access keys to make secure REST or Query protocol requests to AWS service APIs.

For users who need access to the AWS Management Console, create a password in the Users panel after completing this wizard.

5. Ensure that the **Generate an access key for each user** checkbox is selected.

6. Click **Create**.

7. Click **Download Credentials** to download the user security credentials.

Note: The `credentials.csv` contains the IDs for your user security access key and secret access key. Be sure to keep the `credentials.csv` file for your currently active key pairs in a secure directory.

8. Click **Close**.

9. On the User dashboard, click the user name to access the user details page.

Create New Users		User Actions ▾
Search		
<input type="checkbox"/>	User Name ▾	Groups
<input type="checkbox"/>	pcf-user	0

10. In the **Inline Policies** region, click the down arrow to display the inline policies. Click the **click here** link to create a new policy.

▼ Permissions

Managed Policies

There are no managed policies attached to this user.

Attach Policy

Inline Policies

There are no inline policies to show. To create one, [click here](#).

11. On the Set Permissions page, click **Custom Policy** and click **Select**.

Manage User Permissions

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

Policy Generator

Custom Policy

Use the policy editor to customize your own set of permissions.

Select

12. On the Review Policy page, enter `pcf-iam-policy` in **Policy Name**.

13. Copy and paste the policy document referenced in [this topic](#) in the **Policy Document** field.
14. Ensure that the **Use autoformatting for policy editing** checkbox is selected.
15. Click **Apply Policy** and review. The Inline Policies region now displays a list of available policies and actions.

Create a VPC

1. Navigate to the VPC Dashboard.

2. Click **Start VPC Wizard**.

The screenshot shows the VPC Dashboard with the following details:

- Resources:** Start VPC Wizard, Launch EC2 Instances.
- Note:** Your Instances will launch in the US East (N. Virginia) region.
- Virtual Private Cloud:**
 - Your VPCs: 2 VPCs
 - Subnets: 4 Subnets
 - Route Tables: 4 Route Tables
 - Internet Gateways: 2 Internet Gateways
 - DHCP Options Sets: 0 DHCP Options Sets
 - Elastic IPs: 0 Elastic IPs
- Amazon VPC Resources:**
 - 2 Network ACLs
 - 6 Security Groups
 - 0 VPC Peering Connections
 - 0 VPN Connections
 - 3 Elastic IPs
 - 18 Running Instances
 - 0 Customer Gateways
 - 0 Virtual Private Gateways

3. Select **VPC with Public and Private Subnets** and click **Select**.

The screenshot shows the "Step 1: Select a VPC Configuration" screen with the following options:

- VPC with a Single Public Subnet
- VPC with Public and Private Subnets** (selected)
- VPC with Public and Private Subnets and Hardware VPN Access
- VPC with a Private Subnet Only and Hardware VPN Access

Creates:

A /16 network with two /24 subnets. Public subnet instances use Elastic IPs to access the Internet. Private subnet instances access the Internet via a Network Address Translation (NAT) instance in the public subnet. (Hourly charges for NAT instances apply.)

Select

Diagram illustrating the VPC configuration: An external cloud icon (Internet, S3, DynamoDB, SNS, SCS, etc.) connects to an "Amazon Virtual Private Cloud". Inside the cloud, there are two subnets: "Public Subnet" and "Private Subnet". A "NAT" instance is shown between them, indicating that private subnet instances access the internet via the NAT instance.

4. Specify the following details for your VPC:

- **IP CIDR block:** Enter `10.0.0.0/16`.
- Enter a VPC name.
- **Public subnet:** Enter `10.0.0.0/24`.
- Set both **Availability Zone** fields to an available zone within the region you have selected. You must set both subnets to the same AZ, such as `us-east-1a`.
- **Public subnet name:** Enter `public-az1`.
- **Private subnet:** Enter `10.0.16.0/20`.
- **Private subnet name:** Enter `pcf-private-az1`.
- Under **Specify the details of your NAT instance**, set the **Instance type** to `m1.small`.
- Select the `pcf` SSH key you created for **Key Pair name**.
- **Enable DNS hostnames:** Click `Yes`.
- **Hardware tenancy:** Select `Default`.
- Click **Create VPC**.

Configure a Security Group for Ops Manager

1. Return to the EC2 Dashboard.
2. Select **Security Groups>Create Security Group**.
3. Enter a security group name and description: `OpsManager`.
4. Select the VPC to which to deploy Ops Manager.

5. Click the **Inbound** tab and add rules according to the table below.

Note: You may relax the IP restrictions after setting the password for OpsManager. Pivotal recommends limiting access to Ops Manager to IP ranges within your organization.

Type	Protocol	Port Range	Source
HTTP	TCP	80	My IP
HTTPS	TCP	443	My IP
SSH	TCP	22	My IP
BOSH Agent	TCP	6868	10.0.0.0/16
BOSH Director	TCP	25555	10.0.0.0/16

6. Click **Create**.

Configure a Security Group for PCF VMs

- On the EC2 Dashboard, select **Security Groups>Create Security Group**.
- Enter a security group name and description: `pcfVMs`.
- Select the VPC to which to deploy the PCF VMs.
- Click the **Inbound** tab and add rules for all traffic from your public and private subnets to your private subnet, as the table and image show. This rule configuration does the following:
 - Enables BOSH to deploy ERS and other services.
 - Enables application VMs to communicate through the router.
 - Allows the load balancer to send traffic to Elastic Runtime.

Type	Protocol	Port Range	Source
All traffic	All	0 - 65535	Custom IP 10.0.0.0/16

5. Click **Create**.

The screenshot shows the AWS EC2 Security Group creation interface. The 'Inbound' tab is selected. A single rule is defined: Type: All traffic, Protocol: All, Port Range: 0 - 65535, and Source: Custom IP 10.0.0.0/16.

Configure a Security Group for the Elastic Load Balancer

- On the EC2 Dashboard, select **Security Groups>Create Security Group**.
- Enter a security group name and description: `PCF_ELB_SecurityGroup`.
- Select the VPC to which to deploy the ELB.
- Click the **Inbound** tab and add rules to allow traffic to ports 80, 443, and 4443 from 0.0.0.0/0, as the table and image show.

Note: You can change the 0.0.0.0/0 to be more restrictive if you want finer control over what can reach the Elastic Runtime. This security group governs external access to the Elastic Runtime from applications such as the cf CLI and application URLs.

Type	Protocol	Port Range	Source
Custom TCP rule	TCP	4443	Anywhere 0.0.0.0/0
HTTP	TCP	80	Anywhere 0.0.0.0/0
HTTPS	TCP	443	Anywhere 0.0.0.0/0

5. Click **Create**.

Security group name	PCF_ELB_SecurityGroup		
Description	PCF_ELB_SecurityGroup		
VPC	vpc-8ec593eb (10.0.0.0/16) pcf-vpc		
Security group rules:			
<input checked="" type="radio"/> Inbound <input type="radio"/> Outbound			
Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	4443	Anywhere 0.0.0.0/0
HTTP	TCP	80	Anywhere 0.0.0.0/0
HTTPS	TCP	443	Anywhere 0.0.0.0/0

Configure a Security Group for the Outbound NAT

1. On the EC2 Dashboard, select **Security Groups>Create Security Group**.
2. Enter a security group name and description: **OutboundNAT**.
3. Select the VPC to which to deploy the Outbound NAT.
4. Click the **Inbound** tab and add a rule to allow all traffic from your VPCs, as the table and image show.

Type	Protocol	Port Range	Source
All traffic	All	All	Custom IP 10.0.0.0/16

5. Click **Create**.

Security group name	OutboundNAT		
Description	OutboundNAT		
VPC	vpc-8ec593eb (10.0.0.0/16) pcf-vpc		
Security group rules:			
<input checked="" type="radio"/> Inbound <input type="radio"/> Outbound			
Type	Protocol	Port Range	Source
All traffic	All	0 - 65535	Custom IP 10.0.0.0/16

Configure a Security Group for MySQL

Note: If you plan to use an internal database, you can skip this step. If you are using RDS, you must configure a security group that enables the Ops Manager VM and Ops Manager Director VM to access the database.

1. On the EC2 Dashboard, select **Security Groups>Create Security Group**.
2. Enter a security group name and description: **MySQL**.
3. Select the VPC to which to deploy MySQL.
4. Click the **Inbound** tab. Add a rule of type MYSQL and specify the subnet of your VPC in **Source**, as the table and image show.

Type	Protocol	Port Range	Source
MySQL	TCP	3306	Custom IP 10.0.0.0/16

5. Click **Create**.

The screenshot shows the AWS Security Groups configuration for a MySQL security group. It includes fields for Security group name (MySQL), Description (MySQL), and VPC (vpc-8ec593eb (10.0.0.0/16) | pcf-vpc). Below these are the security group rules, which are currently empty.

Launch a Pivotal Ops Manager AMI

1. Return to the EC2 Dashboard.
2. Click **AMIs**. Locate the Pivotal Ops Manager public AMI and click **Launch**. You can find the AMI ID in the *Ops Manager for AWS* PDF that you downloaded from [Pivotal Network](#).

Note: There is a different AMI for each region. If you cannot locate the AMI for your region, verify that you have set your AWS Management Console to your desired region. If you still cannot locate the AMI, log in to the [Pivotal Network](#) and file a support ticket.

The screenshot shows the AWS AMI search results for the term "pivotal-ops". The search bar shows "AMI Name : pivotal-ops" and the results table shows one item: "pivotal-ops-manager-20150323T224150-dcce5db" with AMI ID "ami-...".

3. Choose **m3.large** for your instance type and click **Next: Configure Instance Details**.

The screenshot shows a table of AWS instance types. The columns are Family, Type, vCPUs, Memory (GiB), and Instance Storage (GB). The table includes rows for Micro instances (t1.micro, t2.micro), General purpose (t2.small, t2.medium, m3.medium, m3.large), and other instance types. The m3.large row is selected.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
Micro instances	t1.micro Free tier eligible	1	0.613	EBS only
General purpose	t2.micro Free tier eligible	1	1	EBS only
General purpose	t2.small	1	2	EBS only
General purpose	t2.medium	2	4	EBS only
General purpose	m3.medium	1	3.75	1 x 4 (SSD)
General purpose	m3.large	2	7.5	1 x 32 (SSD)

4. Configure the following for your instance:
 - Network:** Select the VPC that you created.
 - Subnet:** Select your public subnet instance.
 - Auto-assign for Public IP:** Select **Enable**.
 - For all other fields, accept the default values.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances	<input type="text" value="1"/>
Purchasing option	<input type="checkbox"/> Request Spot Instances
Network	vpc-8ec593eb (10.0.0.0/16) pcf-vpc <input type="button" value="Create new VPC"/>
Subnet	subnet-fe8129d5(10.0.0.0/24) public-az1 <input type="button" value="Create new subnet"/> 250 IP Addresses available
Auto-assign Public IP	<input type="button" value="Enable"/>
IAM role	<input type="button" value="None"/> <input type="button" value="Create new IAM role"/>
Shutdown behavior	<input type="button" value="Stop"/>
Enable termination protection	<input type="checkbox"/> Protect against accidental termination
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring <small>Additional charges apply.</small>
Tenancy	<input type="button" value="Shared tenancy (multi-tenant hardware)"/> <small>Additional charges will apply for dedicated tenancy.</small>

- Click **Next: Add Storage** and adjust the **Size (GiB)** value. The default persistent disk value is 50 GB. We recommend increasing this value to a minimum of 100 GB.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	/dev/sda1	snap-f1ce6d7e	<input type="text" value="100"/>	<input type="button" value="General Purpose (SSD)"/>	150 / 3000 <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Not Encrypted
Instance Store 0	/dev/sdb	N/A	N/A	N/A	N/A	N/A	Not Encrypted
<input type="button" value="Add New Volume"/>							

- Click **Next: Tag Instance** and **Next: Configure Security Group**.
- Select the Ops Manager security group that you created in [Configure a Security Group for Ops Manager](#).
- Click **Review and Launch** and confirm the instance launch details.
- Click **Launch**.
- Select the **pcf** key pair, confirm that you have access to the private key file, and click **Launch Instances**. You use this key pair only to access the Ops Manager VM.

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

<input type="button" value="Choose an existing key pair"/>
Select a key pair
<input type="text" value="pcf"/>
<input checked="" type="checkbox"/> I acknowledge that I have access to the selected private key file (pcf.pem), and that without this file, I won't be able to log into my instance.
<input type="button" value="Cancel"/> <input type="button" value="Launch Instances"/>

- Click **View Instances** to access the **Instances** page on the EC2 Dashboard.

Create Load Balancer

1. On the EC2 Dashboard, click **Load Balancers**.
2. Click **Create Load Balancer** and configure a load balancer with the following information:
 - Enter a load balancer name.
 - **Create LB Inside:** Select the pcf-vpc VPC that you created in [Create a VPC](#).
 - Ensure that the **Internal Load Balancer** checkbox is not selected.

[1. Define Load Balancer](#) [2. Assign Security Groups](#) [3. Configure Security Settings](#) [4. Configure Health Check](#) [5. Add EC2 Instances](#) [6. Add Tags](#) [7. Review](#)

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name:	pcf-aws-lb		
Create LB Inside:	vpc-e2210d87 (10.0.0.0/16) pcf-vpc		
Create an internal load balancer:	<input type="checkbox"/> (what's this?)		
Enable advanced VPC configuration:	<input checked="" type="checkbox"/>		
Listener Configuration:			
Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80
Add			

Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-e2210d87 (10.0.0.0/16) | pcf-vpc

A Please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

Available Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
+	us-east-1c	subnet-ac5fc2f5	10.0.1.0/24	Private subnet

Selected Subnets				
Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
-	us-east-1c	subnet-af5fc2f6	10.0.0.0/24	Public subnet

3. Under **Select Subnets**, select the public subnet you configured in [Create a VPC](#), and click **Next: Assign Security Groups**.
4. On the **Assign Security Groups** page, select the security group you configured in [Configure a Security Group for the Elastic Load Balancer](#), and click **Next: Configure Security Settings**.

[1. Define Load Balancer](#) [2. Assign Security Groups](#) [3. Configure Security Settings](#) [4. Configure Health Check](#) [5. Add EC2 Instances](#) [6. Add Tags](#) [7. Review](#)

Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group:	<input type="radio"/> Create a new security group <input checked="" type="radio"/> Select an existing security group		
Filter VPC security groups			
Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-41693b25	default	default VPC security group	Copy to new
<input type="checkbox"/> sg-f6c29092	MySQL	MySQL	Copy to new
<input type="checkbox"/> sg-10dd8f74	OpsManager	OpsManager	Copy to new
<input type="checkbox"/> sg-3ec2905a	OutboundNAT	OutboundNAT	Copy to new
<input type="checkbox"/> sg-88dd8fec	PCF VMs	PCF VMs	Copy to new
<input checked="" type="checkbox"/> sg-48c2902c	PCF_ELB_SecurityGroup	PCF_ELB_SecurityGroup	Copy to new

5. The **Configure Security Settings** page displays a security warning because your load balancer is not using a secure listener. You can ignore this warning. Click **Next: Configure Health Check**.

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings **4. Configure Health Check** 5. Add EC2 Instances 6. Add Tags 7. Review

Step 3: Configure Security Settings

A Improve your load balancer's security. Your load balancer is not using any secure listener.

If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

6. Select **TCP** in **Ping Protocol** on the **Configure Health Check** page. Ensure that the **Ping Port** value is **80** and set the **Health Check Interval** to 10 seconds. Click **Next: Add EC2 Instances**.

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings **4. Configure Health Check** 5. Add EC2 Instances 6. Add Tags 7. Review

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Ping Protocol	TCP
Ping Port	80

Advanced Details

Response Timeout	5	seconds
Health Check Interval	10	seconds
Unhealthy Threshold	2	
Healthy Threshold	10	

7. Accept the defaults on the **Add EC2 Instances** page and click **Next: Add Tags**.

Note: AWS dynamically scales the number of load balancer instances, but you must associate the ELB with your routers. Configure this on the [Resource Config page](#) of the Elastic Runtime tile.

8. Accept the defaults on the **Add Tags** page and click **Review and Create**.

9. Review and confirm the load balancer details, and click **Create**.

Note: Pivotal recommends that you change the timeout setting on your ELB to 10 minutes from the default 60 seconds.

Create Wildcard DNS Record

Create a CNAME wildcard record with your DNS provider to point to the load balancer domain, as the following image shows.

Note: You finish setting up the load balancer with a HTTPS listener and SSL certification information when you configure Elastic Runtime. For more information, see the [Finalize the Load Balancer Setup](#) sections of the *Configuring Elastic Runtime for AWS* topic.

CName (Alias) <small>ⓘ</small>	Host	Points To	TTL	Actions
	*	pcf-aws-lb-1109259214.us-east-1.elb.amazonaws.com	1 Hour	

Secure the NAT Instance

- On the EC2 Dashboard, click **Instances**. Select the NAT instance, which has an instance type of **m1.small**.
- From the **Actions** menu, select **Networking>Change Security Groups**.
- Change the NAT security group from the default group to the outbound NAT security group that you created in

Configure a Security Group for the Outbound NAT, and click **Assign Security Groups**.

Security Group ID	Name	Description
<input type="checkbox"/> sg-04bd8da60	default	default VPC security group
<input type="checkbox"/> sg-6d85e209	ElasticRuntime	ElasticRuntime
<input type="checkbox"/> sg-7e80e71a	MySQL	MySQL
<input type="checkbox"/> sg-37b8df53	OpsManager	OpsManager
<input checked="" type="checkbox"/> sg-1482e570	OutboundNAT	OutboundNAT
<input type="checkbox"/> sg-0283e466	PCF_ELB_SecurityGroup	PCF_ELB_SecurityGroup

[Cancel](#) [Assign Security Groups](#)

Set up Subnets for RDS

1. Navigate to the VPC Dashboard and click **Subnets**.
2. Click **Create Subnet** and enter the following information for the first subnet:
 - **Name tag:**
 - **VPC:** Select the pcf-vpc VPC you created in [Create a VPC](#).
 - **Availability Zone:** For performance reasons, set this AZ value to the same AZ value that you defined for the private subnet.
 - **CIDR block:** Define a unique network range, such as .

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag	<input type="text" value="rds-1"/>
VPC	<input type="text" value="vpc-8ec593eb (10.0.0.0/16) pcf-vpc"/>
Availability Zone	<input type="text" value="us-east-1a"/>
CIDR block	<input type="text" value="10.0.2.0/24"/>

[Cancel](#) [Yes, Create](#)

3. Click **Create Subnet** and enter the following information for the second subnet:
 - **Name tag:**
 - **VPC:** Select the pcf-vpc VPC you created in [Create a VPC](#).
 - **Availability Zone:** Select a different AZ than what you specified in rds-1.
 - **CIDR block:** Define a unique network range, such as .

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

Name tag	<input type="text" value="rds-2"/>
VPC	<input type="text" value="vpc-8ec593eb (10.0.0.0/16) pcf-vpc"/>
Availability Zone	<input type="text" value="us-east-1b"/>
CIDR block	<input type="text" value="10.0.3.0/24"/>

Create RDS Subnet Group

1. Navigate to the RDS Dashboard.

2. Create a RDS Subnet Group for the two RDS subnets.

- Navigate to the RDS Dashboard, click **Subnet Groups>Create DB Subnet Group**.
- Enter a name and description.
- **VPC ID:** Select pcf-vpc.
- **Availability Zone and Subnet ID:** Choose the AZ and subnet for rds-1 and click **Add**.
- Repeat the process above to add rds-2 to the group.
- Click **Create**.

The image shows a completed subnet group.

Create DB Subnet Group

To create a new Subnet Group give it a name, description, and select an existing VPC below. Once you select an existing VPC, you will be able to add subnets related to that VPC.

Name	PCF_RDSGroup	
Description	PCF_RDSGroup	
VPC ID	pcf-vpc (vpc-8ec593eb)	

Add Subnet(s) to this Subnet Group. You may add subnets one at a time below or [add all the subnets](#) related to this VPC. You may make additions/edits after this group is created. A minimum of 2 subnets is required.

Availability Zone	us-east-1b	
Subnet ID	subnet-970765e0 (10.0.3.0/	
Add		

Availability Zone	Subnet ID	CIDR Block	Action
us-east-1b	subnet-970765e0	10.0.3.0/24	Remove
us-east-1a	subnet-119c343a	10.0.2.0/24	Remove

[Cancel](#) [Create](#)

Note: On the Subnet Group dashboard page, you might need to refresh the page to view the new group.

Create a MySQL Database using AWS RDS

Note: You must have an empty MySQL database when you install or reinstall PCF for AWS.

1. Navigate to the RDS Dashboard. Click **Instances>Launch DB Instance** to launch the wizard.
2. Select **MySQL**.
3. Select **Yes** to create a database for production environments. Click **Next Step**.
4. Specify the following database details:
 - **DB Instance Class:** Select **db.m3.large - 2 vCPU, 7.5 GiB RAM**.
 - **Multi-AZ Deployment:** Select **Yes**.
 - **Allocated Storage:** Enter **100 GB**.
 - **DB Instance Identifier:** Enter **pcf-bosh**.
 - Define a secure **Master Username** and **Master Password**.
 - Click **Next Step**.

Note: Record the username and password that you entered. You need this later when configuring the **Director Config** page in the Ops Manager Director tile.

Specify DB Details

Instance Specifications

DB Engine	mysql
License Model	general-public-license
DB Engine Version	5.6.22

Review the Known Issues/Limitations to learn about potential compatibility issues with specific database versions.

DB Instance Class	db.m3.large – 2 vCPU, 7.5 GiB RA
Multi-AZ Deployment	Yes
Storage Type	General Purpose (SSD)
Allocated Storage*	100 GB

Settings

DB Instance Identifier*	pcf-bosh
Master Username*	admin
Master Password*	*****
Confirm Password*	*****

* Required
Retype the value you specified for Master Password.

Cancel
Previous
Next Step

5. Configure advanced settings:

- **VPC:** Select pcf-vpc.
- **Subnet Group:** Select the subnet group you created in [Set up Subnets for RDS](#).
- **Publicly Accessible:** Select **No**.
- **VPC Security Groups:** Select the security group that you created in [Configure a Security Group for MySQL](#).
- **Database Name:** Enter `bosh`.
- Accept the default values for the remaining fields.

Configure Advanced Settings

Network & Security

This instance will be created with the new Certificate Authority rds-ca-2015. If you are using SSL to connect to this instance, you should use the [new certificate bundle](#). Learn more [here](#)

VPC*	pcf-vpc (vpc-e2210d87)
Subnet Group	pcf_rdsgroup
Publicly Accessible	No
Availability Zone	No Preference
VPC Security Group(s)	Create new Security Group MySQL (VPC) OpsManager (VPC) OutboundNAT (VPC)

Database Options

Database Name	bosh
---------------	------

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Database Port	3306
DB Parameter Group	default.mysql5.6
Option Group	default:mysql-5-6
Enable Encryption	No

Select to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service(KMS) console. [Learn More](#).

Backup

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail [here](#).

Backup Retention Period	7 days
Backup Window	No Preference

Maintenance

Auto Minor Version Upgrade	Yes
Maintenance Window	No Preference

* Required [Cancel](#) [Previous](#) [Launch DB Instance](#)

- Click **Launch DB Instance**. The instance generation process might take several minutes.

Configure Ops Manager Director for AWS

After you complete this procedure, complete all of the steps in the [Configuring Ops Manager Director for AWS](#) topic.

Configuring Elastic Runtime for AWS

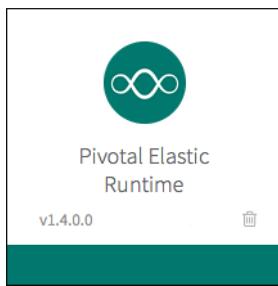
This topic describes how to configure the Pivotal Elastic Runtime components that you need to run [Pivotal Cloud Foundry \(PCF\)](#) on Amazon Web Services (AWS).

Prerequisites

Before following this procedure, complete all of the steps in the [Configuring AWS for PCF](#) and [Configuring Ops Manager Director for AWS](#) topics.

Step 1: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the Pivotal Network link on the left to add Elastic Runtime to Ops Manager.
For more information, refer to the [Adding and Deleting Products](#) topic.
3. Click the Elastic Runtime tile in the Installation Dashboard.



Step 2: Assign Networks and Availability Zones

1. Select **Assign Network**.
2. Select the network that you created in the [Networks Pages](#) section of the *Configuring Ops Manager Director for AWS* topic, and click **Save**.

Note: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.



3. Select **Assign Availability Zones**.
4. Select the option button and checkbox to define the availability zone for singleton jobs and job balancing.
5. Click **Save**.

The screenshot shows the 'Pivotal Elastic Runtime' interface with a navigation bar at the top. Below the navigation bar, there is a sidebar with several configuration items, each with a checked checkbox. To the right of the sidebar, there is a main content area titled 'Availability Zone Assignments'. In this area, there are two sections: 'Place singleton jobs in' and 'Balance other jobs in'. Under 'Place singleton jobs in', there is a radio button labeled 'default' which is selected. Under 'Balance other jobs in', there is a checkbox labeled 'default' which is checked. At the bottom right of the content area is a blue 'Save' button.

Step 3: Configure System Databases

Select **System Database Config**.

If you want to use PCF internal databases, select **Internal** and click **Save**.

If you want to use the external RDS instance for your Elastic Runtime databases, complete the following procedure.

First, create the databases that Elastic Runtime requires. These instructions create those databases on the same RDS instance that you created in [Manually Configuring AWS for PCF](#).

1. Add your key pair to your ssh profile so that you can access the machine:

```
ssh-add pcf.pem
```

2. SSH into your Ops Manager VM using its IP address and the username `ubuntu`:

```
ssh ubuntu@OPS_MANAGER_IP
```

You can find the IP address for the Ops Manager VM on the AWS EC2 console.

3. Log in to your RDS instance through the MySQL client, using the hostname from the RDS console and the username that you created the RDS instance with. This command prompts you to enter the password for your RDS user:

```
mysql -h RDS-HOSTNAME -u RDS-USERNAME -p
```

4. Create databases for each of the six Cloud Foundry components that require a database. Run `CREATE database DB-NAME` for each of the following: `uaa`, `ccdb`, `console`, `notifications`, `autoscale`, `app_usage_service`

5. You have now created the databases on the RDS instance. Exit the mysql client and close your connection to the Ops Manager VM.

6. Navigate to the **System Database Config** page of the Elastic Runtime product tile.

7. Select **External Databases**.

8. Enter the RDS hostname, port (`3306`), username, and password.

9. Click **Save**.

System Database Config **System Database Config***

File Storage Config Internal Databases
 External Databases

IPs and Ports

MySQL Proxy Config

Cloud Controller

External Endpoints

SSO Config

LDAP Config

SMTP Config

Errands

Resource Config

Stemcell

Hostname DNS Name * RDS-HOSTNAME

TCP Port * 3306

Username * RDS-USERNAME

Password * *****

Save

10. (Optional) Select **MySQL Proxy Config**. Enter one or more IP addresses for the MySQL proxy instances configured on your external load balancer.

Use of a proxy permits transparent failover to other nodes within the cluster in the event of a node failure. When a node becomes unhealthy, the proxy re-routes all subsequent connections to a healthy node and all existing connections to the unhealthy node are closed. See [MySQL for PCF: Proxy](#) for more information.

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign Networks A proxy tier routes MySQL connections from internal components to healthy cluster nodes. Configure DNS and/or your own load balancer to point to multiple proxy instances for increased availability. TCP healthchecks can be configured against port 1936.

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

Security Config

MySQL Proxy Config MySQL Proxy IPs

Enter the IP address(es) for the MySQL proxy instances configured on your external load balancer.

MySQL Service Hostname

Cloud Controller

Save

Step 4: Configure File Storage

Select **File Storage Config**.

To use the PCF internal filestore, select **Internal** and click **Save**.

To use an external S3-compatible filestore for your Elastic Runtime file storage, complete the following procedure:

1. Select **External S3-Compatible Filestore**.

2. Enter the **URL Endpoint** for your filestore. If you manually configured Amazon Web Services for PCF, this is the value that you entered into the Director config page in [Configuring Ops Manager Director for AWS](#).
3. Enter the **Bucket Name**. If you manually configured Amazon Web Services for PCF, this is the name that you chose in [Create S3 Buckets](#).
4. Enter your **Access Key** and **Secret Key**. If you are using Amazon Web Services, retrieve your key information with the IAM credentials that you generated in [Create an IAM User for PCF](#).
5. Click **Save**.

<input checked="" type="checkbox"/> Assign Networks <input checked="" type="checkbox"/> Assign Availability Zones <input checked="" type="checkbox"/> System Database Config <input checked="" type="checkbox"/> File Storage Config <input checked="" type="checkbox"/> IPs and Ports <input checked="" type="checkbox"/> Security Config <input checked="" type="checkbox"/> MySQL Proxy Config <input checked="" type="checkbox"/> Cloud Controller <input checked="" type="checkbox"/> External Endpoints <input checked="" type="checkbox"/> SSO Config <input checked="" type="checkbox"/> LDAP Config <input checked="" type="checkbox"/> SMTP Config <input checked="" type="checkbox"/> Errands	<p>Select internal if you want to keep using the system-provided filestore. Otherwise select external if you want to use your own S3-compatible filestore.</p> <p>File Storage Config*</p> <p><input type="radio"/> Internal</p> <p><input checked="" type="radio"/> External S3-Compatible Filestore</p> <p>URL Endpoint*</p> <p><input type="text" value="https://s3.amazonaws.com"/></p> <p>URL endpoint of your S3-compatible blobstore</p> <p>Bucket Name *</p> <p><input type="text"/></p> <p>Access Key *</p> <p><input type="text"/></p> <p>Secret Key *</p> <p><input type="text" value="Secret"/></p> <p style="text-align: right;">Save</p>
--	--

Step 5: Configure the Loggregator Port

1. Select **IPs and Ports**.
2. Leave the **Router IPs** and **HAProxy IPs** fields blank.
3. Set **Loggregator Port** to **4443**.

Step 6: Complete Security Configuration

1. Click on **Security Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configure security settings

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIICKTCAZICCQCVjs8Jtc84cTANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJV
MTUwNTAzMTkyMjEyWjBZMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExiTafBgNV
BAMTCGJhY2tldCBvZiBhbmQgcmVhZG9ubHkgQ0EwHhcNMTIwMjAxMDAwMjUwWjBZ
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC5CWbiXc86fFy4mpAf6PmxQTn36d+bj8fUDo6NZQ7B0twTW5iQ
AoGBAKsRCylqUkSi5uHwW2B09isuv1X4zTa3daoSDy6fIEfGEF5RBKAAnPrd7Xv
-----END RSA PRIVATE KEY-----
```

[Generate Self-Signed RSA Certificate](#)

Ignore SSL certificate verification

HAProxy SSL Ciphers

Router SSL Ciphers

Disable HTTP traffic to HAProxy

Enable cross-container traffic

Enable TLS on the Router

[Save](#)

2. Provide an **SSL Termination Certificate** for your SSL Termination Point.

- In a production environment, use a signed **SSL Certificate** from a known certificate authority (CA). Copy and paste the values for **Certificate PEM** and **Private Key PEM** from the signed certificate into the appropriate text fields.
- In a development or testing environment, you may use a self-signed certificate.

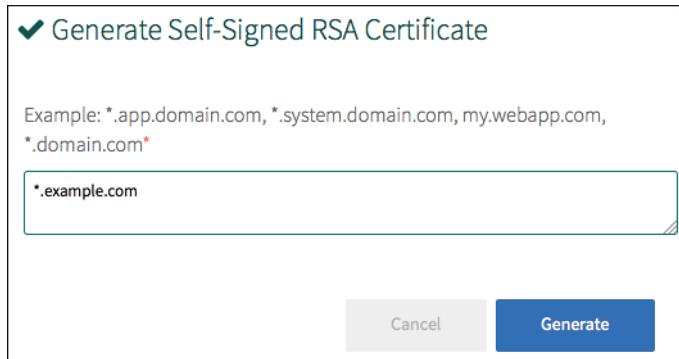
Note: Pivotal does not recommend using a self-signed certificate for production deployments.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

To use a self-signed certificate, follow the steps below: * Click the **Generate Self-Signed RSA Certificate** link to launch the RSA certificate wizard. * Enter your system and app domains in wildcard format. Optionally, also add any custom domains in wildcard format. The example below uses `*.example.com`.

Note: Wildcard DNS records only work for a single domain name component or component fragment. For example, `*.domain.com` works for `apps.domain.com` and `system.domain.com`, but not for `apps.domain.com` or `system.domain.com`.

Note: You can generate a single certificate for two domains separated by a comma. For example, `.apps.domain.com, *.system.domain.com`.



* Click **Generate**. * Elastic Runtime populates the **SSL Certificate** fields with RSA certificate and private key information.

3. Configure **Ignore SSL certificate verification**. Select this option if you are using self-signed certificates or certificates generated from Ops Manager.
4. Configure **HAProxy SSL Ciphers** and **Router SSL Ciphers**. Leave these fields blank unless you want to use a specific set of SSL ciphers for HAProxy or the Router.
5. Configure **Disable HTTP traffic to HAProxy**. If you select the **Disable HTTP traffic to HAProxy** checkbox, your deployment rejects all port 80 traffic to HAProxy. Additionally, this option sets the secure flag in the `VCAP_ID` cookie that the Router generates.

Note: If you enable this checkbox and your deployment is not using HAProxy, configure your external load balancer to reject port 80 traffic. If you do not do this, traffic to port 80 is forwarded to applications but loses session stickiness.

6. Configure **Enable cross container traffic**. By selecting this checkbox, you disable the restriction that prevents containers in the same DEA or Diego Cell from communicating with each other. Pivotal does not recommend selecting this checkbox in multi-tenant environments. You should select this option for microservices such as Pivotal Spring Cloud.
7. Configure **Enable TLS on the Router**. Selecting this enables [SSL termination](#) on the Router.
8. Click **Save**.

Step 7: Finalize the Load Balancer Setup

In this step, you complete your load balancer configuration that you began when setting up AWS components for PCF. For more information, see the [Create Load Balancer](#) section of the *Manually Configuring AWS for PCF* topic.

1. On the EC2 Dashboard, click **Load Balancers**.
 2. Select the load balancer that you created in the [Create Load Balancer](#) section of the *Manually Configuring AWS for PCF* topic. The **Load balancer** detail tabs display at the bottom of the page.
- Note:** You must associate this ELB with your routers. [Configure this](#) on the **Resource Config** page of the Elastic Runtime tile.
3. Select **Listeners**.

4. Click **Edit** to define a second listener with the following values:
 - **Load Balancer Protocol:** **HTTPS (Secure HTTP)**
 - **Load Balancer Port:** **443**
 - **Instance Protocol:** **HTTP**
 - **Instance Port:** **80**
5. Click **Change** in the **SSL Certificate** column of the second listener.
6. On the Select Certificate page, click **Upload a new SSL Certificate** and complete the following information:
 - Enter a **Certificate Name**.
 - **Private Key and Public Key Certificate:** Paste the RSA certificate and private key information from the **SSL Certificate** fields of the IP and Ports page to the appropriate field.
 - Click **Save**.

An SSL Certificate allows you to configure the HTTPS/SSL listeners of your load balancer. You may select a previously uploaded certificate below, or define a new SSL Certificate. [Learn more](#) about setting up HTTPS load balancers and certificate management.

Certificate Type: Choose an existing SSL Certificate Upload a new SSL Certificate

Certificate Name: pcfawscert

Private Key: -----END RSA PRIVATE KEY-----
(pem encoded)
-----BEGIN RSA PRIVATE KEY-----
MIIEvQIBAAKCAQD...
-----END RSA PRIVATE KEY-----
(pem encoded)

Public Key Certificate: -----END CERTIFICATE-----
(pem encoded)
-----BEGIN CERTIFICATE-----
MIID...
-----END CERTIFICATE-----
(pem encoded)

Certificate Chain: Optional

7. Click **Edit** to define a third listener with the following values for use with websockets:
 - **Load Balancer Protocol:** **SSL (Secure TCP)**
 - **Load Balancer Port:** **4443**
 - **Instance Protocol:** **TCP**
 - **Instance Port:** **80**
8. Click **Change** in the **SSL Certificate** column of the third listener.
When you click **Save**, AWS prompts you to enter SSL certificates for these listeners.

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
HTTP	80	HTTP	80	N/A	N/A
HTTPS (Secure HTTP)	443	HTTP	80	Change	pcfawscert Change
SSL (Secure TCP)	4443	TCP	80	Change	pcfawscert Change

- Select **Choose an existing SSL certificate** and select the certificate that you previously uploaded for the second listener.

Note: There might be a brief delay before the certificate that you uploaded for the second listener appears. If you do not see the certificate that you uploaded, try refreshing the page.

The image shows the completed listener information.

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate	
HTTP	80	HTTP	80	N/A	N/A	X
HTTPS (Secure HTTP)	443	HTTP	80	Change	pcfawscert Change	X
SSL (Secure TCP)	4443	TCP	80	Change	pcfawscert Change	X

- Click **Save** to complete the load balancer configuration.

Step 8: Configure the Cloud Controller

- Select **Cloud Controller** and enter your system and application domains.
 - The **System Domain** defines your target when you push apps to Elastic Runtime.
 - The **Apps Domain** defines where Elastic Runtime should serve your apps.

Pivotal recommends that you use the same domain name, but different subdomain names, for your system domain and your app domain. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes.

For example, name your system domain `system.MYDOMAIN.COM` and your apps domain `apps.MYDOMAIN.COM`.

- Click **Save**.

The screenshot shows the 'Settings' tab selected in the Pivotal Elastic Runtime interface. Under the 'Cloud Controller' section, the 'System Domain' is set to 'system.mydomain.com' and the 'Apps Domain' is set to 'apps.mydomain.com'. Other settings like 'Maximum File Upload Size (MB)' and 'Default Quota App Memory (MB)' are also visible.

Step 9: (Optional) Configure SMTP

Elastic Runtime uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the Configuration for SMTP page if you want to enable end-user self-registration and the [Notifications with UI](#) feature.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See [Creating and Managing Users with the cf CLI](#) for more information.

1. Select **SMTP Config**.
2. Enter your reply-to and SMTP email information
3. Verify your authentication requirements with your email administrator and use the **SMTP Authentication Mechanism** drop-down menu to select `None`, `Plain`, or `CRAMMD5`. If you have no SMTP authentication requirements, select `None`.
4. Click **Save**.

Step 10: (Optional) Enable Experimental Features

Use caution when enabling experimental features if you have other Pivotal Cloud Foundry service tiles installed in your Pivotal Cloud Foundry deployment. Not all of the services are guaranteed to work as expected with these features enabled.

This release does not contain any experimental features.

Step 11: Configure Errands

Errands are scripts that Ops Manager runs to automate tasks. By default, Ops Manager runs the post-install errands listed below when you deploy Elastic Runtime. However, you can prevent a specific post-install errand from running by deselecting its checkbox on the **Errands** page.

Note: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, deselecting the checkbox for the corresponding errand on a subsequent deployment does not stop the app.

Errands

Errands are scripts that run at designated points during an installation.

Post Install

- | | |
|---|--|
| <input checked="" type="checkbox"/> Push Apps Manager | Pushes the Pivotal Apps Manager application to your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Run Smoke Tests | Runs Smoke Tests against your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Push App Usage Service | Monitors usage of an application pushed to your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Notifications | Notifications release for PCF |
| <input checked="" type="checkbox"/> Notifications-UI | Notifications UI Component for PCF |
| <input checked="" type="checkbox"/> Deploy CF Autoscaling App | Deploys the CF Autoscaling App |
| <input checked="" type="checkbox"/> Register autoscaling service broker | Register autoscaling service broker |

There are no pre-delete errands for this product.

[Save](#)

- **Push Apps Manager** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the cf CLI. After Apps Manager has been deployed, we recommend deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the Apps Manager, see [Getting Started with the Apps Manager](#).
- **Run Smoke Tests** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Push App Usage Service** deploys the Usage Service, a dashboard that provides resource usage data and analytics for your apps.
- **Notifications** deploys an API for sending email notifications to your PCF platform users.

Note: The Notifications app requires that you [configure SMTP](#) with a username and password, even if [SMTP Authentication Mechanism](#) is set to none.

- **Notifications with UI** deploys a dashboard for users to manage notification subscriptions.
- **Deploy CF Autoscaling App** enables your deployment to automatically scale the number of instances of an app in response to changes in its usage load. To enable Autoscaling for an app, you must also bind the Autoscaling service to it. For more information, see the [Bind a Service Instance](#) section of the *Managing Service Instances with the CLI* topic.

Note: The Autoscaling app requires the Notifications app to send scaling action alerts by email.

- **Register Autoscaling Service Broker** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.

Step 12: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

Resource Config		
JOB	INSTANCES	EPHEMERAL DISK (MB)
NATS	<input type="text" value="1"/>	2048
consul	1	2048
etcd	1	2048
Diego etcd	1	2048
NFS Server	0	2048
Router	1	2048
MySQL Proxy	0	4096
MySQL Server	0	30000
Cloud Controller Database (Postgres)	0	2048
UAA Database (Postgres)	0	2048
Apps Manager Database (Postgres)	0	2048
Cloud Controller	1	20480
HAProxy	0	2048

2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **NFS Server**: Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy**: Enter in **Instances**.
 - **MySQL Server**: Enter in **Instances**.
 - **Cloud Controller Database**: Enter in **Instances**.
 - **UAA Database**: Enter in **Instances**.
 - **Apps Manager Database**: Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter 0 in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 13: Configure Router to Elastic Load Balancer

1. In the **Elastic Runtime** tile, click **Resource Config**.
2. In the **ELB Name** field of the **Router** row, enter the name of your load balancer. You may configure multiple load balancers by entering the names separated by commas.
3. In the **ELB Name** field of the **Diego Brain** row, enter the name of your SSH load balancer. You may configure multiple load balancers by entering the names separated by commas.
4. Click **Save**.

Resource Config					
JOB	INSTANCES	EPHEMERAL DISK (MB)	PERSISTENT DISK (MB)	INSTANCE TYPE	ELB NAMES
NATS	1	2048	0	t2.micro	
consul	1	2048	1024	t2.micro	
etcd	1	2048	1024	t2.micro	
Diego etcd	1	2048	1024	t2.micro	
NFS Server	0	2048	102400	t2.micro	
Router	1	2048	0	t2.micro	pcf-stack-pcf-elb
MySQL Proxy	0	4096	0	t2.micro	
MySQL Server	0	30000	100000	r3.large	
Cloud Controller Database (Postgres)	0	2048	2048	t2.micro	
UAA Database (Postgres)	0	2048	8192	t2.micro	
Apps Manager Database (Postgres)	0	2048	1024	t2.micro	
Cloud Controller	1	20480	0	m3.large	
HAProxy	0	2048	0	t2.micro	
Health Manager	1	2048	0	t2.micro	
Clock Global	1	2048	0	t2.micro	
Cloud Controller Worker	1	2048	0	t2.micro	
Collector	0	2048	0	t2.micro	
UAA	1	2048	0	t2.micro	
Diego Brain	1	4096	1024	t2.small	
Diego Cell	3	131072	0	r3.xlarge	
DEA	0	32768	0	r3.xlarge	
Doppler Server	1	2048	0	t2.micro	
Loggregator Trafficcontroller	1	2048	0	t2.micro	
Push Apps Manager	1	1024	0	t2.micro	
Push App Usage Service	1	1024	0	t2.micro	
Run Smoke Tests	1	1024	0	t2.micro	
Notifications	1	1024	0	t2.micro	
Notifications-UI	1	1024	0	t2.micro	
Deploy CF Autoscaling App	1	1024	0	t2.micro	
Register autoscaling service broker	1	1024	0	t2.micro	
Destroy autoscaling service broker	1	1024	0	t2.micro	
Run Diego Acceptance Tests	1	1024	0	t2.micro	
Run CF Acceptance Tests	1	1024	0	t2.micro	
Run CF Acceptance Tests without internet	1	1024	0	t2.micro	
Compilation	4	20480	0	c4.large	

Save

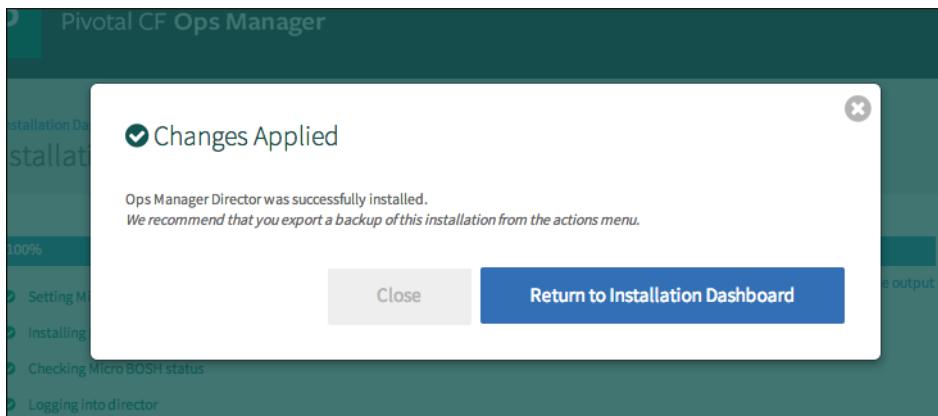
Step 14: Complete the Elastic Runtime Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes** to begin your installation of Elastic Runtime.

If an ICMP error message appears, click **Ignore errors and start the install**.



The install process generally requires a minimum of 90 minutes to complete. The image shows the **Changes Applied** window that displays when the installation process successfully completes.



[Return to the Getting Started Guide](#)

Configuring Ops Manager Director for AWS

This topic describes how to configure the Pivotal Cloud Foundry Operations Manager components that you need to run [Pivotal Cloud Foundry](#) (PCF) on Amazon Web Services (AWS).

Prerequisites

Ensure that you have successfully completed all steps in the [Manually Configuring AWS for PCF](#) topic before beginning this procedure.

Step 1: Access Ops Manager

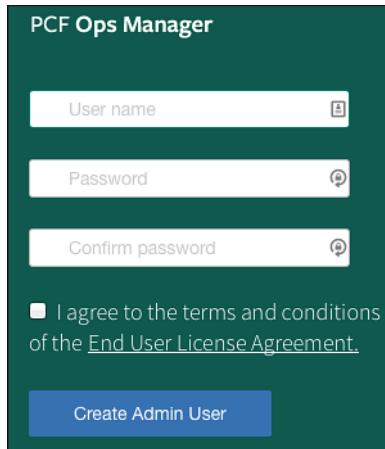
1. Locate the public DNS address address for Ops Manager on the **Instances** page of the AWS EC2 Dashboard. Sort the **Instance Type** column to find the **m3.large** instance that you created in [Configure the Pivotal Ops Manager AMI](#), then note the Public DNS for that instance. For example:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-89db0d5e	m3.large	us-east-1a	running	2/2 checks ...	None	ec2- compute-1.amazonaws.com

2. Use the public DNS address of your Ops Manager instance to launch Ops Manager in a browser.

Note: On your first login attempt, an error message that the connection is untrusted appears because you are attempting to connect securely to a website with a self-signed certificate. Add Ops Manager as an exception to bypass this message on subsequent logins.

3. Enter a username and password and accept the EULA to create an admin user.



Step 2: AWS Config Page

1. Click the **Ops Manager Director powered by AWS** tile.



2. Select **AWS Config**.
3. Enter your **Access Key ID** and **AWS Secret Key** information. To retrieve your AWS key information, use the IAM

credentials that you generated. Refer to the [Obtain AWS Credentials](#) section of the *Manually Configuring AWS for PCF* topic.

4. **VPC ID:** Enter your PCF VPC ID. You can find the VPC ID on the AWS VPC Dashboard page next to the VPC name.
5. **Security Group Name:** Enter `pcfVMs`.
For more information, refer to the [Configure a Security Group for PCF VMs](#).
6. Choose **pcf** as the key pair name.
7. Paste the contents of the generated key file `pcf.pem` into the **SSH Private Key** field.
8. Select your region from the **Region** dropdown menu.
9. Click **Save**.

AWS Management Console Config

Access Key ID*
AKIAIBGY7CQ6MS5WOUUA

AWS Secret Key*

VPC ID*
vpc-8ec593eb

Security Group Name*
pcfVMs

Key Pair Name*
pcf

SSH Private Key*
-----BEGIN RSA PRIVATE KEY-----
p0g+1m\$J8qrAKV4GiDQqcKIH06HH6Z2FwGNJ
n/RTA4RgPitZ+wXfhBTG9BdAtMzOqJQfQEyupsf
/XA2J1GudQ8mLeji+Hf8kpyp3BJdL/mXM=
====END RSA PRIVATE KEY=====

Save

Step 3: Director Config Page

1. Select **Director Config**.
2. Enter at least two of the following NTP servers in the **NTP Servers (comma delimited)** field, separated by a comma:
 - 0.amazon.pool.ntp.org
 - 1.amazon.pool.ntp.org
 - 2.amazon.pool.ntp.org
 - 3.amazon.pool.ntp.org
3. Select the **Enable VM Resurrector Plugin** checkbox to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability.
For more information, see the introduction and Limitation sections of the [Using Ops Manager Resurrector on VMware](#).

vSphere topic.

4. Select **S3 Compatible Blobstore** for **Blobstore Location** and complete the associated fields:
 - **S3 Endpoint:** Navigate to the [Regions and Endpoints](#) topic in the AWS documentation. Locate the endpoint for your region in the **Amazon Simple Storage Service (S3)** table and construct a URL using your region's endpoint. For example, if you are using the us-west-1 region the URL you create would be `https://s3-us-west-1.amazonaws.com`. Enter this URL into the **S3 Endpoint** field in Ops Manager.
 - **Bucket Name:** Enter the Ops Manager bucket name that you defined in the [Create an S3 Bucket](#) section of the *Manually Configuring PCF for AWS* topic.
 - **Access Key and Secret Key:** Enter your **Access Key ID** and **AWS Secret Key** information. To retrieve your AWS key information, use the IAM credentials that you generated in the [Obtain AWS Credentials](#) section of the *Manually Configuring AWS for PCF* topic.

5. Select **External MySQL Database** for **Database Location** and complete the associated fields.

The details page for your RDS instance contains values required for these fields. On the RDS Dashboard, select **Instances**, then click the arrow to the left of your instance to display this page.

The table lists the database instance field name with its equivalent field on the Director Config page, and also provides the values for the **Port** and **Database** fields.

RDS Instance Field	Ops Manager Director Field
Endpoint	Host
Port	Port, which is <code>3306</code> .
DB Name	Database, which is <code>bosh</code> .
Username	Username

For **Password**, enter the password that you defined for your MySQL database. For more information, refer to the [Create a MySQL Database using AWS RDS](#) section of the *Manually Configuring PCF for AWS* topic .

6. **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. For AWS, the default value is 6. Leave the Max Threads field blank to use this default value.
Pivotal recommends that you use the default value unless doing so results in rate limiting or errors on your IaaS.
7. Click **Save**.

Director Config

NTP Servers (comma delimited)*

Metrics IP Address

Enable VM Resurrector Plugin

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*

Bucket Name *

Access Key*

Secret Key*

Database Location
 Internal
 External MySQL Database

Host*

Port*

Username *

Password*

Database*

Max Threads

Save

Step 4: Availability Zones Pages

1. Select **Create Availability Zones**.
 2. **Amazon Availability Zone:** Enter the name of the AWS AZ that contains the private subnet CIDR (10.0.16.0/20) for your VPC, such as us-east-1b.
The image shows where to find the **Availability Zone** name on the **Summary** tab of the VPC Subnets page. For more information, refer to the [Create a VPC](#) section of the *Manually Configuring PCF for AWS* topic.

Name	Subnet ID	State	VPC	CIDR	Available IPs	Availability Zone	Route Table
rds-1	subnet-a2973489	available	vpc-76feb013 (10.0.0.0/16) PC...	10.0.20.0/25	122	us-east-1a	rtb-7a78231f
pcf-private-az1	subnet-ff8129d4	available	vpc-8ec593eb (10.0.0.0/16) pc...	10.0.16.0/20	4091	us-east-1a	rtb-5a2c013f
rds-2	subnet-b20065c5	available	vpc-76feb013 (10.0.0.0/16) PC...	10.0.30.0/25	122	us-east-1b	rtb-7a78231f
rds-2	subnet-970765e0	available	vpc-8ec593eb (10.0.0.0/16) pc...	10.0.3.0/24	250	us-east-1b	rtb-5a2c013f
public-az1	subnet-fe8129d5	available	vpc-8ec593eb (10.0.0.0/16) pc...	10.0.0.0/24	248	us-east-1a	rtb-242c0141
rds-1	subnet-119c343a	available	vpc-8ec593eb (10.0.0.0/16) pc...	10.0.2.0/24	250	us-east-1a	rtb-5a2c013f

subnet-ff8129d4 (10.0.16.0/20) | pcf-private-az1

Summary Route Table Network ACL Tags

Subnet ID:	subnet-ff8129d4 pcf-private-az1	Availability Zone:	us-east-1a
CIDR:	10.0.16.0/20	Route table:	rtb-5a2c013f
State:	available	Network ACL:	acl-54672f31
VPC:	vpc-8ec593eb (10.0.0.0/16) pcf-vpc	Default subnet:	no
Available IPs:	4091	Auto-assign Public IP:	no

3. Click **Save**.
 4. Select **Assign Availability Zones**.
 5. **Singleton Availability Zone:** Select the same AZ value that you specified for **Amazon Availability Zone**.

- AWS Config
- Director Config
- Create Availability Zones
- Assign Availability Zones
- Create Networks
- Assign Networks
- Resource Config

Assign Availability Zones

The Ops Manager Director is a single instance.
Choose the availability zone in which to place that instance. It is highly recommended that you backup this VM on a regular basis to preserve settings.

Singleton Availability Zone

Save

6. Click **Save**.

Step 5: Networks Pages

1. Select **Create Networks**.
 2. Enter a unique name for this network.
 3. Enter your private subnet ID in **VPC Subnet ID**. This ID displays on the **Subnets** page of the **VPC Dashboard**.
 4. **Subnet (CIDR Range)**: Enter `10.0.16.0/20`.

Note: You can assign a single CIDR block to a VPC. The allowed block size is between a /28 netmask and /16 netmask.

5. **Excluded IP Ranges:** Exclude Amazon IPs used at the bottom of the private range. Enter `10.0.16.1-10.0.16.9`.
 6. **DNS:** Enter `10.0.0.2`.
 7. **Gateway:** Enter `10.0.16.1`.

AWS Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Create Network

Name*

VPC Subnet ID*

Subnet (CIDR Range)*

Excluded IP Ranges

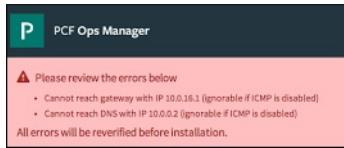
DNS*

Gateway*

Save

8. Click **Save**.

Note: When you save this form, a verification error displays because the PCF security group blocks ICMP. You can ignore this error.



9. Select **Assign Networks** and assign the Director to the network that you created in the previous step.

AWS Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

The Ops Manager director can be configured to have an IP on a network.

Network

Save

Assign Networks

Resource Config

Step 6: Resource Config Page

1. Select **Resource Config**.
2. Adjust any values as necessary for your deployment, such as increasing the persistent disk size.
3. Click **Save**.

JOB	INSTANCES	EPHEMERAL DISK (MB)	PERSISTENT DISK (MB)	INSTANCE TYPE	ELB NAME
Ops Manager Director	1	16384	20480	c4.large	

Step 7: Complete the Ops Manager Director Installation

1. Return to the **Installation Dashboard**.
2. Click **Apply Changes**. An ICMP error message appears.

3. Click **Ignore errors and start the install**.
Ops Manager Director begins to install. The image shows the **Changes Applied** window that displays when the installation process successfully completes.

Step 8: Configure Elastic Runtime for AWS

After you complete this procedure, follow the instructions in the [Configuring Elastic Runtime for AWS](#) topic.

Installing Pivotal Cloud Foundry on OpenStack

This guide describes how to install [Pivotal Cloud Foundry](#) on OpenStack.

Complete the following procedures to install PCF on OpenStack:

1. [Provisioning the OpenStack Infrastructure](#)
2. [Configuring Ops Manager Director after Deploying PCF on OpenStack](#)
3. [Installing Elastic Runtime after Deploying PCF on OpenStack](#)

Supported Versions

Pivotal's automated testing environments currently run on Mirantis OpenStack 6.1 (Juno). Pivotal Cloud Foundry has also been installed on OpenStack releases and distributions based on Havana, Icehouse, Juno, and Kilo from different vendors including SUSE, Red Hat, EMC, and Canonical. The nature of OpenStack as a collection of interoperable components requires OpenStack expertise to troubleshoot issues that may occur when installing Pivotal Cloud Foundry on a particular releases and distributions.

Prerequisites

In order to deploy Pivotal Cloud Foundry on OpenStack, you must have a dedicated OpenStack tenant (formerly known as an OpenStack project) that meets the following requirements.

- You must have keystone credentials for the OpenStack tenant, including:
 - Auth URL
 - API key
 - Username
 - Project name
 - Region
 - SSL certificate for your wildcard domain (see below).
- The following must be enabled for the tenant:
 - The ability to upload custom images to [Glance](#)
 - The ability to create and modify VM flavors. See the [VM flavor configuration table](#).
 - DHCP
 - The ability to allocate floating IPs.
 - The ability for VMs inside a tenant to send messages via the floating IP.
 - Permissions for VMs to boot directly from image.
 - One wildcard DNS domain. Pivotal recommends using two wildcard domains if system and apps need to be separated.
- Your OpenStack tenant must have the following resources before you install Pivotal Cloud Foundry:
 - 70 GB of RAM
 - 22 available instances
 - 16 small VMs (1 vCPU, 1024 MB of RAM, 10 GB of root disk)
 - 3 large VMs (4 vCPU, 16384 MB of RAM, 10 GB root disk)
 - 32 vCPUs
 - 1 TB of storage
 - Neutron networking with floating IP support
- Requirements for your Cinder back end:
 - PCF requires RAW root disk images. The Cinder back end for your OpenStack tenant must support RAW.
 - Pivotal recommends that you use a Cinder back end that supports snapshots. This is required for some BOSH functionalities.
 - Pivotal recommends enabling your Cinder back end to delete block storage asynchronously. If this is not possible, it

 **Note:** It is possible to avoid using wildcard DNS domains by using a service such as xip.io. However, this option requires granting external internet access from inside VMs.

Copyright Pivotal Software Inc, 2013-2015

must be able to delete multiple 20BG volumes within 300 seconds.

- Miscellaneous:

- If any overlay network is being used with VXLAN or GRE protocols, the MTU of the created VMs must be adjusted to the best practices recommended by the plugin vendor (if any). If Neutron is configured with VXLAN via the Open vSwitch mechanism, the MTU should be 1400. For GRE, the recommended number is 1460. DHCP must be enabled in the internal network for the MTU to be assigned to the VMs automatically.
- Pivotal recommends granting complete access to the OpenStack logs to the operator managing the installation process.
- Your OpenStack environment should be thoroughly tested and considered stable before deploying PCF.

Configure your OpenStack VM flavors as follows:

ID	Name	Memory_MB	Disk	Ephemeral	VCPUs
1	m1.small	2048	20	0	1
2	m1.medium	4096	40	0	2
3	m1.large	8192	80	0	4
4	m1.xlarge	16384	160	0	8

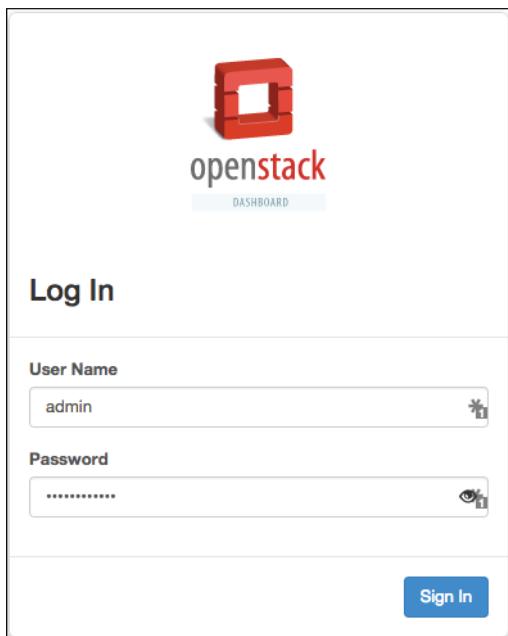
Provisioning the OpenStack Infrastructure

This guide describes how to provision the OpenStack infrastructure that you need to install [Pivotal Cloud Foundry](#) on Mirantis Express hosted OpenStack. Use this topic when [Installing Pivotal Cloud Foundry on OpenStack](#).

After completing this procedure, complete all of the steps in the [Configuring Ops Manager Director after Deploying PCF on OpenStack](#) and [Installing Elastic Runtime after Deploying PCF on OpenStack](#) topics.

Step 1: Log In to the OpenStack Horizon Dashboard

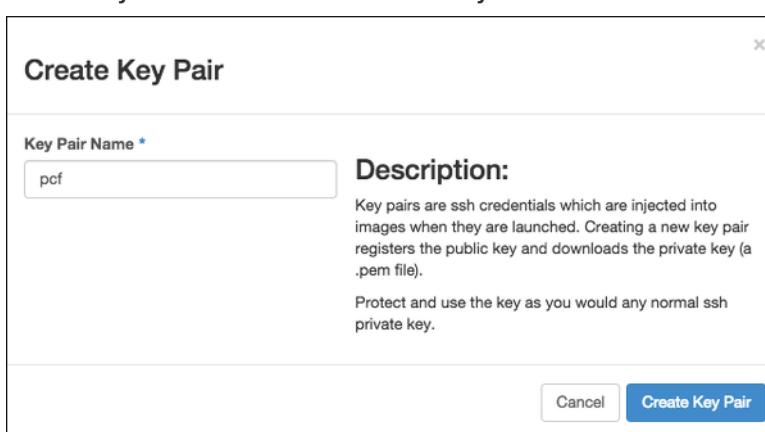
1. Log in to the OpenStack Horizon Dashboard.



The screenshot shows the OpenStack Horizon Dashboard log-in interface. At the top is a red OpenStack logo with the word "openstack" below it. Below the logo is a "DASHBOARD" link. The main area is titled "Log In". It contains two input fields: "User Name" with "admin" typed in and a required indicator (*), and "Password" with several dots and a visibility icon. At the bottom is a blue "Sign In" button.

Step 2: Configure Security

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**.
2. Select the **Key Pairs** tab on the **Access & Security** page.
3. Click **Create Key Pair**.
4. Enter a **Key Pair Name** and click **Create Key Pair**.



The screenshot shows a modal dialog titled "Create Key Pair". It has a "Key Pair Name" field containing "pcf" and a "Description" text area with the following text:
Key pairs are ssh credentials which are injected into images when they are launched. Creating a new key pair registers the public key and downloads the private key (a .pem file).
Protect and use the key as you would any normal ssh private key.
At the bottom are "Cancel" and "Create Key Pair" buttons.

5. In the left navigation, click **Access & Security** to refresh the page.
6. Select the **Security Groups** tab. Click **Create Security Group** and create a group with the following properties:

- **Name:**
- **Description:**

Create Security Group

Name *
opsmanager

Description *
Ops Manager

Description:
Security groups are sets of IP filter rules that are applied to the network settings for the VM. After the security group is created, you can add rules to the security group.

Create Security Group

7. Select the checkbox for the opsmanager Security Group and click **Manage Rules**.

<input type="checkbox"/>	Name	Description	Actions
<input type="checkbox"/>	default	default	<button>Manage Rules</button>
<input checked="" type="checkbox"/>	opsmanager	Ops Manager	<button>Manage Rules</button> ▾

8. Add the access rules for HTTP, HTTPS, and SSH as shown in the table below. The rules with 'opsmanager' in the Remote column have restricted access to that particular Security Group.

Note: Adjust the remote sources as necessary for your own security compliance. Pivotal recommends limiting remote access to Ops Manager to IP ranges within your organization.

Direction	Ether Type	IP Protocol	Port Range	Remote
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	25555	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	1-65535	opsmanager
Ingress	IPv4	UDP	1-65535	opsmanager

Step 3: Create Ops Manager Image

You can create the Ops Manager image in OpenStack using the OpenStack GUI or using the [Glance CLI](#) client.

Note: If your Horizon Dashboard does not support file uploads, you must use the Glance client.

OpenStack GUI

1. Download the **Pivotal Cloud Foundry Ops Manager for OpenStack** image file from [Pivotal Network](#).
2. In the left navigation of your OpenStack dashboard, click **Project > Compute > Images**.
3. Click **Create Image**. Complete the **Create An Image** page with the following information:
 - **Name:** Enter .
 - **Image Source:** Select **Image File**.

- **Image File:** Click **Choose File**. Browse to and select the image file that you downloaded from [Pivotal Network](#).
- **Format:** Select **Raw**.
- **Minimum Disk (GB):** Enter **40**.
- **Minimum RAM (MB):** Enter **4096**.
- Deselect the **Public** checkbox.
- Select the **Protected** checkbox.

4. Click **Create Image**.

Name *
Ops Manager

Description:
Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)
Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

Image Source
Image File

Image File ⓘ
Choose File vm-2015-05-...14fa18e.raw

Format *
Raw

Architecture

Minimum Disk (GB) ⓘ
40

Minimum RAM (MB) ⓘ
4096

Public
 Protected

Create Image

Glance CLI

1. Download the **Pivotal Cloud Foundry Ops Manager for OpenStack** image file from [Pivotal Network](#).

2. In a terminal window, run the following command to install the Glance CLI client:

```
$ apt-get install python-glanceclient
```

3. Run `admin-openrc.sh` to download your `openstack.rc` file and target your OpenStack environment.

```
$ ./admin-openrc.sh
Please enter your OpenStack Password:
```

4. Run the following command to use the Glance CLI client to upload the image file that you downloaded from [Pivotal Network](#):

```
$ glance image-create --progress --disk-format raw --name "Ops Manager" --container-format bare --file PATH/DOWNLOADED-
```

Step 4: Launch Ops Manager VM

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Images**.

2. Click **Launch**.

Images									
	Image Name	Type	Status	Public	Protected	Format	Size	Actions	
<input checked="" type="checkbox"/>	Ops Manager	Image	Active	No	No	RAW	3.0 GB	<button>Launch</button> <button>▼</button>	
<input type="checkbox"/>	DELL Reference 2.0	Image	Active	All	All	RAW	4.0 GB	<button>Launch</button>	

3. Complete the **Details**, **Access & Security**, and **Networking** tabs of the **Launch Instance** form with the information below.

Details Tab

Select the **Details** tab and specify the following details:

- **Availability Zone:** Use the drop-down menu to select an availability zone. You use this availability zone when you [Complete the Availability Zones Pages](#) when [Configuring Ops Manager Director](#).
- **Instance Name:** Enter `Ops Manager`.
- **Flavor:** Select `m1.large`.
- **Instance Count:** Do not change from the default.
- **Instance Boot Source:** Select `Boot from image`.
- **Image Name:** Select the `Ops Manager` image.

Launch Instance

Details *	Access & Security *	Networking *	Post-Creation	Advanced Options												
Availability Zone <input type="text" value="nova"/>	Specify the details for launching an instance. The chart below shows the resources used by this project in relation to the project's quotas.															
Instance Name * <input type="text" value="Ops Manager"/>	Flavor Details <table border="1"> <tr> <td>Name</td> <td>m1.large</td> </tr> <tr> <td>VCPUs</td> <td>4</td> </tr> <tr> <td>Root Disk</td> <td>80 GB</td> </tr> <tr> <td>Ephemeral Disk</td> <td>0 GB</td> </tr> <tr> <td>Total Disk</td> <td>80 GB</td> </tr> <tr> <td>RAM</td> <td>8,192 MB</td> </tr> </table>				Name	m1.large	VCPUs	4	Root Disk	80 GB	Ephemeral Disk	0 GB	Total Disk	80 GB	RAM	8,192 MB
Name	m1.large															
VCPUs	4															
Root Disk	80 GB															
Ephemeral Disk	0 GB															
Total Disk	80 GB															
RAM	8,192 MB															
Instance Count * ⓘ <input type="text" value="1"/>	Project Limits <table border="1"> <tr> <td>Number of Instances</td> <td>0 of No Limit Used</td> </tr> <tr> <td>Number of VCPUs</td> <td>0 of No Limit Used</td> </tr> <tr> <td>Total RAM</td> <td>0 of No Limit MB Used</td> </tr> </table>				Number of Instances	0 of No Limit Used	Number of VCPUs	0 of No Limit Used	Total RAM	0 of No Limit MB Used						
Number of Instances	0 of No Limit Used															
Number of VCPUs	0 of No Limit Used															
Total RAM	0 of No Limit MB Used															
Instance Boot Source * ⓘ <input type="text" value="Boot from image"/>																
Image Name <input type="text" value="Ops Manager (3.0 GB)"/>																
<input type="button" value="Cancel"/> <input type="button" value="Launch"/>																

Access & Security Tab

Select the **Access & Security** tab and specify the following details:

- **Key Pair:** Select the key pair that you created in [Step 2: Configure Security](#). You need this key pair to log in to the Ops Manager instance from your workstation.
- **Security Groups:** Select the `opsmanager` checkbox. Deselect all other Security Groups.

Launch Instance

Details *		Access & Security *	Networking *	Post-Creation	Advanced Options
Key Pair <small>?</small> <input type="text" value="pcf"/> <input type="button" value="+"/>		Control access to your instance via key pairs, security groups, and other mechanisms.			
Security Groups <small>?</small> <input type="checkbox"/> default <input checked="" type="checkbox"/> opsmanager					
<input type="button" value="Cancel"/> <input type="button" value="Launch"/>					

Networking Tab

1. Select the **Networking** tab.
2. Under **Available networks**, select a private subnet. You add a Floating IP to this network in a later step.
3. Click **Launch**.

Launch Instance

Details *		Access & Security *	Networking *	Post-Creation	Advanced Options
Selected networks <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">NIC:1: net04 (fb1dd3e-c20b-42d5-8949-6994c09b11a) <input type="button" value="+"/></div>		Choose network from Available networks to Selected networks by push button or drag and drop, you may change NIC order by drag and drop as well.			
Available networks <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">net04_ext (fa35971-51bb-4d0f-83aa-eac3332cc06a) <input type="button" value="+"/></div>					
<input type="button" value="Cancel"/> <input type="button" value="Launch"/>					

Step 5: Associate a Floating IP Address

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Instances**.
2. Wait until the **Power State** of the Ops Manager instances shows as *Running*.
3. Record the private **IP Address** of the Ops Manager instance. You use this IP Address when you [Complete the Networks Pages](#) in Ops Manager.

Instance Name	Image Name	IP Address	Size	Key Pair	Power State	Actions
██████████	██████████	██████████	m1.medium	miran	Running	<input type="button" value="Associate Floating IP"/>
Ops Manager	Ops Manager	192.168.111.54	m1.large	pcf	Running	<input type="button" value="Associate Floating IP"/>

4. Select the **Ops Manager** checkbox. Click the **Actions** drop-down menu and select **Associate Floating IP**.
5. Under IP Address, click **+**.

Manage Floating IP Associations

IP Address *

Select an IP address

Port to be associated *

Select a port

Select the IP address you wish to associate with the selected instance.

Cancel Associate

- Under **Pool**, select an IP Pool and click **Allocate IP**.

Pool *

net04_ext

Description:

Allocate a floating IP from a given floating IP pool.

Project Quotas

Floating IP (5) 45 Available

Cancel Allocate IP

- Under **Port to be associated**, select your **Ops Manager** instance. Click **Associate**.

IP Address *

216.221.229.32

Port to be associated *

Ops Manager: 192.168.111.54

Select the IP address you wish to associate with the selected instance.

Cancel Associate

Step 6: Add Blob Storage

- In the left navigation of your OpenStack dashboard, click **Project > Object Store > Containers**.
- Click **Create Container**. Create a container with the following properties:
 - Container Name:** Enter `pcf`.
 - Container Access:** Select **private**.

Create Container

Container Name *

Container Access *

Description:

A container is a storage compartment for your data and provides a way for you to organize your data. You can think of a container as a folder in Windows ® or a directory in UNIX ®. The primary difference between a container and these other file system concepts is that containers cannot be nested. You can, however, create an unlimited number of containers within your account. Data must be stored in a container so you must have at least one container defined in your account prior to uploading data.

Note: A Public Container will allow anyone with the Public URL to gain access to your objects in the container.

Return to [Installing Pivotal Cloud Foundry on OpenStack](#).

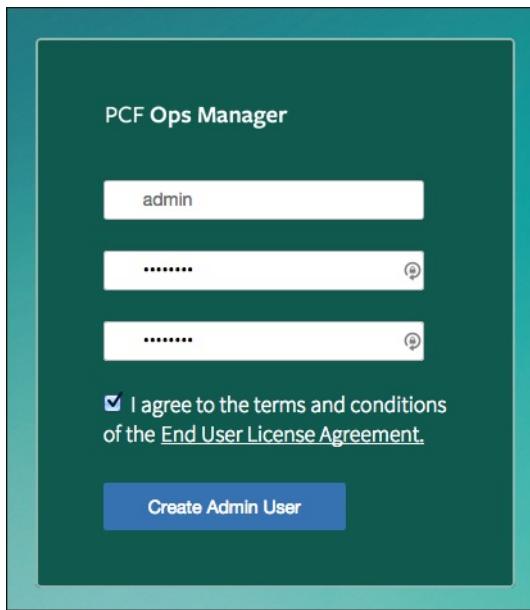
Configuring Ops Manager Director after Deploying PCF on OpenStack

This topic describes how to configure the Ops Manager Director after deploying [Pivotal Cloud Foundry](#) (PCF) on OpenStack. Use this topic when [Installing Pivotal Cloud Foundry on OpenStack](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Provisioning the OpenStack Infrastructure](#) topic. After you complete this procedure, follow the instructions in the [Installing Elastic Runtime after Deploying PCF on OpenStack](#) topic.

Step 1: Log in to Ops Manager

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Instances** to view the public IP address listed for Ops Manager.
2. Use the public IP address of your Ops Manager instance to launch Ops Manager in a browser.
3. Enter a username and password, and accept the EULA to create an admin user.



4. Enter a **Username** and a **Password**.
5. Read the **End User License Agreement**, and select the checkbox to accept.
6. Click **Create Admin User** to create an Ops Manager administrator account.

Step 2: Complete the OpenStack Config Page

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**. Select the **API Access** tab.
2. Record the **Service Endpoint** for the **Identity** service. You use this Service Endpoint as the Authentication URL for Ops Manager in a later step.

Access & Security	
Security Groups Key Pairs Floating IPs API Access	
API Endpoints	
Download OpenStack RC File Download EC	
Service	Service Endpoint
Compute	http://2...:8774/v2.1
Network	http://2...:8775/
Volumev2	http://2...:8776/v2.1
S3	http://2...:80
Image	http://2...:8080
Cloudformation	http://2...:8080/v1/
Volume	http://2...:8772/v2.1
EC2	http://2...:8773/services/Cloud
Orchestration	http://2...:8774/v2.1
Object Store	http://2...:8775/v2.1
Identity	http://2...:5000/v2.0
Displaying 11 items	

3. In your browser, click the **Ops Manager Director** tile.



4. Select **OpenStack Config**.

5. Complete the **OpenStack Management Console Config** page with the following information:

Openstack Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

VM Passwords

Resource Config

Openstack Management Console Config

Authentication URL*

Username*

Password*

Tenant*

Region*

Ignore Server Availability Zone
Checking this box will set the volume AZ to the default AZ.

Security Group Name

Key Pair Name*

SSH Private Key*

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAE53+T5BRrrooGXpm7
TgjTgo0hz0bGquhr0ktJZuBrKqav21xPR
0h8dnVYT54TJysgTTTfTKNjm3eP88kMJ/
67NGMMFNFK824H4H11qH9VBsjMSx
-----
```

Save

- **Authentication URL:** Enter the Service Endpoint for the Identity service that you recorded in a previous step.
- **Username:** Enter your OpenStack Horizon username.
- **Password:** Enter your OpenStack Horizon password.
- **Tenant:** Enter your OpenStack tenant name
- **Region:** Enter `RegionOne`, or another region if recommended by your OpenStack administrator.
- **Ignore Server Availability Zone:** Do not select the checkbox.
- **Security Group Name:** Enter `opsmanager`. You created this Security Group when [Provisioning the OpenStack Infrastructure](#).
- **Key Pair Name:** Enter the name of the key pair that you created in the [Configure Security](#) step of the [Provisioning the OpenStack Infrastructure](#) topic.
- **SSH Private Key:** In a text editor, open the key pair file that you downloaded in the [Configure Security](#) step of the [Provisioning the OpenStack Infrastructure](#) topic. Copy and paste the contents of the key pair file into the SSH Private Key field.

6. Click **Save**.

Step 3: (Optional) Complete the Advanced Config Page

1. In Ops Manager, select **Advanced Infrastructure Config**.
2. If your OpenStack environment requires specific connection options, enter them in the **Connection Options** field in JSON format. For example:

```
'connection_options' => { 'read_timeout' => 200 }
```

Note: This is an advanced option. Most users leave this field blank.

3. Click **Save**.

Step 4: Complete the Director Config Page

1. In the left navigation of your OpenStack dashboard, click **Project > Compute > Access & Security**. Select the **API Access** tab.

Access & Security

Security Groups Key Pairs Floating IPs API Access

API Endpoints

Download OpenStack RC File Download EC2 Credentials View Credentials

Service	Service Endpoint
Compute	http://205.16.10.102
Network	http://205.16.10.103/

2. Click **Download EC2 Credentials**.
3. Unzip the downloaded credentials. In a text editor, open the `ec2rc.sh` file. You use the contents of this file to complete the Ops Manager **Director Config** page.
4. In Ops Manager, select **Director Config**.

- Openstack Config
- Director Config
- Create Availability Zones
- Assign Availability Zones
- Create Networks
- Assign Networks
- VM Passwords
- Resource Config

Director Config

NTP Servers (comma delimited)*
 [x]

Metrics IP Address

Enable VM Resurrector Plugin

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*

Bucket Name*

Access Key*

Secret Key*
 [x]

Database Location
 Internal
 External MySQL Database

Host*

Port*

Username*

Password*
 [x]

Database*

Max Threads

Save

5. Enter one or more of the following NTP servers in the **NTP Servers (comma delimited)** field:
- 0.amazon.pool.ntp.org
 - 1.amazon.pool.ntp.org

- 2.amazon.pool.ntp.org
 - 3.amazon.pool.ntp.org
6. Complete the **Director Config** page using information from the `ec2rc.sh` file:
- **Blobstore Location:** Select the **S3 Compatible Blobstore** option.
 - **S3 Endpoint:** Use **S3_URL** from the `ec2rc.sh` file.
 - **Bucket Name:** Enter `pcf`.
 - **Access Key:** Use **EC2_ACCESS_KEY** from the `ec2rc.sh` file.
 - **Secret Key:** Use **EC2_SECRET_KEY** from the `ec2rc.sh` file.
 - **Max Threads:** Enter `1`.
7. Click **Save**.

Step 5: Complete the Availability Zones Pages

1. In Ops Manager, select **Create Availability Zones**.

2. Enter the name of the **Availability Zone** that you selected when [Provisioning the OpenStack Infrastructure](#).
3. Click **Save**.
4. Select **Assign Availability Zones**.

5. From the **Singleton Availability Zone** drop-down menu, select the availability zone that you just created.
6. Click **Save**.

Step 6: Complete the Networks Pages

1. In the left navigation of your OpenStack dashboard, click **Project > Network > Networks**.

	Name	Subnets Associated	Shared	Status	Admin State	Actions
<input type="checkbox"/>	net04_ext	net04_ext_subnet 192.168.111.0/24	No	ACTIVE	UP	<button>Edit Network</button>
<input type="checkbox"/>	net04	net04_subnet 192.168.111.0/24	No	ACTIVE	UP	<button>Edit Network</button>

2. Click the name of the network that contains the private subnet where you deployed the Ops Manager VM. The OpenStack Network Detail page displays your network settings.

Network Overview

Name	net04
ID	8db1dc3e- XXXXXXXXXX @b11a
Project ID	fe XXXXXXXXXX 32
Status	ACTIVE
Admin State	UP
Shared	No
External Network	No
Provider Network	Network Type: vlan Physical Network: physnet2 Segmentation ID: 1003

Subnets

<input type="checkbox"/>	Name	Network Address	IP Version	Gateway IP	Actions
<input type="checkbox"/>	net04_subnet	192.168.111.0/24	IPv4	192.168.111.1	<button>Edit Subnet</button> ▾

Displaying 1 item

Ports

Name	Fixed IPs	Attached Device	Status	Admin State	Actions
(3906d7ec)	192.168.111.2	network:dhcp	ACTIVE	UP	<button>Edit Port</button>
(472a6c20)	192.168.111.1	network:router_interface	ACTIVE	UP	<button>Edit Port</button>
(67904dc6)	192.168.111.10	compute:None	ACTIVE	UP	<button>Edit Port</button>
(70b7af50)	192.168.111.54	compute:nova	ACTIVE	UP	<button>Edit Port</button>
(7232bd69)	192.168.111.3	network:dhcp	ACTIVE	UP	<button>Edit Port</button>

3. Complete the Ops Manager **Create Networks** page using information from your OpenStack network as follows:

Ops Manager Director

Settings Status Credentials

Openstack Config Create Network

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

VM Passwords

Resource Config

Name*

Network ID*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

Save

- **Name:** Enter a unique name for this network.
- **Network ID:** Use the **ID** from the OpenStack page.
- **Subnet (CIDR Range):** Use the **Network Address** from the OpenStack page.
- **Excluded IP Ranges:** Use the first 10 IP addresses of the **Network Address** Range, and the private IP Address of the Ops Manager instances that you recorded in the [Associate a Floating IP Address](#) step of the [Provisioning the OpenStack Infrastructure](#) topic.
- **DNS:** Use .
- **Gateway:** Use the **Gateway IP** from the OpenStack page.

4. Click **Save**.

5. Select **Assign Networks**.

Openstack Config Assign Networks

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

The Ops Manager director can be configured to have an IP on a network.

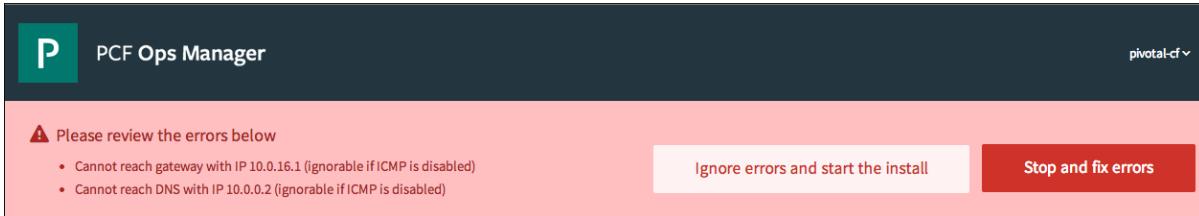
Network

Save

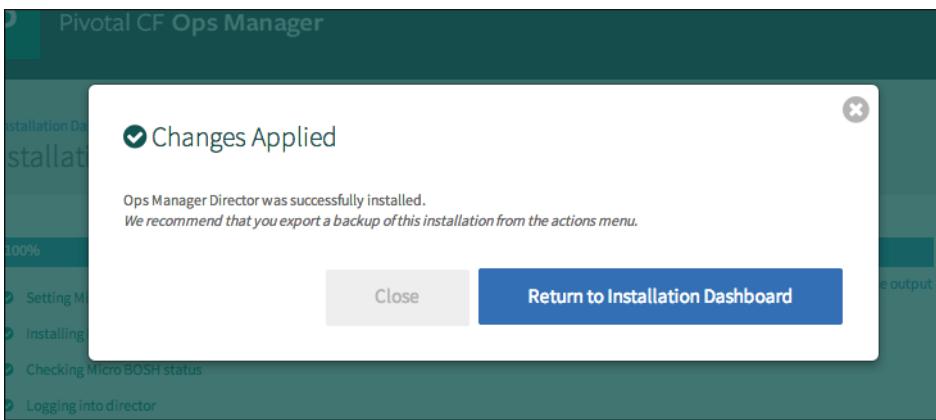
6. Select the network that you created and click **Save**.

Step 7: Complete Ops Manager Director Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.



3. Ops Manager Director installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.



Return to [Installing Pivotal Cloud Foundry on OpenStack](#).

Installing Elastic Runtime after Deploying PCF on OpenStack

This topic describes how to install and configure Elastic Runtime after deploying [Pivotal Cloud Foundry](#) (PCF) on OpenStack.

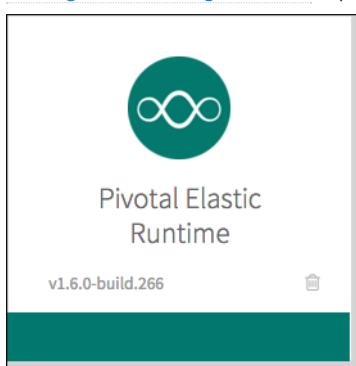
Use this topic when [Installing Pivotal Cloud Foundry on OpenStack](#).

Before beginning this procedure, ensure that you have successfully completed all steps in the [Provisioning the OpenStack Infrastructure](#) topic and the [Configuring Ops Manager Director after Deploying PCF on OpenStack](#) topics.

Note: If you are performing an upgrade from PCF 1.5 to 1.6, please review [Upgrading Ops Manager](#) for critical upgrade information.

Step 1: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the Pivotal Network link on the left to add Elastic Runtime to Ops Manager. For more information, refer to the [Adding and Deleting Products](#) topic.



Step 2: Complete Security Configuration

1. Click on **Security Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configure security settings

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIICKTCAZICCCVjs8Jtc84cTANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJV
MTUwNTAzMTkyMjEyWjBZMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExiTafBgNV
BAMTCGJhY2tldCBvZiBhbmQgcmVhZG9ubHkgQ0EwHhcNMTIwMjAxMDAwMjUwWjBZ
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC5CWbiXc86fFy4mpAf6PmxQTn36d+bj8fUDo6NZQ7B0twTW5iQ
AoGBAKsRCylqUkSi5uHwW2B09isuv1X4zTa3daoSdy6fIEfGEF5RBKAAnPrd7Xv
-----END RSA PRIVATE KEY-----
```

[Generate Self-Signed RSA Certificate](#)

Ignore SSL certificate verification

HAProxy SSL Ciphers

Router SSL Ciphers

Disable HTTP traffic to HAProxy

Enable cross-container traffic

Enable TLS on the Router

Save

2. Provide an **SSL Termination Certificate** for your SSL Termination Point.

- In a production environment, use a signed **SSL Certificate** from a known certificate authority (CA). Copy and paste the values for **Certificate PEM** and **Private Key PEM** from the signed certificate into the appropriate text fields.
- In a development or testing environment, you may use a self-signed certificate.

Note: Pivotal does not recommend using a self-signed certificate for production deployments.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

To use a self-signed certificate, follow the steps below: * Click the **Generate Self-Signed RSA Certificate** link to launch the RSA certificate wizard. * Enter your system and app domains in wildcard format. Optionally, also add any custom domains in wildcard format. The example below uses `*.example.com`.

Note: Wildcard DNS records only work for a single domain name component or component fragment. For example, `*.domain.com` works for `apps.domain.com` and `system.domain.com`, but not for `apps.domain.com` or `system.domain.com`.

Note: You can generate a single certificate for two domains separated by a comma. For example, `.apps.domain.com, *.system.domain.com`.

✓ Generate Self-Signed RSA Certificate

Example: `*.app.domain.com, *.system.domain.com, my.webapp.com, *.domain.com*`

`*.example.com`

* Click **Generate**.

* **Elastic Runtime populates**

the **SSL Certificate fields with RSA certificate and private key information.

3. Configure **Ignore SSL certificate verification**. Select this option if you are using self-signed certificates or certificates generated from Ops Manager.
4. Configure **HAProxy SSL Ciphers** and **Router SSL Ciphers**. Leave these fields blank unless you want to use a specific set of SSL ciphers for HAProxy or the Router.
5. Configure **Disable HTTP traffic to HAProxy**. If you select the **Disable HTTP traffic to HAProxy** checkbox, your deployment rejects all port 80 traffic to HAProxy. Additionally, this option sets the secure flag in the `VCAP_ID` cookie that the Router generates.

Note: If you enable this checkbox and your deployment is not using HAProxy, configure your external load balancer to reject port 80 traffic. If you do not do this, traffic to port 80 is forwarded to applications but loses session stickiness.

6. Configure **Enable cross container traffic**. By selecting this checkbox, you disable the restriction that prevents containers in the same DEA or Diego Cell from communicating with each other. Pivotal does not recommend selecting this checkbox in multi-tenant environments. You should select this option for microservices such as Pivotal Spring Cloud.
7. Configure **Enable TLS on the Router**. Selecting this enables [SSL termination](#) on the Router.
8. Click **Save**.

Step 3: Complete the Cloud Controller Page

1. Select **Cloud Controller** and enter your system and application domains.

<input checked="" type="checkbox"/> Assign Networks <input checked="" type="checkbox"/> Assign Availability Zones <input checked="" type="checkbox"/> System Database Config <input checked="" type="checkbox"/> File Storage Config <input checked="" type="checkbox"/> IPs and Ports <input checked="" type="checkbox"/> Security Config <input checked="" type="checkbox"/> MySQL Proxy Config <input type="checkbox"/> Cloud Controller <input checked="" type="checkbox"/> External Endpoints <input checked="" type="checkbox"/> SSO Config <input checked="" type="checkbox"/> LDAP Config <input checked="" type="checkbox"/> SMTP Config <input checked="" type="checkbox"/> Errands <input checked="" type="checkbox"/> Resource Config <input checked="" type="checkbox"/> Stemcell	<p>Coordinates Pivotal CF Elastic Runtime application lifecycles</p> <p>System Domain * system.example.com <input type="button" value=""/></p> <p>Apps Domain * apps.example.com</p> <p>Cloud Controller DB Encryption Key Secret <input type="button" value=""/></p> <p>Maximum File Upload Size (MB) (min: 1024, max: 2048) * 1024</p> <p><input type="checkbox"/> Disable Custom Buildpacks</p> <p>Default Quota App Memory (MB) (min: 10240, max: 102400) * 10240</p> <p>Default Quota Service Instances (min: 0, max: 1000) * 100</p> <p><input type="checkbox"/> Enable Diego Docker support</p> <p>Save</p>
--	---

- The **System Domain** defines your target when you push apps to Elastic Runtime.
- The **Apps Domain** defines where Elastic Runtime should serve your apps.

Note: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes. For example, name your system domain `system.EXAMPLE.COM` and your apps domain `apps.EXAMPLE.COM`.

2. Click **Save**.

Step 4: (Optional) Configure SMTP

Elastic Runtime uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the **Configuration for SMTP** page if you want to enable end-user self-registration.

1. Select **SMTP Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configuration for Simple Mail Transfer Protocol

From Email

Address of SMTP Server

Port of SMTP Server

SMTP Server Credentials

Username

Password

SMTP Enable Automatic STARTTLS

SMTP Authentication Mechanism*

SMTP CRAMMD5 secret

2. Enter your reply-to and SMTP email information.
3. For **SMTP Authentication Mechanism**, select .
4. Click **Save**.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See [Creating and Managing Users with the cf CLI](#) for more information.

Step 5: Configure Diego

In the PCF 1.6 release, Diego replaces the [DEAs](#) and the [Health Manager](#). Diego is installed and enabled in PCF 1.6 by default.

For more information about Diego, see the [Diego Architecture](#) and [Diego Components](#) topics.

Note: Before you install PCF 1.6, you must uninstall any Diego Beta installations. For more information on upgrading to PCF 1.6, see [Upgrading Operations Manager](#).

Pivotal Elastic Runtime

- Settings
- Status
- Credentials
- Logs

Assign Networks Diego is a new replacement for DEAs.

Assign Availability Zones Use Diego by default instead of DEAs

System Database Config Allow SSH access to apps

File Storage Config Application Containers Subnet Pool *

IPs and Ports 10.254.0.0/22

Security Config **Save**

MySQL Proxy Config

Cloud Controller

System Logging

SSO Config

LDAP Config

SMTP Config

Diego

1. Select **Diego**.
2. Select the **Use Diego by default instead of DEAs** checkbox to ensure that all applications pushed to your PCF deployment use the Diego container management system. For new installations, this option is selected by default. If you are upgrading to PCF 1.6, you must select this option to automatically enable Diego for newly pushed applications.

Note: If you do not select this option, application developers must explicitly target Diego when pushing their applications.
3. (Optional) Select the **Allow SSH access to apps** checkbox to allow application developers to `ssh` directly into their application instances on Diego. See the [Diego-SSH Overview](#) topic for more details.

Step 6: (Optional) Enable Experimental Features

Use caution when enabling experimental features if you have other Pivotal Cloud Foundry service tiles installed in your Pivotal Cloud Foundry deployment. Not all of the services are guaranteed to work as expected with these features enabled.

This release does not contain any experimental features.

Step 7: Enable Traffic to Private Subnet

Unless you are using your own load balancer, you must enable traffic flow to the OpenStack private subnet as follows. Give each HAProxy a way of routing traffic into the private subnet by providing public IPs as floating IPs.

1. Click **Resource Config**.

Resource Config					
	JOB	INSTANCES	PERSISTENT DISK (MB)	INSTANCE TYPE	FLOATING IPs
✓ Assign Networks	NATS	1	0	m1.small	
✓ Assign Availability Zones	consul	1	1024	m1.small	
✓ System Database Config	etcd	1	1024	m1.small	
✓ File Storage Config	NFS Server	1	102400	m1.small	
✓ IPs and Ports	Cloud Controller Database	1	2048	m1.small	
✓ Security Config	UAA Database	1	8192	m1.small	
✓ MySQL Proxy Config	Apps Manager Database	1	1024	m1.small	
✓ Cloud Controller	Cloud Controller	1	0	m1.medium	
✓ HAProxy	HAProxy	1	0	m1.small	23.16.33.43
✓ Router	Router	1	0	m1.small	
✓ External Endpoints	Health Manager	1	0	m1.small	
✓ Clock Global	Clock Global	1	0	m1.small	
✓ SSO Config	Cloud Controller Worker	1	0	m1.small	
✓ LDAP Config	Collector	0	0	m1.small	
✓ UAA	UAA	1	0	m1.small	
✓ SMTP Config	MySQL Proxy	1	0	m1.small	
✓ MySQL Server	MySQL Server	1	109000	m1.large	
✓ Experimental Features	DEA	1	0	m1.xlarge	
✓ Doppler Server	Doppler Server	1	0	m1.small	
✓ Loggregator Trafficcontroller	Loggregator Trafficcontroller	1	0	m1.small	
✓ Push Apps Manager	Push Apps Manager	1	0	m1.small	
✓ Push App Usage Service	Push App Usage Service	1	0	m1.small	
✓ Stemcell	Run Smoke Tests	1	0	m1.small	

2. Enter one or more IP addresses in **Floating IPs** for each HAProxy.

3. Click **Save**.

Refer to the [Configuring PCF SSL Termination](#) topic for more information about configuring traffic depending on your load balancer.

Step 8: (Optional) Disable Unused Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

Resource Config			
JOB	INSTANCES	EPHEMERAL DISK (MB)	
NATS	1	2048	
consul	1	2048	
etcd	1	2048	
Diego etcd	1	2048	
NFS Server	0	2048	
Router	1	2048	
MySQL Proxy	0	4096	
MySQL Server	0	30000	
Cloud Controller Database (Postgres)	0	2048	
UAA Database (Postgres)	0	2048	
Apps Manager Database (Postgres)	0	2048	
Cloud Controller	1	20480	
HAProxy	0	2048	

2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **NFS Server:** Enter 0 in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy:** Enter 0 in **Instances**.
 - **MySQL Server:** Enter 0 in **Instances**.
 - **Cloud Controller Database:** Enter 0 in **Instances**.
 - **UAA Database:** Enter 0 in **Instances**.
 - **Apps Manager Database:** Enter 0 in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter 0 in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 9: Complete Elastic Runtime Installation

1. Click the **Installation Dashboard** link to return to the Installation Dashboard.
2. Click **Apply Changes**. If the following ICMP error message appears, click **Ignore errors and start the install**.

The screenshot shows the PCF Ops Manager interface. At the top, there's a navigation bar with a 'P' icon and the text 'PCF Ops Manager'. On the right, it says 'pivotal-cf'. Below the bar, a red alert box contains the text: '⚠ Please review the errors below' followed by two bullet points: '• Cannot reach gateway with IP 10.0.16.1 (ignorable if ICMP is disabled)' and '• Cannot reach DNS with IP 10.0.0.2 (ignorable if ICMP is disabled)'. To the right of the alert are two buttons: 'Ignore errors and start the install' (in white) and 'Stop and fix errors' (in red).

3. Elastic Runtime installs. The image shows the **Changes Applied** message that Ops Manager displays when the installation process successfully completes.

The screenshot shows the Pivotal CF Ops Manager interface. The main area has a green header with the text 'Pivotal CF Ops Manager'. Below it, a modal window is open with the title 'Changes Applied'. Inside the modal, there's a checkmark icon and the text 'Ops Manager Director was successfully installed. We recommend that you export a backup of this installation from the actions menu.' At the bottom of the modal are two buttons: 'Close' (gray) and 'Return to Installation Dashboard' (blue). In the background, there's a progress bar at 100% completion and a list of tasks: 'Setting Micro BOSH', 'Installing', 'Checking Micro BOSH status', and 'Logging into director'. The 'Installing' task is currently highlighted.

Return to [Installing Pivotal Cloud Foundry on OpenStack](#).

Installing Pivotal Cloud Foundry on vSphere and vCloud Air

This guide describes how to install Pivotal Cloud Foundry on vSphere and vCloud Air. It will walk you through how to deploy the installation virtual machine known as Ops Manager, configure your vSphere settings, and install PCF Elastic Runtime.

If you experience a problem while following the steps below, check the [Known Issues](#), or refer to the [PCF Troubleshooting Guide](#). Once you have completed the steps in this guide, explore the documentation on [docs.pivotal.io](#) to learn more about [Pivotal Cloud Foundry](#) (PCF) and the Pivotal product suite.

Note: If you are performing an upgrade from PCF 1.5 to 1.6, please review [Upgrading Ops Manager](#) for critical upgrade information.

Step 1: Confirm System Requirements

Before you begin your PCF deployment, ensure that your system meets the minimum requirements. The requirements include capacity for the virtual machines necessary for the deployment, a supported version of the cf command line interface tool, and certain user privileges. Refer to the [Prerequisites to Deploying Operations Manager and Elastic Runtime](#) topic for the complete list.

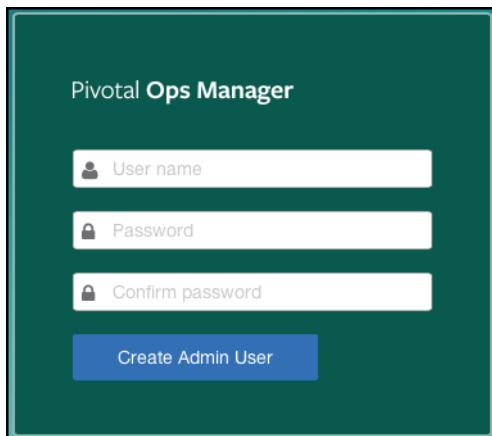
Step 2: Deploy PCF Ops Manager

1. Log in to [Pivotal Network](#) and download the Ops Manager virtual appliance file that matches your infrastructure.
2. Deploy Ops Manager using the procedure that corresponds to the VMware environment that you use:
 - [Deploying Operations Manager to vSphere](#)
 - [Deploying Operations Manager to vCloud Air and vCloud](#)

Step 3: Configure and Install PCF

Goal: Configure and install Ops Manager Director (included), Elastic Runtime, and MySQL for PCF. Skip the import steps for Elastic Runtime or MySQL for PCF if you do not want to import these products.

1. Browse to the interface IP address that you specified in Step 2.



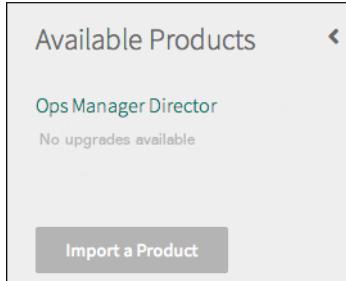
2. Create a **User name** and **Password** and log in to access the interface.

Note: On your first login attempt, an error message that the connection is untrusted appears because you are attempting to connect securely to a website with a self-signed certificate. Add Ops Manager as an exception to bypass this message on subsequent logins.

Import Products

1. Download Elastic Runtime and MySQL for PCF at [Pivotal Network](#).
 - Elastic Runtime: Click the **Pivotal Cloud Foundry** banner to access the PCF product page. Use the drop-down menu to select an Elastic Runtime release.
 - MySQL for PCF: Select the **All Products** tab to view available products. Search for “MySQL for PCF” and select a release.

2. From the Available Products view, click **Import a Product**.



3. Select the Elastic Runtime .pivotal file that you downloaded from Pivotal Network and click **Open**. After the import completes, Elastic Runtime appears in the Available Products view.
4. Repeat the previous step for MySQL for PCF.
5. In the Available Products view, hover over Elastic Runtime and click **Add**. Repeat this step for MySQL for PCF.

Configure Ops Manager Director

Your Ops Manager download includes a tile for the version of Ops Manager Director that corresponds to your IaaS. Refer to one of the following topics for help configuring your Ops Manager Director product tile:

- [Configuring Ops Manager Director for VMware vSphere](#)
- [Configuring Ops Manager Director for vCloud Air and vCloud](#)

Configure Elastic Runtime

See [Configuring Elastic Runtime for vSphere and vCloud Air](#).

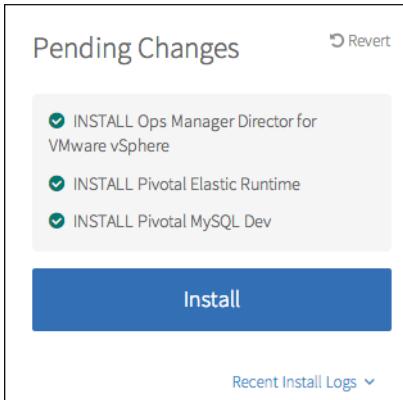
Configure MySQL

1. Click the **MySQL Dev** tile.
2. Configure the MySQL Service Plan, Lifecycle Errands, and resource sizes. Assign Availability Zones and an Ops Manager network, then click **Save**.

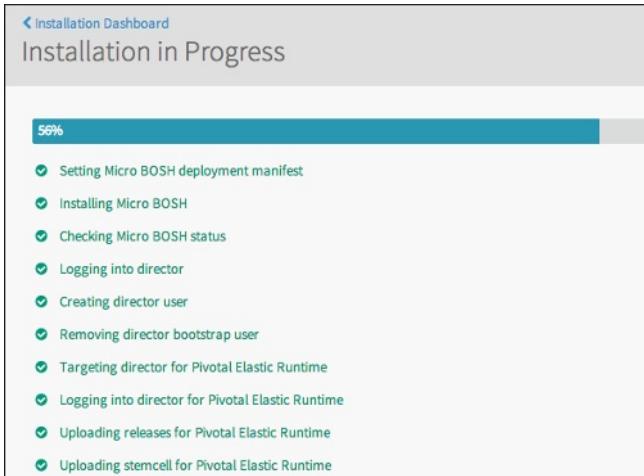
3. Click the **Installation Dashboard** link to return to the Installation Dashboard.

Install Products

1. Click **Install**.



2. Your updated PCF installation begins deploying.



3. When the deployment is finished, a success message appears.

Note: On the recommended hardware infrastructure, deployment should take less than one hour and require no user intervention.

You now have a fully functional installation of PCF and Pivotal MySQL Service.

Step 4: Create New User Accounts

Once you have successfully deployed PCF, add users to your account. Refer to the [Creating New Elastic Runtime User Accounts](#) topic for more information.

Step 5: Target Elastic Runtime

The next step is to use the cf CLI tool to target your Elastic Runtime installation. Make sure that you have [installed the cf CLI tool](#). Refer to the PCF documentation for more information about [using the cf command line tool](#).

Note: In Pivotal Ops Manager, refer to **Elastic Runtime > Credentials** for the UAA admin name and password. You can also use the user that you created in Apps Manager, or create another user with the `create-user` command.

Understanding Availability Zones in VMware Installations

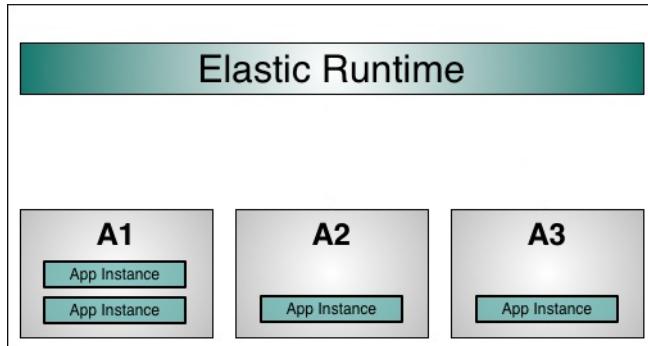
Pivotal defines an Availability Zone (AZ) as a functionally independent segment of network infrastructure that you assign. In cases of partial infrastructure failure, [Pivotal Cloud Foundry \(PCF\) Elastic Runtime](#) distributes and balances all instances of running applications across remaining AZs. Strategic use of Availability Zones contributes to the fault tolerance and high availability of a Elastic Runtime deployment.

AZ Differences Between VMware Products

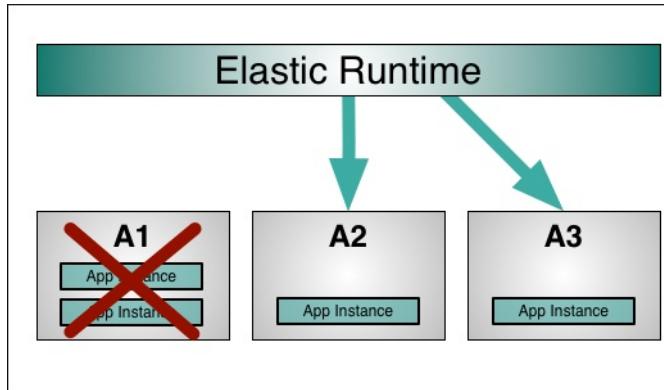
Elastic Runtime on VMware vSphere supports distributing deployments across multiple AZs. See the section on AZs in [Configuring Ops Manager Director for VMware vSphere](#). VMware vCloud and vCloud Air support only one AZ, which is the virtual data center you access with [vShield Edge Gateway Services interface](#).

Balancing Across AZs During Failure: Example Scenario

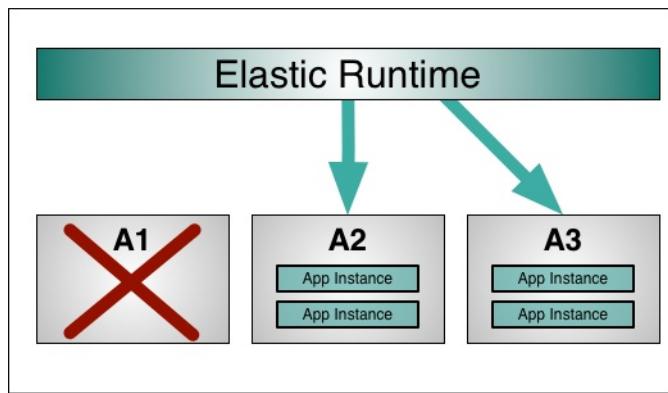
An operator scales an application to four instances in an Elastic Runtime environment distributed across three availability zones: A1, A2, and A3. The environment allocates the instances according to the [DEA Placement Algorithm](#):



If A1 experiences a power outage or hardware failure, the two application instances running in A1 terminate while the application instances in zones A2 and A3 continue to run:



If A1 remains unavailable, Elastic Runtime balances new instances of the application across the remaining availability zones:



Provisioning a Virtual Disk in vSphere

When you create a virtual machine in VMware vSphere, vSphere creates a new virtual hard drive for that virtual machine. The virtual hard drive is contained in a virtual machine disk (VMDK). The disk format you choose for the new virtual hard drive can have a significant impact on performance.

You can choose one of three formats when creating a virtual hard drive:

- Thin Provisioned
- Thick Provisioned Lazy Zeroed
- Thick Provisioned Eager Zeroed

Thin Provisioned

Advantages:

- Fastest to provision
- Allows disk space to be overcommitted to VMs

Disadvantages:

- Slowest performance due to metadata allocation overhead and additional overhead during initial write operations
- Overcommitment of storage can lead to application disruption or downtime if resources are actually used
- Does not support clustering features

When vSphere creates a thin provisioned disk, it only writes a small amount of metadata to the datastore. It does not allocate or zero out any disk space. At write time, vSphere first updates the allocation metadata for the VMDK, then zeros out the block or blocks, then finally writes the data. Because of this overhead, thin provisioned VMDKs have the lowest performance of the three disk formats.

Thin provisioning allows you to overcommit disk spaces to VMs on a datastore. For example, you could put 10 VMs, each with a 50 GB VMDK attached to it, on a single 100 GB datastore, as long as the sum total of all data written by the VMs never exceeded 100 GB. Thin provisioning allows administrators to use space on datastores that would otherwise be unavailable if using thick provisioning, possibly reducing costs and administrative overhead.

Thick Provisioned Lazy Zeroed

Advantages:

- Faster to provision than Thick Provisioned Eager Zeroed
- Better performance than Thin Provisioned

Disadvantages:

- Slightly slower to provision than Thin Provisioned
- Slower performance than Thick Provisioned Eager Zero
- Does not support clustering features

When vSphere creates a thick provisioned lazy zeroed disk, it allocates the maximum size of the disk to the VMDK, but does nothing else. At the initial access to each block, vSphere first zeros out the block, then writes the data. Performance of a thick provisioned lazy zeroed disk is not as good a thick provisioned eager zero disk because of this added overhead.

Thick Provisioned Eager Zeroed

Advantages:

- Best performance
- Overwriting allocated disk space with zeros reduces possible security risks
- Supports clustering features such as Microsoft Cluster Server (MSCS) and VMware Fault Tolerance

Disadvantages:

- Longest time to provision

When vSphere creates a thick provisioned eager zeroed disk, it allocates the maximum size of the disk to the VMDK, then zeros out all of that space.

Example: If you create an 80 GB thick provisioned eager zeroed VMDK, vSphere allocates 80 GB and writes 80 GB of zeros.

By overwriting all data in the allocated space with zeros, thick provisioned eager zeroed eliminates the possibility of reading any residual data from the disk, thereby reducing possible security risks.

Thick provisioned eager zeroed VMDKs have the best performance. When a write operation occurs to a thick provisioned eager zeroed disk, vSphere writes to the disk, with none of the additional overhead required by thin provisioned or thick provisioned lazy zeroed formats.

Deploying Operations Manager to vSphere

Refer to this topic for help deploying Ops Manager to VMware vSphere. For help deploying Ops Manager to vCloud Air or vCloud, see the [Deploying Operations Manager to vCloud Air and vCloud](#) topic.

1. Refer to the [Known Issues](#) topic before getting started.
2. Download the [Pivotal Cloud Foundry](#) Ops Manager .ova file at [Pivotal Network](#) . Click the **Pivotal Cloud Foundry** region to access the PCF product page. Use the dropdown menu to select an Ops Manager release.

The screenshot shows the Pivotal Network interface. At the top, there's a navigation bar with a 'P' icon, 'Pivotal Network', 'PRODUCTS', 'SUPPORT', 'Join', and 'SIGN IN'. Below this, a 'CF' logo is displayed with the text 'Pivotal CF' and the subtext 'Rapidly deploy and scale applications on private clouds with no down time'. A 'Follow' button is present. The main content area has tabs for 'RELEASES' and 'PRODUCT DETAILS'. Under 'RELEASES', it shows 'Ops Manager 1.3.4.0' with a dropdown menu. The 'RELEASE DESCRIPTION' section indicates a security release from Dec 10, 2014, resolving vulnerabilities identified in CVE-2014-6271 & CVE-2014-7169. Additional info is provided at <http://www.pivotal.io/security/CVE-2014-3566>. The 'PRODUCT DETAILS' table lists the file version as 1.3.4.0 and the size as 1.63 GB, with download and details icons.

3. Log into vCenter.
4. Select the **VM and Templates** view.

The screenshot shows the vCenter navigation bar with options: File, Edit, View, Inventory, Administration, Plug-ins, Help. Below the bar, there are navigation icons: Home, Inventory, VMs and Templates. The 'VMs and Templates' icon is highlighted, indicating the current view.

5. Right click on your datacenter and select **New Folder**.

The screenshot shows a right-click context menu on a datacenter named 'sandboxesdc'. The menu items include 'New Folder' (highlighted with a red box), 'New Cluster...', and 'New Datastore Cluster'. A keyboard shortcut 'Ctrl+F' is also shown next to 'New Folder'.

6. Name the folder "pivotal_cf" and select it.

The screenshot shows the 'sandboxesdc' datacenter list. A new folder named 'pivotal_cf' is visible under the 'localhost' entry. Other entries include 'pml1_microsoft_vms' and 'pml1_microsoft_templates'.

7. Select **File > Deploy OVF Template**.

The screenshot shows the vCenter 'File' menu with the 'Deploy OVF Template...' option highlighted (indicated by a red box). Other menu items include 'New', 'Deploy OVF Template...', 'Export', 'Report', 'Browse VA Marketplace...', 'Print Maps', and 'Exit'. A tooltip for 'Deploy OVF Template...' shows the path: 'Inventory > VMs and Temp'.

8. Select the .ova file and click **Next**.

Deploy from a file or URL

Browse...

Enter a URL to download and install the OVF package from the Internet, or specify a location accessible from your computer, such as a local hard drive, a network share, or a CD/DVD drive.

9. Review the product details and click **Next**.
10. Accept the license agreement and click **Next**.
11. Name the virtual machine and click **Next**.

Name:
Pivotal CF vSphere Edition

The name can contain up to 80 characters and it must be unique within the inventory folder.

Inventory Location:

- anchovy-dc
 - birchwood_templates
 - birchwood_vms

Note: The selected folder is the one you created.

12. Select a vSphere cluster and click **Next**.

- anchovy-dc
 - anchovy-cluster

13. If prompted, select a resource pool and click **Next**.
14. If prompted, select a host and click **Next**.

Note: Hardware virtualization must be off if your vSphere host does not support VT-X/EPT. Refer to the [Prerequisites](#) topic for more information.

Choose a specific host within the cluster.
On clusters that are configured with vSphere HA or Manual mode vSphere DRS, each virtual machine must be assigned to a specific host, even when powered off.

Select a host from the list below:

Host Name
172.16.64.2

15. Select a storage destination and click **Next**.

Select a destination storage for the virtual machine files:

VM Storage Profile:

Name	Drive Type	Capacity	Provisioned	Free	Type	Thin Pro
anchovy-ds	Non-SSD	5.41 TB	1.62 TB	3.98 TB	VMFS5	Supports
anchovy-down	SSD	447.00 GB	345.66 GB	71.50 GB	VMFS5	Supports

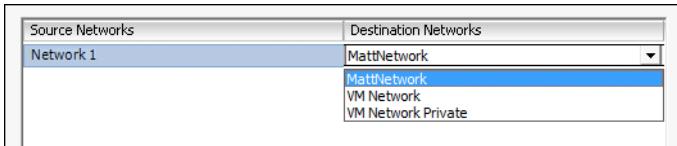
16. Select a disk format and click **Next**. For information about disk formats, see [Provisioning a Virtual Disk](#).

Datastore:

Available space (GB):

Thick Provision Lazy Zeroed
 Thick Provision Eager Zeroed
 Thin Provision

17. Select a network from the drop down list and click **Next**.



18. Enter network information and passwords for the VM admin user and click **Next**.

Note: The IP Address you enter will be the location of the PCF Operations Manager interface.

IP Address The IP address for the Pivotal CF Installer. Leave blank if DHCP is desired. <input type="text"/>
Netmask The netmask for the Pivotal CF Installer's network. Leave blank if DHCP is desired. <input type="text"/>
Default Gateway The default gateway address for the Pivotal CF Installer's network. Leave blank if DHCP is desired. <input type="text"/>
DNS The domain name servers for the Pivotal CF Installer (comma separated). Leave blank if DHCP is desired. <input type="text"/>
NTP Servers Comma-delimited list of NTP servers <input type="text"/>
Admin Password This password is used to SSH into the Pivotal CF Installer. The username is 'tempest'. Enter password <input type="password"/> Confirm password <input type="password"/>

19. Check the **Power on after deployment** checkbox and click **Finish**. Once the VM boots, the interface is available at the IP address you specified.

Note: It is normal to experience a brief delay before the interface is accessible while the web server and VM start up.

[Return to the Getting Started Guide](#)

Deploying Operations Manager to vCloud Air and vCloud

This topic is a prerequisite to [Configuring Ops Manager Director for vCloud Air and vCloud](#).

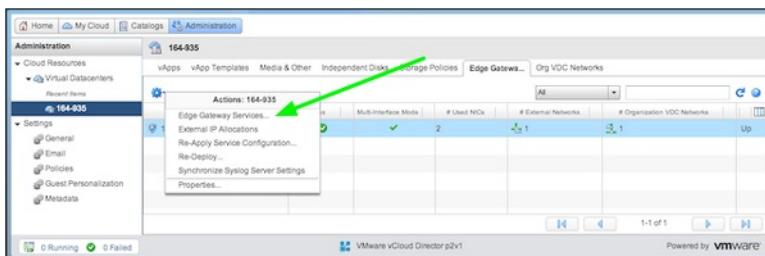
This topic describes how to configure the vCloud or vCloud Air Edge Gateways Configure Services screen and install Ops Manager for your Elastic Runtime environment.

Note: Pivotal Cloud Foundry does not currently support vCloud Air On Demand.

Accessing the vShield Edge Gateway Services Interface

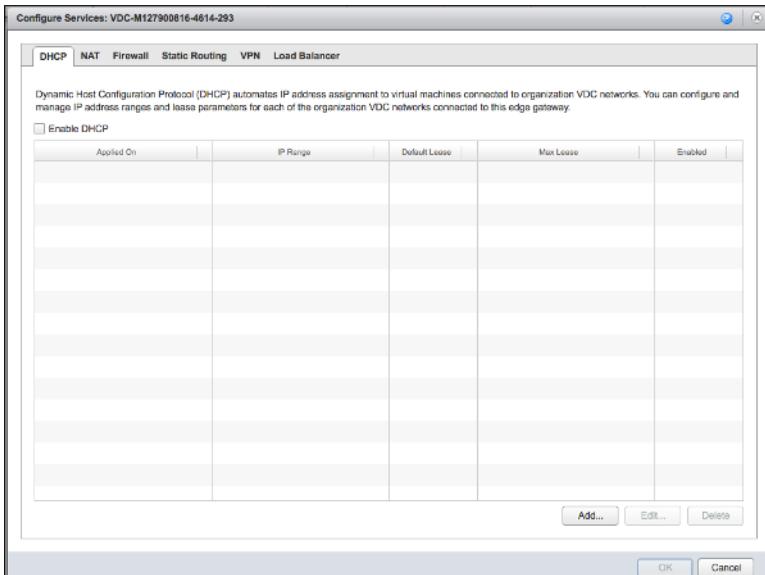
Follow these steps to access the vCloud or vCloud Air Edge Gateways Configure Services screen. For more information about edge gateway services, see the [VMware vCloud Director](#) documentation.

1. Log into vCloud or vCloud Air.
2. Click the **Gateways** tab and your virtual datacenter on the Gateways page. The **Gateways > Gateways Details** page appears.
3. Click **Manage Advanced Gateway Settings** on the right side of the Gateways > Gateways Details page. The **vCloud Director > Administration > Edge Gateways** page appears.
4. Select the gateway you want to configure, then click the gear icon and select **Edge Gateway Services**.



The **Configure Services** screen for your virtual datacenter displays with the following tabs:

- **DHCP**
- **NAT**
- **Firewall**
- **Static Routing**
- **VPN**
- **Load Balancer**



Note: The following sections describe how to perform the minimum configuration steps: setting up NAT rules, firewalls, static routing, and a load balancer. Ensure that you configure the Edge Gateways **Configure Services** screen with any additional settings that your environment requires.

Configuring NAT Rules

The following section describe how to configure your vCloud or vCloud Air Edge Gateway to ensure that Elastic Runtime can access the web.

To do this, you configure the single source NAT rule (SNAT) and three destination NAT (DNAT) rules that Elastic Runtime requires:

- Elastic Runtime accesses the Internet using an SNAT rule.
- Elastic Runtime's API endpoint, which is fronted by HAProxy, requires a DNAT rule to forward traffic from a public IP.
- Ops Manager also requires a DNAT rule to connect external sources on any port to its public IP, as illustrated:

vCloud or vCloud Air evaluates NAT rules in the order you list them in, from top to bottom, on the **NAT** tab of the Edge Gateways **Configure Services** screen. The image is an example of the configured SNAT rule and a DNAT rule.

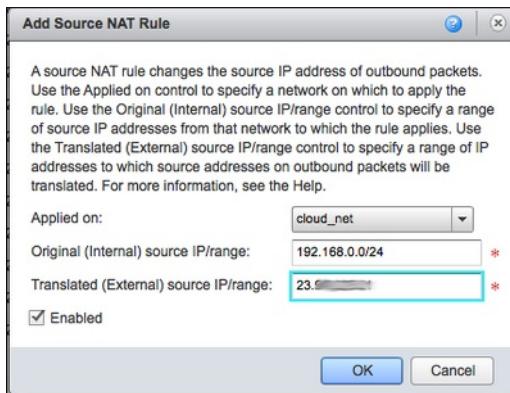
Applied On	Type	Original IP	Original Port	Translated IP	Translated Port	Protocol	Enabled
d2p3-ext	SNAT	192.168.1.10	any	23.92.245.13	any	ANY	✓
d2p3-ext	DNAT	23.92.245.13	80	192.168.1.10	80	TCP & UDP	✓
d2p3-ext	DNAT	23.92.245.13	443	192.168.1.10	443	TCP & UDP	✓
d2p3-ext	DNAT	23.92.245.13	22	192.168.1.10	22	TCP & UDP	✓

Create SNAT and DNAT Rules

To allow outbound connections through Ops Manager public IP address, configure an SNAT rule. To enable inbound traffic over SSH to your Ops Manager VM, create a DNAT rule.

Note: Using the Elastic Runtime IP address for outbound connections can be problematic for DNS resolution.

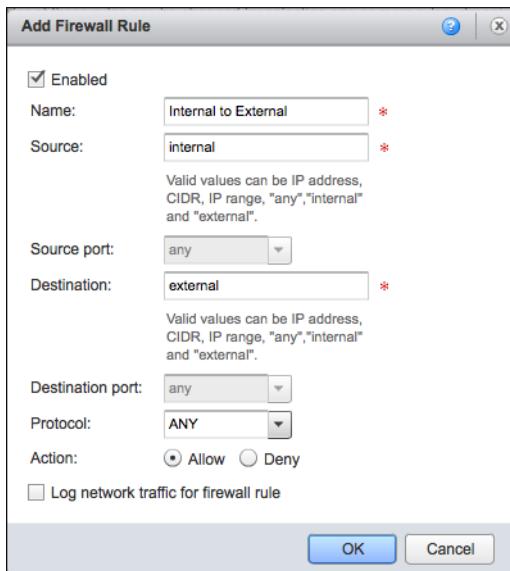
1. In the Edge Gateways **Configure Services** screen, select the **NAT** tab.
2. Configure an SNAT rule:
 - a. From the **Applied on** drop down menu, select the network where you want to apply the NAT rule.
 - b. In the **Original (Internal) source IP/range** field, enter the IP range/subnet mask.
 - c. In the **Translated (External) source IP/range** field, enter the Ops manager public IP.
 - d. Ensure the checkbox **Enabled** is checked.



3. Create a destination NAT (DNAT) rule by following the same procedure, using the following configuration:
 - Applied on: Select your external network
 - Original (External) IP/range: Enter the public IP address for Ops Manager
 - Protocol: Select **TCP & UDP**
 - Original Port: Select **22**
 - Translated (Internal) IP/range: Enter the private IP address of your Ops Manager
 - Translated port: **22**

Create Firewall Rules for SNAT and DNAT

1. In the Edge Gateways **Configure Services** screen, select the **Firewall** tab.
2. Create a SNAT firewall rule allowing outbound traffic from all internal IP addresses to all IP external addresses.



3. Create a DNAT firewall rule allowing inbound traffic from the public IP to the private IP address of your Ops Manager.

Allow Inbound Web Traffic for Ops Manager

Repeat the steps above for ports 80 and 443 for the same public address.

Allow Inbound Web Traffic for Elastic Runtime

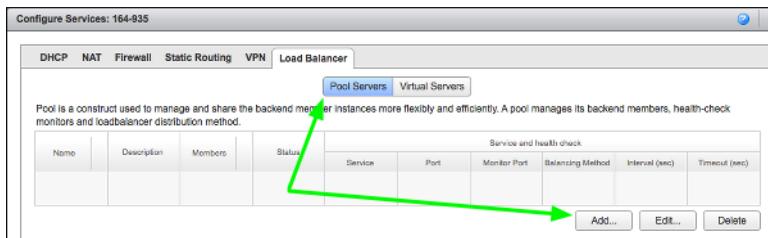
Repeat the steps above for ports 80 and 443 for the Elastic Runtime public IP address.

Setting up Static Routing

Select the **Enable static routing** checkbox.

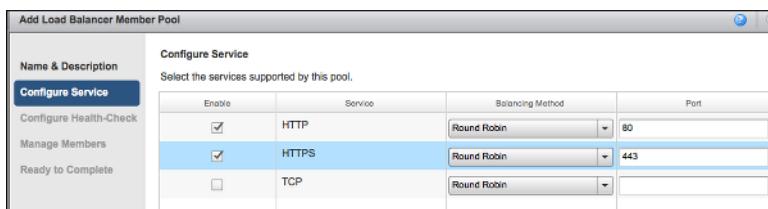
Setting up Network Rules for Elastic Runtime DNS Resolution

- In the Edge Gateways **Configure Services** screen, select the **Load Balancer** tab.
- Click **Pool Servers**, then click **Add**.

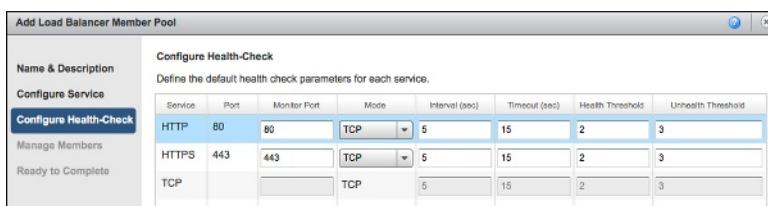


The **Add Load Balancer Member Pool** wizard appears.

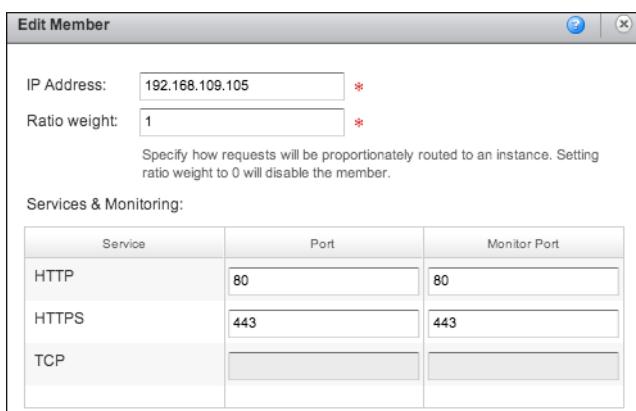
- Name the pool **Load Balancer to Elastic Runtime**.
- In the **Configure Service** step, enable the pool to support HTTP port **80** and HTTPS port **443**. We recommend using the default balancing method, **Round Robin**.



- In the **Configure Health-Check** step, enter Monitor Port **80** for HTTP and **443** for HTTPS. For both HTTP and HTTPS, change the Mode to **TCP**.



- In the **Manage Members** step, click **Add**. Enter the IP address of the HAProxy VM. Specify **80** for the HTTP port values and **443** for the HTTPS port values.



- Click **Finish**.
- Click **Virtual Servers**.

DHCP	NAT	Firewall	Static Routing	VPN	Load Balancer
Pool Servers Virtual Servers (highlighted)					
Pool is a construct used to manage and share the backend member instances more flexibly and efficiently. A pool manages its backend members, health-check monitors and loadbalancer distribution method.					
Name	Description	Members	Status	Service and health check	
				Service	Port
Load Balancer		1	✖	HTTP HTTPS	80 443
				Monitor Port	80 443
				Balancing Method	Round Robin Round Robin
				Interval (sec)	5 5
				Timeout (sec)	15 15

9. Click **Add**.
10. Complete the new virtual server form with the following information:
- Name: Load Balancer
 - Applied On: Select your external network
 - IP Address: Enter the public IP address of your Elastic Runtime instance
 - Pool: Select the **Load Balancer to Elastic Runtime** pool
 - Services: Enable HTTP on port **80** with a Persistence Method of **None**, and HTTPS on port **443** with a Persistence Method of **Session Id**
 - Enabled: Select this checkbox

Edit Virtual Server

Name:	Load Balancer	*			
Description:					
Applied on:	d2p3-ext				
IP address:					
Pool:	Load Balancer to Elastic Runtime	Supports (HTTP,HTTPS)			
Services:					
Enabled	Name	Port	Persistence Method	Cookie name	Cookie mode
<input checked="" type="checkbox"/>	HTTP	80	None		
<input checked="" type="checkbox"/>	HTTPS	443	Session Id		
<input type="checkbox"/>	TCP		None		
<input checked="" type="checkbox"/> Enabled <input type="checkbox"/> Log network traffic for virtual server					

11. Click **OK** to complete.

Deploying Ops Manager to vCloud or vCloud Air

The following procedures guide you through uploading and deploying Ops Manager as a vApp on vCloud or vCloud Air. Refer to the [Known Issues](#) topic before getting started.

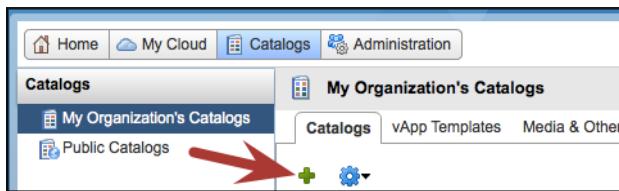
Note: vCloud and vCloud Air use the vCloud Director Web Console, which only supports 32-bit browsers like Firefox. It does not support Chrome. Refer to [Article 2034554](#) in the VMware Knowledge Base for more information about browser versions that the vCloud Director supports.

Upload Ops Manager

You must either upload the Ops Manager vApp into your catalog or use a vApp that your cloud administrator uploaded to your organization's catalog.

Note: The first time you upload software to vCloud Director, you must install the **Client Integration Plug-in** and restart all browsers. If the plug-in does not work and you continue to receive a message prompting you to download it, check the plug-in permissions for your browsers.

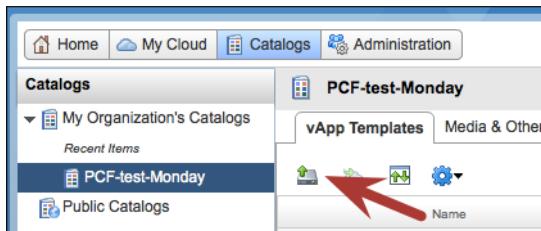
1. Download Pivotal Cloud Foundry Operations Manager for vCloud Air and vCloud Director from [Pivotal Network](#).
2. Log into vCloud Director.
3. Navigate to **Catalogs > My Organization's Catalogs** and select a catalog or click **Add** to create a new catalog.



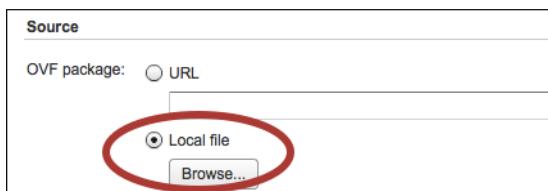
If you are creating a new catalog:

- Enter a name for the new catalog and click **Next**.
- Select a storage type and click **Next**.
- Specify sharing (if needed) and click **Next**.
- Review your settings and click **Finish**.

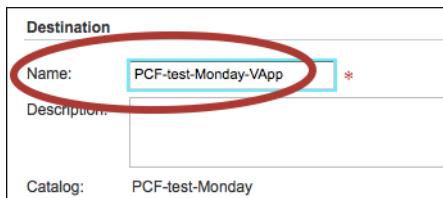
4. Navigate to the **vApp Templates** tab for your catalog and click **Upload**.



5. Select **Local file** and browse to your **.ovf** file.



6. Enter a name for your Ops Manager vApp, enter a description, and click **Upload**.

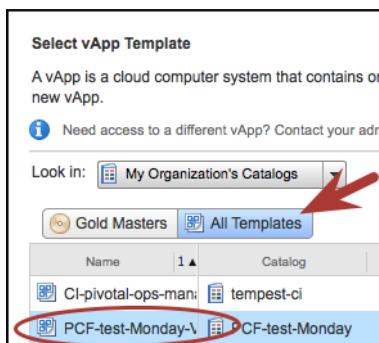


vCloud Director transfers the OVF package to a staging environment, then uploads it to your catalog.

7. Navigate to the **Home** view and click **Add vApp from Catalog**.



8. Select your Ops Manager vApp and click **Next**.



9. Complete the **Add vApp from Catalog** wizard, changing the default settings as necessary for your environment. See [Complete the vApp Wizard and Deploy Ops Manager](#) for more information.

Complete the vApp Wizard and Deploy Ops Manager

After adding the Ops Manager vApp to your vCloud Director, you can finish the set up and deploy as follows:

1. Check the **I agree** checkbox to accept licenses and click **Next**.
2. Enter the name of your Ops Manager vApp, select the virtual data center where the vApp should run, and click **Next**.

Select Name and Location

A vApp is a cloud computer system that contains one or more virtual machines. Describe this vApp and select its Virtual Datacenter.

Name:	PCF-test-Monday-vApp	
Description:		

Virtual Datacenter

Select the Virtual Datacenter (VDC) in which this vApp is stored and runs when it is started.

164-935	
---------	--

3. Choose a storage policy and click **Next**.

Configure Resources

Select what Storage Policies this vApp's virtual machines will use when deployed.

Virtual Machine	Computer Name	Storage Policy
PCF-test-Monday-VApp *	PivotalOpsM-001 *	SSD-Accelerated

4. Set the network mapping **Destination** to the network name, set **IP allocation** to **Static - Manual**, and click **Next**.

Network Mapping

Map networks used in the OVF template to networks in your inventory.

Source	Destination	IP allocation
Network 1	164-935-default-routed	Static - Manual

Source: Network 1 - Description
Logical network used by this appliance.

Destination: 164-935-default-routed - Protocol settings

Gateway: 192.168.109.1
Netmask: 255.255.255.0
DNS:
DNS suffix:

5. Enter the desired networking information, set an admin password for the Ops Manager vApp, and click **Next**.

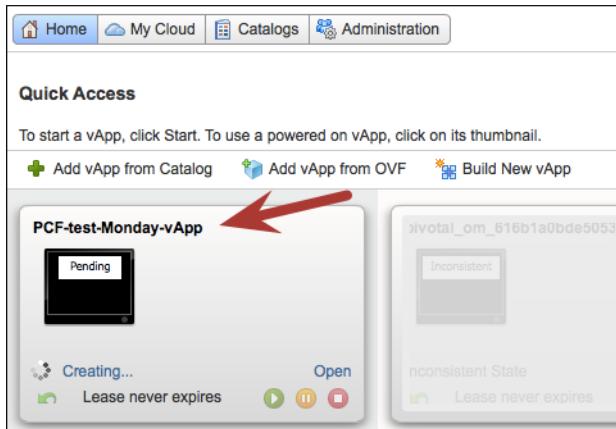
Uncategorized	
DNS:	<input type="text"/>
The domain name servers for the Pivotal Ops Manager (comma separated). Leave blank if DHCP is used.	
Admin Password:	<input type="password"/> Enter password <input type="password"/> Confirm password
This password is used to SSH into the Pivotal Ops Manager. The username is 'tempst'.	
Netmask:	<input type="text"/>
The netmask for the Pivotal Ops Manager's network. Leave blank if DHCP is desired.	
Default Gateway:	<input type="text"/>
The default gateway address for the Pivotal Ops Manager's network. Leave blank if DHCP is desired.	
IP Address:	<input type="text"/>
The IP address for the Pivotal Ops Manager. Leave blank if DHCP is desired.	
NTP Servers:	<input type="text"/>
Comma-delimited list of NTP servers	

Note: The order of the items on your screen may vary from the order shown in this image.

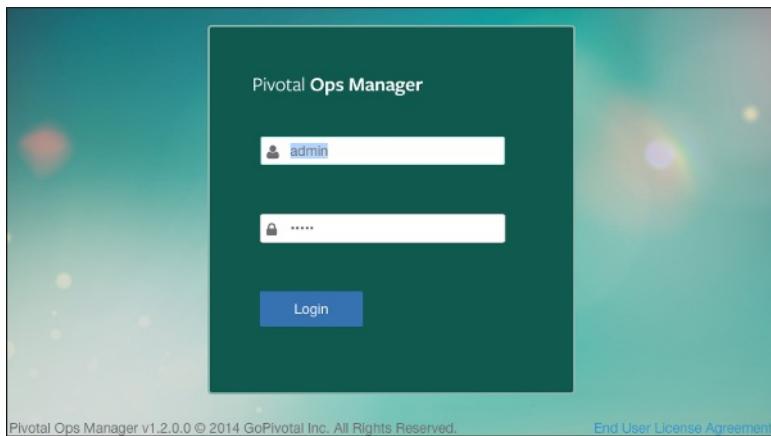
The following list contains tips on entering specific networking information:

- **DNS:** If you are unsure of your Pivotal Ops Manager DNS, you can use the Google Public DNS value **8.8.8.8**. For more information, refer to the [Using Google Public DNS](#) topic.
- **Default Gateway:** On the vCloud Air or vCloud Dashboard, click the **Gateways** tab and copy the **GATEWAY IP** value.
- **IP Address:** Navigate to the **My Clouds > VMs** page, locate the Pivotal Ops Manager VM, and copy the IP address from the **IP Address** column. If this column does not display, click the **Customize Columns** icon on the right side to set your column display preferences.

6. Review the hardware specifications of the virtual machine and click **Next**.
7. In the **Ready to Complete** dialog, check the **Power on vApp After This Wizard is Finished** checkbox and click **Finish**.
8. Navigate to the **Home** view to verify that your Ops Manager vApp is being created.



9. Once the VM boots, the Ops Manager login screen displays at the IP address you specified.



[Return to the Getting Started Guide](#)

Configuring Ops Manager Director for VMware vSphere

Refer to the following procedure for help configuring the Ops Manager Director for VMware vSphere product tile.

Step 1: Access Ops Manager

1. Log in to [Pivotal Cloud Foundry](#) Operations Manager.
2. Click the **Ops Manager Director for VMware vSphere** tile.



Step 2: vCenter Config Page

1. Select **vCenter Config** and enter the following information:

- a. vCenter IP address
- b. vCenter credentials
- c. Datacenter name
- d. A comma-delimited list of datastore names
- e. Click **Save**.

Installation Dashboard

Ops Manager Director

Settings Status Credentials Logs

vCenter Config

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

vCenter IP Address
192.168.0.1

vCenter Username
root

vCenter Password

Datacenter Name
private

Datastore Names (comma delimited)
private

NOTE: Removing a Datastore after an initial deploy can result in a system outage and/or data loss.

Save



Note: The vCenter credentials must grant create and delete privileges for VMs and folders.

Step 3: Director Config Page

1. Select **Director Config**.
 - a. Enter a comma-delimited list of time server addresses.
 - b. If you have installed and configured the Ops Metrics product, enter the Metrics IP address.
 - c. Click **Enable VM Resurrector Plugin** to enable the Ops Manager Resurrector functionality and increase Elastic Runtime availability. For more information, see the introduction and Limitation sections of the [Using Ops Manager Resurrector on VMware vSphere](#) topic.
 - d. Pivotal recommends that you select **Internal** for your **Blobstore Location**. However, if you choose **S3 Compatible Blobstore** for your blobstore location, complete the **S3 Endpoint**, **Bucket Name**, **Access Key**, and **Secret Key** with information from your blobstore provider.
 - e. Select a **Database Location**. By default, Pivotal Cloud Foundry deploys and manages a database for you. If you choose to use an **External MySQL Database**, complete the associated fields with information obtained from your external MySQL Database provider.
 - f. **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. For vSphere, the default value is 32. Leave the Max Threads field blank to use this default value. Pivotal recommends that you use the default value unless doing so results in rate limiting or errors on your IaaS.
 - g. Click **Save**.

Director Config

NTP Servers (comma delimited)*

Metrics IP Address

Enable VM Resurrector Plugin

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*

Bucket Name *

Access Key*

Secret Key*

Database Location
 Internal
 External MySQL Database

Host*

Port*

Username *

Password*

Database*

Max Threads

Save

Step 4: Availability Zones Pages

- Select **Create Availability Zones**. Ops Manager Availability Zones correspond to your vCenter clusters and resource pools.

Having multiple Availability Zones allows you to provide high-availability and load balancing to your applications. When an application runs more than one instance of a job, Ops Manager deterministically balances the instances across all of the Availability Zones that you assign to the application.

Use the following steps to create one or more Availability Zones for your products to use:

- Click **Add**.
- Enter a unique name for the Availability Zone.
- Enter the name of an existing vCenter cluster to use as an Availability Zone.
- (Optional) Enter the name of a resource pool in the vCenter cluster that you specify. The jobs running in this Availability Zone share the CPU and memory resources defined by the pool.
- Click **Save**.

Installation Dashboard

Ops Manager Director for VMware vSphere

- Settings
- Status
- Credentials
- Logs

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Create Availability Zones

Availability Zones
Clusters and resource pools to which you will deploy Pivotal products

az-1

Name* az-1

Cluster* drinks-cl

Resource Pool classic

Save

- Select **Assign Availability Zones**. Use the drop-down menu to select an Availability Zone. The Ops Manager Director installs in this Availability Zone.

Installation Dashboard

Ops Manager Director for VMware vSphere

- Settings
- Status
- Credentials
- Logs

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Assign Availability Zones

The Ops Manager Director is a single instance.

Choose the availability zone in which to place that instance.
It is highly recommended that you snapshot this VM on a regular basis to preserve settings.
Singleton Availability Zone

az-1

Save

Step 5: Networks Pages

1. Select **Create Networks**. Having multiple networks allows you to place vCenter on a private network and the rest of your deployment on a public network. Isolating vCenter in this manner denies access to it from outside sources and reduces possible security vulnerabilities.

Use the following steps to create one or more Ops Manager networks:

- a. Click **Add**.
- b. Enter a unique name for the network.
- c. Enter the vSphere network name as it displays in vCenter.
- d. For **Subnet**, enter a valid CIDR block.
- e. For **Excluded IP Ranges**, enter any IP addresses from the **Subnet** that you want to blacklist from the installation.
- f. Enter DNS and Gateway IP addresses.
- g. Click **Save**.

Note: If you are using the Cisco Nexus 1000v Switch, refer to the [Using the Cisco Nexus 1000v Switch with Ops Manager](#) topic for more information.

2. Select **Assign Networks**. You can configure the Ops Manager Director to have an IP address on two networks to give the Director access to both networks. Use the drop-down menu to select an **Infrastructure** network and a **Deployment** network for Ops Manager.

When using multiple networks to isolate vCenter, install vCenter on the **Infrastructure** network and the rest of your deployment on the **Deployment** network. This gives the Director access to both vCenter and the rest of your deployment.

[◀ Installation Dashboard](#)

Ops Manager Director for VMware vSphere

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Assign Networks

The Ops Manager director can be configured to have an IP on two networks. Typically one is routable to the IaaS API (*Infrastructure*) and the other is routable to deployed products (*Deployment*).

Infrastructure Network:

Deployment Network:

Save

Step 6: Resource Config Page

Select **Resource Config**, accept the defaults or make necessary changes, and click **Save**.

[◀ Installation Dashboard](#)

Ops Manager Director for VMware vSphere

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

vCenter credentials

JOB	INSTANCES	CPU	RAM (MB)	EPHEMERAL DISK (MB)	PERSISTENT DISK (MB)
Ops Manager Director	1	4	3072	16384	20480

vSphere configuration

Network configuration

Save

Step 7: Complete the Ops Manager Director Installation

Click the **Installation Dashboard** link to return to the Installation Dashboard.

[Return to the Getting Started Guide](#)

Configuring Ops Manager Director for vCloud Air and vCloud

Before following these instructions you must [deploy Ops Manager and configure NAT and Firewall rules](#).

Refer to the following procedure for help configuring the Ops Manager Director for VMware vCloud Air and vCloud product tile.

1. Log in to [Pivotal Cloud Foundry](#) Operations Manager.
2. Click the **Ops Manager Director for VMware vCloud** tile.



3. Select **vCloud Config** and enter the following information:

- a. The URL of the vCloud Director.
- b. The **Organization name**.

Note: vCloud Air and vCloud Director use case-sensitive organization names. The name that you supply in the Ops Manager **Organization name** field must match the vCD organization name exactly.

- c. Your credentials.

Note: This user must have create and delete privileges for VMs and folders.

- d. The **Virtual Datacenter name** and **Storage Profile name**. The name that you supply in the Ops Manager **Virtual Datacenter name** field must be the name of the virtual datacenter as it appears in vCloud Director. This name is an alphanumeric string with a “VDC” prefix. See the example in the image below.
- e. Click **Save**.

[Installation Dashboard](#)

Ops Manager Director

Settings Status Credentials Logs

vCloud Config

Director Config

Create Networks

Assign Networks

Resource Config

vCloud Director Config

vCloud API URL*

Organization name*

Username*

Password*

Virtual Datacenter name*

Storage Profile name*

Save

4. Select **Director Config**.

- a. Enter a comma-delimited list of time server addresses.
- b. If you have installed and configured the Ops Metrics product, enter the **Metrics IP Address**.
- c. Check or uncheck **Enable VM resurrector plugin**.
- d. Pivotal recommends that you select **Internal** for your **Blobstore Location** and **Database Location**.
- e. **Max Threads** sets the maximum number of threads that the Ops Manager Director can run simultaneously. For vCloud, the default value is 4. Leave the Max Threads field blank to use this default value. Pivotal recommends that you use the default value unless doing so results in rate limiting or errors on your IaaS.
- f. Click **Save**.

Director Config

NTP Servers (comma delimited)*

Metrics IP Address

Enable VM Resurrector Plugin

Blobstore Location
 Internal
 S3 Compatible Blobstore

S3 Endpoint*

Bucket Name *

Access Key*

Secret Key*

Database Location
 Internal
 External MySQL Database

Host*

Port*

Username *

Password*

Database*

Max Threads

Save

5. Select **Create Networks**. Use the following steps to create one or more Ops Manager networks:
 - a. Define a unique name for the network.

- b. Enter the **vCloud Network Name** as it appears in vCloud Director.
- c. Enter a valid CIDR block in **Subnet**.
- d. Enter any IP addresses from the **Subnet** that you want to blacklist from the installation in **Excluded IP Ranges**.
- e. Enter the **DNS** and **Gateway** IP addresses.
- f. Click **Save**.

Installation Dashboard

Ops Manager Director

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

Create Networks

Networks
One or many IP ranges upon which your products will be deployed

my network

Name*

vCloud Network Name*

Subnet*

Excluded IP Ranges

DNS*

Gateway*

Save

6. Select **Assign Networks**. You can configure the Ops Manager Director to have an IP address on one network. Use the drop-down menu to select the network that acts as the infrastructure and deployment network for Ops Manager.

Installation Dashboard

Ops Manager Director

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

Assign Networks

The Ops Manager director can be configured to have an IP on a network.

Network

Save

7. Select **Resource Config**, accept the defaults or make necessary changes, and click **Save**.

The screenshot shows the 'Resource Config' section of the 'Ops Manager Director' configuration interface. On the left, a sidebar lists several configuration steps: vCloud Config, Director Config, Create Networks, Assign Networks, and Resource Config. The 'Resource Config' step is currently selected. The main panel displays resource allocation details for the 'Ops Manager Director' job. A table provides the following information:

JOB	INSTANCES	CPU	RAM [MB]	TEMPORAL DISK [MB]	PERSISTENT DISK [MB]
Ops Manager Director	1	2	3072	1536	20480

A blue 'Save' button is located at the bottom right of the main panel.

8. Click the **Installation Dashboard** link to return to the Installation Dashboard.

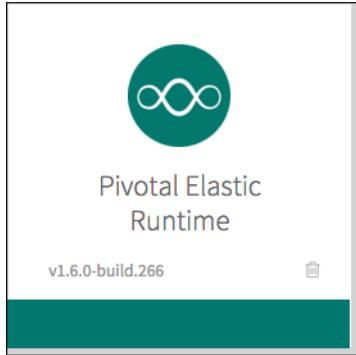
[Return to the Getting Started Guide](#)

Configuring Elastic Runtime for vSphere and vCloud

This topic describes how to configure the Pivotal Elastic Runtime components that you need to run [Pivotal Cloud Foundry \(PCF\)](#) for VMware vSphere, vCloud, or vCloud Air.

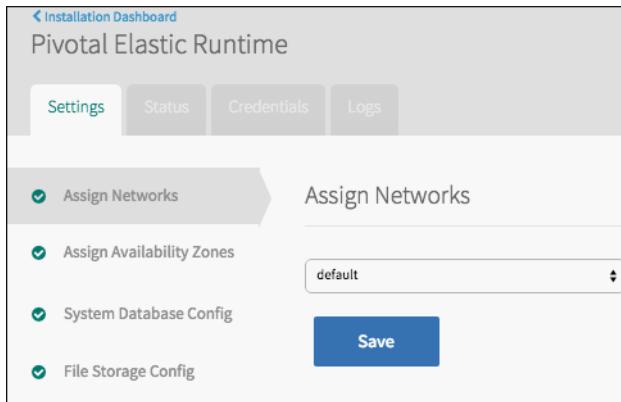
Step 1: Add Elastic Runtime to Ops Manager

1. Navigate to the Pivotal Cloud Foundry Operations Manager Installation Dashboard.
2. Click the Pivotal Network link on the left to add Elastic Runtime to Ops Manager.
For more information, refer to the [Adding and Deleting Products](#) topic.
3. Click the Elastic Runtime tile in the Installation Dashboard.



Step 2: Assign Networks and Availability Zones

1. Select **Assign Networks**. Elastic Runtime runs on the network that you select.



2. **(vSphere Only)** Select **Assign Availability Zones**. These are the Availability Zones that you [create](#) when configuring Ops Manager Director.
 - Select an Availability Zone under **Place singleton jobs**. Ops Manager runs any job with a single instance in this Availability Zone.
 - Select one or more Availability Zones under **Balance other jobs**. Ops Manager balances instances of jobs with more than one instance across the Availability Zones that you specify.

[◀ Installation Dashboard](#)

Pivotal Elastic Runtime

- [Settings](#)
- [Status](#)
- [Credentials](#)
- [Logs](#)

Assign Networks

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

IPs and Ports

Availability Zone Assignments

Place singleton jobs in

default

Balance other jobs in

default

[Save](#)

Step 3: Configure System Databases

1. Select **System Database Config**. By default, this page specifies that PCF uses an internal database for UAADB, Apps Manager DB, CCDB, App Usage Events DB, Autoscaling, and Notifications. Alternatively, you can configure an external AWS RDS database. If you choose to use an external database, complete the associated fields with information obtained from AWS.

Note: Pivotal recommends that you accept the default internal database unless you require the functionality of AWS RDS.

Assign Networks

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

Security Config

MySQL Proxy Config

Cloud Controller

System Logging

SSO Config

LDAP Config

SMTP Config

Diego

Experimental Features

Errands

Configurations should only be made on the initial installation of Elastic Runtime (do NOT change anything on this section when upgrading). Once Elastic Runtime is installed, a manual data migration is required in order to change this configuration safely. Consult Pivotal Cloud Foundry documentation for details.

Configure your system's databases*

Internal Databases - MySQL and Postgres (the Postgres DBs are not highly-available, but this selection is required if you want to keep your system data from a pre-1.6.0 Elastic Runtime that you upgraded)

Internal Databases - MySQL (preferred for enabling complete high-availability)

External Databases (preferred if, for example, you can use AWS RDS)

Hostname DNS Name *

pp19dd339auydlw.cpdtp8njpud.us-west-1

TCP Port *

3306

Username *

docs

Password *

[Save](#)

Note: The screenshot depicts the option selected for a new installation. If you are performing an upgrade from Elastic Runtime 1.5 to 1.6, do not modify your existing internal DB configuration or you may lose data. You must migrate your existing data first before changing the configuration. See [Upgrading Ops Manager](#) for additional upgrade information.

2. (**Optional**) Select **MySQL Proxy Config**. Enter one or more IP addresses for the MySQL proxy instances configured on your external load balancer.

Use of a proxy permits transparent failover to other nodes within the cluster in the event of a node failure. When a node becomes unhealthy, the proxy re-routes all subsequent connections to a healthy node and all existing connections to the unhealthy node are closed. See [MySQL for PCF: Proxy](#) for more information.

The screenshot shows the 'Pivotal Elastic Runtime' interface with the 'Settings' tab selected. On the left, a sidebar lists configuration sections: Assign Networks, Assign Availability Zones, System Database Config, File Storage Config, IPs and Ports, Security Config, MySQL Proxy Config (which is highlighted), and Cloud Controller. The main content area has two input fields: 'MySQL Proxy IPs' and 'MySQL Service Hostname'. A note above the 'MySQL Proxy IPs' field says: 'Enter the IP address(es) for the MySQL proxy instances configured on your external load balancer.' A blue 'Save' button is at the bottom right.

Step 4: Configure File Storage

1. Select **File Storage Config**.
2. To use the PCF internal filestore, select the **Internal** option and click **Save**.
3. To use an external S3-compatible filestore for your Elastic Runtime file storage, select the **External S3-Compatible Filestore** option and complete the following procedure:
 - a. Enter the **URL Endpoint** for your filestore.
 - b. Enter your **Access Key** and **Secret Key**.

Note: If you are performing an upgrade from Elastic Runtime 1.5 to 1.6, do not modify the existing Bucket Name configuration or you may lose data. You must migrate your existing data before changing the configuration. See the [Upgrading Ops Manager](#) topic for additional upgrade information.
 - c. Enter a **Buildpacks Bucket Name**.
 - d. Enter a **Droplets Bucket Name**.
 - e. Enter a **Packages Bucket Name**.
 - f. Enter a **Resources Bucket Name**.
 - g. Click **Save**.

Assign Networks

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

Security Config

MySQL Proxy Config

Cloud Controller

System Logging

SSO Config

LDAP Config

SMTP Config

Diego

Experimental Features

Errands

Resource Config

Stemcell

Configurations should only be made on the initial installation of Elastic Runtime (do NOT change anything on this section when upgrading). Once Elastic Runtime is installed, a manual data migration is required in order to change this configuration safely. Consult Pivotal Cloud Foundry documentation for details.

Configure your Cloud Controller's filesystem*

Internal

External S3-Compatible Filestore

URL Endpoint *

URL endpoint of your S3-compatible blobstore

Access Key *

Secret Key *

Buildpacks Bucket Name *

Droplets Bucket Name *

Packages Bucket Name *

Resources Bucket Name *

Save

Step 5: Set IPs and Loggregator Port

1. Select **IPs and Ports**.

Assign Networks

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

Security Config

MySQL Proxy Config

Cloud Controller

System Logging

HAProxy is the default load balancer for SSL termination. Alternatively, you can use your own load balancer and forward traffic to the Cloud Foundry router IP.

Router IPs

Enter the IP address(es) for the Pivotal CF Elastic Runtime Router. This IP must be in your subnet.

HAProxy IPs

Loggregator Port

Save

1. **(Optional)** The values you enter in the **Router IPs** and **HAProxy IPs** fields depends on whether you are using your own load balancer or the HAProxy load balancer. Find your load balancer type in the table below to determine how to complete these fields.

 **Note:** If you choose to assign specific IP addresses in either the **Router IPs** or **HAProxy IPs** field, ensure that these IPs are in your subnet.

LOAD BALANCER	ROUTER IP FIELD VALUE	HAProxy IP FIELD VALUE
Your own load balancer	Enter the IP address or addresses for PCF that you registered with your load balancer. Refer to the Using Your Own Load Balancer topic for help using your own load balancer with PCF.	Leave this field blank.
HAProxy load balancer	Leave this field blank.	Enter at least one HAProxy IP address. Point your DNS to this address.

For more information, refer to the [Configuring PCF SSL Termination](#) topic. For help understanding the Elastic Runtime architecture, refer to the [Architecture](#) topic.

- The **Loggregator Port** field specifies an alternate port for Loggregator. If you leave this field blank, Elastic Runtime uses the default port 443. Pivotal recommends that you accept the default value unless you require Loggregator to use a different port.

Step 6: Configure Security and SSL Certificates

- Click on **Security Config**.

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configure security settings

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIICKTCAZICCCVjs8Jtc84cTANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJV
MTUwNTAzMTkyMjEyWjBZMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0ExiTafBgNV
BAMTCGJhZG1pbnQgQ2VudHJhbCBvZiB0aGUgUmVhZCwgQ0EwHhcNMTIwMDIwMDIw
MjUwWzBzMDIwMDIwMDIwMDIwWzBzMDIwMDIwMDIwMDIwWzBzMDIwMDIwMDIwMDIw
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC5CWbiXc86fFy4mpAf6PmxQTn36d+bj8fUDo6NZQ7B0twTW5iQ
AoGBAKsRCylqUkSi5uHwW2B09isuv1X4zTa3daoSdy6fIEfGEF5RBKAAnPrd7Xv
-----END RSA PRIVATE KEY-----
```

[Generate Self-Signed RSA Certificate](#)

Ignore SSL certificate verification

HAProxy SSL Ciphers

Router SSL Ciphers

Disable HTTP traffic to HAProxy

Enable cross-container traffic

Enable TLS on the Router

Save

2. Provide an **SSL Termination Certificate** for your SSL Termination Point.

- In a production environment, use a signed **SSL Certificate** from a known certificate authority (CA). Copy and paste the values for **Certificate PEM** and **Private Key PEM** from the signed certificate into the appropriate text fields.
- In a development or testing environment, you may use a self-signed certificate.

Note: Pivotal does not recommend using a self-signed certificate for production deployments.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

To use a self-signed certificate, follow the steps below: * Click the **Generate Self-Signed RSA Certificate** link to launch the RSA certificate wizard. * Enter your system and app domains in wildcard format. Optionally, also add any custom domains in wildcard format. The example below uses `*.example.com`.

Note: Wildcard DNS records only work for a single domain name component or component fragment. For example, `*.domain.com` works for `apps.domain.com` and `system.domain.com`, but not for `apps.domain.com` or `system.domain.com`.

Note: You can generate a single certificate for two domains separated by a comma. For example, `.apps.domain.com, *.system.domain.com`.

✓ Generate Self-Signed RSA Certificate

Example: *.app.domain.com, *.system.domain.com, my.webapp.com,
.domain.com

`*.example.com`

Cancel Generate

* Click **Generate**. * Elastic Runtime populates the **SSL Certificate** fields with RSA certificate and private key information.

3. Configure **Ignore SSL certificate verification**. Select this option if you are using self-signed certificates or certificates generated from Ops Manager.
4. Configure **HAProxy SSL Ciphers** and **Router SSL Ciphers**. Leave these fields blank unless you want to use a specific set of SSL ciphers for HAProxy or the Router.
5. Configure **Disable HTTP traffic to HAProxy**. If you select the **Disable HTTP traffic to HAProxy** checkbox, your deployment rejects all port 80 traffic to HAProxy. Additionally, this option sets the secure flag in the `VCAP_ID` cookie that the Router generates.

Note: If you enable this checkbox and your deployment is not using HAProxy, configure your external load balancer to reject port 80 traffic. If you do not do this, traffic to port 80 is forwarded to applications but loses session stickiness.

6. Configure **Enable cross container traffic**. By selecting this checkbox, you disable the restriction that prevents containers in the same DEA or Diego Cell from communicating with each other. Pivotal does not recommend selecting this checkbox in multi-tenant environments. You should select this option for microservices such as Pivotal Spring Cloud.
7. Configure **Enable TLS on the Router**. Selecting this enables [SSL termination](#) on the Router.
8. Click **Save**.

Step 7: Configure the Cloud Controller

1. Select **Cloud Controller** and enter the system and application domains.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign Networks	Coordinates Pivotal CF Elastic Runtime application lifecycles
Assign Availability Zones	
System Database Config	System Domain * system.mydomain.com
File Storage Config	Apps Domain * apps.mydomain.com
IPs and Ports	
MySQL Proxy Config	Cloud Controller DB Encryption Key Secret
Cloud Controller	Maximum File Upload Size (MB) (min: 1024, max: 2048) * 1024
External Endpoints	
SSO Config	<input type="checkbox"/> Disable Custom Buildpacks
LDAP Config	Default Quota App Memory (MB) (min: 10240, max: 102400) * 10240
SMTP Config	
Errands	Default Quota Service Instances (min: 0, max: 1000) * 100
Resource Config	
Stemcell	Save

- The **System Domain** defines your target when you push apps to Elastic Runtime.
- The **Apps Domain** defines where Elastic Runtime should serve your apps.

Note: Pivotal recommends that you use the same domain name but different subdomain names for your system and app domains. Doing so allows you to use a single wildcard certificate for the domain while preventing apps from creating routes that overlap with system routes. For example, name your system domain `system.EXAMPLE.COM` and your apps domain `apps.EXAMPLE.COM`.

Note: You configured wildcard DNS records for these domains in an earlier step.

- Leave the **Cloud Controller DB encryption key** field blank unless:

- You deployed Elastic Runtime earlier
- You then stopped Elastic Runtime, or it crashed
- You are re-deploying Elastic Runtime with a backup of your Cloud Controller database

Enter your Cloud Controller database encryption key only if these conditions apply. See [Backing Up Pivotal Cloud Foundry](#) for more information.

- Enter your intended maximum file upload size.
- (Optional) Check the **Disable Custom Buildpacks** checkbox. By default, the cf command line tool (cf CLI) gives developers the option of using a custom buildpack when they deploy apps to Elastic Runtime. To do so, they use the `-b` option to provide a custom buildpack URL with the `cf push` command. The **Disable Custom Buildpacks** checkbox disables the `-b` option. For more information about custom buildpacks, refer to the [buildpacks](#) section of the PCF documentation.
- Enter your default app memory and service instances quotas.
- Click **Save**.

Step 8: (Optional - Advanced) Configure System Logging

If you are forwarding syslog messages via TCP to a RELP syslog server, complete this step.

1. Select System Logging.

Configure system logging. Leave the External Syslog fields blank unless you wish to use an external syslogd server.

Assign Networks	External Syslog Aggregator Host IP
Assign Availability Zones	External Syslog Aggregator Port
System Database Config	External Syslog Network Protocol
File Storage Config	Syslog Drain Buffer Size (# of messages) *
IPs and Ports	100
Security Config	
MySQL Proxy Config	
Cloud Controller	
System Logging	
SSO Config	

Save

2. Enter the IP address and port of the **Syslog Aggregator Host**.

Note: The host must be reachable from the Elastic Runtime network, accept TCP connections, and use the RELP protocol. Ensure `syslogd` listens on external interfaces.

3. Configure the **Syslog drain Buffer Size**. This number configures how many messages the Doppler server can hold from Metron agents before the server starts to drop them. See [Loggregator Guide for Cloud Foundry Operators](#) for more details.
4. Click **Save**.

Step 9: (Optional - Advanced) Configure VMware SSO

If you are using the VMware SSO appliance for integration with Active Directory, follow the instructions in [Configuring Single Sign-On](#).

The screenshot shows the 'Settings' tab selected in the top navigation bar. On the left, a vertical list of configuration items is shown with checkboxes: Assign Networks (checked), Assign Availability Zones (checked), System Database Config (checked), File Storage Config (checked), IPs and Ports (checked), MySQL Proxy Config (checked), Cloud Controller (checked), External Endpoints (checked), SSO Config (checked), and LDAP Config (unchecked). A grey arrow points from the 'SSO Config' item towards the 'LDAP Config' item. To the right, the 'Configure Identity Provider' section is displayed, containing fields for 'Provider Name' (empty) and 'Provider Metadata' (empty), along with a placeholder '(OR) Provider Metadata URL'. A blue 'Save' button is located at the bottom right of this section.

Step 10: (Optional - Advanced) Configure LDAP

If you are using the LDAP endpoint for UAA, you must configure Elastic Runtime with your LDAP endpoint information. See [Configuring LDAP](#) for more information.

Step 11: (Optional) Configure SMTP

Elastic Runtime uses SMTP to send invitations and confirmations to Apps Manager users. You must complete the Configuration for SMTP page if you want to enable end-user self-registration.

1. Select **SMTP Config**.

Configuration for SMTP

Assign Networks	From Email admin@my.cf.com
Assign Availability Zones	Address of SMTP Server 172.16.64.5
System Database Config	Port of SMTP Server 25
File Storage Config	SMTP Server Credentials Username Password
IPs and Ports	<input type="checkbox"/> SMTP Enable Automatic STARTTLS
MySQL Proxy Config	SMTP Authentication Mechanism* None
Cloud Controller	SMTP CRAMMD5 secret
External Endpoints	
SSO Config	
LDAP Config	
SMTP Config	
Errands	
Resource Config	
Stemcell	
Save	

2. Enter your reply-to and SMTP email information
3. Verify your authentication requirements with your email administrator and use the **SMTP Authentication Mechanism** drop-down menu to select `None`, `Plain`, or `CRAMMD5`. If you have no SMTP authentication requirements, select `None`.
4. Click **Save**.

Note: If you do not configure the SMTP settings using this form, the administrator must create orgs and users using the cf CLI tool. See [Creating and Managing Users with the cf CLI](#) for more information.

Step 12: Configure Diego

In the PCF 1.6 release, Diego replaces the [DEAs](#) and the [Health Manager](#). Diego is installed and enabled in PCF 1.6 by default.

For more information about Diego, see the [Diego Architecture](#) and [Diego Components](#) topics.

Note: Before you install PCF 1.6, you must uninstall any Diego Beta installations. For more information on upgrading to PCF 1.6, see [Upgrading Operations Manager](#).

Pivotal Elastic Runtime

- Settings
- Status
- Credentials
- Logs

<input checked="" type="checkbox"/> Assign Networks	Diego is a new replacement for DEAs.
<input checked="" type="checkbox"/> Assign Availability Zones	<input checked="" type="checkbox"/> Use Diego by default instead of DEAs
<input checked="" type="checkbox"/> System Database Config	<input checked="" type="checkbox"/> Allow SSH access to apps
<input checked="" type="checkbox"/> File Storage Config	Application Containers Subnet Pool *
<input checked="" type="checkbox"/> IPs and Ports	10.254.0.0/22
<input checked="" type="checkbox"/> Security Config	
<input checked="" type="checkbox"/> MySQL Proxy Config	
<input checked="" type="checkbox"/> Cloud Controller	
<input checked="" type="checkbox"/> System Logging	
<input checked="" type="checkbox"/> SSO Config	
<input checked="" type="checkbox"/> LDAP Config	
<input checked="" type="checkbox"/> SMTP Config	
<input checked="" type="checkbox"/> Diego	

Save

1. Select **Diego**.

2. Select the **Use Diego by default instead of DEAs** checkbox to ensure that all applications pushed to your PCF deployment use the Diego container management system. For new installations, this option is selected by default. If you are upgrading to PCF 1.6, you must select this option to automatically enable Diego for newly pushed applications.

Note: If you do not select this option, application developers must explicitly target Diego when pushing their applications.

3. (Optional) Select the **Allow SSH access to apps** checkbox to allow application developers to `ssh` directly into their application instances on Diego. See the [Diego-SSH Overview](#) topic for more details.

Step 13: (Optional) Enable Experimental Features

Use caution when enabling experimental features if you have other Pivotal Cloud Foundry service tiles installed in your Pivotal Cloud Foundry deployment. Not all of the services are guaranteed to work as expected with these features enabled.

This release does not contain any experimental features.

Step 14: Configure Errands

Errands are scripts that Ops Manager runs to automate tasks. By default, Ops Manager runs the post-install errands listed below when you deploy Elastic Runtime. However, you can prevent a specific post-install errand from running by deselecting its checkbox on the **Errands** page.

Note: Several errands deploy apps that provide services for your deployment, such as Autoscaling and Notifications. Once one of these apps is running, deselecting the checkbox for the corresponding errand on a

subsequent deployment does not stop the app.

Errands

Errands are scripts that run at designated points during an installation.

Post Install

- | | |
|---|--|
| <input checked="" type="checkbox"/> Push Apps Manager | Pushes the Pivotal Apps Manager application to your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Run Smoke Tests | Runs Smoke Tests against your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Push App Usage Service | Monitors usage of an application pushed to your Elastic Runtime installation |
| <input checked="" type="checkbox"/> Notifications | Notifications release for PCF |
| <input checked="" type="checkbox"/> Notifications-UI | Notifications UI Component for PCF |
| <input checked="" type="checkbox"/> Deploy CF Autoscaling App | Deploys the CF Autoscaling App |
| <input checked="" type="checkbox"/> Register autoscaling service broker | Register autoscaling service broker |

There are no pre-delete errands for this product.

Save

- **Push Apps Manager** deploys the Apps Manager, a dashboard for managing apps, services, orgs, users, and spaces. Until you deploy Apps Manager, you must perform these functions through the cf CLI. After Apps Manager has been deployed, we recommend deselecting the checkbox for this errand on subsequent Elastic Runtime deployments. For more information about the Apps Manager, see [Getting Started with the Apps Manager](#).
- **Run Smoke Tests** verifies that your deployment can do the following:
 - Push, scale, and delete apps
 - Create and delete orgs and spaces
- **Push App Usage Service** deploys the Usage Service, a dashboard that provides resource usage data and analytics for your apps.
- **Notifications** deploys an API for sending email notifications to your PCF platform users.

Note: The Notifications app requires that you [configure SMTP](#) with a username and password, even if [SMTP Authentication Mechanism](#) is set to none.

- **Notifications with UI** deploys a dashboard for users to manage notification subscriptions.
 - **Deploy CF Autoscaling App** enables your deployment to automatically scale the number of instances of an app in response to changes in its usage load. To enable Autoscaling for an app, you must also bind the Autoscaling service to it. For more information, see the [Bind a Service Instance](#) section of the *Managing Service Instances with the CLI* topic.
- Note:** The Autoscaling app requires the Notifications app to send scaling action alerts by email.
- **Register Autoscaling Service Broker** makes the Autoscaling service available to your applications. Without this errand, you cannot bind the Autoscaling app to your apps.

Step 15: (Optional) Configure Resources

By default, Elastic Runtime uses an internal filestore and internal databases. If you configure Elastic Runtime to use external resources, you can disable the corresponding system-provided resources in Ops Manager to reduce costs and

administrative overhead.

Complete the following procedures to disable specific VMs in Ops Manager:

1. Click **Resource Config**.

Resource Config		
JOB	INSTANCES	EPHEMERAL DISK (MB)
NATS	1	2048
consul	1	2048
etcd	1	2048
Diego etcd	1	2048
NFS Server	0	2048
Router	1	2048
MySQL Proxy	0	4096
MySQL Server	0	30000
Cloud Controller Database (Postgres)	0	2048
UAA Database (Postgres)	0	2048
Apps Manager Database (Postgres)	0	2048
Cloud Controller	1	20480
HAProxy	0	2048

2. If you configure Elastic Runtime to use an external S3-compatible filestore, edit the following fields:
 - **NFS Server:** Enter in **Instances**.
3. If you configure Elastic Runtime to use an external Relational Database Service (RDS), edit the following fields:
 - **MySQL Proxy:** Enter in **Instances**.
 - **MySQL Server:** Enter in **Instances**.
 - **Cloud Controller Database:** Enter in **Instances**.
 - **UAA Database:** Enter in **Instances**.
 - **Apps Manager Database:** Enter in **Instances**.
4. If you are using an External Load Balancer instead of HAProxy, enter 0 in the **Instances** field for **HAProxy**.
5. Click **Save**.

Step 16: Configure Stemcell

1. Select **Stemcell**. This page displays the stemcell version that shipped with Ops Manager.

The screenshot shows the 'Installation Dashboard' section of the Ops Manager Director. On the left, there's a sidebar with links: 'vCenter Config', 'Director Config', 'Create Availability Zones', 'Assign Availability Zones', 'Create Networks', 'Assign Networks', 'Resource Config', and 'Stemcell'. The 'Stemcell' link is highlighted with a green arrow pointing to it. The main content area has a title 'Stemcell' and a sub-section titled 'A stemcell is a template from which Ops Manager creates the VMs needed for a wide variety of components and products.' Below this, it says 'microbosh requires BOSH stemcell version 2859 ubuntu-trusty' and provides a download link: 'Using bosh-stemcell-2859-vsphere-esxi-ubuntu-trusty-go_agent.tgz'. A large 'Import Stemcell' button is at the bottom of this section.

You can also use this page to import a new stemcell version. You only need to import a new Stemcell if your Ops Manager does not already have the Stemcell version required by Elastic Runtime.

2. Click the **Installation Dashboard** link to return to the Installation Dashboard.

[Return to Installing Pivotal Cloud Foundry on vSphere and vCloud Air](#)

Using Ops Manager Resurrector on VMware vSphere

The Ops Manager Resurrector increases Pivotal Cloud Foundry Elastic Runtime availability in the following ways:

- Reacts to hardware failure and network disruptions by restarting virtual machines on active, stable hosts
- Detects operating system failures by continuously monitoring virtual machines and restarting them as required
- Continuously monitors the BOSH Agent running on each virtual machine and restarts the VMs as required

The Ops Manager Resurrector continuously monitors the status of all virtual machines in an Elastic Runtime deployment. The Resurrector also monitors the BOSH Agent on each VM. If either the VM or the BOSH Agent fail, the Resurrector restarts the virtual machine on another active host.

Limitations

The following limitations apply to using the Ops Manager Resurrector:

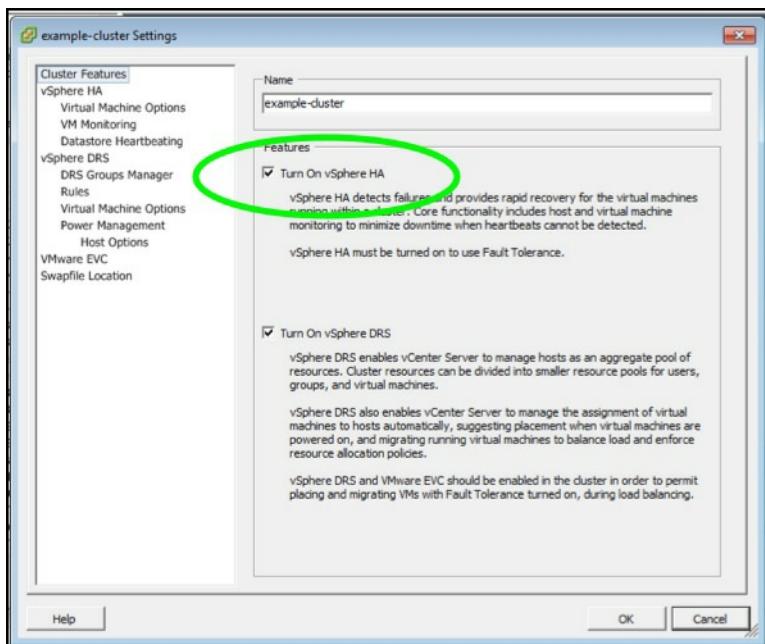
- The Resurrector does not monitor or protect the Ops Manager VM or the BOSH Director VM.
- The Resurrector might not be able to resolve issues caused by the loss of an entire host.
- The Resurrector does not monitor or protect data storage.

For increased reliability, Pivotal recommends that you use vSphere High Availability to protect all of the VMs in your deployment, and that you use a highly-available storage solution.

Enabling vSphere High Availability

Follow the steps below to enable vSphere High Availability:

1. Launch the vSphere Management Console.
2. Right-click the cluster that contains the [Pivotal Cloud Foundry](#) deployment and select **Edit Settings**.
3. Check the **Turn on vSphere High Availability** checkbox.



4. Click **OK** to enable vSphere High Availability on the cluster.

Enabling Ops Manager Resurrector

To enable the Ops Manager Resurrector:

1. Log into the Ops Manager web interface.
2. On the Product Dashboard, select **Ops Manager Director**.



3. In the left navigation menu, select **System Settings**.
4. Check **Enable VM resurrector plugin**. Click **Save**.

A screenshot of the Director Config page under System Settings. On the left, there's a sidebar with several options: "vCenter Config" (selected), "Director Config" (highlighted with a grey arrow pointing to the right), "Create Availability Zones", "Assign Availability Zones", "Create Networks", and "Assign Networks". On the right, the "Director Config" section is displayed. It has two input fields: "NTP Servers (comma delimited)*" containing "time.apple.com" and "Metrics IP Address" which is empty. At the bottom right of the form, there is a checked checkbox labeled "Enable VM Resurrector Plugin".

5. Ops Manager Resurrector is now enabled.

Configuring PCF SSL Termination for vSphere Deployments

To use SSL termination in [Pivotal Cloud Foundry](#) (PCF), you must configure the Pivotal-deployed HAProxy load balancer or your own load balancer.

We recommend using HAProxy in lab and test environments only. Production environments should instead use a highly-available customer-provided load balancing solution.

Select an SSL termination method to determine the steps you must take to configure Elastic Runtime.

Using the Pivotal HAProxy Load Balancer

PCF deploys with a single instance of HAProxy for use in lab and test environments. You can use this HAProxy instance for SSL termination and load balancing to the PCF Routers. HAProxy can generate a self-signed certificate if you do not want to obtain a signed certificate from a well-known certificate authority.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

To use the HAProxy load balancer, you must create a wildcard A record in your DNS and configure three fields in the Elastic Runtime product tile.

1. Create an A record in your DNS that points to the HAProxy IP address. The A record associates the **System Domain** and **Apps Domain** that you configure on the Cloud Controller page in Ops Manager with the HAProxy IP address.

For example, with `cf.example.com` as the main subdomain for your CF install and an HAProxy IP address `172.16.1.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `172.16.1.1`.

Name	Type	Data	Domain
<code>*.cf</code>	A	<code>176.16.1.1</code>	<code>example.com</code>

2. Use the Linux `host` command to test your DNS entry. The `host` command should return your HAProxy IP address.

Example:

```
$ host cf.example.com
cf.example.com has address 172.16.1.1
$ host anything.example.com
anything.cf.example.com has address 172.16.1.1
```

3. From the PCF Ops Manager Dashboard, click on the Elastic Runtime tile.
4. Select **IPs and Ports**.
5. Enter the IP address for HAProxy in the **HAProxy IPs** field.
6. Leave the **Router IPs** field blank. HAProxy assigns the router IPs internally.
7. Provide your SSL certificate on the **Security Config** page. See [Providing a Certificate for your SSL Termination Point](#).

[Return to the Getting Started Guide](#)

Using Another Load Balancer

Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides SSL termination with wildcard DNS location
- Provides load balancing to each of the PCF Router IPs

- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers

You must register static IP addresses for PCF with your load balancer and configure three fields in the Elastic Runtime product tile.

1. Register one or more static IP address for PCF with your load balancer.
2. Create an A record in your DNS that points to your load balancer IP address. The A record associates the **System Domain** and **Apps Domain** that you configure on the Cloud Controller page in Ops Manager with the IP address of your load balancer.

For example, with `cf.example.com` as the main subdomain for your CF install and a load balancer IP address `10.10.1.1`, you must create an A record in your DNS that serves `example.com` and points `*.cf` to `10.10.1.1`.

Name	Type	Data	Domain
<code>*.cf</code>	A	<code>10.10.1.1</code>	<code>example.com</code>

3. From the PCF Ops Manager Dashboard, click on the Elastic Runtime tile.
4. Select **IPs and Ports**.
5. Leave the **HAProxy IPs** field blank.
6. In the **Router IPs** field, enter the static IP address for PCF that you have registered with your load balancer.
7. Provide your SSL certificate on the **Security Config** page. See [Providing a Certificate for your SSL Termination Point](#).

 **Note:** When adding or removing PCF routers, you must update your load balancing solution configuration with the appropriate IP addresses.

[Return to the Getting Started Guide](#)

Identifying Elastic Runtime Jobs Using vCenter

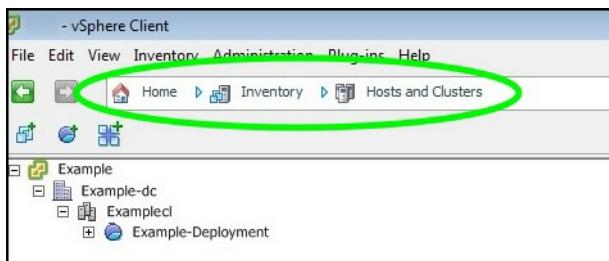
To effectively monitor, control, and manage the virtual machines making up your Elastic Runtime deployment, you may need to identify which VM corresponds to a particular job in Elastic Runtime. You can find the CID of a particular VM from [Pivotal Cloud Foundry](#) Operations Manager by navigating to **Elastic Runtime > Status**.

If you have deployed Elastic Runtime to VMware vSphere, you can also identify which Elastic Runtime job corresponds to which VM using the vCenter vSphere client. This option is not available if you have deployed Elastic Runtime to VMware vCloud Air / vCloud.

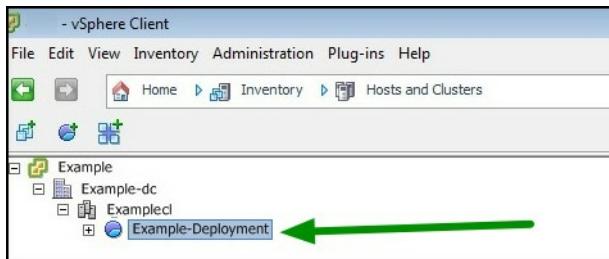
 **Note:** The CID shown in Ops Manager is the name of the machine in vCenter.

Identifying Elastic Runtime Jobs Using vCenter

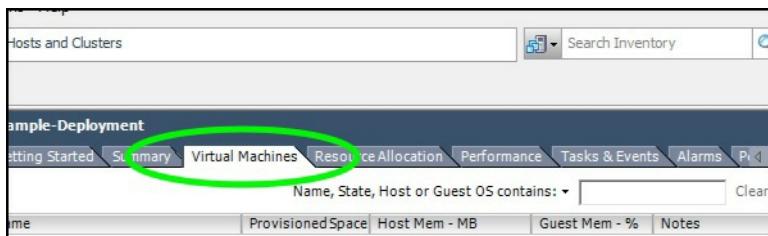
1. Launch the vSphere client and log in to the vCenter Server system.
2. Select the **Inventory > Hosts and Clusters** view.



3. Select the Resource Pool containing your Elastic Runtime deployment.



4. Select the **Virtual Machines** tab.



5. Right-click the column label heading and check **job**.

The screenshot shows the 'Example-Deployment' section of the Pivotal Ops Manager interface. On the left, there's a tree view under 'Status' with nodes like 'Completed' and 'Completed'. The main pane displays a table of virtual machines with columns: Name, Provisioned Space, Host Mem - MB, Guest Mem - MB, Guest OS, VM Version, Memory Size, Reservation - MB, CPU Count, NIC Count, Uptime, IP Address, VMware Tools Running Status, and VMware Tools Version Status. A green arrow points from the 'job' node in the status tree to the 'job' column in the table.

Name	Provisioned Space	Host Mem - MB	Guest Mem - MB	Guest OS	VM Version	Memory Size	Reservation - MB	CPU Count	NIC Count	Uptime	IP Address	VMware Tools Running Status	VMware Tools Version Status
vm-5d98bbf5-813f-4612-8	9.09 GB	1040											
vm-a98b916f-c295-4a41-8	11.09 GB	597											
Pivotal Ops Manager(zero)	151.09 GB	1036											
vm-3e13d9e3-2323-457d-8	10.09 GB	598											
vm-56012bf-e627-4651-8	19.09 GB	1023											
sc-a49692cf-18e2-41ed-9	6.69 GB	0											
vm-a0aaea51-6b7d-4639-8	9.09 GB	1045											
vm-d26d43a2-11f9-4b52-8	9.09 GB	470											
vm-e5bfff2f-e357-44ea-9	9.09 GB	468											
vm-8e5030ae-f9d1-4548-8	20.09 GB	3130											
sc-a1b24b3-dca3-4418-8	6.69 GB	0											
vm-95662aee-5ec5-4e62-9	9.09 GB	551											
sc-449692cf-18e2-41ed-9	6.69 GB	0											
sc-b095e033-bba7-4e2d-8	6.69 GB	0											

6. The job column displays the Elastic Runtime job associated with each virtual machine.

This screenshot shows the same interface as the previous one, but the 'job' column is now populated with specific job names for each virtual machine. A green arrow points from the 'job' column to the 'job' node in the status tree. The table includes columns for Name, job, and other metrics.

Name	job
vm-347a89c6-115f-433a-83d1-38cd2c31451b	ccdb
sc-d3520bac-11c2-460f-aa02-60fc11cba375	cloud_controller
vm-d537f816-fc51-450c-8ada-cb536a9e76f5	consoledb
vm-85d43147-7cd2-4cc5-ac6c-47cce18ccd69	dea
vm-5c04a27a-ca70-4f67-a221-767efb76922a	ha_proxy
vm-e9b3e319-690f-44be-87fb-c9e7f90c9b2	health_manager
vm-af8bd3d1-7e31-47f3-800f-3f81ff23ed81	loggregator
vm-ee8b5374-bbc9-436b-bfe5-1a7e2b5fee9a	loggregator_trafficcontroller
vm-5896908f-15c5-4989-8c90-36a36e0eb4fd	nats
vm-61fc0d19-01f5-4434-82e0-a11e15b38848	nfs_server
vm-e88725cd-36fe-4a79-b8ea-f35b2d2cd85b	riak-cs
vm-9ab66b39-4cae-443a-842c-3651c9bf228c	router
vm-2726dcaa-7f92-440c-906e-3e5e544fe628	saml_login
vm-273df76f-7141-432d-9e53-9cb6549d870a	uaa
vm-307a34f4-ed7f-4c48-94d1-bb7a1a695bc	uaadb
vm-e6c2dfcd-ef2e-40d8-a0b-06e5f3e79fa9	
vm-0e9e9384-0057-4efc-9cea-234b5956c510	
vm-5e33e793-0118-44bb-948d-3ea526b4b2f	

Troubleshooting Ops Manager for VMware vSphere

This guide provides help with diagnosing and resolving issues that are specific to [Pivotal Cloud Foundry](#) (PCF) deployments on VMware vSphere.

For infrastructure-agnostic troubleshooting help, refer to the [PCF Troubleshooting Guide](#).

Common Issues

The following sections list common issues you might encounter and possible resolutions.

PCF Installation Fails

If you modify the vCenter Statistics Interval Duration setting from its default setting of 5 minutes, the PCF installation might fail at the MicroBOSH deployment stage, and the logs might contain the following error message:

`[The specified parameter is not correct, interval.]` This failure happens because Ops Manager expects a default value of 5 minutes, and the call to this method fails when the retrieved value does not match the expected default value.

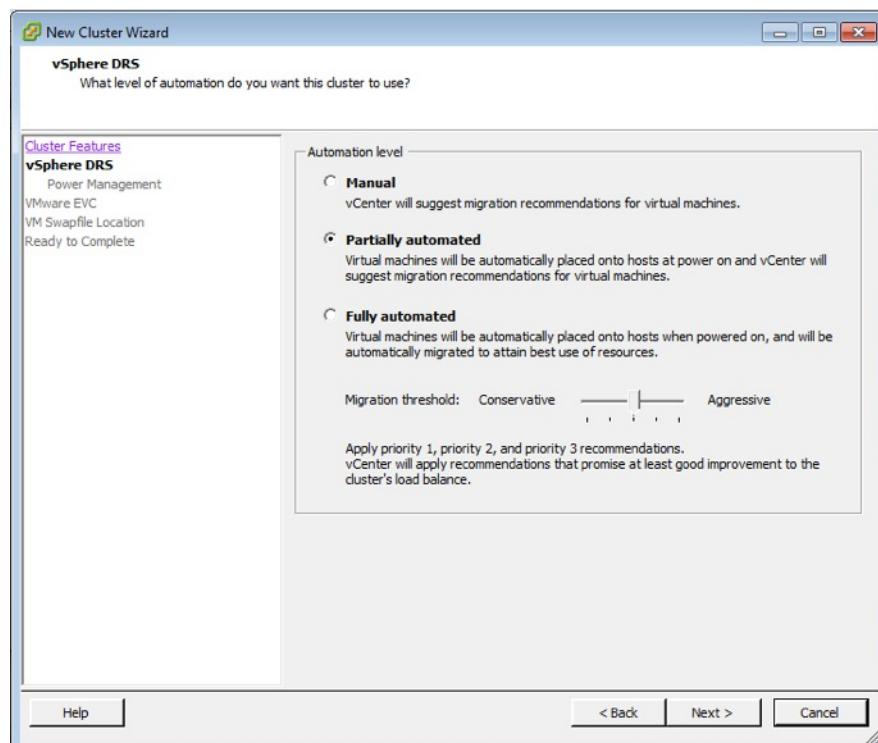
To resolve this issue, launch vCenter, navigate to **Administration > vCenter Server Settings > Statistics**, and reset the vCenter Statistics Interval Duration setting to 5 minutes.

BOSH Automated Installation Fails

Before starting an Elastic Runtime deployment, you must set up and configure a vSphere cluster.

If you enable vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to **Partially automated** or **Fully automated**.

If you set the Automation level to **Manual**, the BOSH automated installation will fail with a `[power_on_vm]` error when BOSH attempts to create virtual VMs.



Ops Manager Loses Its IP Address After HA or Reboot

Ops Manager can lose its IP address and use DHCP due to an issue in the open source version of VMware Tools. To resolve this issue you must log in to the Ops Manager VM and edit `/etc/network/interfaces` as follows:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

# Generated with set_dynamic_ip script
# iface eth0 inet static

# Manually configured
iface eth0 inet static
    address 10.70.128.16
    netmask 255.255.255.0
    network 10.70.128.0
    broadcast 10.70.128.255
    gateway 10.70.128.1

    dns-nameservers 10.70.0.3
```

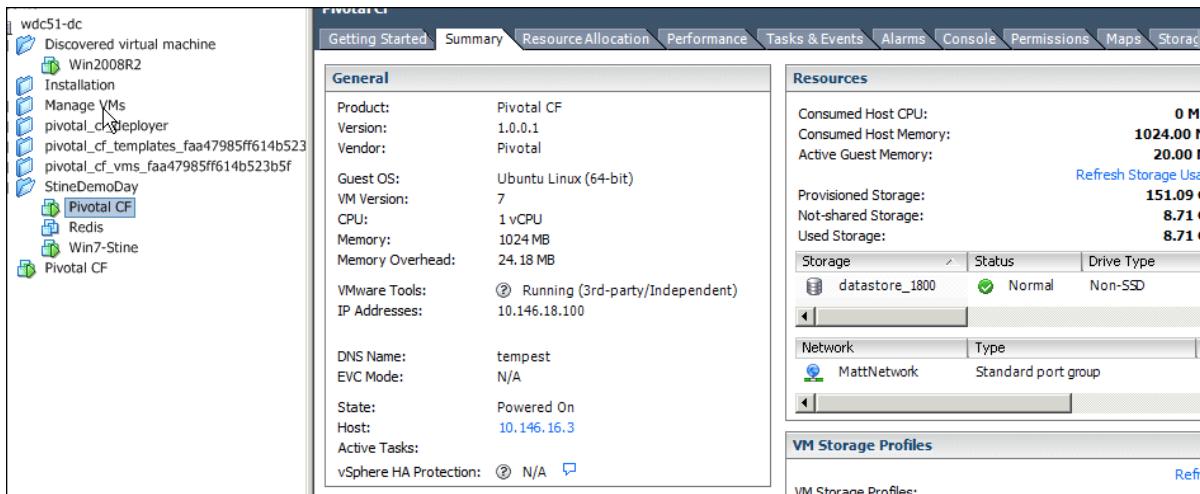
After editing the file run `sudo /etc/init.d/networking restart`. The interfaces files will be overwritten on reboot.

Cannot Connect to the OVF Via a Browser

If you deployed the OVF file but cannot connect to it via a browser, check that the network settings you entered in the wizard are correct.

1. Access the PCF installation VM using the vSphere Console. If your network settings are misconfigured, you will not be able to SSH into the installation VM.
2. Log in using the credentials you provided when you imported the PCF .ova in vCenter.
3. Confirm that the network settings are correct by checking that the ADDRESS, NETMASK, GATEWAY, and DNS-NAMESERVERS entries are correct in `/etc/network/interfaces`.
4. If any of the settings are wrong, run `sudo vi /etc/network/interfaces` and correct the wrong entries.

5. In vSphere, navigate to the **Summary** tab for the VM and confirm that the network name is correct.



6. If the network name is wrong, right click on the VM, select **Edit Settings > Network adapter 1**, and select the correct network.
7. Reboot the installation VM.

Installation Fails with Failed Network Connection

If you experience a communication error while installing Ops Manager or MicroBOSH Director, check the following settings.

- Ensure that the routes are not blocked. vSphere environments use [VCNS](#). All communication between PCF VMs and vCenter or ESXi hosts route through the VCNS firewall and are blocked by default.
- Open port 443. Ops Manager and MicroBOSH Director VMs require access to vCenter and all ESX through port 443.
- Allocate more IP addresses. BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.

Using Operations Manager

Operations Manager is a web application that you use to deploy and manage a [Pivotal Cloud Foundry](#) PaaS. This is a guide to deploying and using Ops Manager.

Before You Deploy

- [Prerequisites to Deploying Operations Manager and Elastic Runtime](#)
- [Preparing Your Firewall for Deploying PCF](#)
- [Using Your Own Load Balancer](#)
- [Using Ops Manager Resurrector on VMware vSphere](#)
- [Understanding Availability Zones and Regions in Pivotal Cloud Foundry](#)
- [Configuring Network Isolation Options in Ops Manager](#)

Deploying Operations Manager

- [Deploying Operations Manager to vSphere](#)
- [Provisioning a Virtual Disk in vSphere](#)
- [Deploying Operations Manager to vCloud Air and vCloud](#)
- [Installing Pivotal Cloud Foundry on OpenStack](#)
- [Provisioning the OpenStack Infrastructure](#)
- [Installing Pivotal Cloud Foundry on AWS](#)
- [Deploying the CloudFormation Template for PCF on AWS](#)
- [Launching an Ops Manager Director Instance on AWS using CloudFormation](#)
- [Manually Configuring AWS for PCF](#) (Warning: Not Recommended)
- [Using the Cisco Nexus 1000v Switch with Ops Manager](#)

Operations Manager API

Use the Ops Manager API to automate any Ops Manager task. To view the Ops Manager API documentation, browse to <https://YOUR-OPS-MANAGER-IP-ADDRESS/docs>.

 **Warning:** The Ops Manager API is an experimental feature that is not fully implemented and could change without notice. Pivotal is developing an officially supported Ops Manager API that will replace many of these endpoints in a subsequent release.

- [Identifying the API Endpoint for your Elastic Runtime Instance](#)

Using Operations Manager and Installed Products

- [Understanding the Ops Manager Interface](#)
- [Adding and Deleting Products](#)
- [Configuring Ops Manager Director on AWS](#)
- [Configuring Ops Manager Director for VMware vSphere](#)
- [Configuring Ops Manager Director for vCloud Air and vCloud](#)
- [Configuring Ops Manager Director after Deploying PCF on OpenStack](#)
- [Configuring Elastic Runtime for AWS](#)
- [Configuring Elastic Runtime for vSphere and vCloud](#)
- [Installing Elastic Runtime after Deploying PCF on OpenStack](#)
- [Starting and Stopping PCF VMs](#)

- [Creating New Elastic Runtime User Accounts](#)
- [Logging into the Apps Manager](#)
- [Controlling Apps Manager User Activity with Environment Variables](#)
- [Configuring Your App Autoscaling Instance](#)
- [Managing Scheduled Scaling in the App Autoscaling Service](#)
- [Deleting an AWS Installation from the Console](#)

Backing Up and Upgrading

- [Backing Up Pivotal Cloud Foundry](#)
- [Restoring Pivotal Cloud Foundry from Backup](#)
- [Upgrading Operations Manager](#)
- [Upgrading Products in a PCF Deployment](#)

Monitoring, Logging, and Troubleshooting

- [Monitoring VMs in PCF](#)
- [Deploying Pivotal Ops Metrics](#)
- [Using SSL with a Self-Signed Certificate in Pivotal Ops Metrics](#)
- [Using Pivotal Ops Metrics](#)
- [PCF Troubleshooting Guide](#)
- [Troubleshooting Ops Manager for VMware vSphere](#)
- [Advanced Troubleshooting with the BOSH CLI](#)
- [PCF Security Overview and Policy](#)

Prerequisites to Deploying Operations Manager and Elastic Runtime

This topic explains system requirements for deploying the Pivotal Operations Manager and Elastic Runtime applications.

vSphere/vCenter Requirements

The following are the minimum resource requirements for maintaining a [Pivotal Cloud Foundry](#) deployment with Ops Manager and Elastic Runtime on vSphere:

- vSphere 6.0, 5.5, or 5.1.
- vSphere editions: standard and above.
- Ops Manager Director must have HTTPS access to vCenter and ESX hosts on TCP ports 443, 902, and 903.
- A configured vSphere cluster:
 - If you enable vSphere DRS (Distributed Resource Scheduler) for the cluster, you must set the Automation level to **Partially automated** or **Fully automated**. If you set the Automation level to **Manual**, the BOSH automated installation will fail with a `power_on_vm` error when BOSH attempts to create virtual VMs.
 - Turn hardware virtualization off if your vSphere hosts do not support VT-X/EPT. If you are unsure whether the VM hosts support VT-x/EPT, then you can turn this setting off. If you leave this setting on and the VM hosts do not support VT-x/EPT, then each VM requires manual intervention in vCenter to continue powering on without the Intel virtualized VT-x/EPT. Refer to the vCenter help topic at [Configuring Virtual Machines > Setting Virtual Processors and Memory > Set Advanced Processor Options](#) for more information.
- Ops Manager requires read/write permissions to the datacenter level of the [vSphere Inventory Hierarchy](#) to successfully install. Pivotal recommends using the default [VMware Administrator System Role](#) to achieve the appropriate permission level, or a custom role that has all privileges for all objects in the datacenter, including propagating privileges to children. Be advised that Ops Manager might indicate that you do not have the appropriate rights to create/delete folders when this is untrue. If so, click **Ignore errors and start the install** to continue.

Note: If you are using the Cisco Nexus 1000v Switch, refer to the [Using the Cisco Nexus 1000v Switch with Ops Manager](#) topic for more information.

Note: When installing Ops Manager on a vSphere environment with multiple ESXi hosts, you must use network-attached or shared storage devices. Local storage devices do not support sharing across multiple ESXi hosts.

vCD/vCloud Air Requirements

The following are the minimum resource requirements for maintaining a [Pivotal Cloud Foundry](#) deployment with Ops Manager and Elastic Runtime on vCloud Air:

- vCD 5.1, 5.2, or 5.6 (vCloud Air)
- Disk space: 120GB
- Memory: 60GB
- Two public IP addresses: One for Elastic Runtime and one for Ops Manager
- vCPU cores: 28
- Overall CPU: 17 GHz
- Virtual infrastructure administrator privileges to enable Elastic Runtime to automatically power VMs on and off

Note: For more information about user privileges, refer to the “User Privileges by Role” section in the [VMware vCloud Air User’s Guide](#).

Amazon Web Services

The following are the minimum resource requirements for maintaining a [Pivotal Cloud Foundry](#) deployment with Ops Manager and Elastic Runtime on Amazon Web Services infrastructure:

- 1 Elastic Load Balancer
- 1 Relational Database Service. We recommend at least a db.m3.large instance with 100 GB of allocated storage.
- 2 S3 Buckets
- EC2 Instances:
 - 10 t2.micros
 - 1 r3.xlarge (1 per DEA)
 - 1 c4.xlarge
 - 2 m3.large

See [Installing Pivotal Cloud Foundry on AWS using CloudFormation](#) for more detailed installation requirements.

OpenStack

Pivotal has tested and certified Pivotal Cloud Foundry on Mirantis OpenStack versions 5.1 (IceHouse) and 6.1 (Juno). Other OpenStack releases and distributions based on Havana, Icehouse, and Juno may also function properly.

See [Installing Pivotal Cloud Foundry on OpenStack](#) for detailed requirements.

General Requirements

The following are general requirements for deploying and managing the Pivotal Operations Manager and Elastic Runtime applications:

- The following user privileges:
 - Datastore (Allocate space, Browse datastore, Low-level file operations, Remove file, Update virtual machine files)
 - Folder (All)
 - Network (Assign network)
 - Resource (All)
 - vApp (All)
 - Virtual machine (All)
- **(Recommended)** Ability to create a wildcard DNS record to point to your router or load balancer. Alternatively, you can use a service such as xip.io. (Example: 172.16.64.xip.io).

Elastic Runtime gives each application its own hostname in your app domain. With a wildcard DNS record, every hostname in your domain resolves to the IP address of your router or load balancer, and you do not need to configure an A record for each app hostname. For example, if you create a DNS record `*.example.com` pointing to your router, every application deployed to the `example.com` domain resolves to the IP address of your router.

- **(Recommended)** A network without DHCP available for deploying the Elastic Runtime VMs.

 **Note:** If you have DHCP, refer to the Troubleshooting Guide to avoid issues with your installation.

- Sufficient IP Allocation:
 - One IP address for each VM instance.
 - An additional IP address for each instance that requires static IPs.
 - An additional IP address for each errand.
 - An additional IP address for each compilation worker. IPs needed = VM instances + static IPs + errands + compilation workers

 **Note:** BOSH requires that you allocate a sufficient number of additional dynamic IP addresses when configuring a reserved IP range during installation. BOSH uses these IPs during installation to compile and deploy VMs, install Elastic Runtime, and connect to services. We recommend that you allocate at least 36 dynamic IP addresses when deploying Ops Manager and Elastic Runtime.

- The cf command line interface (cf CLI) tool version 6.1.1 or higher.
- One or more NTP servers.
- Capacity for the following virtual machines:

Virtual Machine	Instances	CPU	RAM (MB)	Ephemeral Disk (MB)	Persistent Disk (MB)	Static IP
Message Bus (NATS)	1	1	1024	1024	0	✓
consul	1	1	1024	1024	1024	✓
etcd	1	1	1024	1024	1024	✓
Blob Store (NFS Server)	1	1	1024	2048	10240	✓
Cloud Controller Database	1	1	1024	2048	2048	✓
OAuth2 Server Database (UAA Database)	1	1	1024	2048	8192	✓
Apps Manager Database	1	1	1024	2048	1024	✓
Cloud Controller	1	1	4096	20480	0	
HAProxy	0	1	1024	2048	0	✓
Router	1	1	1024	2048	0	✓
Health Manager	1	1	1024	1024	0	
Clock Global	1	1	1024	2048	0	
Cloud Controller Worker	1	1	1024	2048	0	
Collector	0	1	1024	2048	0	
OAuth2 Server (UAA)	1	1	1024	2048	0	
MySQL Proxy	0	1	1024	4096	0	✓
MySQL Server	1	1	4096	10000	10000	✓
Application Execution (DEA)	1	1	4096	10240	0	
Doppler Server	1	1	1024	2048	0	✓
Loggregator Trafficcontroller	1	1	1024	2048	0	✓
Push Apps Manager	1	1	1024	1024	0	
Push App Usage Service	1	1	1024	1024	0	
Run Smoke Tests	1	1	1024	1024	0	
Notifications with UI	1	1	1024	2048	0	
Deploy CF Autoscaling App	1	1	512	1024	0	
Register Autoscaling Service Broker	1	1	512	1024	0	
Destroy Autoscaling Service Broker	1	1	512	1024	0	
Run CF Acceptance Tests	1	1	512	1024	0	
Run CF Acceptance Tests without internet	1	1	1024	1024	0	
Compilation	1	1	1024	20480	0	
TOTALS	27	30	37 GB	102 GB	33 GB	13

[Return to the Getting Started Guide](#)

Preparing Your Firewall for Deploying PCF

This topic describes how to configure your firewall for [Pivotal Cloud Foundry](#) and how to verify that PCF resolves DNS entries behind your firewall.

Configure Your Firewall for PCF

Ops Manager and Elastic Runtime require the following open TCP ports:

- **25555**: Routes to the Ops Manager VM
- **443**: Routes to HAProxy or, if configured, your own load balancer
- **80**: Routes to HAProxy or, if configured, your own load balancer
- **22 (Optional)**: Only necessary if you want to connect using SSH

For more information about required ports for additional installed products, refer to the product documentation.

The following example procedure uses iptables commands to configure a firewall.

Note: `GATEWAY_EXTERNAL_IP` is a placeholder. Replace this value with your `PUBLIC_IP`.

1. Open `/etc/sysctl.conf`, a file that contains configurations for Linux kernel settings, with the command below:

```
$ sudo vi /etc/sysctl.conf
```

2. Add the line `net.ipv4.ip_forward=1` to `/etc/sysctl.conf` and save the file.

3. If you want to remove all existing filtering or Network Address Translation (NAT) rules, run the following commands:

```
$ iptables --flush  
$ iptables --flush -t nat
```

4. Add environment variables to use when creating the IP rules:

```
$ export INTERNAL_NETWORK_RANGE=10.0.0.0/8  
$ export GATEWAY_INTERNAL_IP=10.0.0.1  
$ export GATEWAY_EXTERNAL_IP=88.198.252.242  
$ export PIVOTALCF_IP=10.0.0.2  
$ export HA_PROXY_IP=10.0.0.254
```

5. Run the following commands to configure IP rules for the specified chains:

- **FORWARD:**

```
$ iptables -A FORWARD -i eth1 -j ACCEPT  
$ iptables -A FORWARD -o eth1 -j ACCEPT
```

- **POSTROUTING:**

```
$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE  
$ iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \  
    -p tcp --dport 80 -j SNAT --to $GATEWAY_INTERNAL_IP  
$ iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \  
    -p tcp --dport 443 -j SNAT --to $GATEWAY_INTERNAL_IP
```

- **PREROUTING:**

```
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
25555 -j DNAT --to $PIVOTALCF_IP
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
443 -j DNAT --to $HA_PROXY_IP
$ iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport \
80 -j DNAT --to $HA_PROXY_IP
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8443 -j DNAT \
--to $PIVOTALCF_IP:443
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
--to $HA_PROXY_IP:80
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8022 -j DNAT \
--to $PIVOTALCF_IP:22
$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8080 -j DNAT \
--to $PIVOTALCF_IP:80
```

- Run the following command to save the iptables:

```
$ service iptables save
```

For more information about administering IP tables with `iptables`, refer to the [iptables documentation](#).

Verify PCF Resolves DNS Entries Behind a Firewall

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. For example, if you use [xip.io](#) to test your DNS configuration, the tests will fail without warning if the firewall prevents Elastic Runtime from accessing `*.xip.io`.

To verify that Elastic Runtime can correctly resolve DNS entries:

- SSH into the Pivotal Ops Manager VM.
For more information, refer to the [SSH into Ops Manager](#) section of the Advanced Troubleshooting with the BOSH CLI topic.
- Run any of the following network administration commands with the IP address of the VM:
 - `nslookup`
 - `dig`
 - `host`
 - The appropriate `traceroute` command for your OS
- Review the output of the command and fix any blocked routes.
If the output displays an error message, review the firewall logs to determine which blocked route or routes you need to clear.
- Repeat steps 1-3 with the Ops Manager Director VM and the HAProxy VM.

Using Your Own Load Balancer

This guide describes how to use your own load balancer and forward traffic to your Elastic Runtime router IP address.

Pivotal Cloud Foundry (PCF) deploys with a single instance of HAProxy for use in lab and test environments. Production environments should use a highly-available customer-provided load balancing solution that does the following:

- Provides load balancing to each of the PCF Router IPs
- Supports SSL termination with wildcard DNS location
- Adds appropriate `x-forwarded-for` and `x-forwarded-proto` HTTP headers to incoming requests
- **(Optional)** Supports WebSockets

Note: Application logging with [Loggregator](#) requires WebSockets. To use another logging service, see the [Using Third-Party Log Management Services](#) topic.

Prerequisites

To integrate your own load balancer with PCF, you must ensure the following:

- WebSocket connections are not blocked for Loggregator functionality.
- The load balancer must be able to reach the gorouter IPs.

Follow the instructions below to use your own load balancer.

Step 1: Deploy PCF Installation VM

Deploy a PCF Installation virtual machine. The procedure you follow depends on the IaaS you use:

- [Deploying Operations Manager to vSphere](#)
- [Deploying Operations Manager to vCloud Air and vCloud](#)

Step 2: Register PCF IP Address

In your load balancer, register the IP addresses that you assigned to PCF.

Step 3: Configure Pivotal Ops Manager and Ops Manager Director

Configure your Pivotal Operations Manager and Ops Manager Director as described in [Getting Started with Pivotal Cloud Foundry](#), then add Elastic Runtime.

Do not click **Install** after adding Elastic Runtime.

Step 4: Configure IPs

1. In Pivotal Operations Manager, click the **Elastic Runtime** tile.
2. In the left column, select **IPs and Ports** if not already selected.
3. In the **HAProxy IPs** field, delete any existing IP addresses. This field should be blank.



4. Select **Router IPs**.
 5. In the **Router IPs** field, enter the IP address or addresses for PCF that you registered with your load balancer in Step 2. Save this change.

<input checked="" type="checkbox"/> System Database Config	Router IPs
	<input type="text" value="192.168.0.1"/>
<input checked="" type="checkbox"/> File Storage Config	

Step 5: Configure Security Settings

- ## 1. Click on **Security Config.**

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config
- IPs and Ports
- Security Config
- MySQL Proxy Config
- Cloud Controller
- System Logging
- SSO Config
- LDAP Config
- SMTP Config
- Diego
- Experimental Features
- Errands
- Resource Config
- Stemcell

Configure security settings

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIICKTCCAZICCCQCYjs8Jtc84cTANBgkqhkiG9w0BAQUFADBZMQswCQYDVQQGEwJV
-----END CERTIFICATE-----
```

-----BEGIN RSA PRIVATE KEY-----
MIGfMA0GCSqGSIb3DQEBCwQADQHgQC5CWbiXc86ffy4mpAf6PmxQTn36d+bj8fUDo6NZQ7B0twTW5iQ
-----END RSA PRIVATE KEY-----

[Generate Self-Signed RSA Certificate](#)

Ignore SSL certificate verification

HAProxy SSL Ciphers

Router SSL Ciphers

Disable HTTP traffic to HAProxy

Enable cross-container traffic

Enable TLS on the Router

[Save](#)

2. Provide an **SSL Termination Certificate** for your SSL Termination Point.

- In a production environment, use a signed **SSL Certificate** from a known certificate authority (CA). Copy and paste the values for **Certificate PEM** and **Private Key PEM** from the signed certificate into the appropriate

text fields.

- In a development or testing environment, you may use a self-signed certificate.

Note: Pivotal does not recommend using a self-signed certificate for production deployments.

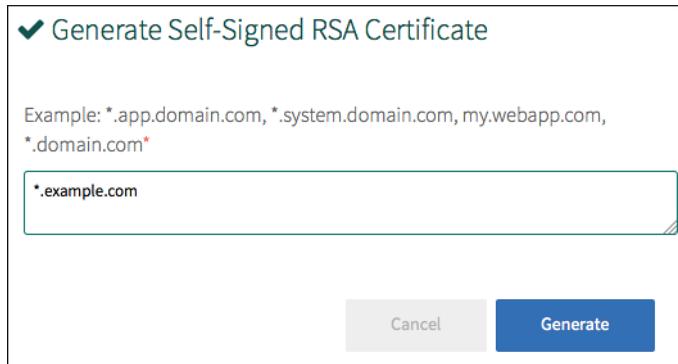
Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

To use a self-signed certificate, follow the steps below:

- * Click the **Generate Self-Signed RSA Certificate** link to launch the RSA certificate wizard.
- * Enter your system and app domains in wildcard format. Optionally, also add any custom domains in wildcard format. The example below uses `*.example.com`.

Note: Wildcard DNS records only work for a single domain name component or component fragment. For example, `*.domain.com` works for `apps.domain.com` and `system.domain.com`, but not for `apps.domain.com` or `system.domain.com`.

Note: You can generate a single certificate for two domains separated by a comma. For example, `.apps.domain.com, *.system.domain.com`.



* Click *Generate.

* Elastic Runtime populates

the **SSL Certificate fields with RSA certificate and private key information.

3. Configure **Ignore SSL certificate verification**. Select this option if you are using self-signed certificates or certificates generated from Ops Manager.
4. Configure **HAProxy SSL Ciphers** and **Router SSL Ciphers**. Leave these fields blank unless you want to use a specific set of SSL ciphers for HAProxy or the Router.
5. Configure **Disable HTTP traffic to HAProxy**. If you select the **Disable HTTP traffic to HAProxy** checkbox, your deployment rejects all port 80 traffic to HAProxy. Additionally, this option sets the secure flag in the `VCAP_ID` cookie that the Router generates.

Note: If you enable this checkbox and your deployment is not using HAProxy, configure your external load balancer to reject port 80 traffic. If you do not do this, traffic to port 80 is forwarded to applications but loses session stickiness.

6. Configure **Enable cross container traffic**. By selecting this checkbox, you disable the restriction that prevents containers in the same DEA or Diego Cell from communicating with each other. Pivotal does not recommend selecting this checkbox in multi-tenant environments. You should select this option for microservices such as Pivotal Spring Cloud.
7. Configure **Enable TLS on the Router**. Selecting this enables [SSL termination](#) on the Router.
8. Click **Save**.

Step 6: Finalize Changes

1. Return to the **Ops Manager Installation Dashboard**

2. Click **Install**.

Configuring Network Isolation Options in Ops Manager

This topic describes configuring different network isolation options in [Pivotal Cloud Foundry](#) Operations Manager.

Once you have deployed Ops Manager Director, configure the following traffic back to the Director for each segmented network:

- TCP traffic on port 4222
- TCP traffic on port 25250
- TCP and UDP traffic on port 53

Prerequisites

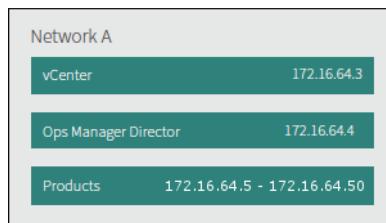
This topic assumes that you have deployed vCenter and created multiple networks.

Option 1: Installing PCF Products to a Common Network

Deploy Ops Manager, Ops Manager Director, and all Pivotal Cloud Foundry (PCF) products to a common network. The vCenter IP must be reachable on or from this network.

 **Note:** Production environments typically require higher security than this option provides.

The following image shows all PCF products and vCenter deployed to a common network.



To configure your PCF deployment for the common network option:

1. Log into the Ops Manager web interface.
2. On the Installation Dashboard, select **Ops Manager Director**.
3. In the left navigation menu, select **Create Networks**.
4. Click **Add**. Add a network that maps to a vCenter network.
The image shows an example network named A.

[Installation Dashboard](#)

Ops Manager Director

- Settings
- Status
- Credentials

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Create Networks

Networks
One or many IP ranges upon which your products will be deployed

A

Name*

vSphere Network Name*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

5. In the left navigation menu, click **Assign Networks**.
6. Click the **Infrastructure Network** drop-down menu and select the network that you created in step 4.
7. Click the **Deployment Network** drop-down menu and select the network that you created in step 4.

[Installation Dashboard](#)

Ops Manager Director

- Settings
- Status
- Credentials

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Assign Networks

The Ops Manager director can be configured to have an IP on two networks. Typically one is routable to the IaaS API (*Infrastructure*) and the other is routable to deployed products (*Deployment*).

Infrastructure Network

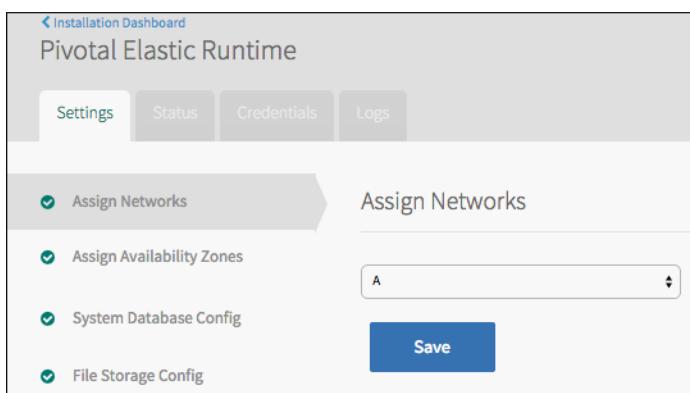
Deployment Network

Save

8. Return to the Installation Dashboard.
9. Perform the following steps for each PCF product, including Pivotal Cloud Foundry Elastic Runtime:
 - a. On the Installation Dashboard, select the PCF product.
 - b. In the left navigation menu, click **Assign Networks**. Assign the product to the network that you created in step 4.

4.

The image shows an example network named A.

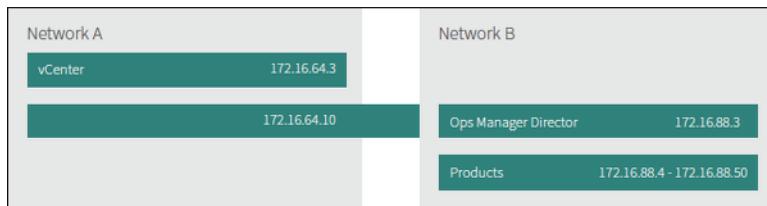


c. Return to the Installation Dashboard.

Option 2: Installing All PCF Products to an Isolated Network

Deploy all PCF products to a Deployment Network. Ensure that you isolate this network from the Infrastructure Network that can reach your vCenter. This configuration provides a higher level of security than [Option 1](#) by reducing the risk of a malicious attack at the IaaS level.

You must configure Ops Manager Director to communicate on two networks, either through routing or by multi-homing. For example, the following image shows vCenter in Network A, all PCF products in Network B, and Ops Manager Director multi-homed in both networks.



To configure your PCF deployment for an isolated network:

1. Log into the Ops Manager web interface.
2. On the Installation Dashboard, select **Ops Manager Director**.
3. In the left navigation menu, select **Create Networks**.
4. Click **Add**. Add a network that maps to a vCenter network.
The image shows an example network named A.

[Installation Dashboard](#)

Ops Manager Director

- Settings
- Status
- Credentials

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

Create Networks

Networks
One or many IP ranges upon which your products will be deployed

A

Name*

vSphere Network Name*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

5. Add another network that maps to a different vCenter network.

The image shows an example network named B.

Create Networks

Networks
One or many IP ranges upon which your products will be deployed

B

Name*

vSphere Network Name*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

- In the Ops Manager Director configuration interface, configure the Director to receive an IP address on both networks, as follows:
 - In the left navigation menu, click **Assign Networks**. Click the **Infrastructure Network** drop-down menu and select the network that you created in step 4.
 - Click the **Deployment Network** drop-down menu and select the network that you created in step 5.

Assign Networks

The Ops Manager director can be configured to have an IP on two networks. Typically one is routable to the IaaS API (*Infrastructure*) and the other is routable to deployed products (*Deployment*).

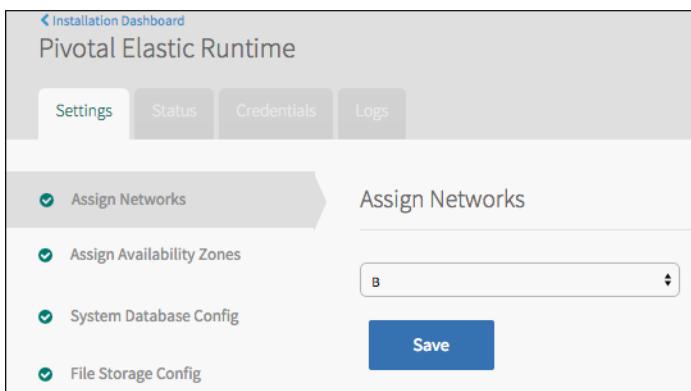
Infrastructure Network

Deployment Network

Save

- Return to the Installation Dashboard.
- Perform the following steps for each PCF product, including Elastic Runtime:

- a. On the Installation Dashboard, select the PCF product.
- b. In the left navigation menu, click **Assign Networks**. Assign the product to the Deployment Network. The image shows an example network named B.

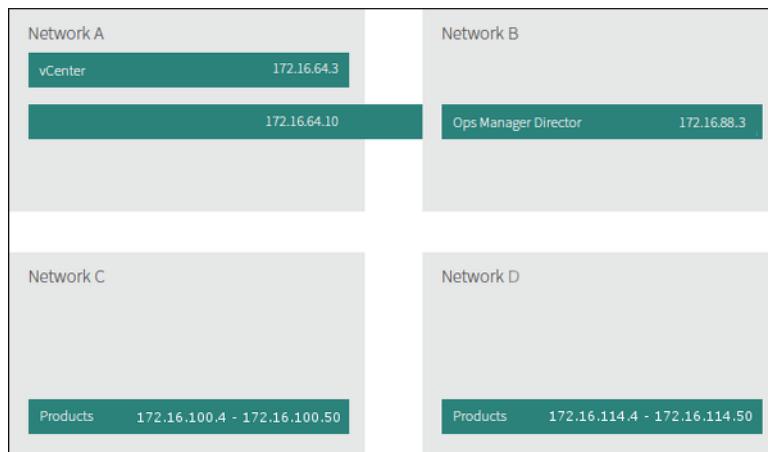


- c. Return to the Installation Dashboard.

Option 3: Installing PCF Products to Multiple Isolated Networks

Deploy PCF products to multiple networks. Ensure that you isolate these networks from each other and from the Infrastructure Network that can reach your vCenter. This configuration provides a higher level of security than [Option 1](#) and [Option 2](#) by reducing the risk of a malicious attack at both the IaaS and product levels.

You must configure Ops Manager Director to communicate on two networks, either through routing or by multi-homing. All other PCF products receive IP addresses on one or more separate networks. For example, the following image shows vCenter in Network A, Ops Manager Director multi-homed in Network A and Network B, and all other PCF products in Network C and Network D.



To configure your PCF deployment for multiple isolated networks:

1. Log into the Ops Manager web interface.
 2. On the Installation Dashboard, select **Ops Manager Director**.
 3. In the left navigation menu, select **Create Networks**.
 4. Click **Add**. Add a network that maps to a vCenter network.
- The image shows an example network named A.

The screenshot shows the 'Ops Manager Director' settings page. On the left, a sidebar lists configuration steps: vCenter Config, Director Config, Create Availability Zones, Assign Availability Zones, Create Networks (which is selected), Assign Networks, and Resource Config. The main area is titled 'Create Networks' and contains fields for creating a network named 'A'. The fields include:

- Name***: A
- vSphere Network Name***: 9048fkm
- Subnet (CIDR Range)***: 172.16.64.0/24
- Excluded IP Ranges**: 172.16.64.1-172.16.64.10,172.16.64.255
- DNS***: 172.16.64.1
- Gateway***: 172.16.64.1

5. Add another network for the Ops Manager Director that maps to a different vCenter network.
The image shows an example network named B.

The screenshot shows the 'Ops Manager Director' interface under the 'Settings' tab. On the left, a vertical list of configuration steps is shown with checkboxes:

- vCenter Config (checked)
- Director Config (checked)
- Create Availability Zones (checked)
- Assign Availability Zones (checked)
- Create Networks (checked)** (highlighted with a grey arrow pointing to it)
- Assign Networks (checked)
- Resource Config (checked)

The main right-hand panel is titled 'Create Networks'. It contains a section for 'Networks' with the sub-instruction: 'One or many IP ranges upon which your products will be deployed'. Below this are two sections labeled 'A' and 'B' with expandable arrows:

- Section A:** Contains a note about networks.
- Section B:** Contains fields for creating a network named 'B' with vSphere Network Name 'ifm349vg', Subnet (CIDR Range) '172.16.88.0/24', Excluded IP Ranges '172.16.88.1-172.16.88.10,172.16.88.255', DNS '172.16.88.1', and Gateway '172.16.88.1'.

- Add a third network specifically for Elastic Runtime that maps to a different vCenter network. Ensure that this network can route to the Ops Manager Director IP address that you created in step 3.
- The image shows an example network named C.

The screenshot shows the 'Create Networks' configuration screen in the Ops Manager Director settings. On the left, a sidebar lists several configuration steps: vCenter Config, Director Config, Create Availability Zones, Assign Availability Zones, Create Networks (which is selected and highlighted with a grey arrow), Assign Networks, and Resource Config. The main panel is titled 'Create Networks' and contains the following fields:

- Networks**: One or many IP ranges upon which your products will be deployed.
- Name***: A text input field containing the value 'C'.
- vSphere Network Name***: A text input field containing the value 'idnm43jf8'.
- Subnet (CIDR Range)***: A text input field containing the value '172.16.100.0/24'.
- Excluded IP Ranges**: A text input field containing the value '172.16.100.1-172.16.100.10,172.16.100.255'.
- DNS***: A text input field containing the value '172.16.100.1'.
- Gateway***: A text input field containing the value '172.16.100.1'.

7. **(Optional)** Add additional networks for other products if you want to isolate them from the applications running in Elastic Runtime and from each other. Ensure that each network can route to the Ops Manager Director IP address.
8. In the Ops Manager Director configuration interface, configure the Director to receive an IP address on the Infrastructure and Deployment networks, as follows:
 - In the left navigation menu, click **Assign Networks**. Click the **Infrastructure Network** drop-down menu and select the network that you created in step 4.
 - Click the **Deployment Network** drop-down menu and select the network that you created in step 5.

Installation Dashboard

Ops Manager Director

Settings **Status** **Credentials**

Assign Networks

The Ops Manager director can be configured to have an IP on two networks. Typically one is routable to the IaaS API (*Infrastructure*) and the other is routable to deployed products (*Deployment*). Infrastructure Network: A Deployment Network: B

Save

- vCenter Config
- Director Config
- Create Availability Zones
- Assign Availability Zones
- Create Networks
- Assign Networks
- Resource Config

9. Return to the Installation Dashboard.

10. Perform the following steps for each PCF product, including Elastic Runtime:

- On the Installation Dashboard, select the PCF product.
 - In the left navigation menu, click **Assign Networks**. Assign the product to a network other than Infrastructure or Deployment.
- The image shows an example network named C.

Installation Dashboard

Pivotal Elastic Runtime

Settings **Status** **Credentials** **Logs**

Assign Networks

c

Save

- Assign Networks
- Assign Availability Zones
- System Database Config
- File Storage Config

- Return to the Installation Dashboard.

Using the Cisco Nexus 1000v Switch with Ops Manager

Refer to the procedure in this topic to use Ops Manager with the Cisco Nexus 1000v Switch. First, configure Ops Manager through Step 4 in [Configuring Ops Manager Director for VMware vSphere](#). Then configure your network according to the following steps.

1. From your PCF Ops Manager **Installation Dashboard**, click the **Ops Manager Director** tile.
2. Select **Create Networks**.
3. Click the network name to configure the network settings. This is **default** if you have not changed the name.

Create Networks

Networks
One or many IP ranges upon which your products will be deployed

default

Name*

vSphere Network Name*

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

Save

4. Find the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured below, a user might want to use the switch configured on the `beer-apple` port group, which is in the `drinks-dc` folder.

Name	VLAN ID	Status	Port Binding	Network Pro
beer-apple	VLAN access: 44	Normal	Static binding (elastic)	
beer-grape	VLAN access: 42	Normal	Static binding (elastic)	
beer-orange	VLAN access: 43	Normal	Static binding (elastic)	
DManagement	VLAN access: 0	Normal	Static binding (elastic)	
DPrivate	VLAN access: 0	Normal	Static binding (elastic)	
ipa-zero	VLAN access: 45	Normal	Static binding (elastic)	

5. In the **vSphere Network Name** field, instead of entering your network name, enter the folder name and port group name for the switch, as you configured them in vCenter. For the example vSphere environment pictured above, you would enter `drinks-dc/beer-apple` to use the switch configured on the `beer-apple` port group.

Installation Dashboard

Ops Manager Director

Settings Status Credentials

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Networks

One or many IP ranges upon which your products will be deployed

Add

YOUR_NETWORK_NAME

Name*

YOUR_NETWORK_NAME

vSphere Network Name*

drinks-dc/beer-apple

The name of the network as it appears in vCenter

Subnet (CIDR Range)*

Excluded IP Ranges

DNS*

Gateway*

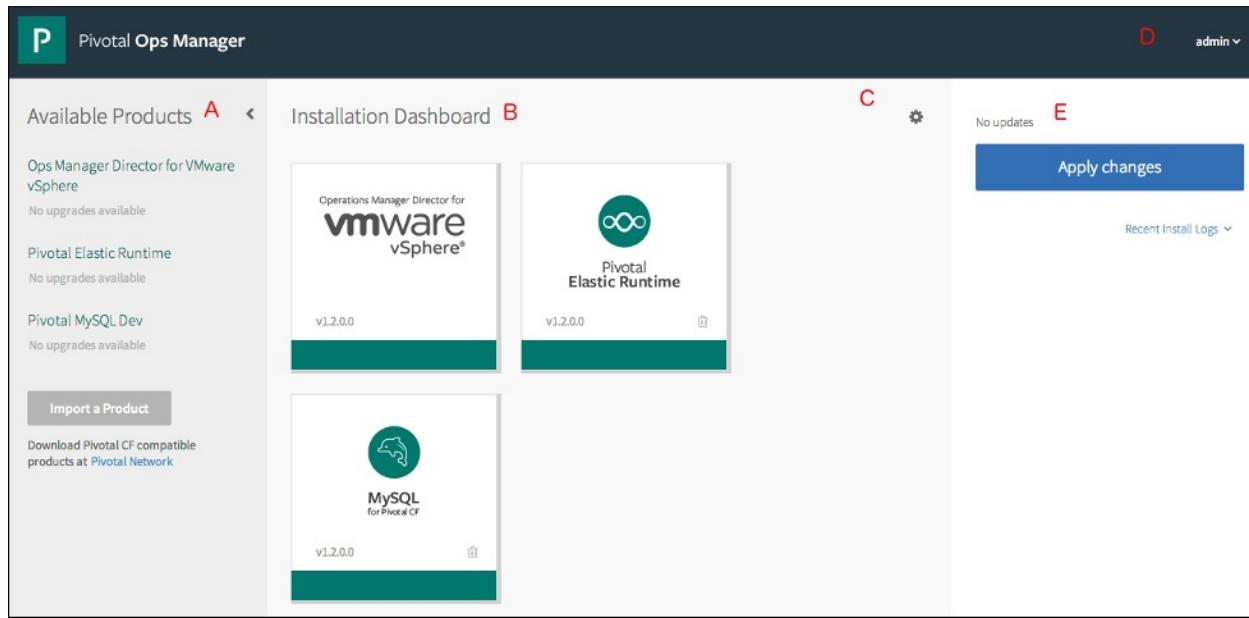
Save

6. Click **Save**.

7. Return to [Configuring Ops Manager Director for VMware vSphere](#) to complete the Ops Manager installation.

Understanding the Ops Manager Interface

This topic describes key features of the [Pivotal Cloud Foundry](#) Operations Manager interface.



- A—Available Products:** Displays a list of products you have imported that are ready for installation. Click the **Import a Product** link to add a new product to Ops Manager.
- B—Installation Dashboard:** Displays a product tile for each installed product.
- C—Actions menu:** Includes the following options:
 - Download activity data:** Downloads a directory containing the config file for the installation, the deployment history, and version information.
 - Import installation settings:** Navigates to the **Import installation settings** view. Select this option to import an existing PCF installation with all of its assets. When you import an installation, the prior installation disappears and is replaced by the newly imported one.
 - Export installation settings:** Exports the current installation with all of its assets. When you export an installation, the exported file contains references to the installation IP addresses. It also contains the base VM images and necessary packages. As a result, an export can be very large (as much as 5 GB or more).
 - Delete this installation:** Deletes the current installation with all of its assets.
- D—User account menu:** Use this menu to change your password or log out.
- E—Pending Changes view:** Displays queued installations and updates that will install during the next deploy.

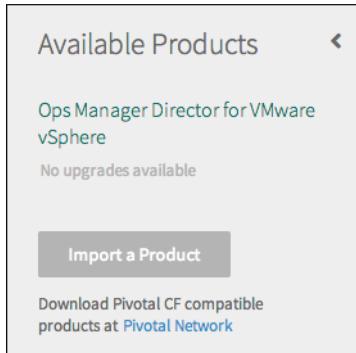
Note: When an update depends on prerequisites, the prerequisites automatically install first.

Adding and Deleting Products

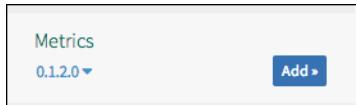
Refer to this topic for help adding and deleting additional products from your [Pivotal Cloud Foundry](#) (PCF) installation, such as [Pivotal HD for PCF](#) and [Pivotal RabbitMQ for PCF](#).

Adding and Importing Products

1. Download PCF-compatible products at [Pivotal Network](#).
2. From the Available Products view, click **Import a Product**.



3. To import a product, select the .zip file that you downloaded from Pivotal Network or received from your software distributor, then click **Open**.
After the import completes, the product appears in the Available Products view.
4. Hover over the product name in the Available Products view to expose the **Add** button, then click **Add**.



5. The product tile appears in the Installation Dashboard. If the product requires configuration, the tile appears orange.



If necessary, configure the product.

6. **(Optional)** In the product configuration view, select the **Lifecycle Errands** tab to configure post-install errands or review the default settings. Post-install errands are scripts that automatically run after a product installs, before Ops Manager makes the product available for use. For more information about post-install errands, see [Understanding Lifecycle Errands](#).

Note: By default, Ops Manager reruns lifecycle errands even if they are not necessary due to settings left from a previous install. Leaving errands checked at all times can cause updates and other processes to take longer. To prevent a lifecycle errand from running, deselect the checkbox for the errand in the **Settings** tab on the product tile before installing the product.

The screenshot shows the 'Lifecycle Errands' section of the Redis for Pivotal CF settings. It includes a description of lifecycle errands as scripts running at designated points during installation. A checkbox for 'Broker Registrar' is checked. Other options like 'Assign Networks', 'Redis', and 'Resource Config' are listed but not checked. A 'Save' button is at the bottom.

The **Broker Registrar** checkbox is an example of a lifecycle errand available for a product. When you select this checkbox, this errand registers service brokers with the Cloud Controller and also updates any broker URL and credential values that have changed since the previous registration.

7. In the Pending Changes view, click **Apply Changes** to start installation and run post-install lifecycle errands for the product.

Deleting a Product

1. From the Installation Dashboard, click the trash icon on a product tile to remove that product. In the **Delete Product** dialog box that appears, click **Confirm**.

Note: You cannot delete the Ops Manager Director product.

2. In the Pending Changes view, click **Apply Changes**.

After you delete a product, the product tile is removed from the installation and the Installation Dashboard. However, the product appears in the Available Products view.

Starting and Stopping PCF VMs

This topic describes starting and stopping the virtual machines (VMs) that make up a [Pivotal Cloud Foundry](#) (PCF) deployment. This procedure uses the BOSH CLI. See [Prepare to Use the BOSH CLI](#) for help setting up this tool.

Dependencies between the components in your PCF deployment require that you start and stop the VMs for those components in a specific order. These orders are specified below in the [start order](#) and [stop order](#) tables.

Finding the Names for Your PCF Virtual Machines

You will need the full names for the VMs to [start](#) or [stop](#) them using the BOSH CLI. The name of each VM corresponds to its job as a CF component, but also includes information specific to your deployment. To find the full names for the VMs running these components, run:

```
o → bosh vms
Deployment `cf-2a2f1a81c2f1d8e1280d'

Director task 88

Task 88 done

+-----+-----+
| Job/index | State | Resource Pool
+-----+-----+
| ccdb-partition-a52fa835989ba15a07de/0 | running | ccdb-partition-a52fa835989ba15a07de
| clock_global-partition-a52fa835989ba15a07de/0 | running | clock_global-partition-a52fa835989ba15a07de
| cloud_controller-partition-a52fa835989ba15a07de/0 | running | cloud_controller-partition-a52fa835989ba15a07de
| cloud_controller_worker-partition-a52fa835989ba15a07de/0 | running | cloud_controller_worker-partition-a52fa835989ba15a07de
| consoledb-partition-a52fa835989ba15a07de/0 | running | consoledb-partition-a52fa835989ba15a07de
| consul_server-partition-a52fa835989ba15a07de/0 | running | consul_server-partition-a52fa835989ba15a07de
| dea-partition-a52fa835989ba15a07de/0 | running | dea-partition-a52fa835989ba15a07de
| doppler-partition-a52fa835989ba15a07de/0 | running | doppler-partition-a52fa835989ba15a07de
| etcd_server-partition-a52fa835989ba15a07de/0 | running | etcd_server-partition-a52fa835989ba15a07de
| ha_proxy-partition-a52fa835989ba15a07de/0 | running | ha_proxy-partition-a52fa835989ba15a07de
| health_manager-partition-a52fa835989ba15a07de/0 | running | health_manager-partition-a52fa835989ba15a07de
| loggregator_trafficcontroller-partition-a52fa835989ba15a07de/0 | running | loggregator_trafficcontroller-partition-a52fa835989ba15a07de
| mysql-partition-a52fa835989ba15a07de/0 | running | mysql-partition-a52fa835989ba15a07de
| mysql_proxy-partition-a52fa835989ba15a07de/0 | running | mysql_proxy-partition-a52fa835989ba15a07de
| nats-partition-a52fa835989ba15a07de/0 | running | nats-partition-a52fa835989ba15a07de
| nfs_server-partition-a52fa835989ba15a07de/0 | running | nfs_server-partition-a52fa835989ba15a07de
| router-partition-a52fa835989ba15a07de/0 | running | router-partition-a52fa835989ba15a07de
| uaa-partition-a52fa835989ba15a07de/0 | running | uaa-partition-a52fa835989ba15a07de
| uaadb-partition-a52fa835989ba15a07de/0 | running | uaadb-partition-a52fa835989ba15a07de
+-----+-----+
VMs total: 19
```

Starting PCF Virtual Machines

In the order specified in the [table](#) below, run `bosh start VM-NAME` for each component in your PCF deployment, using the name as displayed in the [output](#) of the `bosh vms` command under `Job/index`. Complete each VM name by replacing 'FROM-BOSH-VMS-OUTPUT' in the table with specific information from your `bosh vms` [terminal output](#).

For example, in the deployment corresponding to the [example terminal output](#), you would run

```
bosh start nats-partition-a52fa835989ba15a07de
```

```
$ bosh start nats-partition-a52fa835989ba15a07de

Processing deployment manifest
-----
You are about to start nats-partition-a52fa835989ba15a07de/0

Detecting deployment changes
-----
Start nats-partition-a52fa835989ba15a07de/0? (type 'yes' to continue): yes

Performing `start nats-partition-a52fa835989ba15a07de/0'...

...
Started updating job nats-partition-a52fa835989ba15a07de > nats-partition-a52fa835989ba15a07de/0 (canary). Done (00:00:43)

Task 42 done

nats-partition-a52fa835989ba15a07de/0 has been started
```

Note: To start all VM instances of a certain type, run the command as above. You can also start a specific instance by including its index after a '/'. In the example deployment above, to start only the first instance of the NATS VM, run: `$ bosh start nats-partition-a52fa835989ba15a07de/0`

Start Order	Component	VM NAME
1	NATS	nats-FROM-BOSH-VMS-OUTPUT
2	etcd	etcd_server-FROM-BOSH-VMS-OUTPUT
3	NFS Server	nfs_server-FROM-BOSH-VMS-OUTPUT
4	Cloud Controller Database	ccdb-FROM-BOSH-VMS-OUTPUT
5	UAA Database	uaadb-FROM-BOSH-VMS-OUTPUT
6	Apps Manager Database	consoledb-FROM-BOSH-VMS-OUTPUT
7	Cloud Controller	cloud_controller-FROM-BOSH-VMS-OUTPUT
8	HAProxy	ha_proxy-FROM-BOSH-VMS-OUTPUT
9	Router	router-FROM-BOSH-VMS-OUTPUT
10	Health Manager	health_manager-FROM-BOSH-VMS-OUTPUT
11	Clock Global	clock_global-FROM-BOSH-VMS-OUTPUT
12	Cloud Controller Worker	cloud_controller_worker-FROM-BOSH-VMS-OUTPUT
13	UAA	uaa-FROM-BOSH-VMS-OUTPUT
14	Login	login-FROM-BOSH-VMS-OUTPUT
15	MySQL Proxy	mysql_proxy-FROM-BOSH-VMS-OUTPUT
16	MySQL Server	mysql-FROM-BOSH-VMS-OUTPUT
17	DEA	dea-FROM-BOSH-VMS-OUTPUT
18	Doppler Server	doppler-FROM-BOSH-VMS-OUTPUT
19	Loggregator Trafficcontroller	loggregator_trafficcontroller-FROM-BOSH-VMS-OUTPUT

Stopping PCF Virtual Machines

In the order specified in the [table](#) below, run `bosh stop VM-NAME` for each component in your PCF deployment. Use the name as displayed in the [output](#) of the `bosh vms` command under `Job/index` to complete each VM name, i.e. replace 'FROM-BOSH-VMS-OUTPUT' in the table with specific information from your `bosh vms` [terminal output](#).

For example, in the deployment corresponding to the [example terminal output](#), you would run `bosh stop loggregator_trafficcontroller-partition-a52fa835989ba15a07de` to stop the Loggregator Trafficcontroller virtual machine:

```
$ bosh stop loggregator_trafficcontroller-partition-a52fa835989ba15a07de

Processing deployment manifest
-----
You are about to stop loggregator_trafficcontroller-partition-a52fa835989ba15a07de/0

Detecting deployment changes
-----
Stop loggregator_trafficcontroller-partition-a52fa835989ba15a07de/0? (type 'yes' to continue): yes

Performing `stop loggregator_trafficcontroller-partition-a52fa835989ba15a07de/0'...

...
Started updating job loggregator_trafficcontroller-partition-a52fa835989ba15a07de > loggregator_trafficcontroller-partition
loggregator_trafficcontroller-partition-a52fa835989ba15a07de/0 has been stopped
```

Note: To stop all VM instances of a certain type, run the command as above. You can also stop a specific instance by including its index after a '/'. In the example deployment above, to stop only the first instance of the NATS VM, run: `$ bosh stop nats-partition-a52fa835989ba15a07de/0`

Stop Order	Component	VM NAME
1	Loggregator Trafficcontroller	loggregator_trafficcontroller-FROM-BOSH-VMS-OUTPUT
2	Doppler Server	doppler-FROM-BOSH-VMS-OUTPUT
3	DEA	dea-FROM-BOSH-VMS-OUTPUT
4	MySQL Server	mysql-FROM-BOSH-VMS-OUTPUT
5	MySQL Proxy	mysql_proxy-FROM-BOSH-VMS-OUTPUT
6	Login	login-FROM-BOSH-VMS-OUTPUT
7	UAA	uaa-FROM-BOSH-VMS-OUTPUT
8	Cloud Controller Worker	cloud_controller_worker-FROM-BOSH-VMS-OUTPUT
9	Clock Global	clock_global-FROM-BOSH-VMS-OUTPUT
10	Health Manager	health_manager-FROM-BOSH-VMS-OUTPUT
11	Router	router-FROM-BOSH-VMS-OUTPUT
12	HAProxy	ha_proxy-FROM-BOSH-VMS-OUTPUT
13	Cloud Controller	cloud_controller-FROM-BOSH-VMS-OUTPUT
14	Apps Manager Database	consoledb-FROM-BOSH-VMS-OUTPUT
15	UAA Database	uaadb-FROM-BOSH-VMS-OUTPUT
164	Cloud Controller Database	ccdb-FROM-BOSH-VMS-OUTPUT
17	NFS Server	nfs_server-FROM-BOSH-VMS-OUTPUT
18	etcd	etcd_server-FROM-BOSH-VMS-OUTPUT
19	NATS	nats-FROM-BOSH-VMS-OUTPUT

Creating New Elastic Runtime User Accounts

When you first deploy your [Elastic Runtime](#) PaaS, there is only one user: an administrator. At this point, you can add accounts for new users who can then push applications using the cf Command Line Interface (cf CLI).

How to add users depends on whether or not you have SMTP enabled, as described in the options below.

Option 1: Adding New Users when SMTP is Enabled

If you have enabled SMTP, your users can sign up for accounts and create their own orgs. They do this using the Pivotal Cloud Foundry Apps Manager, a self-service tool for managing organizations, users, applications, and application spaces.

Instruct users to complete the following steps to log in and get started using the Apps Manager.

1. Browse to `console.YOUR-SYSTEM-DOMAIN`. Refer to **Elastic Runtime > Cloud Controller** to locate your system domain.
2. Select **Create an Account**.
3. Enter your email address and click **Create an Account**. You will receive an email from the Apps Manager when your account is ready.
4. When you receive the new account email, follow the link in the email to complete your registration.
5. You will be asked to choose your organization name.

You now have access to the Apps Manager. Refer to the Apps Manager documentation at [docs.pivotal.io](#) for more information about using the Apps Manager.

Option 2: Adding New Users when SMTP is Not Enabled

If you have not enabled SMTP, only an administrator can create new users, and there is no self-service facility for users to sign up for accounts or create orgs.

The administrator creates users with the cf CLI. See [Creating and Managing Users with the cf CLI](#).

[Return to the Getting Started Guide](#)

Logging into the Apps Manager

Complete the following steps to log in to the Apps Manager:

1. Browse to `console.YOUR-SYSTEM-DOMAIN`. In Pivotal Ops Manager, refer to **Pivotal Elastic Runtime > Settings > Cloud Controller** to locate your system domain.
2. Log in to the Apps Manager using the UAA Administrator User credentials. In Pivotal Ops Manager, refer to **Pivotal Elastic Runtime > Credentials** for these credentials.

Controlling Apps Manager User Activity with Environment Variables

This topic describes two environment variables you can use to manage users' interactions with the Apps Manager, and how to set these variables using the cf CLI.

Understanding the Apps Manager Environment Variables

You can control which users can create orgs and perform user management actions on the Apps Manager using the following environment variables.

ENABLE_NON_ADMIN_ORG_CREATION

If set to `true`, a user of any role can create an org. On the Org Dashboard, the links to create a new org appear as follows:

- The **Create a New Org** link in the drop-down menu in the left navigation panel
- The **Create Org** link on the right side of the My Account page, which you access from the user name drop-down menu

The image shows the link locations.



If set to `false`, only an Admin can create an org. This is the default value.

ENABLE_NON_ADMIN_USER_MANAGEMENT

If set to `true`, Org Managers and Space Managers can invite users and manage roles, and a user of any role can leave an org, provided at least one Org Manager exists in the org.

If set to `false`, only an Admin can invite users and manage roles. This is the default value. Users cannot remove themselves from an org, and on the Org Dashboard, the Members tab appears as read-only for all users except Admins.

Changing an Environment Variable Value

Note: To run the commands discussed in this section, you must be an administrator for your org and log in to the cf CLI with your UAA Administrator user credentials. In Pivotal Operations Manager, refer to **Elastic Runtime > Credentials** for the UAA admin name and password.

To change an environment variable value:

1. From a terminal window, run `cf api URL` and target your Apps Manager URL. For example:
`cf api api.YOUR-SYSTEM-DOMAIN`.
2. Run `cf login` and provide your UAA Administrator user credentials.
3. Select the `system` org and the `apps-manager` space.
4. Run `cf set-env apps-manager ENVIRONMENT-VARIABLE VALUE`. For example:
`cf set-env apps-manager ENABLE_NON_ADMIN_ORG_CREATION true`
5. Run `cf restart apps-manager` to reinitialize the Apps Manager with the new environment variable value.

Configuring Your App Autoscaling Instance

The App Autoscaling service scales bound applications in response to load.

An instance of the App Autoscaling service examines the CPU usage of an application bound to it every few minutes. In response to load changes, the service scales your app up and down according to the thresholds, minimums, and maximums that you provide.

Follow the steps below to configure your App Autoscaling service instance.

1. Log in to the Apps Manager: [How to log in](#).
2. Select a space containing an App Autoscaling service instance from the org dashboard or from the left navigation bar.
3. In the Services section of the space dashboard, under your App Autoscaling service instance name, click **Manage**.

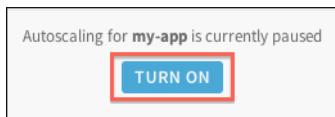
Note: You must specifically have the role of Space Developer to access the **Manage** link for the app autoscaling service. Space Managers, Space Auditors, and all Org roles do not have the permission to make changes to App Autoscaling. See [Managing User Accounts in Spaces Using the Apps Manager](#) for help managing user roles.

The screenshot shows the 'SERVICES' section of the Apps Manager. It lists two service instances:

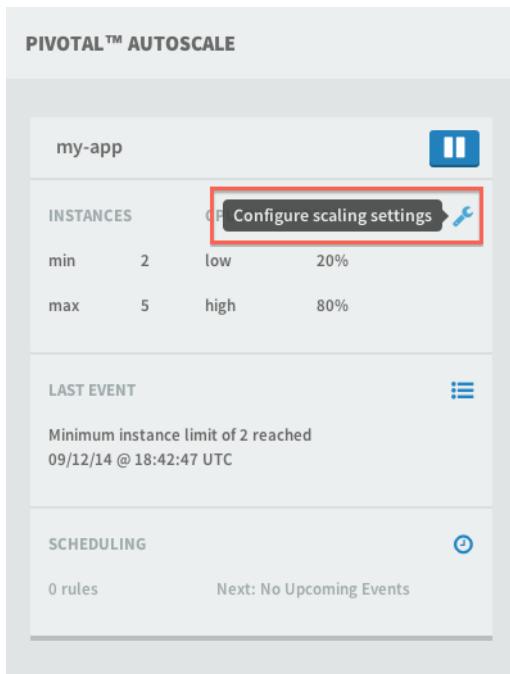
SERVICE INSTANCE	SERVICE PLAN	BOUND APPS
my-database	MySQL Database	1
my-autoscaler	App Autoscaler Gold	1

For each service instance, there are four links: 'Manage', 'Documentation', 'Support', and 'Delete'. The 'Manage' link for the 'my-autoscaler' instance is highlighted with a red box.

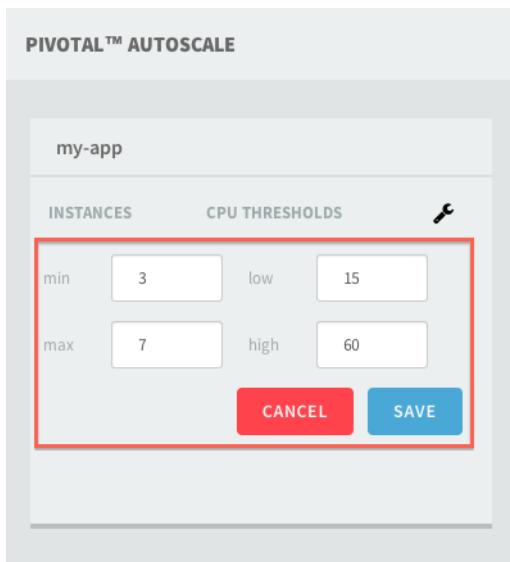
4. By default, new autoscaling instances are paused. If paused, click **Turn On** to enable monitoring of your application and start the autoscaling process.



5. Click the wrench icon on your Autoscaling dashboard.



6. Change the configuration settings and click **Save**. See the [Configuration Options](#) section of this topic for information about the configuration settings.



7. Examine the App Autoscaling service instance dashboard to confirm your changes.

INSTANCES	CPU THRESHOLDS		
min	3	low	15%
max	7	high	60%

LAST EVENT

Minimum instance limit of 2 reached
09/12/14 @ 18:42:47 UTC

SCHEDULING

0 rules Next: No Upcoming Events

Configuration Options

You can set the absolute maximum and minimum number of instances for your app, as well as the CPU thresholds for an app that trigger the autoscaling service.

Instance Counts

The **Instances** values specify the absolute minimum and maximum number of instances autoscaling can set for an application.

- **Min:** Default value: `2`. The minimum number of instances to which autoscaling can scale your app. Autoscaling never scales your application below this number of instances.
- **Max:** Default value: `5`. The maximum number of instances to which autoscaling can scale your app. Autoscaling never scales your application above this number of instances.

Note: **Min** and **Max** cannot be set to less than `1` or greater than `20`. **Min** must be less than or equal to **Max**.

CPU Thresholds

The **CPU thresholds** values specify the upper and lower limits of CPU utilization that trigger the autoscaling service.

The autoscaling service calculates CPU utilization as a moving average across the CPUs of all currently running instances of an application.

- **Low:** Default value: `20`. When the autoscaling service instance detects CPU utilization below this threshold, it reduces the number of instances of the app by one.
- **High:** Default value: `80`. When the autoscaling service instance detects CPU utilization above below this threshold, it increases the number of instances of the app by one.

Manual Scaling

If you manually scale an application bound to an autoscaling service instance, the autoscaling service stops monitoring and autoscaling your application.

To re-enable monitoring and scaling, click **Turn On** on the App Autoscaling service instance dashboard.

Autoscaling for **my-app** is currently paused

[TURN ON](#)

Managing Scheduled Scaling in the App Autoscaling Service

Follow the steps below to manage your App Autoscaling service instance.

1. Log in to the Apps Manager: [How to log in](#).
2. Select a space containing an App Autoscaling service instance from the org dashboard or from the left navigation bar.
3. In the Services section of the space dashboard, under your App Autoscaling service instance name, click **Manage**.

Note: To access the **Manage** link for the app autoscaling service, you must have the role of **Space Developer**. See [Managing User Accounts in Spaces Using the Apps Manager](#) for help managing user roles.

SERVICE INSTANCE	SERVICE PLAN	BOUND APPS
my-database	MySQL Database	1
Manage Documentation Support Delete		
my-autoscaler	App Autoscaler Gold	1
Manage Documentation Support Delete		

4. By default, new autoscaling instances are paused. If paused, click **Turn On** to enable monitoring of your application and start the autoscaling process.

Autoscaling for **my-app** is currently paused

TURN ON

5. Click the clock icon on your Autoscaling dashboard.

PIVOTAL™ AUTOSCALE

my-app

INSTANCES	CPU THRESHOLDS
min 1	low 20%
max 2	high 80%

LAST EVENT

Minimum instance limit of 1 reached
09/12/14 @ 22:01:39 UTC

SCHEDULING

0 rules Scheduled scaling rules →

Next: No Upcoming Events

6. In the Scheduling interface, create a new rule by editing the date and time fields and choosing values for the number of minimum and maximum instances. When finished, click **Save**. See the [Rule Types](#) section of this topic for

more information.

SCHEDULING: MY-APP SERVER TIME: 09/12/14 @ 20:55:25 UTC X

+ New UNSAVED

on at *All times are UTC

repeats every S M T W T F S

min	<input type="text" value="5"/>	low	<input type="text" value="20%"/>
max	<input type="text" value="10"/>	high	<input type="text" value="80%"/>

ADD

- After saving, the left side of the Scheduling interfaces shows your rule. Click your rule to edit it.

SCHEDULING: MY-APP SERVER TIME: 09/12/14 @ 20:55:25 UTC X

+ New

11/21/2014 / 02:00 X

5 to 10 instances

on at *All times are UTC

repeats every S M T W T F S

min	<input type="text" value="2"/>	low	<input type="text" value="20%"/>
max	<input type="text" value="5"/>	high	<input type="text" value="80%"/>

ADD

- Edit your existing rule and click **Save** to save your changes.

SCHEDULING: MY-APP SERVER TIME: 09/12/14 @ 20:55:25 UTC X

+ New

11/21/2014 / 02:00 X

5 to 10 instances

on at *All times are UTC

repeats every S M T W T F S

min	<input type="text" value="5"/>	low	<input type="text" value="20%"/>
max	<input type="text" value="15"/>	high	<input type="text" value="80%"/>

SAVE

- In the left pane of the Scheduling interfaces, click the X for a rule to delete it.

SCHEDULING: MY-APP SERVER TIME: 09/12/14 @ 20:55:25 UTC X

+ New 11/21/2014 / 02:00 5 to 15 instances

on 11/21/2014 at 2:00 ▲
*All times are UTC

repeats every S M T W T F S

min	5	low	20%
max	15	high	80%

SAVE

10. Close the Scheduling interfaces to return to your Autoscaling dashboard. The Scheduling section of the Autoscaling dashboard displays the next occurring rule and summary information about your rules.

PIVOTAL™ AUTOSCALE

my-app II

INSTANCES	CPU THRESHOLDS
min 2	low 20%
max 5	high 80%

LAST EVENT ≡

Minimum instance limit of 2 reached
09/12/14 @ 20:57:55 UTC

SCHEDULING ?

1 rules Next: 11/21/14 @ 02:00:00 UTC

Rule Types

Scheduled scaling rules affect the minimum and maximum instance count values for your application. When the autoscaling service runs a scheduled scaling rule, it changes the **Min** and **Max** values of the [instance count](#) of your application to the values specified in the rule.

One-time Rules

The autoscaling service runs a one-time scheduled scaling rule once only. After running a one-time scheduled scaling rule, the service removes the rule from the list of existing rules.

💡 **Note:** You must schedule one-time rules to occur at a time in the future.

Recurring Rules

The autoscaling service runs a recurring scheduled scaling rule on a regular basis. You select one or more days of the

week for a rule, and the autoscaling service runs the rule on those days every week.

Click **pause** for a particular rule to stop the autoscaling service from running that rule. Click **play** to resume running that rule.

Note: The autoscaling service does not run a recurring rule for the first time until the date specified in the rule.

Scheduling Example

- The rule shown in the image below recurs every Monday and Friday at 4AM, starting on Friday, November 28, 2014. This rule changes the minimum number of instances of the app to 10 and the maximum to 20.

The screenshot shows the 'SCHEDULING: MY-APP' interface. The server time is 09/12/14 @ 21:27:22 UTC. A rule is being configured with the following details:

- On:** 11/28/2014
- At:** 4:00
- Repeats every:** M, F (Monday and Friday are checked)
- Min:** 10
- Max:** 20

- The rule shown in the image below recurs every Wednesday at 4AM, starting on Friday, November 28, 2014. This rule changes the minimum number of instances of the app to 1 and the maximum to 3.

The screenshot shows the 'SCHEDULING: MY-APP' interface. The server time is 09/12/14 @ 21:27:22 UTC. A rule is being configured with the following details:

- On:** 11/28/2014
- At:** 4:00
- Repeats every:** W (Wednesday is checked)
- Min:** 1
- Max:** 3

Based on the two rules above, starting on Friday, November 28, 2014, the autoscaling service scales the minimum and maximum instance counts for the application as follows:

- Every Monday, the autoscaling service scales the minimum up to 10 and the maximum to 20.
- Every Wednesday, the autoscaling service scales the minimum down to 1 and the maximum to 3.
- Every Friday, the autoscaling service scales the minimum back up to 10 and the maximum to 20.

Backing Up Pivotal Cloud Foundry

Pivotal recommends frequently backing up your deployment.

The complete state of an Elastic Runtime deployment is captured by the following:

- PCF installation settings
- Critical databases:
 - Pivotal MySQL Server Database
 - Cloud Controller Database
 - UAA Database
 - Apps Manager Database
 - NFS Server Database

To back up a deployment, you must temporarily stop the Cloud Controller and save all of the above, as well as your database encryption credentials, to backup files. To restore a deployment, you must temporarily stop the Cloud Controller and restore the state of each component from the backup file.

The procedure for backing up Elastic Runtime each component is described below.

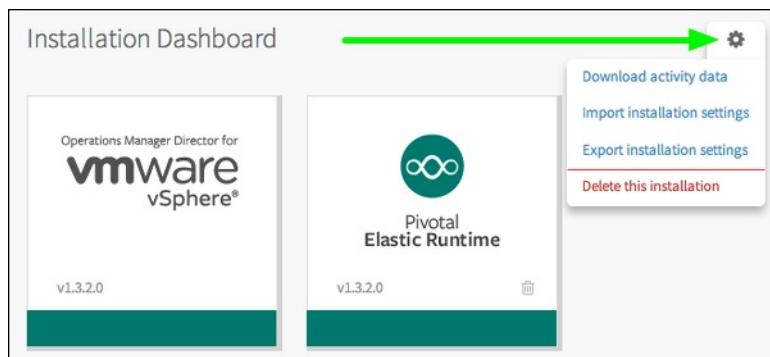
See [Restoring Pivotal Cloud Foundry from Backup](#) for instructions for restoring.

Export Installation Settings

Pivotal recommends that you back up your installation settings by exporting frequently. Always export an installation before importing a new one. Import an installation to restore your settings or to share your settings with another user. See [Import Installation Settings](#)

Note: Exporting your installation only backs up your installation settings. It does not back up your VMs or any external MySQL databases.

From the Product Installation Dashboard in the Ops Manager interface, click the gear icon and select **Export installation settings**. This option is only available after you have deployed at least one time.



Export installation settings exports the current PCF installation settings and assets. When you export an installation, the exported file contains the base VM images, necessary packages, and references to the installation IP addresses. As a result, an export can be 5 GB or more.

Note: For versions of Ops Manager 1.3 or older, the process of archiving files for export may exceed the timeout limit of 600 seconds, causing a 500 error. If this happens, you can fix the problem by manually configuring the timeout value to a longer limit, or by assigning more resources to the Ops Manager VM to improve performance. Refer to the [Knowledge Base](#) for a more in depth discussion of this issue.

Back Up the Cloud Controller Database Encryption Credentials

From the Product Installation Dashboard, select **Pivotal Elastic Runtime > Credentials** and locate the Cloud Controller section. Record the Cloud Controller **DB encryption credentials**. You must provide these credentials if you

contact Pivotal Support for help restoring your installation.

Cloud Controller	Vm Credentials	vcap / e38252ff3dad2d28
	Staging Upload Credentials	staging_upload_user / c845fc832d5cef490674
	Bulk Api Credentials	bulk_api / ad87614112cb7cc82c9c
	Db Encryption Credentials	db_encryption / db54fc70546bbf12eb28
	Encrypt Key	

Download the BOSH Deployment Manifest

1. Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your PCF system deployment.
2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP address listed for the Ops Manager Director. You access the BOSH Director using this IP address.

The screenshot shows the 'Installation Dashboard' with the 'Ops Manager Director' section selected. Under the 'Status' tab, there is a table with columns 'JOB', 'INDEX', 'IPS', and 'CLOUD'. A single row is present for 'Ops Manager Director' with values 0, 10.0.0.3, and some blurred text respectively.

3. Click **Credentials** and record the Director credentials.

The screenshot shows the 'Installation Dashboard' with the 'Ops Manager Director' section selected. Under the 'Credentials' tab, there is a table with columns 'JOB', 'NAME', and 'CREDENTIALS'. A row for 'Ops Manager Director' has three entries: 'VM credentials' (vcap / [REDACTED]), 'BOSH agent credentials' (vcap / [REDACTED]), and 'Director credentials' (director / [REDACTED]). The 'Director credentials' row is circled in red.

4. From the command line, target the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 10.0.0.3
Target set to `microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as `director'
```

Note: If `bosh target` does not prompt you for your username and password, run `bosh login`.

5. Run `bosh deployments` to identify the name of the current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name      | Release(s)          | Stemcell(s)           |
+-----+-----+
| cf-example | cf-mysql1/10       | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|           | cf/183.2             |                      |
+-----+-----+
```

6. Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save the current BOSH deployment manifest. You need this manifest to locate information about your databases. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml  
Deployment manifest saved to `cf.yml'
```

Back Up Critical Databases

This section describes the procedure for backing up databases associated with your PCF installation.

Your Elastic Runtime deployment contains several critical data stores that must be present for a complete restore. You must back up each of the following:

- Cloud Controller Database
- UAA Database
- Apps Manager Database
- NFS Server
- Pivotal MySQL Server

Note: If you are running the default internal databases, follow the instructions below. If you are running your databases or filestores externally, disregard these instructions and ensure that you back up your external databases and filestores.

Stop Cloud Controller

1. If you have not already, download the BOSH manifest as instructed in the [Download the BOSH Manifest](#) section.
2. Run `bosh deployment DEPLOYMENT-MANIFEST` to set your deployment.

```
$ bosh deployment cf.yml  
Deployment set to `/home/working_directory(cf.yml'
```

3. Run `bosh vms` in the terminal to view the list of VMs in your selected deployment.

```
$ bosh vms
```

Notice that all the Cloud Controller VMs begin with “cloud_controller”.

Job/index	State	Resource Pool	IPs
cloud_controller-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
cloud_controller_worker-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
clock_global-partition-bd784aa8b15f/0	running	clock_global-partition-bd784aa8b15f	10.85.x
nats-partition-bd784aa8b15f/0	running	nats-partition-bd784aa8b15f	10.85.x
router-partition-bd784aa8b15f/0	running	router-partition-bd784aa8b15f	10.85.x
uaa-partition-bd784aa8b15f/0	running	uaa-partition-bd784aa8b15f	10.85.x

4. Run `bosh stop SELECTED-VM` for each Cloud Controller VM.

```
$ bosh stop cloud_controller-partition-bd784aa8b15f/0  
Acting as user 'director'...  
  
You are about to stop cloud_controller-partition-bd784aa8b15f/0  
  
Detecting deployment changes  
-----  
Stop cloud_controller-partition-bd784aa8b15f/0? (type 'yes' to continue)
```

You do not need to stop the Cloud Controller worker VMs.

Back Up the Cloud Controller Database

1. In `cf.yml`, locate the Cloud Controller database component under the `ccdb` key and record the IP address:

```
ccdb:
  address: 10.85.52.96
  port: 2544
  db_scheme: postgres
```

- From the Installation Dashboard in Ops Manager, select **Elastic Runtime** and click **Credentials**. Record the Cloud Controller database credentials.

Cloud Controller	Vm Credentials	vcap / [REDACTED]
------------------	----------------	-------------------

- SSH into the Cloud Controller database VM as the admin using the IP address and password recorded in the previous steps.

```
$ ssh vcap@10.85.52.96
Password:*****
```

- Run `find /var/vcap | grep 'bin/psql'` to find the locally installed psql client on the CCDB VM. For example:

```
$ root@10.85.52.96:~# find /var/vcap | grep 'bin/psql'
/var/vcap/data/packages/postgres/5.1/bin/psql
```

- Run `pg_dump` from the locally installed psql client to export the database:

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql/pg_dump -h 10.85.52.96 -U admin -p 2544 ccdb > ccdb.sql
```

- Exit from the Cloud Controller database VM.

- Run `scp` to copy the exported database to your local machine.

```
$ scp vcap@10.85.52.96:ccdb.sql
```

Back Up the UAA Database

- In the BOSH deployment manifest, locate the `uaadb` component and record the IP address:

```
uaadb:
  address: 10.85.52.101
  port: 2544
  db_scheme: postgresql
```

- From the Installation Dashboard in Ops Manager, select **Elastic Runtime** and click **Credentials**. Record the UAA database credentials.

UAA Database	Vm Credentials	vcap / [REDACTED]
--------------	----------------	-------------------

- SSH into the UAA database VM as the admin using the IP address and password recorded in the previous steps.

- Run `find /var/vcap | grep 'bin/psql'` to find the locally installed psql client on the UAA database VM.

```
$ root@10.85.52.101:~# find /var/vcap | grep 'bin/psql'
/var/vcap/data/packages/postgres/5.1/bin/psql
```

- Run `pg_dump` from the locally installed psql client to export the database:

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql/pg_dump -h 10.85.52.101 -U root -p 2544 uaa > uaa.sql
```

- Exit from the UAA database VM.

- Run `scp` to copy the exported database to your local machine.

```
$ scp vcap@10.85.52.101:uaa.sql
```

Back Up the Apps Manager Database

1. In the BOSH deployment manifest, locate the `databases` component and record the IP address and password:

```
databases:  
  address: 10.85.52.104  
  port: 2544  
  db_scheme: postgresql  
  roles:  
    - tag: admin  
      name: root  
      password: *****  
  databases:  
    - tag: console  
      name: console
```

2. From the Installation Dashboard in Ops Manager, select **Elastic Runtime** and click **Credentials**. Record the Apps Manager database credentials.

Apps Manager Database	Vm Credentials	vcap / dd4dd4dd4d44
-----------------------	----------------	---------------------

3. SSH into the Apps Manager database VM as the admin using the IP address and password recorded in the previous steps.

4. Run `find /var/vcap | grep 'bin/psql'` to find the locally installed psql client on the Apps Manager database VM.

```
$ root@10.85.52.104:~# find /var/vcap | grep 'bin/psql'  
  
/var/vcap/data/packages/postgres/5.1/bin/psql
```

5. Run `pg_dump` from the locally installed psql client to export the database:

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql/pg_dump -h 10.85.52.104 -U root -p 2544 console > console.sql
```

6. Exit from the Apps Manager database.

7. Run `scp` to copy the exported database to your local machine.

```
$ scp vcap@10.85.52.104:console.sql
```

Back Up NFS Server

1. In the BOSH deployment manifest, locate the `nfs_server` component and record the address:

```
nfs_server:  
  address: 10.0.0.10  
  network: 10.0.0.0/24  
  syslog_aggregator:  
    address:  
    port:
```

2. From the Installation Dashboard in Ops Manager, select **Elastic Runtime** and click **Credentials**. Record NFS Server credentials.

NFS Server	Vm Credentials	vcap / [REDACTED]
------------	----------------	-------------------

3. SSH into NFS server VM using NFS Server and create a TAR file:

```
$ ssh vcap@10.0.0.10 'cd /var/vcap/store && tar cz shared' > nfs.tar.gz
```

Note: The TAR file that you create to back up NFS server might be large. To estimate the size of the TAR file before you create it, run the following command: `tar -cf - /dir/to/archive/ | wc -c`

Back Up Pivotal MySQL Server

Note: The Elastic Runtime deploy contains an embedded MySQL Server that serves as the data store for the Application Usage Events, Notifications, and Autoscaler services.

- From the Installation Dashboard in Ops Manager, select **Pivotal Elastic Runtime**. Click **Status** and record the IP address of **MySQL Server**.

JOB	INDEX	IPS	CID
NATS	0	10.0.1.12	i-8db9275e
consul	0	10.0.1.53	i-47e57c94
etcd	0	10.0.1.13	i-8eb9275d
NFS Server	0	10.0.1.14	i-02b927d1
Cloud Controller Database	0	10.0.1.15	i-30b628e3
UAA Database	0	10.0.1.16	i-31b628e2
Apps Manager Database	0	10.0.1.17	i-32b628e1
Cloud Controller	0	10.0.1.18	i-aab52b79
Router	0	10.0.1.19	i-e4b42a37
Health Manager	0	10.0.1.20	i-feb42a2d
Clock Global	0	10.0.1.21	i-aeb52b7d
Cloud Controller Worker	0	10.0.1.22	i-41b42a92
UAA	0	10.0.1.23	i-36b52be5
MySQL Proxy	0	10.0.1.25	i-2c9e07ff
MySQL Server	0	10.0.1.26	i-11b22cc2

- Click **Credentials** and record the Mysql Admin Credentials of **MySQL Server**.

MySQL Server	Vm Credentials	vcap / 26c36dffffb49c638
	Mysql Admin Credentials	root / bf45c700a0c74d54d066

- Dump the data from the p-mysql database and save it:

```
$ mysqldump -u root -p -h 10.85.52.105 --all-databases > user_databases.sql
```

Start Cloud Controller

- Run `bosh vms` to view the list of VMs in your selected deployment.

```
$ bosh VMs
```

Notice that all the Cloud Controller VMs begin with “cloud_controller”.

Job/index	State	Resource Pool	IPs
cloud_controller-partition-bd784aa8b15f/0	failing	cloud_controller-partition-bd784aa8b15f	10.85.x
cloud_controller-worker-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
clock_global-partition-bd784aa8b15f/0	running	clock_global-partition-bd784aa8b15f	10.85.x
nats-partition-bd784aa8b15f/0	running	nats-partition-bd784aa8b15f	10.85.x
router-partition-bd784aa8b15f/0	running	router-partition-bd784aa8b15f	10.85.x
uaa-partition-bd784aa8b15f/0	running	uaa-partition-bd784aa8b15f	10.85.x

- Run `bosh start SELECTED-VM` For each stopped Cloud Controller. Stopped VMs have their state listed as `failing`.

```
$ bosh start cloud_controller_worker-partition-bd784aa8b15f/0
Processing deployment manifest
-----
Processing deployment manifest
-----
You are about to start cloud_controller_worker-partition-bd784aa8b15f/0/1
Processing deployment manifest
-----
Detecting deployment changes
-----
Start cloud_controller_worker-partition-bd784aa8b15f/0/1? (type 'yes' to continue): yes
Performing `start cloud_controller_worker-partition-bd784aa8b15f/0/1'...
Director task 1428
Started preparing deployment
Started preparing deployment > Binding deployment. Done (00:00:00)
Started preparing deployment > Binding releases. Done (00:00:00)
Started preparing deployment > Binding existing deployment. Done (00:00:01)
Started preparing deployment > Binding resource pools. Done (00:00:00)
Started preparing deployment > Binding stemcells. Done (00:00:00)
Started preparing deployment > Binding templates. Done (00:00:00)
Started preparing deployment > Binding properties. Done (00:00:00)
Started preparing deployment > Binding unallocated VMs. Done (00:00:00)
Started preparing deployment > Binding instance networks. Done (00:00:00)
Done preparing deployment (00:00:01)
Started preparing package compilation > Finding packages to compile. Done (00:00:01)
Started preparing dns > Binding DNS. Done (00:00:00)
Started preparing configuration > Binding configuration. Done (00:00:13)
Started updating job cloud_controller_worker-partition-bd784aa8b15f/0 > cloud_controller_worker-partition-bd784aa8b15f/
Task 1428 done
Started 2015-02-25 17:54:28 UTC
Finished 2015-02-25 17:56:27 UTC
Duration 00:01:59
cloud_controller_worker-partition-bd784aa8b15f/0/1 has been started
```

Restoring Pivotal Cloud Foundry from Backup

Pivotal recommends frequently backing up your deployment.

The complete state of an Elastic Runtime deployment is captured by the following:

- PCF installation settings
- Critical databases:
 - Pivotal MySQL Server Database
 - Cloud Controller Database
 - UAA Database
 - Apps Manager Database
 - NFS Server Database

To back up a deployment, you must temporarily stop the Cloud Controller and save all of the above, as well as your database encryption credentials, to backup files. To restore a deployment, you must temporarily stop the Cloud Controller and restore the state of each component from the backup file.

The procedure for restoring Elastic Runtime is described here.

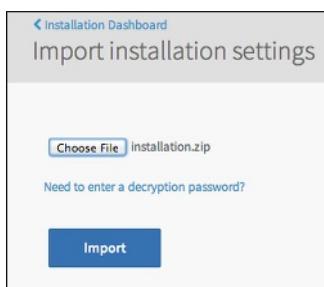
See [Backing Up Pivotal Cloud Foundry](#) for instructions for creating backups.

Note: The procedure described here will restore a running Elastic Runtime deployment to the state captured by backup files. However, it does not deploy any Elastic Runtime components. You must deploy Elastic Runtime before following the procedure described here. See the [Getting Started Guide](#) for help getting your deployment up and running.

Import Installation Settings

Note: Pivotal recommends that you export your installation settings before importing from a backup. See [Export Installation Settings](#)

1. From the Product Installation Dashboard, click the gear icon and select **Import installation settings**.
2. Browse to and select a PCF installation file.
3. If the admin password has changed since the file was exported, select **Need to enter a decryption password?** before you click **Import** to prevent an error. Enter the original password that was used when the installation was exported. If the admin password has not changed, you may skip this step.



4. Click **Import**.

Import installation settings imports the settings and assets of an existing PCF installation. Importing an installation overwrites any existing installation.

Download the BOSH Deployment Manifest

1. Install Ruby and the [BOSH CLI Ruby gem](#) on a machine outside of your PCF system deployment.
2. From the Installation Dashboard in Ops Manager, select **Ops Manager Director > Status** and record the IP

address listed for the Ops Manager Director. You access the BOSH Director using this IP address.

JOB	INDEX	IPS	CLOUD
Ops Manager Director	0	10.0.0.3	v9.0.83 d

- Click **Credentials** and record the Director credentials.

JOB	NAME	CREDENTIALS
Ops Manager Director	VM credentials	vcap / [REDACTED]
	BOSH agent credentials	vcap / [REDACTED]
	Director credentials	director / [REDACTED]
	NATS credentials	nats / [REDACTED]

- From the command line, target the BOSH Director using the IP address and credentials that you recorded:

```
$ bosh target 10.0.0.3
Target set to `microbosh-1234abcd1234abcd1234'
Your username: director
Enter password: *****
Logged in as `director'
```

Note: If `bosh target` does not prompt you for your username and password, run `bosh login`.

- Run `bosh deployments` to identify the name of the current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name      | Release(s)          | Stemcell(s)           |
+-----+-----+
| cf-example | cf-mysql/10        | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|           | cf/183.2            |                      |
+-----+-----+
```

- Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save the current BOSH deployment manifest. You need this manifest to locate information about your databases. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml
Deployment manifest saved to `cf.yml'
```

Restoring Critical Databases

Refer to this section for help restoring databases associated with your PCF installation.

Your Elastic Runtime deployment contains several critical data stores that must be present for a complete restore. You must restore each of the following:

- Cloud Controller Database
- UAA Database
- Apps Manager Database
- NFS Server
- Pivotal MySQL Server

Stop Cloud Controller

1. If you have not already, download the BOSH manifest as instructed in the [Download the BOSH Manifest](#) section.
2. Run `bosh deployment DEPLOYMENT-MANIFEST` to set your deployment.

```
$ bosh deployment cf.yml  
Deployment set to `/home/working_directory(cf.yml`)
```

3. Run `bosh vms` in the terminal to view the list of VMs in your selected deployment.

```
$ bosh vms
```

Notice that all the Cloud Controller VMs begin with “cloud_controller”.

Job/index	State	Resource Pool	IPs
cloud_controller-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
cloud_controller_worker-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
clock_global-partition-bd784aa8b15f/0	running	clock_global-partition-bd784aa8b15f	10.85.x
nats-partition-bd784aa8b15f/0	running	nats-partition-bd784aa8b15f	10.85.x
router-partition-bd784aa8b15f/0	running	router-partition-bd784aa8b15f	10.85.x
uaa-partition-bd784aa8b15f/0	running	uaa-partition-bd784aa8b15f	10.85.x

4. Run `bosh stop SELECTED-VM` for each Cloud Controller VM.

```
$ bosh stop cloud_controller-partition-bd784aa8b15f/0  
Acting as user 'director'...  
  
You are about to stop cloud_controller-partition-bd784aa8b15f/0  
  
Detecting deployment changes  
-----  
Stop cloud_controller-partition-bd784aa8b15f/0? (type 'yes' to continue)
```

You do not need to stop the Cloud Controller worker VMs.

Restore UAA Database

Stop the UAA database

1. Find your UAA database VM id. To view all VM ids, run:

```
$ bosh vms
```

2. Run `bosh stop UAADB-VM` to stop the UAA Database VM. Replace `UAADB-VM` in this command with the name of your UAA Database VM. For example:

```
$ bosh stop uaa-partition-bd784aa8b15f/0
```

Drop the UAA database tables

1. SSH into the UAA database VM using the `vcap` user and password. If you do not have this information recorded, find it in the Ops Manager Installation Dashboard. Click the **Elastic Runtime** tile and select **Credentials**.

```
$ ssh vcap@YOUR-UAADB-VM-IP-ADDRESS
```

2. Run `find /var/vcap | grep 'bin/psql'` to find the locally installed psql client on the UAA Database VM.

```
$ root@10.85.52.101:~# find /var/vcap | grep 'bin/psql'  
/var/vcap/data/packages/postgres/5.1/bin/psql
```

3. Log in to the psql client:

```
$ root@10.85.52.101:~# /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 uaa
```

- Run the following commands to drop the tables:

```
drop schema public cascade;
create schema public;
\q;
```

- Exit the UAA database VM.

```
$ exit
```

Restore the UAA database from its backup state

- Use the UAA database password and IP address to restore the UAA database by running the following commands. You can find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

- Use `scp` to copy the database backup file to the UAA database VM.

```
$ scp uaa.sql vcap@YOUR-UAADB-VM-IP-ADDRESS: #UAADB server
```

- SSH into the UAA database VM.

```
$ ssh vcap@YOUR-UAADB-VM-IP-ADDRESS
```

- Restore the database from the backup file.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 uaa < uaa.sql
```

- Restart the UAA database VM.

```
$ bosh start YOUR-UAA-VM-UUID
```

Restore NFS

Use the NFS password and IP address to restore the NFS by following the steps detailed below. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

- Use `scp` to send the NFS backup tarball to the NFS VM.

```
$ scp nfs.tar.gz vcap@YOUR-NFS-VM-IP-ADDRESS
```

- Run `ssh` to enter the NFS VM.

```
$ ssh vcap@10.0.0.10
```

- Log in as root user. When prompted for a password, enter the vcap password you used to `ssh` into the VM.:

```
$ vcap@10.0.0.10 sudo su
```

- Run the following command to create a store folder if one does not exist:

```
$root@10.0.0.10 [ ! -d /var/vcap/store ] && mkdir -p /var/vcap/store
```

- Record the permissions on `/var/vcap/store` so you can restore them after running the commands below. To view the permissions, run:

```
$root@10.0.0.10 ls -ld /var/vcap/store
```

6. Change the permissions on `/var/vcap/store` to `777`.

```
$root@10.0.0.10 chmod 777 /var/vcap/store
```

7. Move the backup archive to the `store` folder.

```
$root@10.0.0.10 mv nfs.tar.gz /var/vcap/store
```

8. `cd` into the `store` folder.

```
$root@10.0.0.10 cd /var/vcap/store
```

9. Uncompress and extract the contents of the backup archive.

```
$root@10.0.0.10 tar zx < nfs.tar.gz
```

10. Change the permissions on `/var/vcap/store` back to their prior setting.

```
$root@10.0.0.10 chmod YOUR-PRIOR-PERMISSION /var/vcap/store
```

11. Exit the NFS VM.

```
$root@10.0.0.10 exit
```

Restore the Cloud Controller Database

Use the Cloud Controller database password and IP address to restore the Cloud Controller database by following the steps detailed below. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

1. Use `scp` to send the Cloud Controller database backup file to the Cloud Controller database VM.

```
$ scp ccdb.sql vcap@YOUR-CCDB-VM-IP-ADDRESS
```

2. SSH into the Cloud Controller database VM.

```
$ ssh vcap@YOUR-CCDB-VM-IP
```

3. Log in to the `psql` client

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 ccdb
```

4. Drop the database schema and then create a new one to replace it.

```
ccdb=# drop schema public cascade;
ccdb=# create schema public;
```

5. Restore the database from the backup file.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 ccdb < ccdb.sql
```

Restore the Console Database from its backup state

1. Use the Console database password and IP address to restore the Cloud Controller database by running the following command. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

2. Use `scp` to copy the database backup file to the UUA database VM.

```
$ scp console.sql vcap@YOUR-CONSOLE-DB-VM-IP-ADDRESS
```

3. SSH into the UUA database VM.

```
$ ssh vcap@YOUR-CONSOLE-DB-VM-IP-ADDRESS
```

4. Log into the psql client.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 console
```

5. Drop the existing database schema and create a new one to replace it.

```
console=# drop schema public cascade;
console=# create schema public;
```

6. Restore the database from the backup file.

```
$ /var/vcap/data/packages/postgres/5.1/bin/psql -U vcap -p 2544 console < console.sql
```

Restore Pivotal MySQL Dev

Note: Only follow these steps if you have installed the Pivotal MySQL Dev product tile.

1. Use the Pivotal MySQL Dev database password and IP address to restore the Cloud Controller database by running the following command. Find the IP address in your BOSH deployment manifest. To find your password in the Ops Manager Installation Dashboard, select **Elastic Runtime** and click **Credentials**.

```
mysql -h YOUR-MYSQL-SERVER-IP -u root -p < user_databases.sql
```

2. Log in to the MySQL client and flush privileges.

```
$ mysql -u root -p -h
mysql > flush privileges;
```

Start Cloud Controller

1. Run `bosh vms` to view the list of VMs in your selected deployment.

```
$ bosh VMs
```

Notice that all the Cloud Controller VMs begin with “cloud_controller”.

Job/index	State	Resource Pool	IPs
cloud_controller-partition-bd784aa8b15f/0	failing	cloud_controller-partition-bd784aa8b15f	10.85.x
cloud_controller_worker-partition-bd784aa8b15f/0	running	cloud_controller-partition-bd784aa8b15f	10.85.x
clock_global-partition-bd784aa8b15f/0	running	clock_global-partition-bd784aa8b15f	10.85.x
nats-partition-bd784aa8b15f/0	running	nats-partition-bd784aa8b15f	10.85.x
router-partition-bd784aa8b15f/0	running	router-partition-bd784aa8b15f	10.85.x
uaa-partition-bd784aa8b15f/0	running	uaa-partition-bd784aa8b15f	10.85.x

2. Run `bosh start SELECTED-VM` For each stopped Cloud Controller. Stopped VMs have their state listed as `failing`.

```
$ bosh start cloud_controller_worker-partition-bd784aa8b15f/0
Processing deployment manifest
-----
Processing deployment manifest
-----
You are about to start cloud_controller_worker-partition-bd784aa8b15f/0/1
Processing deployment manifest
-----
Detecting deployment changes
-----
Start cloud_controller_worker-partition-bd784aa8b15f/0/1? (type 'yes' to continue): yes
Performing `start cloud_controller_worker-partition-bd784aa8b15f/0/1'...
Director task 1428
Started preparing deployment
Started preparing deployment > Binding deployment. Done (00:00:00)
Started preparing deployment > Binding releases. Done (00:00:00)
Started preparing deployment > Binding existing deployment. Done (00:00:01)
Started preparing deployment > Binding resource pools. Done (00:00:00)
Started preparing deployment > Binding stemcells. Done (00:00:00)
Started preparing deployment > Binding templates. Done (00:00:00)
Started preparing deployment > Binding properties. Done (00:00:00)
Started preparing deployment > Binding unallocated VMs. Done (00:00:00)
Started preparing deployment > Binding instance networks. Done (00:00:00)
  Done preparing deployment (00:00:01)
Started preparing package compilation > Finding packages to compile. Done (00:00:01)
Started preparing dns > Binding DNS. Done (00:00:00)
Started preparing configuration > Binding configuration. Done (00:00:13)
Started updating job cloud_controller_worker-partition-bd784aa8b15f/0 > cloud_controller_worker-partition-bd784aa8b15f/
Task 1428 done
Started    2015-02-25 17:54:28 UTC
Finished   2015-02-25 17:56:27 UTC
Duration   00:01:59
cloud_controller_worker-partition-bd784aa8b15f/0/1 has been started
```

Upgrading Operations Manager

Important: Read the Known Issues section of the [Pivotal Cloud Foundry Release Notes](#) before getting started.

Complete the following steps to upgrade Pivotal Cloud Foundry Operations Manager (Ops Manager). Select the procedure that corresponds to your upgrading scenario.

Pivotal strongly recommends that you [back up all critical data](#) prior to upgrading.

Note: To upgrade your [Pivotal Cloud Foundry](#) (PCF) installation to a target release, you must install all releases from your currently deployed version to the target version in sequential order. For example, if your deployment uses Ops Manager release 1.1 and you are upgrading to 1.3, you must sequentially install 1.2 and 1.3.

Upgrading to PCF 1.6

This section contains important guidelines that you must follow if you are upgrading an existing PCF 1.5 deployment to PCF 1.6. Failure to follow these instructions may jeopardize your existing deployment data or cause your upgrade to fail.

Before You Upgrade

- In Ops Manager, delete the Diego Beta tile, if installed.
- On the **Resource Config** page of your existing Elastic Runtime 1.5 configuration, scale the number of consul servers down to 1 instance. You can scale the number of instances back up after you have upgraded Elastic Runtime. If you are not running any consul servers as part of your deployment, you can ignore this step.
- If you are upgrading PCF on AWS, you should create and configure an ELB for enabling Diego SSH connections.

During the Elastic Runtime Upgrade

- If your existing PCF 1.5 deployment uses internal databases, do not modify the selected Postgres/MySQL configuration on the **System Database Config** page. PCF 1.6 allows you to deploy all of your internal databases to MySQL for improved high-availability. However, you should only select the MySQL option after you have migrated all of your existing Postgres data. If you wish to migrate your data in order to access this feature in PCF 1.6, please contact Pivotal support first.
- If your existing PCF 1.5 deployment uses an external S3 filestore, do not modify the pre-populated value used in the bucket name fields on the **File Storage Config** page. PCF 1.6 allows you to specify four different buckets for various cloud-controller artifacts; however, you should continue to use your existing 1.5 bucket names across all four buckets in order to retain existing data.
- If you wish all new applications pushed to your PCF 1.6 deployment to use [Diego](#) by default, select the **Use Diego by default instead of DEAs** option on the **Diego configuration screen**. If you do not select this option, application developers must explicitly specify to use Diego when pushing their applications. See [Diego Migration](#).

After the Upgrade

- On the **Resource Config** page of Elastic Runtime, scale your consul server instances back up to the desired number.
- Advise your application developers on targeting Diego when pushing their applications. See [Diego Migration](#).

Upgrading with Installed Products

Follow the steps below to keep all installed products when you upgrade Ops Manager.

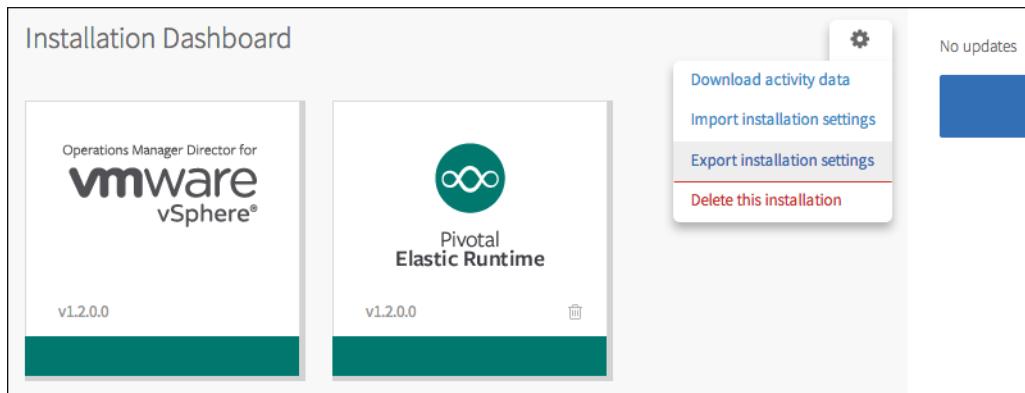
Note: If you have disabled lifecycle errands for any installed product in order to reduce deployment time, Pivotal recommends that you re-enable these errands before upgrading. See [Adding and Deleting Products](#) for more information.

1. Confirm that you have adequate disk space for your upgrades. From your Installation Dashboard, select the **Ops Manager Director** tile. Select **Status**. If you need more space to handle your upgrades, select **Settings > Resource Config**. Increase your persistent disk space to 50 GB, or enough to handle the size of the resources.

2. Ensure that every product tile on the Installation Dashboard is compatible with the new version of Ops Manager. To be compatible, a product must meet the minimum version requirement for that product. If a product does not meet this requirement, you must upgrade the product or remove the tile before upgrading Ops Manager.

For specific compatibility information, refer to the full [Product Version Matrix](#).

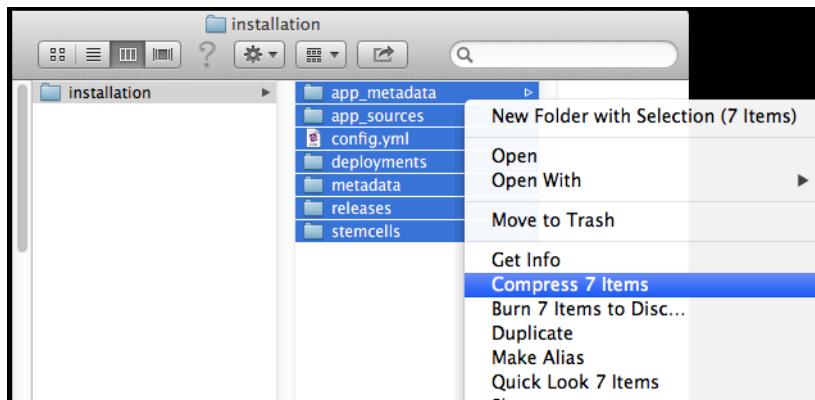
3. From the Product Dashboard, select **Actions > Export installation settings**.



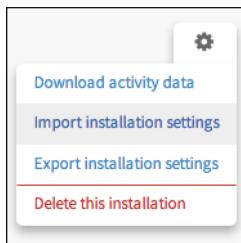
This exports the current PCF installation with all of its assets. When you export an installation, the export contains the base VM images and necessary packages, and references to the installation IP addresses. As a result, an exported file can be very large, as much as 5 GB or more.

- Because of the size of the exported file, exporting can take tens of minutes.
- Some browsers do not provide feedback on the status of the export process, and may appear to hang.
- Some operating systems may automatically unzip the exported installation. If this occurs, create a zip file of the unzipped export.

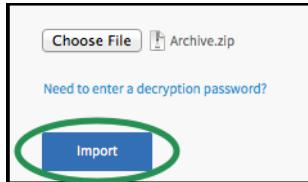
Note: When creating a zip file of an unzipped export, do not start compressing at the “installation” folder level. Instead, start compressing at the level containing the `config.yml` file:



4. Download the latest Ops Manager VM Template from the [Pivotal Network](#) site.
5. Record the IP address of the existing Ops Manager VM.
6. To avoid IP conflicts, power off the existing Ops Manager VM.
7. Deploy the new Ops Manager VM. See [Deploying Operations Manager to vSphere](#) or [Deploying Operations Manager to vCloud Air and vCloud](#).
8. From the Product Dashboard, select **Actions > Import installation settings**.



9. Click **Choose File**, browse to the installation zip file exported in Step 1, and click **Choose**.
10. If the admin password has changed since the file was exported, you must select **Need to enter a decryption password?** before you click **Import** to prevent an error. Enter the original password that was used when the installation was exported. If the admin password has not changed, you may skip this step.



11. Click **Import**.

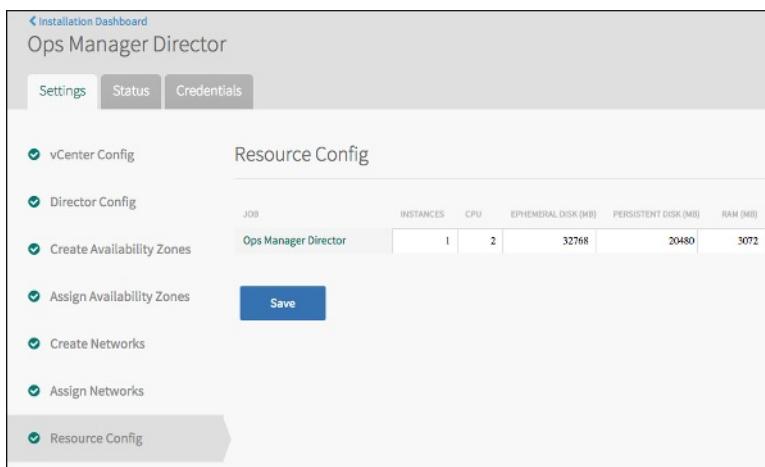
Note: Importing can take tens of minutes. Some browsers do not provide feedback on the status of the import process, and may appear to hang.

12. A “Successfully imported installation” message appears upon completion.



13. Select **Resource Config**.

14. Change the value of the **Ephemeral Disk (MB)** for the Ops Manager Director to **32768**.



15. Click **Save**.
16. Click **Apply Changes**. This immediately imports and applies upgrades to all tiles in a single transaction.
17. Click each service tile, select the **Status** tab, and confirm that all VMs appear and are in good health.
18. Remove the original Ops Manager VM from your IaaS if the new installation functions correctly.

Note: Independently from upgrading Ops Manager, you can upgrade individual products such as Pivotal Cloud Foundry Elastic Runtime, [Pivotal MySQL](#), or [RabbitMQ](#) in your PCF deployment. See [Upgrading Products in a PCF Deployment](#).

Uninstalling and Reinstalling Ops Manager

Follow these steps to upgrade Ops Manager if you have no installed products, or you have no installed products that you want to keep.

Note: If you want to reinstall Ops Manager, and would like to import your information to the new installation, you will need the exported Cloud Controller database and the configuration from the previous installation.

1. Browse to the Ops Manager web interface and select **Delete this installation**.
2. In vSphere, vCloud Air, or vCloud, power off and delete your existing Ops Manager VM.
3. Download the latest Ops Manager VM Template from the [Pivotal Network](#) site.
4. Deploy the new Ops Manager VM. See one of the following topics:
 - [Deploying Operations Manager to vSphere](#)
 - [Deploying Operations Manager to vCloud Air and vCloud](#)

Rolling Back an Upgraded Installation

Note: To roll back an installation, or if upgrading fails, Pivotal recommends that you contact Pivotal Support.

If you want to roll back or troubleshoot a failed installation on your own, consult the table below. The table shows guidelines for compatibility between your exported configuration and available Ops Manager versions.

Current Version	Can import into 1.2	Can import into 1.3	Notes
1.2	✓	✓	After successful import to 1.3, 1.2 export configuration is no longer valid.
1.3		✓	

Upgrading Elastic Runtime and Other PCF Products

Elastic Runtime Snapshot

Current PCF Elastic Runtime Details

Version: 1.6.0

Release Date: 23 October 2015

Software component version: Cloud Foundry 222

Compatible Ops Manager Version(s): 1.6.x

vSphere support? Yes

AWS support? Yes

Openstack support? Yes

Upgrading Elastic Runtime to the Latest Version

Consider the following compatibility information before upgrading Elastic Runtime for Pivotal Cloud Foundry.

 **Note:** Before you upgrade to Ops Manager 1.5.x, you must first upgrade PCF Elastic Runtime to any version in its 1.3.x minor release. This allows PCF Elastic Runtime upgrades after you install OpsManager 1.5.x.

For more information, refer to the full [Product Version Matrix](#).

Ops Manager Version	Supported Upgrades from Imported Elastic Runtime Installation
1.3.x	<ul style="list-style-type: none"> • From 1.2.0 to 1.3.0, 1.3.1, 1.3.2, 1.3.3, 1.3.4, or 1.3.5 • From 1.2.1 to 1.3.0, 1.3.1, 1.3.2, 1.3.3, 1.3.4, or 1.3.5 • From 1.2.2 to 1.3.0, 1.3.1, 1.3.2, 1.3.3, 1.3.4, or 1.3.5 • From 1.3.0 to 1.3.1, 1.3.2, 1.3.3, 1.3.4, or 1.3.5 • From 1.3.1 to 1.3.2, 1.3.3, 1.3.4, or 1.3.5 • From 1.3.2 to 1.3.3, 1.3.4, or 1.3.5 • From 1.3.3 to 1.3.4 or 1.3.5 • From 1.3.4 to 1.3.5
1.4.x	<ul style="list-style-type: none"> • From 1.3.0 to 1.4.0 • From 1.3.1 to 1.4.0 • From 1.3.2 to 1.4.0 • From 1.3.3 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.4 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.5 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.4.0 to 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.4.1 to 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.4.2 to 1.4.3, 1.4.4, or 1.4.5 • From 1.4.3 to 1.4.4 or 1.4.5 • From 1.4.4 to 1.4.5

1.5.x	<ul style="list-style-type: none"> • From 1.3.0 to 1.4.0 • From 1.3.1 to 1.4.0 • From 1.3.2 to 1.4.0 • From 1.3.3 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.4 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.5 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.4.0 to 1.4.1, 1.4.2, 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.1 to 1.4.2, 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.2 to 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.3 to 1.4.4, 1.4.5, or 1.5.0 • From 1.4.4 to 1.4.5 or 1.5.0 • From 1.4.5 to 1.5.0
1.6.x	<ul style="list-style-type: none"> • From 1.3.0 to 1.4.0 • From 1.3.1 to 1.4.0 • From 1.3.2 to 1.4.0 • From 1.3.3 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.4 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.3.5 to 1.4.0, 1.4.1, 1.4.2, 1.4.3, 1.4.4, or 1.4.5 • From 1.4.0 to 1.4.1, 1.4.2, 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.1 to 1.4.2, 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.2 to 1.4.3, 1.4.4, 1.4.5, or 1.5.0 • From 1.4.3 to 1.4.4, 1.4.5, or 1.5.0 • From 1.4.4 to 1.4.5 or 1.5.0 • From 1.4.5 to 1.5.0 • From 1.5.0 to 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, or 1.6.0 • From 1.5.1 to 1.5.2, 1.5.3, 1.5.4, 1.5.5, 1.5.6, or 1.6.0 • From 1.5.2 to 1.5.3, 1.5.4, 1.5.5, 1.5.6, or 1.6.0 • From 1.5.3 to 1.5.4, 1.5.5, 1.5.6, or 1.6.0 • From 1.5.4 to 1.5.5, 1.5.6, or 1.6.0 • From 1.5.5 to 1.5.6 or 1.6.0 • From 1.5.6 to 1.6.0

Install via Pivotal Operations Manager

To install Elastic runtime for PCF, follow the procedure for installing Pivotal Cloud Foundry Operations Manager tiles:

1. Download the product file from [Pivotal Network](#).
2. Upload the product file to your Ops Manager installation.
3. Click **Add** next to the uploaded product description in the Available Products view to add this product to your staging area.
4. Click the newly added tile to review any configurable options.
5. Click **Apply Changes** to install the service.

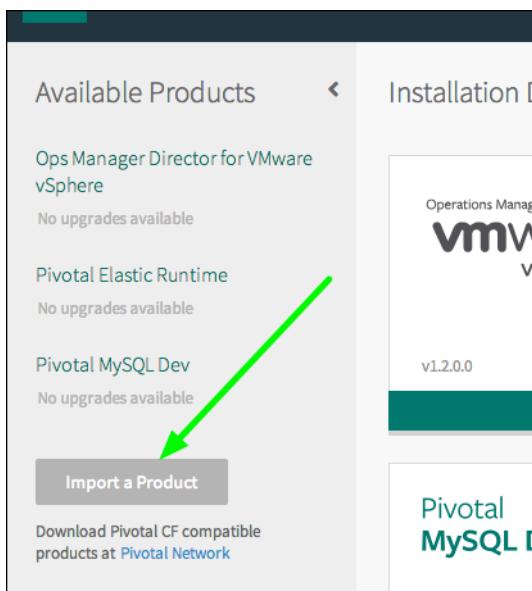
Upgrading PCF Products

Important: Read the Known Issues section of the [Pivotal Cloud Foundry Release Notes](#) before getting started.

This section describes how to upgrade individual products like Pivotal Cloud Foundry Elastic Runtime, [Pivotal MySQL](#), or [RabbitMQ](#) in your [Pivotal Cloud Foundry](#) (PCF) deployment.

Note: To upgrade your PCF product to a target release, you must install all releases from your currently deployed version to the target version in sequential order. For example, if your deployment uses Elastic Runtime release 1.1 and you are upgrading to 1.3, you must sequentially install 1.2 and 1.3.

1. Browse to [Pivotal Network](#) and sign in.
2. Download the latest PCF release for the product or products you want to upgrade. Every product is tied to exactly one stemcell. Download the stemcell that matches your product and version.
3. Confirm that you have adequate disk space for your upgrades. From your Installation Dashboard, select the **Ops Manager Director** tile. Select **Status**. If you need more space to handle your upgrades, select **Settings > Resource Config**. Increase your persistent disk space to 50 GB, or enough to handle the size of the resources.
4. Browse to the Pivotal Cloud Foundry Operations Manager web interface and click **Import a Product**.



Note: As of release 1.4.0.0, Pivotal Ops Manager no longer supports older versions of PCF products. You must update all products to at least version 1.2 (except for RabbitMQ, which must be updated to at least version 1.3.4.0) before importing them into Ops Manager 1.4.0.0.

5. Upload the new version of a product you want to upgrade.
6. Under **Available Products**, click **Upgrade** for the uploaded product.
7. Repeat the import, upload, and **Upgrade** steps for each product you downloaded.
8. If you are upgrading a product that uses a self-signed certificate from version 1.1 to 1.2, you must configure the product to trust the self-signed certificate.
To do this:
 - Click the product tile.
 - In the left-hand column, select the setting page containing the SSL certificate configuration. For example, for Elastic Runtime, select the **HAProxy** page.
 - Check the **Trust Self-Signed Certificates** box.
 - Click **Save**.
9. Click **Apply changes**.

Monitoring VMs in PCF

This topic covers strategies for monitoring VM status and performance in [Pivotal Cloud Foundry](#).

Monitoring VMs Using the Ops Manager Interface

Click any product tile and select the **Status** tab to view monitoring information.

Pivotal Elastic Runtime											
JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
HAProxy	0	10.0.0.254	vm-9985a13c-106a-48d1-a3de-d0e0e816c857	0.06%	0.1%	9.6%	0.0%	41%	5%	N/A	Download
NATS	0	10.0.0.5	vm-dee49615-aea8-4f4f-bf0f-b1060083ddef vm-6d43e59e-	0.12%	0.1%	9.7%	0.0%	41%	21%	N/A	Download

The columns display the following information:

VM Data Point	Details
Job	Each job represents a component running on one or more VMs that Ops Manager deployed.
Index	For jobs that run across multiple VMs, the index value indicates the order in which the job VMs were deployed. For jobs that run on only one VM, the VM has an index value of 0.
IPs	IP address of the job VM.
CID	Uniquely identifies the VM.
Load Avg15	CPU load average over 15 minutes.
CPU	Current CPU usage.
Memory	Current memory usage.
Swap	Swap file percentage.
System Disk	System disk space usage.
Ephem. Disk	Ephemeral disk space usage.
Pers. Disk	Persistent disk space usage.
Logs	Download link for the most recent log files.

Operations Manager VM Disk Space

The Ops Manager stores its logs on the Ops Manager VM in the `/tmp` directory.

Note: The logs collect over time and do not self-delete. To prevent the VM from running out of disk space, restart the VM to clear the log entries from `/tmp`.

Monitoring in vSphere

To monitor VMs using the vSphere client:

1. Connect to a vCenter Server instance using the vSphere client.
2. Navigate to the **Hosts And Clusters** or **VMs And Templates** inventory view.
3. In the inventory tree, select a virtual machine.
4. Select the **Performance** tab from the content pane on the right.

VMware vSphere Server provides alarms that monitor VMs, as well as clusters, hosts, datacenters, datastores, networks, and licensing. To view preconfigured alarms, including disk usage alarms, related to a particular VM:

1. In the vSphere client, select the VM you want to monitor.
2. At the bottom left of the client window, click **Alarms**.
3. If a VM starts to run out of disk space, an alarm appears in the bottom panel.

Monitoring in vCloud Air

[vCenter Operations Manager](#) collects performance data from the virtual machines and disk drives in a deployment.

[vCenter Hyperic](#) specifically monitors operating systems, middleware, and applications.

Use vCenter Operations Manager and vCenter Hyperic to monitor the following services on the vCloud Director cells in your PCF deployment:

- **vmware-vcd-watchdog**: Watchdog service for the cell.
- **vmware-guestd**: VMware Tools service. Provides heartbeat, shutdown, restart, and custom script execution functionality.
- **vmware-vcd-log-collection-agent**: Log collection service for the cell.
- **vmware-vcd-cell**: vCloud services for the cell.

PCF Troubleshooting Guide

This guide provides help with diagnosing and resolving issues encountered during a [Pivotal Cloud Foundry](#) (PCF) installation. For help troubleshooting issues that are specific to PCF deployments on VMware vSphere, refer to the topic on [Troubleshooting Ops Manager for VMware vSphere](#).

An install or update can fail for many reasons. Fortunately, the system tends to heal or work around hardware or network faults. By the time you click the [Install](#) or [Apply Changes](#) button again, the problem may be resolved.

Some failures produce only generic errors like `Exited with 1`. In cases like this, where a failure is not accompanied by useful information, retry clicking [Install](#) or [Apply Changes](#).

When the system does provide informative evidence, review the [Common Problems](#) section at the end of this guide to see if your problem is covered there.

Besides whether products install successfully or not, an important area to consider when troubleshooting is communication between VMs deployed by Pivotal Cloud Foundry. Depending on what products you install, communication takes the form of messaging, routing, or both. If they go wrong, an installation can fail. For example, in an Elastic Runtime installation the PCF VM tries to push a test application to the cloud during post-installation testing. The installation fails if the resulting traffic cannot be routed to the HA Proxy load balancer.

Viewing the Debug Endpoint

The debug endpoint is a web page that provides information useful in troubleshooting. If you have superuser privileges and can view the Ops Manager Installation Dashboard, you can access the debug endpoint.

- In a browser, open the URL:
`https://OPS-MANAGER-IP-ADDRESS/debug`

The debug endpoint offers three links:

- Files* allows you to view the YAML files that Ops Manager uses to configure products that you install. The most important YAML file, `installation.yml`, provides networking settings and describes `microbosh`. In this case, `microbosh` is the VM whose BOSH Director component is used by Ops Manager to perform installations and updates of Elastic Runtime and other products.
- Components* describes the components in detail.
- Rails log* shows errors thrown by the VM where the Ops Manager web application (a Rails application) is running, as recorded in the `production.log` file. See the next section to learn how to explore other logs.

Logging Tips

Identifying Where to Start

This section contains general tips for locating where a particular problem is called out in the log files. Refer to the later sections for tips regarding specific logs (such as those for Elastic Runtime Components).

- Start with the largest and most recently updated files in the job log
- Identify logs that contain 'err' in the name
- Scan the file contents for a "failed" or "error" string

Viewing Logs for Elastic Runtime Components

To troubleshoot specific Elastic Runtime components by viewing their log files, browse to the Ops Manager interface and follow the procedure below.

- In Ops Manager, browse to the **Pivotal Elastic Runtime > Status** tab. In the **Job** column, locate the component of interest.
- In the **Logs** column for the component, click the download icon.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
HAProxy	0	10.0.0.254	vm-9985a13c-106a-48d1-a3de-d0e0e816c857	0.08%	0.1%	8.7%	0.0%	41%	5%	N/A	
NATS	0	10.0.0.5	vm-dee49615-aea8-4f4f-bf0f-b1060083ddef	0.05%	0.2%	8.9%	0.0%	41%	19%	N/A	
			vm-6d43e59e-								

3. Browse to the **Pivotal Elastic Runtime > Logs** tab.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

FILENAME	UPDATED AT
Downloaded:	
/var/tempest/workspaces/default/jobs_logs/dea-0-792ba29b2bc8.zip	2014-02-20 18:27:22 UTC
Pending:	
/var/tempest/workspaces/default/jobs_logs/cloud_controller-0-6d1150865104.zip	2014-02-20 18:27:18 UTC

4. Once the zip file corresponding to the component of interest moves to the **Downloaded** list, click the linked file path to download the zip file.

5. Once the download completes, unzip the file.

The contents of the log directory vary depending on which component you view. For example, the DEA log directory contains subdirectories for the `dea_logging_agent`, `dea_next`, `monit`, and `warden` processes. To view the standard error stream for `warden`, download the DEA logs and open `dea.0.job > warden > warden.stderr.log`.

Viewing Web Application and BOSH Failure Logs in a Terminal Window

You can obtain diagnostic information from the Operations Manager by logging in to the VM where it is running. To log in to the Operations Manager VM, you need the following information:

- The IP address of the PCF VM shown in the `Settings` tab of the Ops Manager Director tile.
- Your **import credentials**. Import credentials are the username and password used to import the PCF `.ova` or `.ovf` file into your virtualization system.

Complete the following steps to log in to the Operations Manager VM:

1. Open a terminal window.
2. Run `ssh IMPORT-USERNAME@PCF-VM-IP-ADDRESS` to connect to the PCF installation VM.
3. Enter your import password when prompted.

4. Change directories to the home directory of the web application:

```
cd /home/tempest-web/tempest/web/
```

5. You are now in a position to explore whether things are as they should be within the web application.

You can also verify that the `microbosh` component is successfully installed. A successful MicroBOSH installation is required to install Elastic Runtime and any products like databases and messaging services.

6. Change directories to the BOSH installation log home:

```
cd /var/tempest/workspaces/default/deployments/micro
```

7. You may want to begin by running a tail command on the `current` log:

```
cd /var/tempest/workspaces/default/deployments/micro
```

If you are unable to resolve an issue by viewing configurations, exploring logs, or reviewing common problems, you can troubleshoot further by running BOSH diagnostic commands with the BOSH Command-Line Interface (CLI).

Note: Do not manually modify the deployment manifest. Operations Manager will overwrite manual changes to this manifest. In addition, manually changing the manifest may cause future deployments to fail.

Viewing Apps Manager Logs in a Terminal Window

The [Apps Manager](#) provides a graphical user interface to help manage organizations, users, applications, and spaces.

When troubleshooting Apps Manager performance, you might want to view the Apps Manager application logs. To view the Apps Manager application logs, follow these steps:

1. Run `cf login -a api.MY-SYSTEM-DOMAIN -u admin` from a command line to log in to PCF using the UAA Administrator credentials. In Pivotal Ops Manager, refer to **Pivotal Elastic Runtime > Credentials** for these credentials.

```
$ cf login -a api.example.com -u admin
API endpoint: api.example.com

Password>*****
Authenticating...
OK
```

2. Run `cf target -o system -s apps-manager` to target the `system` org and the `apps-manager` space.

```
$ cf target -o system -s apps-manager
```

3. Run `cf logs apps-manager` to tail the Apps Manager logs.

```
$ cf logs apps-manager
Connected, tailing logs for app apps-manager in org system / space apps-manager as
admin...
```

Changing Logging Levels for the Apps Manager

The Apps Manager recognizes the `LOG_LEVEL` environment variable. The `LOG_LEVEL` environment variable allows you to filter the messages reported in the Apps Manager log files by severity level. The Apps Manager defines severity levels using the Ruby standard library [Logger class](#).

By default, the Apps Manager `LOG_LEVEL` is set to `info`. The logs show more verbose messaging when you set the `LOG_LEVEL` to `debug`.

To change the Apps Manager `LOG_LEVEL`, run `cf set-env apps-manager LOG_LEVEL` with the desired severity level.

```
$ cf set-env apps-manager LOG_LEVEL debug
```

You can set `LOG_LEVEL` to one of the six severity levels defined by the Ruby Logger class:

- **Level 5:** `unknown` – An unknown message that should always be logged
- **Level 4:** `fatal` – An unhandleable error that results in a program crash

- **Level 3:** `error` – A handleable error condition
- **Level 2:** `warn` – A warning
- **Level 1:** `info` – General information about system operation
- **Level 0:** `debug` – Low-level information for developers

Once set, the Apps Manager log files only include messages at the set severity level and above. For example, if you set `LOG_LEVEL` to `fatal`, the log includes `fatal` and `unknown` level messages only.

Common Issues

Compare evidence that you have gathered to the descriptions below. If your issue is covered, try the recommended remediation procedures.

BOSH Does Not Reinstall

You might want to reinstall BOSH for troubleshooting purposes. However, if PCF does not detect any changes, BOSH does not reinstall. To force a reinstall of BOSH, select **Ops Manager Director > Resource Sizes** and change a resource value. For example, you could increase the amount of RAM by 1.

Creating Bound Missing VMs Times Out

This task happens immediately following package compilation, but before job assignment to agents. For example:

```
cloud_controller/0: Timed out pinging to f690db09-876c-475e-865f-2cece06aba79 after 600 seconds (00:10:24)
```

This is most likely a NATS issue with the VM in question. To identify a NATS issue, inspect the agent log for the VM. Since the BOSH director is unable to reach the BOSH agent, you must access the VM using another method. You will likely also be unable to access the VM using TCP. In this case, access the VM using your virtualization console.

To diagnose:

1. Access the VM using your virtualization console and log in.
2. Navigate to the **Credentials** tab of the **Elastic Runtime** tile and locate the VM in question to find the **VM credentials**.
3. Become root.
4. Run `cd /var/vcap/bosh/log`.
5. Open the file `current`.
6. First, determine whether the BOSH agent and director have successfully completed a handshake, represented in the logs as a “ping-pong”:

```
2013-10-03\14:35:48.58456 #[608] INFO: Message: {"method"=>"ping", "arguments"=>[], "reply\_to"=>"director.f4b7df14-cb8f.19719508-e0dd-4f53-b755-58b6336058ab"}  
2013-10-03\14:35:48.60182 #[608] INFO: reply\_to: director.f4b7df14-cb8f.19719508-e0dd-4f53-b755-58b6336058ab:  
payload: {:value=>"pong"}
```

This handshake must complete for the agent to receive instructions from the director.

7. If you do not see the handshake, look for another line near the beginning of the file, prefixed `INFO: loaded new infrastructure settings`. For example:

```
2013-10-03\_\_14:35:21.83222 # [608] INFO: loaded new infrastructure settings:
{"vm"=>{"name"=>"vm-4d80ede4-b0a5-4992-ae6a0386e18e", "id"=>"vm-360"}, 
"agent\_id"=>"56aea4ef-6aa9-4c39-8019-7024ccfdde4",
"networks"=>{"default"=>{"ip"=>"192.168.86.19",
"netmask"=>"255.255.255.0", "cloud\_properties"=>{"name"=>"VMNetwork"}, 
"default"=>["dns", "gateway"], 
"dns"=>["192.168.86.2", "192.168.86.17"], "gateway"=>"192.168.86.2",
"dns\_record\_name"=>"0.nats.default.cf-d729343071061.microbosh",
"mac"=>"00:50:56:9b:71:67"}, "disks"=>{"system"=>0, "ephemeral"=>1,
"persistent"=>{}}, "ntp"=>[], "blobstore"=>{"provider"=>"dav",
"options"=>{"endpoint"=>"http://192.168.86.17:25250",
"user"=>"agent", "password"=>"agent"}, 
"mbus"=>"nats://nats:nats@192.168.86.17:4222",
"env"=>{"bosh"=>{"password"=>"$6$40ftQ9K4rvvC/8ADZHW0"}}}
```

This is a JSON blob of key/value pairs representing the expected infrastructure for the BOSH agent. For this issue, the following section is the most important:

```
"mbus"=>"nats://nats:nats@192.168.86.17:4222"
```

This key/value pair represents where the agent expects the NATS server to be. One diagnostic tactic is to try pinging this NATS IP address from the VM to determine whether you are experiencing routing issues.

Install Exits With a Creates/Updates Deletes App Failure or With a 403 Error

Scenario 1: Your PCF install exits with the following 403 error when you attempt to log in to the Apps Manager:

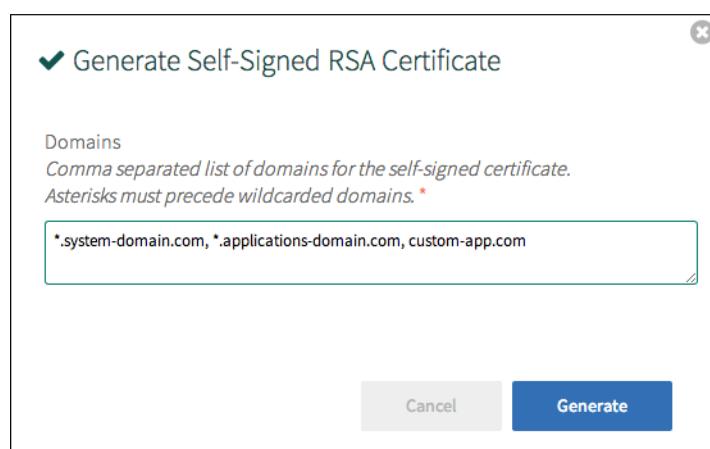
```
{"type": "step_finished", "id": "apps-manager.deploy"}  
/home/tempest-web/tempest/web/vendor/bundle/ruby/1.9.1/gems/mechanize-2.7.2/lib/mechanize/http/agent.rb:306:in  
'fetch': 403 => Net::HTTPForbidden for https://login.api.x.y/oauth/authorizeresponse_type=code&client_id=portal&redirect_uri=https%3...  
-- unhandled response (Mechanize::ResponseCodeError)
```

Scenario 2: Your PCF install exits with a `creates/updates/deletes an app (FAILED - 1)` error message with the following stack trace:

```
1) App CRUD creates/updates/deletes an app  
Failure/Error: Unable to find matching line from backtrace  
CFoundry::TargetRefused:  
  Connection refused - connect(2)
```

In either of the above scenarios, ensure that you have correctly entered your domains in wildcard format:

1. Browse to the Operations Manager interface IP.
2. Click the **Elastic Runtime** tile.
3. Select **HAProxy** and click **Generate Self-Signed RSA Certificate**.
4. Enter your system and app domains in wildcard format, as well as optionally any custom domains, and click **Save**. Refer to **Elastic Runtime > Cloud Controller** for explanations of these domain values.



Install Fails When Gateway Instances Exceed Zero

If you configure the number of Gateway instances to be greater than zero for a given product, you create a dependency on Elastic Runtime for that product installation. If you attempt to install a product tile with an Elastic Runtime dependency before installing Elastic Runtime, the install fails.

To change the number of Gateway instances, click the product tile, then select **Settings > Resource sizes > INSTANCES** and change the value next to the product Gateway job.

To remove the Elastic Runtime dependency, change the value of this field to `0`.

Out of Disk Space Error

PCF displays an `Out of Disk Space` error if log files expand to fill all available disk space. If this happens, rebooting the PCF installation VM clears the tmp directory of these log files and resolves the error.

Installing Ops Manager Director Fails

If the DNS information for the PCF VM is incorrectly specified when deploying the PCF .ova file, installing Ops Manager Director fails at the “Installing Micro BOSH” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

Deleting Ops Manager Fails

Ops Manager displays an error message when it cannot delete your installation. This scenario might happen if the Ops Manager Director cannot access the VMs or is experiencing other issues. To manually delete your installation and all VMs, you must do the following:

1. Use your IaaS dashboard to manually delete the VMs for all installed products, with the exception of the Ops Manager VM.
2. SSH into your Ops Manager VM and remove the `installation.yml` file from `/var/tempest/workspaces/default/`.

Note: Deleting the `installation.yml` file does not prevent you from reinstalling Ops Manager. For future deploys, Ops Manager regenerates this file when you click **Save** on any page in the Ops Manager Director.

Your installation is now deleted.

Installing Elastic Runtime Fails

If the DNS information for the PCF VM becomes incorrect after Ops Manager Director has been installed, installing Elastic Runtime with Pivotal Operations Manager fails at the “Verifying app push” step.

To resolve this issue, correct the DNS settings in the PCF Virtual Machine properties.

Cannot Attach Disk During MicroBOSH Deploy to vCloud

When attempting to attach a disk to a MicroBOSH VM, you might receive the following error:

`The requested operation cannot be performed because disk XXXXXXXX was not created properly.`

Possible causes and recommendations:

- If the account used during deployment lacks permission to access the default storage profile, attaching the disk might fail.
- vCloud Director can incorrectly report a successful disk creation even if the operation fails, resulting in subsequent error messages. To resolve this issue, redeploy MicroBOSH.

Ops Manager Hangs During MicroBOSH Install or HAProxy States “IP Address Already Taken”

During an Ops Manager installation, you might receive the following errors:

- The Ops Manager GUI shows that the installation stops at the “Setting MicroBOSH deployment manifest” task.
- When you set the IP address for the HAProxy, the “IP Address Already Taken” message appears.

When you install Ops Manager, you assign it an IP address. Ops Manager then takes the next two consecutive IP addresses, assigns the first to MicroBOSH, and reserves the second. For example:

```
10.17.108.1 - Ops Manager (User assigned)  
10.17.108.2 - MicroBOSH (Ops Manager assigned)  
10.17.108.3 - Reserved (Ops Manager reserved)
```

To resolve this issue, ensure that the next two subsequent IP addresses from the manually assigned address are unassigned.

Common Issues Caused by Firewalls

This section describes various issues you might encounter when installing Elastic Runtime in an environment that uses a strong firewall.

DNS Resolution Fails

When you install PCF in an environment that uses a strong firewall, the firewall might block DNS resolution. To resolve this issue, refer to the [Troubleshooting DNS Resolution Issues](#) section of the Preparing Your Firewall for Deploying PCF topic.

Advanced Troubleshooting with the BOSH CLI

This page assumes you are running cf CLI v6.

You must log into the BOSH Director and run specific commands using the BOSH Command-Line Interface (CLI) to perform advanced troubleshooting.

The BOSH Director runs on the VM that Ops Manager deploys on the first install of the Ops Manager Director tile. BOSH Director diagnostic commands have access to information about your entire [Pivotal Cloud Foundry](#) (PCF) installation.

 **Note:** For more troubleshooting information, refer to the [Troubleshooting Guide](#).

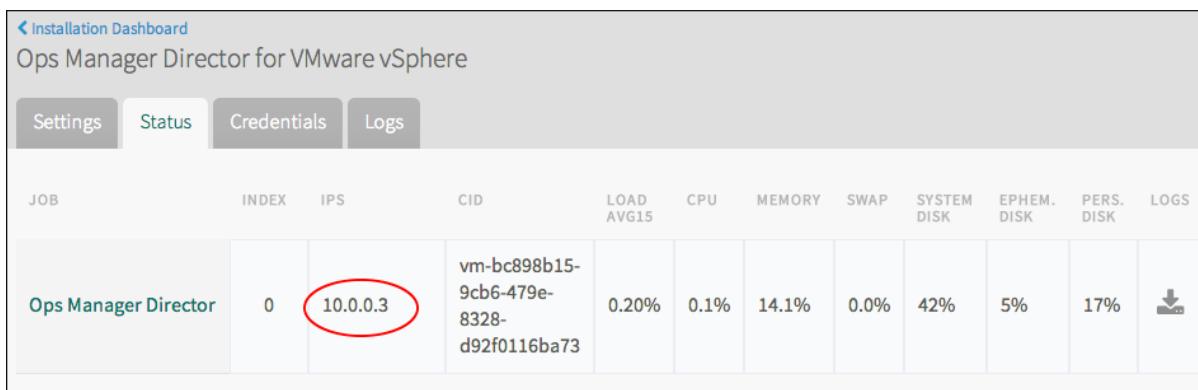
Prepare to Use the BOSH CLI

This section guides you through preparing to use the BOSH CLI.

Gather Information

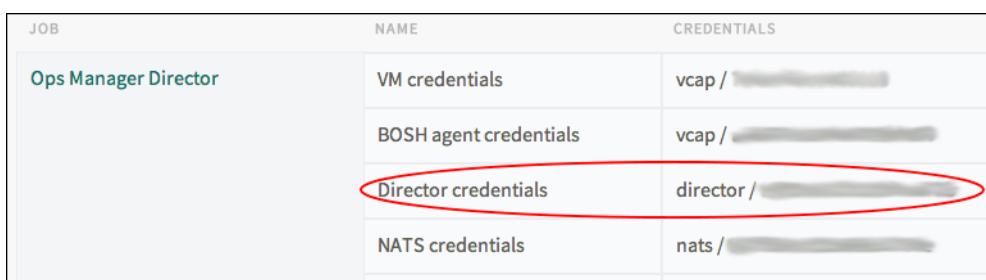
Before you begin troubleshooting with the BOSH CLI, collect the information you need from the Ops Manager interface. You must log out of the Ops Manager interface to use the BOSH CLI.

1. Open the Ops Manager interface. Ensure that there are no installations or updates in progress.
2. Click the **Ops Manager Director** tile and select the **Status** tab.
3. Record the IP address for the Ops Manager Director job. This is the IP address of the VM where the BOSH Director runs.



Ops Manager Director for VMware vSphere											
JOB	INDEX	IPS	CID	LOAD AVG15	CPU	MEMORY	SWAP	SYSTEM DISK	EPHEM. DISK	PERS. DISK	LOGS
Ops Manager Director	0	10.0.0.3	vm-bc898b15-9cb6-479e-8328-d92f0116ba73	0.20%	0.1%	14.1%	0.0%	42%	5%	17%	

4. Select the **Credentials** tab.
5. Record the BOSH Director credentials.



JOB	NAME		CREDENTIALS	
	Ops Manager Director	VM credentials	vcap /	[REDACTED]
		BOSH agent credentials	vcap /	[REDACTED]
		Director credentials	director /	[REDACTED]
		NATS credentials	nats /	[REDACTED]

6. Return to the **Installation Dashboard**.
7. **(Optional)** To prepare to troubleshoot the job VM for any other product, click the product tile and repeat the procedure above to record the IP address and VM credentials for that job VM.
8. Log out of Ops Manager.

SSH into Ops Manager

Use SSH to connect to the Ops Manager web application VM.

To SSH into the Ops Manager VM:

vSphere:

You will need the credentials used to import the PCF .ova or .ovf file into your virtualization system.

1. From a command line, run `ssh ubuntu@OPS-MANAGER-IP-ADDRESS`.
2. When prompted, enter the password you set during the .ova deployment into vCenter:

```
$ ssh ubuntu@10.0.0.6  
Password: *****
```

AWS and OpenStack:

1. Locate the Ops Manager public IP on the AWS EC2 instances page or the OpenStack Access & Security page.
2. Change the permissions on the `.pem` file to be more restrictive:

```
$ chmod 600 ops_mgr.pem
```

3. Run the `ssh` command:

```
ssh -i ops_mgr.pem ubuntu@OPS-MGR-IP
```

Log into the BOSH Director

1. To use the BOSH CLI without installing the BOSH CLI gem, set the `BUNDLE_GEMFILE` environment variable to point to the BOSH Gemfile for the Ops Manager interface. The following command creates an alias that sets `BUNDLE_GEMFILE` and calls `bundle exec bosh`. This alias allows you to run `bosh COMMAND` instead of `bundle exec bosh COMMAND`.

```
$ alias bosh="BUNDLE_GEMFILE=/home/tempest-web/tempest/web/bosh.Gemfile bundle exec bosh"
```

Note: This alias and the manual change to the `BUNDLE_GEMFILE` environment variable last only until the end of the current session.

2. Verify that no BOSH processes are running on the Ops Manager VM. You should not proceed with troubleshooting until all BOSH processes have completed or you have ended them.
3. Run `bosh target OPS-MANAGER-DIRECTOR-IP-ADDRESS` to target your Ops Manager VM using the BOSH CLI.
4. Log in using the BOSH Director credentials:

```
$ bosh target 10.0.0.6  
Target set to 'Ops Manager'  
Your username: director  
Enter password: *****  
Logged in as 'director'
```

Select a Product Deployment to Troubleshoot

When you import and install a product using Ops Manager, you deploy an instance of the product described by a YAML file. Examples of available products include Elastic Runtime, MySQL, or any other service that you imported and installed.

To select a product deployment to troubleshoot:

1. Identify the YAML file that describes the deployment you want to troubleshoot.
You identify the YAML file that describes a deployment by its filename. For example, to identify Elastic Runtime deployments, run:

```
find /var/tempest/workspaces/default/deployments -name cf-*.*yml
```

The table below shows the naming conventions for deployment files.

Product	Deployment Filename Convention
Elastic Runtime	cf-<20-character_random_string>.yml
MySQL Dev	cf_services-<20-character_random_string>.yml
Other	<20-character_random_string>.yml

 **Note:** Where there is more than one installation of the same product, record the release number shown on the product tile in Operations Manager. Then, from the YAML files for that product, find the deployment that specifies the same release version as the product tile.

2. Run `bosh status` and record the UUID value.
3. Open the `DEPLOYMENT-FILENAME.yml` file in a text editor and compare the `director_uuids` value in this file with the UUID value that you recorded. If the values do not match, do the following:
 - a. Replace the `director_uuids` value with the UUID value.
 - b. Run `bosh deployment DEPLOYMENT-FILENAME.yml` to reset the file for your deployment.
4. Run `bosh deployment DEPLOYMENT-FILENAME.yml` to instruct the BOSH Director to apply CLI commands against the deployment described by the YAML file you identified:

```
$ bosh deployment /var/tempest/workspaces/default/deployments/cf-cca18e39287264623408.yml
```

Use the BOSH CLI for Troubleshooting

Many of the BOSH CLI commands are useful in troubleshooting. This section describes three BOSH CLI commands that are particularly useful:

- **VMS:** Lists all VMs in a deployment
- **Cloudcheck:** Cloud consistency check and interactive repair
- **SSH:** Start an interactive session or execute commands with a VM

BOSH VMS

`bosh vms` provides an overview of the virtual machines BOSH is managing as part of the current deployment.

```
$ bosh vms
Deployment `cf-66987630724a2c421061'

Director task 99

Task 99 done

+-----+-----+-----+
| Job/index | State | Resource Pool | IPs   |
+-----+-----+-----+
| unknown/unknown | running | push-apps-manager | 192.168.86.13 |
| unknown/unknown | running | smoke-tests | 192.168.86.14 |
| cloud_controller/0 | running | cloud_controller | 192.168.86.23 |
| collector/0 | running | collector | 192.168.86.25 |
| consolodb/0 | running | consolodb | 192.168.86.29 |
| dea/0 | running | dea | 192.168.86.47 |
| health_manager/0 | running | health_manager | 192.168.86.20 |
| loggregator/0 | running | loggregator | 192.168.86.31 |
| loggregator_router/0 | running | loggregator_router | 192.168.86.32 |
| nats/0 | running | nats | 192.168.86.19 |
| nfs_server/0 | running | nfs_server | 192.168.86.21 |
| router/0 | running | router | 192.168.86.16 |
| saml_login/0 | running | saml_login | 192.168.86.28 |
| syslog/0 | running | syslog | 192.168.86.24 |
| uaa/0 | running | uaa | 192.168.86.27 |
| uaadb/0 | running | uaadb | 192.168.86.26 |
+-----+-----+-----+

VMs total: 15
Deployment `cf_services-2c3c918a135ab5f91ee1`

Director task 100

Task 100 done

+-----+-----+-----+
| Job/index | State | Resource Pool | IPs   |
+-----+-----+-----+
| mysql_gateway/0 | running | mysql_gateway | 192.168.86.52 |
| mysql_node/0 | running | mysql_node | 192.168.86.53 |
+-----+-----+-----+

VMs total: 2
```

When troubleshooting an issue with your deployment, `bosh vms` may show a VM in an **unknown** state. Run `bosh cloudcheck` on VMs in an **unknown** state to have BOSH attempt to diagnose the problem.

You can also use `bosh vms` to identify VMs in your deployment, then use `bosh ssh` to SSH into an identified VM for further troubleshooting.

`bosh vms` supports the following arguments:

- `--details`: Report also includes Cloud ID, Agent ID, and whether or not the BOSH Resurrector has been enabled for each VM
- `--vitals`: Report also includes load, CPU, memory usage, swap usage, system disk usage, ephemeral disk usage, and persistent disk usage for each VM
- `--dns`: Report also includes the DNS A record for each VM

Note: The **Status** tab of the **Elastic Runtime** product tile displays information similar to the `bosh vms` output.

BOSH Cloudcheck

`bosh cloudcheck` attempts to detect differences between the VM state database maintained by the BOSH Director and the actual state of the VMs. For each difference detected, `bosh cloudcheck` offers repair options:

- `Reboot VM`: Instructs BOSH to reboot a VM. Rebooting can resolve many transient errors.
- `Ignore problem`: Instructs BOSH to do nothing. You may want to ignore a problem in order to run `bosh ssh` and attempt troubleshooting directly on the machine.
- `Reassociate VM with corresponding instance`: Updates the BOSH Director state database. Use this option if you believe that the BOSH Director state database is in error and that a VM is correctly associated with a job.

- `Recreate VM using last known apply spec` : Instructs BOSH to destroy the server and recreate it from the deployment manifest the installer provides. Use this option if a VM is corrupted.
- `Delete VM reference` : Instructs BOSH to delete a VM reference in the Director state database. If a VM reference exists in the state database, BOSH expects to find an agent running on the VM. Select this option only if you know this reference is in error. Once you delete the VM reference, BOSH can no longer control the VM.

Example Scenarios

Unresponsive Agent

```
$ bosh cloudcheck  
ccdb/0 (vm-3e37133c-bc33-450e-98b1-f86d5b63502a) is not responding:  
- Ignore problem  
- Reboot VM  
- Recreate VM using last known apply spec  
- Delete VM reference (DANGEROUS!)
```

Missing VM

```
$ bosh cloudcheck  
VM with cloud ID `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` missing:  
- Ignore problem  
- Recreate VM using last known apply spec  
- Delete VM reference (DANGEROUS!)
```

Unbound Instance VM

```
$ bosh cloudcheck  
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` reports itself as `ccdb/0` but does not have a bound instance:  
- Ignore problem  
- Delete VM (unless it has persistent disk)  
- Reassociate VM with corresponding instance
```

Out of Sync VM

```
$ bosh cloudcheck  
VM `vm-3e37133c-bc33-450e-98b1-f86d5b63502a` is out of sync:  
expected `cf-d7293430724a2c421061: ccdb/0`, got `cf-d7293430724a2c421061: nats/0`:  
- Ignore problem  
- Delete VM (unless it has persistent disk)
```

BOSH SSH

Use `bosh ssh` to SSH into the VMs in your deployment.

To use `bosh ssh`:

1. Run `ssh-keygen -t rsa` to provide BOSH with the correct public key.
2. Accept the defaults.
3. Run `bosh ssh`.
4. Select a VM to access.
5. Create a password for the temporary user `bosh ssh` creates. Use this password if you need sudo access in this session.

Example:

```
$ bosh ssh
1. ha_proxy/0
2. nats/0
3. etcd_and_metrics/0
4. etcd_and_metrics/1
5. etcd_and_metrics/2
6. health_manager/0
7. nfs_server/0
8. ccdb/0
9. cloud_controller/0
10. clock_global/0
11. cloud_controller_worker/0
12. router/0
13. uaadb/0
14. uaa/0
15. login/0
16. consoledb/0
17. dea/0
18. loggregator/0
19. loggregator_traffic_controller/0
20. push-apps-manager/0
21. smoke-tests/0

Choose an instance: 17
Enter password (use it to sudo on remote host): *****
Target deployment `cf_services-2c3c918a135ab5f91ee1'

Setting up ssh artifacts
```

PCF Security Overview and Policy

This document outlines our security policy and is addressed to operators deploying [Pivotal Cloud Foundry](#) (PCF) using Pivotal Cloud Foundry Operations Manager.

For a comprehensive overview of the security architecture of each PCF component, refer to the [Cloud Foundry Security](#) topic.

How Pivotal Monitors for Security Vulnerabilities

Pivotal receives private reports on vulnerabilities from customers and from field personnel via our secure disclosure process. We also monitor public repositories of software security vulnerabilities to identify newly discovered vulnerabilities that might affect one or more of our products.

How to Report a Vulnerability

Pivotal encourages users who become aware of a security vulnerability in our products to contact Pivotal with details of the vulnerability. Please send descriptions of any vulnerabilities found to security@pivotal.io. Please include details on the software and hardware configuration of your system so that we can reproduce the issue.

 **Note:** We encourage use of encrypted email. Our public PGP key is located at <http://www.pivotal.io/security>.

Notification Policy

PCF has many customer stakeholders who need to know about security updates. When there is a possible security vulnerability identified for a PCF component, we do the following:

1. Assess the impact to PCF.
2. If the vulnerability would affect a PCF component, we schedule an update for the impacted component(s).
3. Update the affected component(s) and perform system tests.
4. Announce the fix publicly via the following channels:
 - a. Automated notification to end users who have downloaded or subscribed to a PCF product on [Pivotal Network](#) when a new, fixed version is available.
 - b. Adding a new post to <http://www.pivotal.io/security>.

Classes of Vulnerabilities

Attackers can exploit vulnerabilities to compromise user data and processing resources. This can affect data confidentiality, integrity, and availability to different degrees.

Pivotal reports the severity of vulnerabilities using the following severity classes:

Critical

Critical vulnerabilities are those that can be exploited by an unauthenticated attacker from the Internet or those that break the guest/host Operating System isolation. The exploitation results in the complete compromise of confidentiality, integrity, and availability of user data and/or processing resources without user interaction. Exploitation could be leveraged to propagate an Internet worm or execute arbitrary code between Virtual Machines and/or the Host Operating System.

Important

Important vulnerabilities are those that are not rated critical but whose exploitation results in the complete compromise of confidentiality and/or integrity of user data and/or processing resources through user assistance or by authenticated attackers. This rating also applies to those vulnerabilities that could lead to the complete compromise of availability.

when the exploitation is by a remote unauthenticated attacker from the Internet or through a breach of virtual machine isolation.

Moderate

Moderate vulnerabilities are those in which the ability to exploit is mitigated to a significant degree by configuration or difficulty of exploitation, but in certain deployment scenarios could still lead to the compromise of confidentiality, integrity, or availability of user data and/or processing resources.

Low

Low vulnerabilities are all other issues that have a security impact. These include vulnerabilities for which exploitation is believed to be extremely difficult, or for which successful exploitation would have minimal impact.

Alerts/Actions Archive

<http://www.pivotal.io/security> ↗

Cloud Foundry Concepts

Cloud Foundry is an open platform as a service, providing a choice of clouds, developer frameworks, and application services. Cloud Foundry makes it faster and easier to build, test, deploy and scale applications. It is an [open source project](#) and is available through a variety of private cloud distributions and public cloud instances.

This guide presents an overview of how Cloud Foundry works, a discussion of key concepts, and a glossary of terms. Refer to this guide to learn more about Cloud Foundry fundamentals.

Table of Contents

- [Overview](#)
- [Cloud Foundry Components](#)
- [How Applications are Staged](#)
- [Zero Downtime Deployment and Scaling in CF](#)
- [Orgs, Spaces, Roles, and Permissions](#)
- [Cloud Foundry Security](#)
- [Diego Architecture](#)
- [Diego Components](#)
- [How Diego Allocates Work](#)
- [Stacks](#)
- [Glossary](#)

Cloud Foundry Overview

What is an open PaaS?

Each generation of computing ushers in a new application platform. In the cloud era, the application platform will be delivered as a service, often described as Platform as a Service (PaaS). PaaS makes it much easier to deploy, run, and scale applications.

Not all PaaS offerings are created equal. Some have limited language and framework support, do not deliver key application services needed for cloud applications, or restrict deployment to a single cloud. By offering an open PaaS, you get a choice of clouds for deployment, frameworks for development, and application services for running your application. And as an open source project, there is a community both contributing and supporting Cloud Foundry.

What is Cloud Foundry?

Cloud Foundry is an open platform as a service, providing a choice of clouds, developer frameworks, and application services. Cloud Foundry makes it faster and easier to build, test, deploy, and scale applications. It is an [open source project](#) and is available through a variety of private cloud distributions and public cloud instances.

Cloud Foundry takes an open approach to Platform as a Service. Most PaaS offerings restrict developer choices of frameworks, application services, and deployment clouds. The open and extensible nature of Cloud Foundry means developers will not be locked into a single framework, single set of application services, or a single cloud.

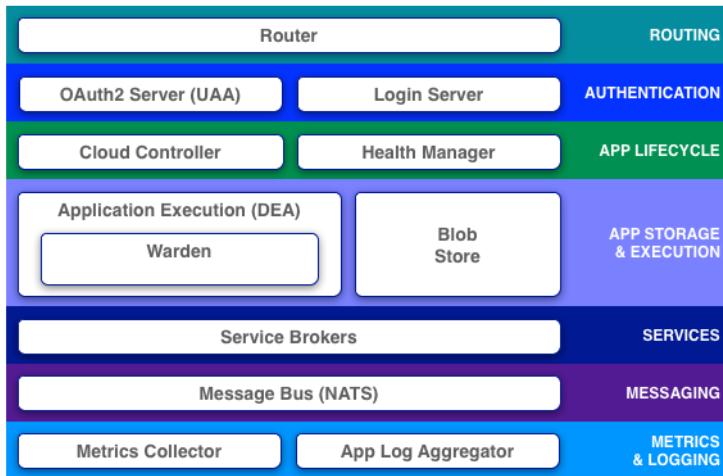
Who should use Cloud Foundry?

Cloud Foundry is ideal for any developer interested in removing the cost and complexity of configuring infrastructure for their applications. Cloud Foundry allows developers to deploy and scale their applications without being locked in to a single cloud. Developers can deploy their applications to Cloud Foundry using their existing tools and with zero modification to their code.

Cloud Foundry Components

Cloud Foundry components include a self-service application execution engine, an automation engine for application deployment and lifecycle management, and a scriptable command line interface (CLI), as well as integration with development tools to ease deployment processes. Cloud Foundry has an open architecture that includes a buildpack mechanism for adding frameworks, an application services interface, and a cloud provider interface.

Refer to the descriptions below for more information about Cloud Foundry components. Some descriptions include links to more detailed documentation.



Router

The [router](#) routes incoming traffic to the appropriate component, usually the Cloud Controller or a running application on a DEA node.

OAuth2 Server (UAA) and Login Server

The OAuth2 server (the [UAA](#)) and Login Server work together to provide identity management.

Cloud Controller

The [Cloud Controller](#) is responsible for managing the lifecycle of applications. When a developer pushes an application to Cloud Foundry, she is targeting the Cloud Controller. The Cloud Controller then stores the raw application bits, creates a record to track the application metadata, and directs a DEA node to stage and run the application. The Cloud Controller also maintains records of orgs, spaces, service instances, user roles, and more.

Health Manager (HM9000)

HM9000 has four core responsibilities:

- Monitor applications to determine their state (e.g. running, stopped, crashed, etc.), version, and number of instances. HM9000 updates the actual state of an application based on heartbeats and `droplet.exited` messages issued by the DEA running the application.
- Determine applications' expected state, version, and number of instances. HM9000 obtains the desired state of an application from a dump of the Cloud Controller database.
- Reconcile the actual state of applications with their expected state. For instance, if fewer than expected instances are running, HM9000 will instruct the Cloud Controller to start the appropriate number of instances.
- Direct Cloud Controller to take action to correct any discrepancies in the state of applications.

HM9000 is essential to ensuring that apps running on Cloud Foundry remain available. HM9000 restarts applications whenever the DEA running an app shuts down for any reason, when Warden kills the app because it violated a quota, or

when the application process exits with a non-zero exit code.

Refer to the [HM9000 readme](#) for more information about the HM9000 architecture.

Application Execution (DEA)

The [Droplet Execution Agent](#) manages application instances, tracks started instances, and broadcasts state messages.

Application instances live inside [Warden](#) containers. Containerization ensures that application instances run in isolation, get their fair share of resources, and are protected from noisy neighbors.

Blob Store

The blob store holds:

- Application code
- Buildpacks
- Droplets

Service Brokers

Applications typically depend on [services](#) such as databases or third-party SaaS providers. When a developer provisions and binds a service to an application, the service broker for that service is responsible for providing the service instance.

Message Bus (NATS)

Cloud Foundry uses [NATS](#), a lightweight publish-subscribe and distributed queueing messaging system, for internal communication between components.

Metrics Collector and App Log Aggregator: Logging and Statistics

The metrics collector gather metrics from the components. Operators can use this information to monitor an instance of Cloud Foundry.

The application log aggregator streams application logs to developers.

Diego Architecture

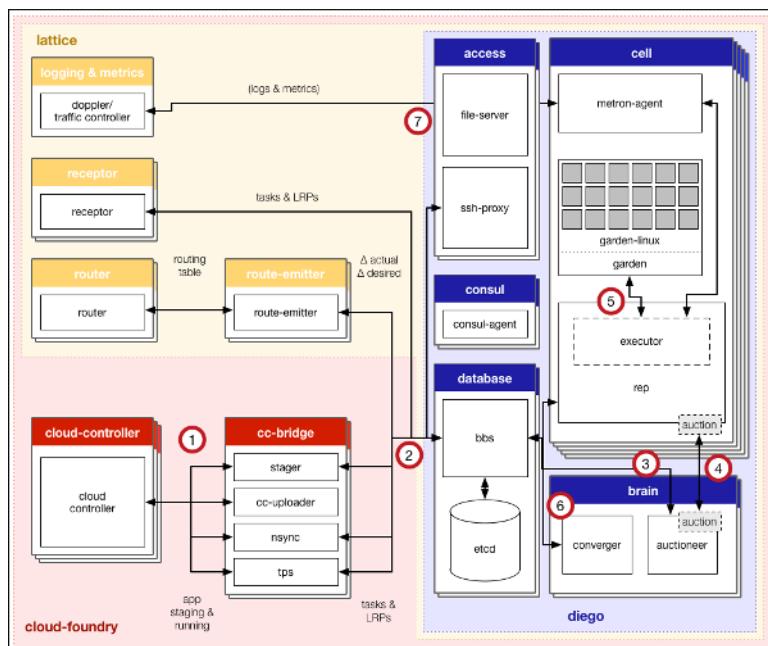
This topic provides an overview of the high-level structure of Diego, the next-generation container management system for Cloud Foundry (CF).

- See the [Using Diego in Pivotal Cloud Foundry](#) topic for information about using Diego in Pivotal Cloud Foundry.
- For more information about the major components of Diego and how they interact with other Cloud Foundry components, see the [Diego Components](#) topic.

Overview

In a Cloud Foundry deployment without Diego, the [Cloud Controller](#) schedules and manages applications on the [Droplet Execution Agents](#) (DEAs). Diego replaces the DEAs and the [Health Manager \(HM9000\)](#), and assumes application scheduling and management responsibility from the Cloud Controller.

Refer to the diagram and descriptions below for information about the way Diego handles application requests.



1. The Cloud Controller passes requests to stage and run applications to the [Cloud Controller Bridge](#) (CC-Bridge).
2. The CC-Bridge translates staging and running requests into [Tasks and Long Running Processes](#) (LRPs), then submits these to the [Bulletin Board System](#) (BBS) through an RPC-style API over HTTP.
3. The BBS submits the Tasks and LRPs to the [Auctioneer](#), part of the [Diego Brain](#).
4. The Auctioneer distributes these Tasks and LRPs to [Cells](#) through an [Auction](#).
5. Once the Auctioneer assigns a Task or LRP to a Cell, an in-process [Executor](#) creates a [Garden](#) container in the Cell. The Task or LRP runs in the container.
6. The BBS tracks desired LRPs, running LRP instances, and in-flight Tasks for the [Converger](#), part of the [Diego Brain](#). The Converger periodically analyzes this information and corrects discrepancies to ensure consistency between `ActualLRP` and `DesiredLRP` counts.
7. [Metron Agents](#), part of the Cells, forward application logs, errors, and metrics to the [Loggregator Doppler](#) component.

Monitoring and Testing

The following releases monitor and test runtime deployment:

Inigo

Inigo is an integration test suite that launches the Diego components through various test cases, including component failures and other exceptional scenarios. Inigo validates a given set of component versions to ensure mutual compatibility, robustness, and graceful performance degradation in failure conditions.

Refer to the [Inigo repo](#) on GitHub for more information.

Auction

The [auction package](#) encodes behavioral details about Task and LRP allocation to cells during a Diego [Auction](#). It includes a simulation test suite that validates the optimal performance of the auction algorithm. You can run the simulation for different algorithm variants at various scales, and in the following ways:

- In-process, for short feedback loops
- Across multiple processes, to reveal the impact of communication in the auction
- Across multiple machines in a cloud-like infrastructure, to reveal the impact of latency on the auction

Refer to the [Auction repo](#) on GitHub for more information.

Diego Acceptance Tests

Diego Acceptance Tests (DAT) is a suite of acceptance-level tests that run against Cloud Foundry and Diego deployment releases. DAT executes a number of “happy-path” test cases across an entire deployment.

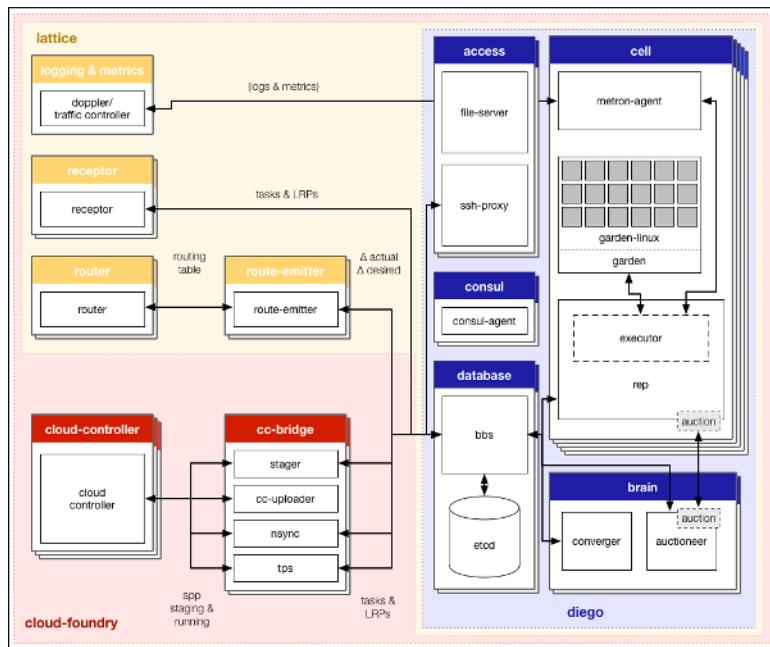
Refer to the [Diego Acceptance Test repo](#) on GitHub for more information.

Diego Components

This topic describes the components that make up the Diego container management system for Cloud Foundry (CF).

For an overview of the high-level structure of Diego, see the [Diego Architecture](#) topic.

The diagram below illustrates the primary Diego components and how they interact with other Cloud Foundry components.



[Enlarge image ↗](#)

This table describes the major areas in the image above:

Diagram Color	Scope of Responsibility	Source Code Location	Notes
Blue	Running and monitoring Tasks and Long Running Processes (LRPs)	Diego Release ↗	
Red	Translating app-specific messages from the Cloud Controller to the Bulletin Board System (BBS)	Stager ↗ CC-Uploader ↗ Nsync ↗ TPS ↗	
Yellow	Supporting streaming logs and routing to containers managed by Diego	Diego Release ↗ and CF-Release ↗	The Lattice ↗ open source project distribution includes these components and offers developers an environment for interacting with Diego.

Diego Core Components

Components in the Diego core run and monitor [Tasks](#) and [Long Running Processes](#) (LRPs). The core consists of the following major areas:

- [Brain](#)
- [Cells](#)
- [Database VMs](#)
- [Access VMs](#)
- [Consul](#)

Diego Brain

Diego Brain components distribute Tasks and LRPs to Diego [Cells](#), and correct discrepancies between [ActualLRP](#) and [DesiredLRP](#).

`DesiredLRP` counts to ensure fault-tolerance and long-term consistency. The Diego Brain consists of the [Auctioneer](#) and the [Converger](#).

Auctioneer

- Uses the [auction package](#) to run [Diego Auctions](#) for Tasks and LRPCs
- Communicates with Cell [Reps](#) over HTTP
- Maintains a lock in the [BBS](#) that restricts auctions to one Auctioneer at a time

Refer to the [Auctioneer repo](#) on GitHub for more information.

Converger

- Uses converge methods from the [Runtime Schema](#) to analyze snapshots from the [BBS](#) to ensure consistency and fault tolerance for Tasks and LRPCs
- Acts to keep `DesiredLRP` count and `ActualLRP` count synchronized in the following ways:
 - If the `DesiredLRP` count exceeds the `ActualLRP` count, the Converger requests a start auction from the [Auctioneer](#)
 - If the `ActualLRP` count exceeds the `DesiredLRP` count, the Converger sends a stop message to the [Rep](#) on the [Cell](#) hosting an instance
- Monitors for potentially missed messages, resending them if necessary
- Maintains a lock in the [BBS](#) that limits converge methods to one Converger at a time

Refer to the [Converger repo](#) on GitHub for more information.

Diego Cell Components

Diego Cell components manage and maintain Tasks and LRPCs.

Rep

- Represents a Cell in [Diego Auctions](#) for Tasks and LRPCs
- Mediates all communication between the Cell and the BBS
- Ensures synchronization between the set of Tasks and LRPCs in the BBS with the containers present on the Cell
- Maintains the presence of the Cell in the BBS
- Runs Tasks and LRPCs by asking the in-process [Executor](#) to create a container and `RunAction` recipes

Refer to the [Rep repo](#) on GitHub for more information.

Executor

- Runs as a logical process inside the [Rep](#)
- Implements the generic Executor actions detailed in the [API documentation](#)
- Streams `STDOUT` and `STDERR` to the [Metron agent](#) running on the Cell

Refer to the [Executor repo](#) on GitHub for more information.

Garden

- Provides a platform-independent server and clients to manage Garden containers
- Defines the [garden-linux](#) interface for container implementation

Refer to the [Garden repo](#) on GitHub for more information.

Metron Agent

Fowards application logs, errors, and application and Diego metrics to the [Loggregator](#) Doppler component

Refer to the [Metron repo](#) on GitHub for more information.

Database VMs

Diego Bulletin Board System

- Maintains a real-time representation of the state of the Diego cluster, including all desired LRPCs, running LRP instances, and in-flight Tasks
- Provides an RPC-style API over HTTP to [Diego Core](#) components and external clients, including the [SSH Proxy](#), [CC-Bridge](#), and [Route Emitter](#).

Refer to the [Bulletin Board System repo](#) on GitHub for more information.

etcd

- Provides a consistent key-value data store to Diego

Access VMs

File Server

- Serves static assets used by various Diego components, particularly the [App Lifecycle binaries](#)

Refer to the [File Server repo](#) on GitHub for more information.

SSH Proxy

- Brokers connections between SSH clients and SSH servers running inside instance containers

Refer to the [Diego SSH repo](#) on GitHub for more information.

Consul

- Provides dynamic service registration and load balancing through DNS resolution
- Provides a consistent key-value store for maintenance of distributed locks and component presence

Refer to the [Consul repo](#) on GitHub for more information.

Consuladapter

Consuladapter provides a driver for interfacing with etcd.

Refer to the [Consuladapter repo](#) on GitHub for more information.

Cloud Controller Bridge Components

The Cloud Controller Bridge (CC-Bridge) components translate app-specific requests from the Cloud Controller to the BBS. These components include the following:

Stager

- Translates staging requests from the Cloud Controller into generic Tasks and LRPCs
- Sends a response to the Cloud Controller when a Task completes

Refer to the [Stager repo](#) on GitHub for more information.

CC-Uploader

- Mediates uploads from the [Executor](#) to the Cloud Controller
- Translates simple HTTP POST requests from the Executor into complex multipart-form uploads for the Cloud Controller

Refer to the [CC-Uploader repo](#) on GitHub for more information.

Nsync

- Listens for app requests to update the `DesiredLRPs` count and updates `DesiredLRPs` through the BBS
- Periodically polls the Cloud Controller for each app to ensure that Diego maintains accurate `DesiredLRPs` counts

Refer to the [Nsync repo](#) on GitHub for more information.

TPS

- Provides the Cloud Controller with information about currently running LRPCs to respond to `cf apps` and `cf app APP_NAME` requests
- Monitors `ActualLRP` activity for crashes and reports them to the Cloud Controller

Refer to the [TPS repo](#) on GitHub for more information.

Platform-specific Components

Garden Backends

Garden contains a set of interfaces that each platform-specific backend must implement. These interfaces contain methods to perform the following actions:

- Create and delete containers
- Apply resource limits to containers
- Open and attach network ports to containers
- Copy files into and out of containers
- Run processes within containers
- Stream `STDOUT` and `STDERR` data out of containers
- Annotate containers with arbitrary metadata
- Snapshot containers for redeploys without downtime

Refer to the [Garden repo](#) on GitHub for more information.

Current Implementations

- [Garden-Linux](#) provides a Linux-specific implementation of a Garden interface.

App Lifecycles

App Lifecycles implement deployment strategies. Each App Lifecycle contains the following three binaries:

- The **Builder**, which stages a CF application. The [CC-Bridge](#) runs the Builder as a Task on every staging request. The Builder performs static analysis on the application code and does any necessary pre-processing before the application is first run.
- The **Launcher**, which runs a CF application. The [CC-Bridge](#) sets the Launcher as the Action on the `DesiredLRP` for the application. The Launcher executes the start command with the correct system context, including working directory and environment variables.
- The **Healthcheck**, which performs a status check on running CF application from inside the container. The [CC-Bridge](#) sets the Healthcheck as the Monitor action on the `DesiredLRP` for the application.

Current Implementations

- [Buildpack App Lifecycle](#) implements the Cloud Foundry buildpack-based deployment strategy.
- [Docker App Lifecycle](#) implements a Docker deployment strategy.

Other Components

Route-Emitter

- Monitors `DesiredLRP` and `ActualLRP` states, emitting route registration and unregistration messages to the Cloud Foundry [router](#) when it detects changes
- Periodically emits the entire routing table to the Cloud Foundry router

Refer to the [Route-Emitter repo](#) on GitHub for more information.

How Diego Allocates Work

The Diego Auction is the process by which [Diego](#) allocates tasks and long running processes (LRPs) to [Cells](#), which execute them. The auction replaces the [Cloud Controller DEA placement algorithm](#), which performed this function in the pre-Diego Cloud Foundry architecture.

[Auction on GitHub ↗](#)

Tasks and Long Running Processes

Diego runs and monitors two types of processes: **Tasks** and **Long Running Processes** (LRPs).

- **Tasks:** Tasks run once, for a finite amount of time.
Example Tasks: Making a database schema change, bulk importing data to initialize a database, setting up a connected service
- **Long Running Processes** (LRPs): LRPs run continuously, for an indefinite time. LRPs only terminate if stopped or killed, or in the case of failure. Diego can run multiple instances of any LRP.
Example LRPs: A web-server, a service that continuously accepts input and processes it, an application

Initiating an Auction

An [Auctioneer](#) initiates an auction when it receives a request to allocate a batch of work (tasks and LRPs) either from:

- A Cell [Rep](#), in response to a request from the Cloud Controller via the CC-Bridge.
- The [Converger](#), in response to a mismatch between the desired LRPs and actual LRPs as represented in the [BBS\(.diego-components.html#bbs\)](#).

Only one auction can take place at a time.

Querying the Cells

The Auctioneer queries the state of each Cell via a request to the Cell [Rep](#). The information provided by the Rep includes:

- the Cell's available capacity
- the Cell's stack
- the set of Actual LRPs currently running on the Cell

Prioritizing the Work to be Auctioned

When the Auctioneer receives a batch of work it must strategically decide the order in which to distributed it.

Two pitfalls must be avoided:

- The Auctioneer could incorrectly fill the Cells with small units of work before attempting to place large units of work. As a result, these large units could fail to place even if there is sufficient capacity in the cluster.
- The Auctioneer could incorrectly prioritize large units of work over small units of work. As a result, each Cell could be filled with a single large app, preventing smaller apps from having any instances running.

To avoid these bad results, the Auctioneer first sorts the batch of LRPs to allocate into priority groups based on their global indices. Each LRP has an index for its type, which is unique across a deployment. Within each priority group, work is sorted in order of decreasing memory demand so that larger units of work are placed first.

For example, suppose a batch contains the following 5 LRPs:

LRP type	LRP index
type 1	0
type 1	1

type 1	2
type 2	0
type 2	1

The Auctioneer would allocate the LRP in the following order:

- Type 1/ Index 0
- Type 2/ Index 0
- Type 1/ Index 1
- Type 2/ Index 1
- Type 1/ Index 2

This sorting by index ensures that LRP of each type are placed before all of the LRP of any type are, preventing both large and small LRP from dominating the allocation process.

Allocating Work

The Auctioneer calculates the distribution of Actual LRP across available Cells.

For each Task and LRP in the batch, The Auctioneer does the following:

1. It filters the pool of candidate cells to those with the correct stack and sufficient resources to host the actual LRP.
2. It chooses the winning cell by optimizing for the following, in increasing priority:
 - a. an even distribution of memory, disk, and container usage across Cells (all with equal weighting)
 - b. minimizing colocation of ActualLRP instances of a given DesiredLRP on the same Cell (takes precedence over the distribution of memory/disk/etc.).
 - c. minimizing colocation of ActualLRP instances of a given DesiredLRP in the same Availability Zone

After calculating the distribution of Tasks and LRP to Cells, the Auctioneer submits requests to the Cells to execute their allotted work.

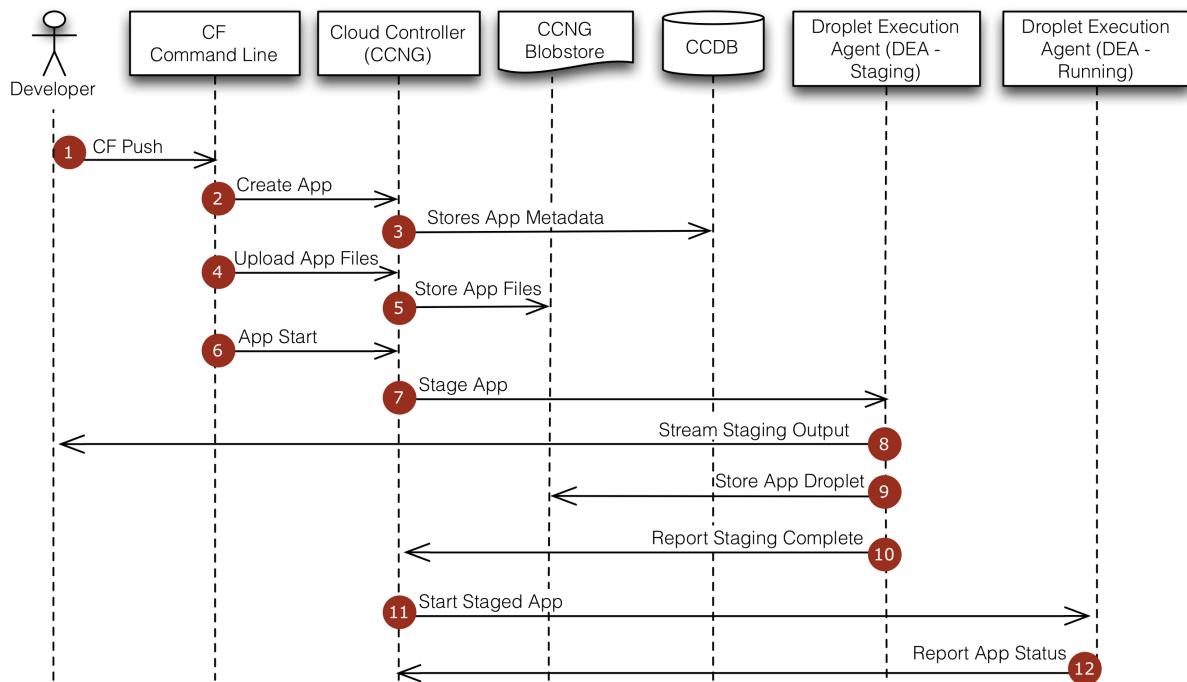
Any work that could not be allocated is carried over into the next batch.

If a Cell responds by saying the work can not be performed, the Auctioneer carries the failed work over into the next batch.

If a Cell fails to respond, for example if a connection times out, the Auctioneer does *not* carry the work over into the next batch. In this case the Auctioneer defers to the Converger to figure out what to do.

Any work carried over into the next batch is merged with work that came in during the previous auction, and the loop is repeated.

How Applications Are Staged



1. At the command line, the developer enters the directory containing her application and uses the cf command line tool to issue a push command.
2. The cf command line tool tells the Cloud Controller to create a record for the application.
3. The Cloud Controller stores the application metadata (e.g. the app name, number of instances the user specified, and the buildpack).
4. The cf command line tool uploads the application files.
5. The Cloud Controller stores the raw application files in the blobstore.
6. The cf command line tool issues an app start command.
7. Because the app has not already been staged, the Cloud Controller chooses a DEA instance from the DEA pool to stage the application. The staging DEA uses the instructions in the buildpack to stage the application.
8. The staging DEA streams the output of the staging process so the developer can troubleshoot application staging problems.
9. The staging DEA packages the resulting staged application into a tarball called a “droplet” and stores it in the blobstore. The results are cached and used next time the application is staged.
10. The staging DEA reports to the Cloud Controller that staging is complete.
11. The Cloud Controller chooses one or more DEAs from the pool to run the staged application.
12. The running DEAs report the status of the application to the Cloud Controller.

Zero Downtime Deployment and Scaling in CF

To increase the capacity and availability of the Cloud Foundry platform, and to decrease the chances of downtime, you can scale a deployment up using the strategies described below.

This topic also describes the requirements for a zero downtime deployment. A zero downtime deployment ensures that if individual components go down, your deployment continues to run.

Zero Downtime Deployment

This section describes the required configurations for achieving a zero downtime deployment.

Application Instances

Deploy at least two instances of every application.

Components

Scale your components as described in the [Scaling Platform Availability](#) section below. Components should be distributed across two or more [availability zones](#) (AZs).

Space

Ensure that you allocate and maintain enough of the following:

- Free space on DEAs so that apps expected to deploy can successfully be staged and run.
- Disk space and memory in your deployment such that if one DEA is down, all instances of apps can be placed on the remaining DEAs.
- Free space to handle one AZ going down if deploying in multiple AZs.

Resource pools

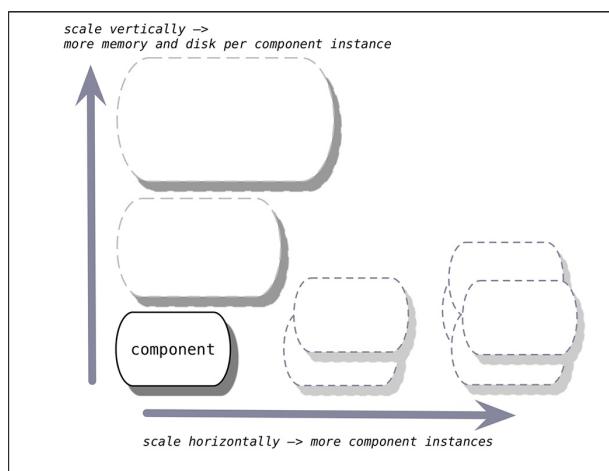
Configure your [resource pools](#) according to the requirements of your deployment.

Ops Manager Resurrector

Enable the [Ops Manager Resurrector](#).

Scaling Platform Capacity

You can scale platform capacity vertically by adding memory and disk, or horizontally by adding more VMs running instances of Cloud Foundry components.



Trade-offs and Benefits

The nature of a particular application should determine whether you scale vertically or horizontally.

DEAs:

The optimal sizing and CPU/memory balance depends on the performance characteristics of the apps that will run on the DEA.

- The more DEAs are horizontally scaled, the higher the number of NATS messages the DEAs generate. There are no known limits to the number of DEA nodes in a platform.
- The denser the DEAs (the more vertically scaled they are), the larger the NATS message volume per DEA, as each message includes details of each app instance running on the DEA.
- Larger DEAs also make for larger points of failure: the system takes longer to rebalance 100 app instances than to rebalance 20 app instances.

Router:

Scale the router with the number of incoming requests. In general, this load is much less than the load on DEA nodes.

Health Manager:

The Health Manager works as a failover set, meaning that only one Health Manager is active at a time. For this reason, you only need to scale the Health Manager to deal with instance failures, not increased deployment size.

Cloud Controller:

Scale the Cloud Controller with the number of requests to the API and with the number of apps in the system.

Scaling Platform Availability

To scale the Cloud Foundry platform for high availability, the actions you take fall into three categories.

- For components that support multiple instances, increase the number of instances to achieve redundancy.
- For components that do not support multiple instances, choose a strategy for dealing with events that degrade availability.
- For database services, plan for and configure backup and restore where possible.

 **Note:** Data services may have single points of failure depending on their configuration.

Scalable Processes

You can think of components that support multiple instances as scalable processes. If you are already scaling the number of instances of such components to increase platform capacity, you need to scale further to achieve the redundancy required for high availability.

Component	Number	Notes
Load Balancer	1	
MySQL Server	≥ 3	Set this to an odd number to support cluster quorum. For more information about cluster quorum, see Cluster Scaling, Node Failure, and Quorum .
MySQL Proxy	≥ 2	
NATS Server	≥ 2	If you lack the network bandwidth, CPU utilization, or other resources to deploy two stable NATS servers, Pivotal recommends that you use one NATS server.
HM9000	≥ 2	
Cloud Controller	≥ 2	More Cloud Controllers help with API request volume.
Gorouter	≥ 2	Additional Gorouters help bring more available bandwidth to ingress and egress.
Collector	1	
UAA	≥ 2	
DEA	≥ 3	More DEAs add application capacity.

Doppler Server (formerly Loggregator Server)	≥ 2	Deploying additional Doppler servers splits traffic across them.
Loggregator Traffic Controller	≥ 2	Deploying additional Loggregator Traffic Controllers allows you to direct traffic to them in a round-robin manner.

Single-node Processes

You can think of components that do not support multiple instances as single-node processes. Since you cannot increase the number of instances of these components, you should choose a different strategy for dealing with events that degrade availability.

First, consider the components whose availability affects the platform as a whole.

HAProxy:

Cloud Foundry deploys with a single instance of HAProxy for use in lab and test environments. Production environments should use your own highly-available load balancing solution.

NATS:

You might run NATS as a single-node process if you lack the resources to deploy two stable NATS servers.

Cloud Foundry continues to run any apps that are already running even when NATS is unavailable for short periods of time. The components publishing messages to and consuming messages from NATS are resilient to NATS failures. As soon as NATS recovers, operations such as health management and router updates resume and the whole Cloud Foundry system recovers.

Because NATS is deployed by BOSH, the BOSH resurrector will recover the VM if it becomes non-responsive.

NFS Server:

For some deployments, an appropriate strategy would be to use your infrastructure's high availability features to immediately recover the VM where the NFS Server runs. In others, it would be preferable to run a scalable and redundant blobstore service. Contact Pivotal PSO if you need help.

SAML Login Server:

Because the Login Server is deployed by BOSH, the BOSH resurrector will recover the VM if it becomes non-responsive.

Secondly, there are components whose availability does not affect that of the platform as a whole. For these, recovery by normal IT procedures should be sufficient even in a high availability Cloud Foundry deployment.

Syslog:

An event that degrades availability of the Syslog VM causes a gap in logging, but otherwise Cloud Foundry continues to operate normally.

Because Syslog is deployed by BOSH, the BOSH resurrector will recover the VM if it becomes non-responsive.

Collector:

This component is not in the critical path for any operation.

Compilation:

This component is active only during platform installation and upgrades.

Etcdb:

Etcdb is a highly-available key value store used for shared configuration and service discovery. You can find more information about running etcd on single nodes in the [etcd GitHub repository](#).

Databases

For database services deployed outside Cloud Foundry, plan to leverage your infrastructure's high availability features and to configure backup and restore where possible.

Contact Pivotal PSO if you require replicated databases or any assistance.

Orgs, Spaces, Roles, and Permissions

Cloud Foundry uses role-based access control (RBAC), with each role granting permissions in either an org or a space.

Orgs

An org is a development account that an individual or multiple collaborators can own and use. All collaborators access an org with user accounts. Collaborators in an org share a resource quota plan, applications, services availability, and custom domains.

User Accounts

A user account represents an individual person within the context of a Cloud Foundry installation. A user can have different roles in different spaces within an org, governing what level and type of access they have within that space.

Spaces

Every application and service is scoped to a space. Each org contains at least one space. A space provides users with access to a shared location for application development, deployment, and maintenance. Each space role applies only to a particular space.

Roles and Permissions

A user can have one or more roles. The combination of these roles defines the user's overall permissions in the org and within specific spaces in that org.

Org Roles and Permissions

Org Manager

Assign this role to managers or other users who need to administer the account.

An Org Manager can do the following:

- Add and manage users
- View users and edit org roles
- View the org quota
- Create, view, edit, and delete spaces
- Invite and manage users in spaces
- View the status, number of instances, service bindings, and resource use of each application in every space in the org
- Add domains

 **Note:** An Org Manager needs explicit administrator permissions to perform certain actions. Refer to the [Creating and Managing Users with the UAA CLI \(UAAC\)](#) topic to learn how to create a user with admin rights.

Org Auditor

Assign this role to people who need to view but not edit user information and org quota usage information.

An Org Auditor can do the following:

- View users and org roles
- View the org quota

Billing Manager

Assign this role to people who need to create and manage billing account and payment information.

A Billing Manager can do the following:

- Set the org spending limit
- Create and set payment information
- Read invoices and payment history
- Create and edit the invoice notification email addresses

 **Note:** The Billing Manager role is only relevant for Cloud Foundry environments deployed with a billing engine.

Space Roles and Permissions

Space Manager

Assign this role to managers or other users who need to administer a space.

A Space Manager can do the following:

- Add and manage users in the space
- View the status, number of instances, service bindings, and resource use of each application in the space

Space Developer

Assign this role to application developers or other users who need to manage applications and services in a space.

A Space Developer can do the following:

- Deploy an application
- Start or stop an application
- Rename an application
- Delete an application
- Create, view, edit, and delete services in a space
- Bind or unbind a service to an application
- Rename a space
- View the status, number of instances, service bindings, and resource use of each application in the space
- Change the number of instances, memory allocation, and disk limit of each application in the space
- Associate an internal or external URL with an application

Space Auditor

Assign this role to people who need to view but not edit the space.

A Space Auditor can do the following:

- View the status, number of instances, service bindings, and resource use of each application in the space

Understanding Cloud Foundry Security

Pivotal protects customers from security threats by applying security controls and by isolating customer applications and data.

Cloud Foundry:

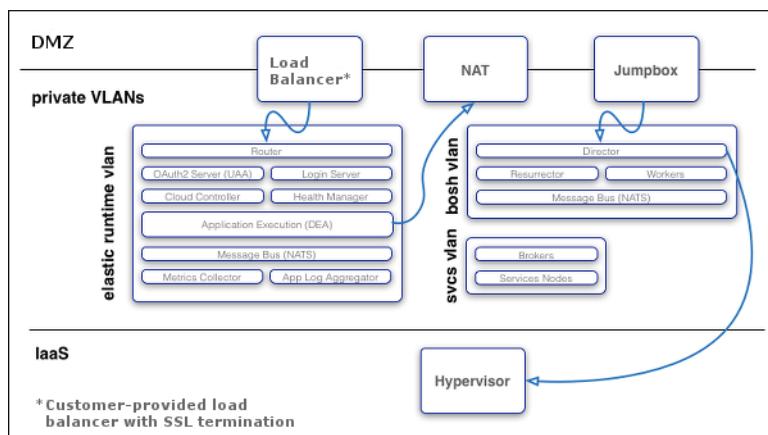
- Implements role-based access controls, applying and enforcing roles and permissions to ensure that users can only view and affect the spaces for which they have been granted access.
- Ensures security of application bits in a multi-tenant environment.
- Prevents possible denial of service attacks through resource starvation.

Before you read this document, you might want to review the [general system architecture](#).

System Boundaries and Access

As the image below shows, in a typical deployment of Cloud Foundry, the components run on virtual machines (VMs) that exist within a VLAN. In this configuration, the only access points visible on a public network are a load balancer that maps to one or more Cloud Foundry routers and, optionally, a NAT VM and a jumpbox. Because of the limited number of contact points with the public internet, the surface area for possible security vulnerabilities is minimized.

Note: Pivotal recommends that you also install a NAT VM for outbound requests and a jumpbox to access the BOSH Director, though these access points are optional depending on your network configuration.



Protocols

All traffic from the public internet to the Cloud Controller and UAA happens over HTTPS. Inside the boundary of the system, components communicate over a publish-subscribe (pub-sub) message bus, [NATS](#), and also HTTP.

Application Traffic

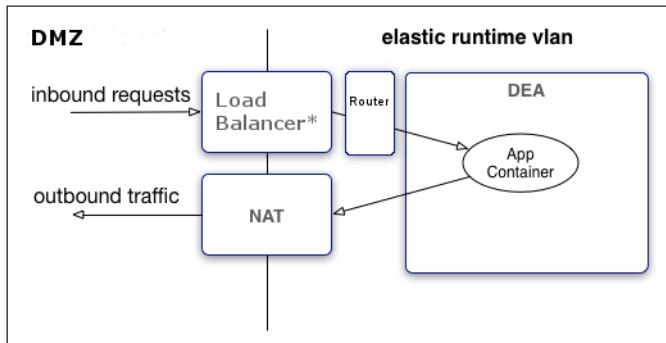
When an app instance starts, the DEA allocates an IP address and also assigns an arbitrary port to the Application Container. The application uses the PORT environment variable provided in the container environment to determine which port to listen on. Because the DEA assigns a random value to the PORT environment variable, the value is generally unique for each application instance.

The router handles all inbound traffic to applications, routing traffic to one of the application instances.

A DEA has a single IP address. If you configure the deployment with the cluster on a VLAN, as recommended, then all traffic goes through the following levels of network address translation:

- Inbound: From the load balancer through the router to the DEA, then from the DEA to the App Container.
- Outbound: From the App Container to the DEA, then to the gateway on the DEA virtual network interface. This gateway might be a NAT to external networks depending on your IaaS.

The image below shows traffic flow through the recommended deployment configuration.



Network Traffic Rules

Cloud Foundry implements network traffic rules using Linux iptables on the component VMs. Operators can configure rules to prevent system access from external networks and between internal components, and to restrict applications from establishing connections over the DEA network interface.

Operators can use either of the following to configure these rules:

- Cloud Foundry [Application Security Groups](#): An Application Security Group (ASG) consists of a list of access rules to control application outbound traffic. Operators can define Cloud Foundry ASGs both in a manifest and on the cf Command Line Interface (CLI), however, Cloud Foundry ASGs defined in a manifest are valid only until operators update them on the CLI. For this reason, Pivotal recommends that operators use the CLI to create and bind Cloud Foundry ASGs to the running and staging deployment phases for greater flexibility.
- [DEA Network Properties](#): Operators can configure the `allow_networks` and `deny_networks` parameters for DEAs to prohibit communication between system components and applications. Any subsequent Cloud Foundry ASG configurations will overwrite these configurations.

Note: Pivotal recommends that you use Cloud Foundry ASGs to specify egress access rules for your applications. This functionality enables you to more securely restrict application outbound traffic to predefined routes.

Spoofing

If an IP, MAC, or ARP spoofing attack bypasses the physical firewall for your deployment, Cloud Foundry network traffic rules help prevent the attack from accessing application containers. Cloud Foundry uses application isolation, operating system restrictions, and encrypted connections to further mitigate risk.

BOSH

Operators deploy Cloud Foundry with BOSH. The BOSH Director is the core orchestrating component in BOSH: it controls VM creation and deployment, as well as other software and service lifecycle events. You use HTTPS to ensure secure communication to the BOSH Director.

Note: Pivotal recommends that you deploy the BOSH Director on a subnet that is not publicly accessible, and access the BOSH Director from a jumpbox on the subnet or through VPN.

BOSH includes the following functionality for security:

- Communicates with the VMs it launches over NATS. Because NATS cannot be accessed from outside Cloud Foundry, this ensures that published messages can only originate from a component within your deployment.
- Provides an audit trail through the `bosh tasks` command. This command shows all actions that an operator has taken with BOSH.
- Allows you to set up individual login accounts for each operator. BOSH operators have root access.

Note: BOSH does not encrypt data stored on BOSH VMs. Your IaaS might encrypt this data.

Authentication and Authorization

User Account and Authentication (UAA) is the central identity management service for the Elastic Runtime platform and its various components.

UAA acts as an [OAuth2 ↗](#) Authorization Server and issues access tokens for applications that request platform resources. The tokens are based on the [JSON Web Token ↗](#) and are digitally signed by UAA.

Operators can configure the identity store in UAA. If users register an account with the Cloud Foundry platform, UAA acts as the user store and stores user passwords in the UAA database using [bcrypt ↗](#). UAA also supports connecting to external user stores through LDAP and SAML. Once an operator has configured the external user store, such as a corporate Microsoft Active Directory, users can use their LDAP credentials to gain access to the Cloud Foundry platform instead of registering a separate account. Alternatively, operators can use SAML to connect to an external user store and enable single sign-on for users into the Cloud Foundry platform.

Managing User Access with Role-Based Access Control

Applications that users deploy to Cloud Foundry exist within a space. Spaces exist within orgs. To view and access an org or a space, a user must be a member of it. Cloud Foundry uses role-based access control (RBAC), with each role granted permissions to either an org or a specified space. For more information about roles and permissions, refer to the [Orgs, Spaces, Roles, and Permissions](#) topic.

For more information, see [Understanding Apps Manager Permissions ↗](#) and [Managing User Spaces in Accounts using the Apps Manager ↗](#).

Security for Service Broker Integration

The Cloud Controller authenticates every request with the Service Broker API using HTTP or HTTPS, depending on which protocol that you specify during broker registration. The Cloud Controller rejects any broker registration that does not contain a username and password.

Service instances bound to an app contain credential data. Users specify the binding credentials for [user-provided service instances](#), while third-party brokers specify the binding credentials for managed service instances. The VCAP_SERVICES environment variable contains credential information for any service bound to an app. Cloud Foundry constructs this value from encrypted data that it stores in the Cloud Controller Database (CCDB).

 **Note:** The selected third-party broker controls how securely to communicate managed service credentials.

A third-party broker might offer a dashboard client in its catalog. Dashboard clients require a text string defined as a `client_secret`. Cloud Foundry does not store this secret in the CCDB. Instead, Cloud Foundry passes the secret to the UAA component for verification using HTTP or HTTPS.

Software Vulnerability Management

Cloud Foundry manages software vulnerability using releases and BOSH stemcells. New Cloud Foundry releases are created with updates to address code issues, while new stemcells are created with patches for the latest security fixes to address any underlying operating system issues.

Ensuring Security for Application Artifacts

Cloud Foundry secures both the code and the configuration of an application using the following functionality:

- Application developers push their code using the [Cloud Foundry API ↗](#). Cloud Foundry secures each call to the CF API using the [UAA](#) and SSL.
- The Cloud Controller uses [RBAC](#) to ensure that only authorized users can access a particular application.
- The Cloud Controller stores the configuration for an application in an encrypted database table. This configuration data includes user-specified environment variables and service credentials for any services bound to the app.
- Cloud Foundry runs the app inside a secure container. For more information, see the [Application Isolation with Containers](#) section.
- Cloud Foundry operators can configure network traffic rules to control inbound communication to and outbound communication from an app. For more information, see the [Network Traffic Rules](#) section.

Application Isolation with Containers

Each application deployed to Cloud Foundry runs within its own self-contained environment, a Linux [Warden container](#). Cloud Foundry operators can configure whether contained applications can directly interact with other applications or other Cloud Foundry system components.

Applications are typically allowed to invoke other applications in Cloud Foundry by leaving the system and re-entering through the load balancer positioned in front of the Cloud Foundry routers. The Warden containers isolate processes, memory, and the file system. Each Warden container also provides a dedicated virtual network interface that establishes IP filtering rules for the app, which are derived from network traffic rules. This restrictive operating environment is designed for security and stability.

Isolation is achieved by namespacing kernel resources that would otherwise be shared. The intended level of isolation is set such that multiple containers present on the same host cannot detect each other. This means that each container is given its own Process ID (PID), namespace, network namespace, and mount namespace.

Resource control is managed using Linux Control Groups ([cgroups](#)). Each container is placed in its own cgroup, which requires the container to use a fair share of CPU compared to the relative CPU share of other containers. This placement also determines the maximum amount of memory the container may use.

 **Note:** BOSH does not support a RedHat Enterprise Linux OS stemcell. This is due to an inherent security issue with the way RedHat handles user namespacing and container isolation.

Networking

The [DEA](#) for each Warden container assigns the container a dedicated virtual network interface. This interface is one side of a virtual ethernet pair created on the host VM. The other side of the virtual ethernet pair is only visible on the host from the root namespace. The pair is configured to use IPs in a small and static subnet. Traffic to and from the container is forwarded using NAT. Operators can configure global and container-specific [network traffic rules](#) that become Linux iptable rules to filter and log outbound network traffic.

Filesystem

Every container includes a private root filesystem. For Linux containers, this filesystem is created by stacking a read-only base filesystem and a container-specific read-write filesystem, commonly known as an overlay filesystem. The read-only filesystem contains the minimal set of Linux packages and Warden-specific modifications common to all containers. Containers can share the same read-only base filesystem because all writes are applied to the read-write filesystem. The read-write filesystem is unique to each container and is created by formatting a large sparse file of a fixed size. This fixed size prevents the read-write filesystem from overflowing into unallocated space.

Security Event Logging and Auditing

For operators, Cloud Foundry provides an audit trail through the `bosh tasks` command. This command shows all actions that an operator has taken with the platform. Additionally, operators can redirect Cloud Foundry component logs to a standard syslog server using the `syslog_daemon_config` [property](#) in the `metron_agent` job of `cf-release`.

For users, Cloud Foundry records an audit trail of all relevant API invocations of an app. The CLI command `cf events` returns this information.

Running a Secure Deployment

To help run a secure deployment, Pivotal recommends the following:

- Configure UAA clients and users using a BOSH manifest. Limit and manage these clients and users as you would any other kind of privileged account.
- Deploy within a VLAN that limits network traffic to individual VMs. This reduce the possibility of unauthorized access to the VMs within your BOSH-managed cloud.
- Enable HTTPS for applications and SSL database connections to protect sensitive data transmitted to and from applications.
- Ensure that the jumpbox is secure, along with the load balancer and NAT VM.

- Encrypt stored files and data within databases to meet your data security requirements. Deploy using industry standard encryption and the best practices for your language or framework.
- Prohibit promiscuous network interfaces on the trusted network.
- Review and monitor data sharing and security practices with third-party services that you use to provide additional functionality to your application.
- Store SSH keys securely to prevent disclosure, and promptly replace lost or compromised keys.
- Use Cloud Foundry's RBAC model to restrict your users' access to only what is necessary to complete their tasks.
- Use a strong passphrase for both your Cloud Foundry user account and SSH keys.

Stacks

A stack is a prebuilt root filesystem (rootfs) which works in tandem with a buildpack and is used to support running applications.

 **Note:** You must ensure compatibility for buildpacks on the stacks they are running against.

Building Stacks

A Droplet Execution Agent (DEA) can support multiple stacks. The scripts for building the available Cloud Foundry stacks and for building other stacks reside in the GitHub [Stacks](#) project.

Available Stacks

cflinuxfs2 Stack

The cflinuxfs2 stack is derived from Ubuntu Trusty 14.04.

CLI Commands

Use the `cf stacks` command to list all the stacks available in a deployment.

```
$ cf stacks
Getting stacks in org MY-ORG / space development as developer@example.com...
OK

name      description
cflinuxfs2  Cloud Foundry Linux-based filesystem
```

Use the `cf push APPNAME -s STACKNAME` to change your stack and restage your application.

```
$ cf push testappmx1 -s cflinuxfs2
Using stack cflinuxfs2...
OK
Creating app testappmx1 in org MY-ORG / space development as developer@example.com...
OK

Creating route testappmx1.cfapps.io...
OK

Binding testappmx1.cfapps.io to testappmx1...
OK

Uploading testappmx1...
Uploading app files from: /Users/pivotal/workspace/app_files
Uploading 13.4M, 544 files
Done uploading
OK

Starting app testappmx1 in org MY-ORG / space development as developer@example.com...
----> Downloaded app package (21M)
-----> Buildpack version 1.3.0
-----> Compiling Ruby/Rack
-----> Using Ruby version: ruby-2.0.0
-----> Installing dependencies using 1.7.12
      Running: bundle install --without development:test --path vendor/bundle --binstubs vendor/bundle/bin -j4 --deployment
      Fetching gem metadata from http://rubygems.org/.....
      Installing rack-rewrite 1.5.0
      Installing rack 1.5.2
      Using bundler 1.7.12
      Installing ref 1.0.5
      Installing libv8 3.16.14.3
      Installing therubyracer 0.12.1
      Your bundle is complete!
      Gems in the groups development and test were not installed.
      It was installed into ./vendor/bundle
      Bundle completed (16.63s)
      Cleaning up the bundler cache.
##### WARNING:
      No Procfile detected, using the default web server (webrick)

----> Uploading droplet (45M)

1 of 1 instances running

App started

OK

App testappmx1 was started using this command `bundle exec rackup config.ru -p $PORT`

Showing health and status for app testappmx1 in org MY-ORG / space development as developer@example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: testappmx1.cfapps.io
last uploaded: Wed Apr 8 23:40:57 UTC 2015

      state      since          cpu    memory         disk
#0  running   2015-04-08 04:41:54 PM  0.0%  57.3M of 1G  128.8M of 1G
```

For API information, review the Stacks section of the [Cloud Foundry API Documentation](#).

Glossary

Term	Definition
API	Application Programming Interface
BOSH	BOSH is an open framework for managing the full development and deployment life cycle of large-scale distributed software applications.
CLI	Command Line Interface
DEA	Droplet Execution Agent. The DEA is the component in Elastic Runtime responsible for staging and hosting applications.
Domains	A domain is a domain name like <code>example.com</code> . Domains can also be multi-level and contain sub-domains like the “myapp” in <code>myapp.example.com</code> . Domain objects belong to an org and are not directly bound to apps.
Droplet	An archive within Elastic Runtime that contains the application ready to run on a DEA. A droplet is the result of the application staging process.
Management	You can manage spaces and orgs with the cf command line interface, the Cloud Controller API, and the Cloud Foundry Eclipse Plugin.
Ops Manager or Operations Manager	Operations Manager is a web application that you use to deploy and manage a Pivotal Cloud Foundry PaaS.
Org	An org is the top-most meta object within the Elastic Runtime infrastructure. Only an account with administrative privileges on a Elastic Runtime instance can manage its orgs.
Routes	A route, based on a domain with an optional host as a prefix, may be associated with one or more applications. For example, <code>myapp</code> is the host and <code>example.com</code> is the domain when using the route <code>myapp.example.com</code> . It is possible to have a route that represents <code>example.com</code> without a host. Routes are children of domains and are directly bound to apps.
Service	A “factory” which produces service instances.
Service Instance	A reserved resource provisioned by a service. The resource provisioned will differ by service; could be a database or an account on a multi-tenant application.
Spaces	An org can contain multiple spaces. You can map a domain to multiple spaces, but you can map a route to only one space.
Staging	The process in Elastic Runtime by which the raw bits of an application are transformed into a droplet that is ready to execute.
Warden	The mechanism for containerization on DEAs that ensures applications running on Elastic Runtime have a fair share of computing resources and cannot access either the system code or other applications running on the DEA.

Operator's Guide

This guide covers networking and user management for [Pivotal Cloud Foundry](#) operators.

Table of Contents

- [Isolating a PCF Deployment with a Private Network](#)
- [Understanding the Elastic Runtime Network Architecture](#)
- [Configuring PCF SSL Termination for vSphere Deployments](#)
- [Changing the Quota Plan of an Organization with cf CLI](#)
- [Identifying Elastic Runtime Jobs Using vCenter](#)
- [Understanding the Effects of Single Components on a PCF Upgrade](#)
- [Configuring Single Sign-On](#)
- [Configuring LDAP](#)
- [Switching Application Domains](#)
- [Monitoring Instance Usage with the Accounting Report](#)
- [Using Diego in Pivotal Cloud Foundry](#)
- [Deploying Diego for Windows](#)
- [Key Metrics for Monitoring a Pivotal Cloud Foundry Deployment](#)

Isolating a PCF Deployment with a Private Network

You may want to place your [Pivotal Cloud Foundry](#) (PCF) deployment on a private network, then route requests to PCF from a router or a load balancer on a public network. Isolating PCF in this way increases security and allows you to use as many IP addresses as you need by subnetting the private network.

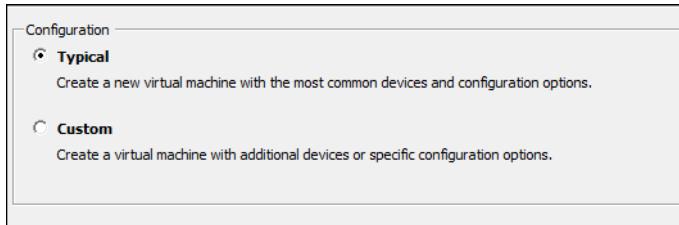
One way to accomplish this is to configure Network Address Translation (NAT) on a VM. The following example explains how to set up a virtual CentOS NAT box in vSphere, and use that to isolate a PCF deployment.

Step 1: Deploy a CentOS image to vSphere

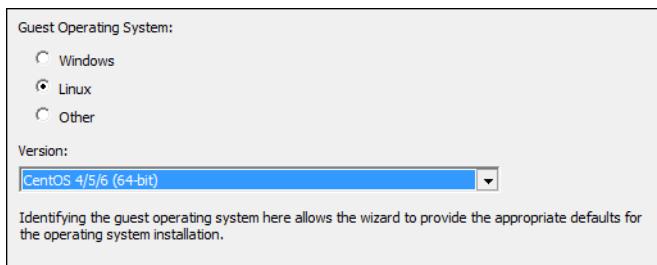
Connect to vCenter using the vSphere client. Assign two network adapters to the VM. This results in the VM having two IP addresses:

- A public-facing IP address: This IP address connects to the public network VLAN (WAN) through `GATEWAY_EXTERNAL_IP`.
- An internal VLAN IP address to communicate with other BOSH/Cloud Foundry components: This IP address connects to the private network (LAN) for the Ops Manager deployment using `GATEWAY_INTERNAL_IP`.

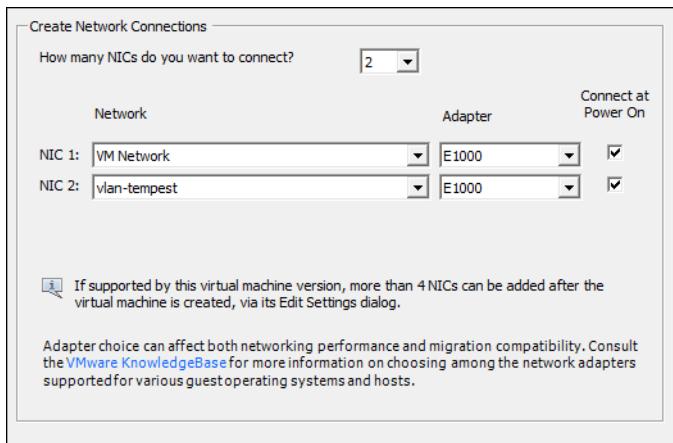
1. Download the latest CentOS image from <http://wiki.centos.org/Download>.
2. In vSphere, select **File > New > Virtual Machine**.
3. On the **Configuration** page, select the **Typical** configuration and click **Next**.



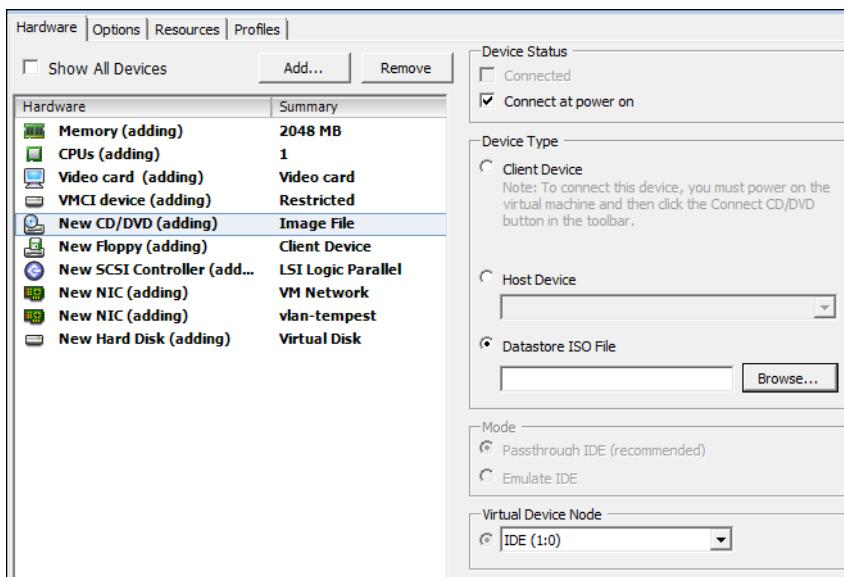
4. On the **Name and Location** page, name the virtual machine and click **Next**.
5. On the **Resource Pool** page, select a resource pool and click **Next**.
6. On the **Storage** page, select a destination storage for the VM and click **Next**.
7. On the **Guest Operating System** page, select **Linux** as the Guest Operating System and **CentOS 4/5/6 (64-bit)** as the version, then click **Next**.



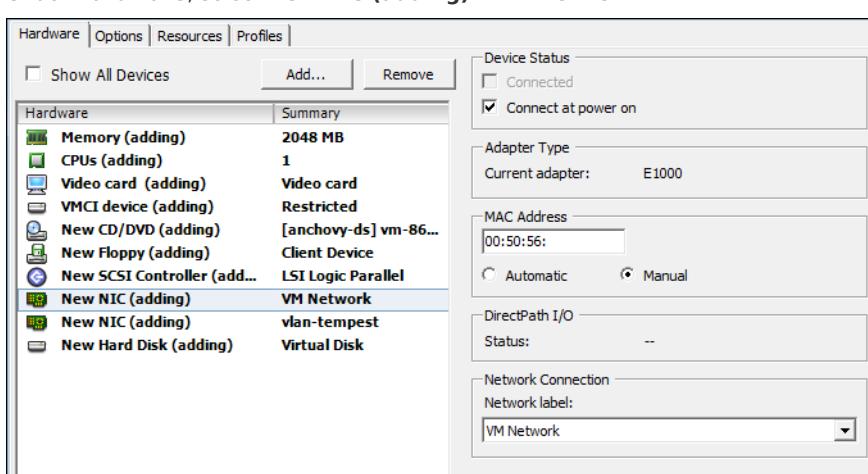
8. On the **Network** page, select **2** from the drop down menu to specify how many NICs you want to connect. Under **Network**, select **VM Network** for NIC 1 and the name of your private network for NIC 2. Then, click **Next**.



9. On the **Create a Disk** page, click **Next**.
10. On the **Ready to Complete** page, check the **Edit the virtual machine settings before completion** checkbox, then click **Continue**.
11. Under **Hardware**, select **New CD/DVD**, then select **Datastore ISO File** and click **Browse**. Browse to **ISO > CentOS-6.5-x86_64-minimal.iso** in your datastore.



12. Under **Device Status**, check the **Connect at power on** checkbox.
13. Under **Hardware**, select **New NIC (adding) - VM Network**.



14. Under **MAC Address**, select **Manual** and specify the MAC address, then click **Finish**.

Step 2: Set up iptables

1. Log in to the CentOS VM as root.
2. Run the following script to set up iptables, replacing the example IPs as appropriate.

```
# the following is an example proxy configuration
sudo vi /etc/sysctl.conf
set net.ipv4.ip_forward=1

# reset
iptables --flush

# example IPs--replace as appropriate
export INTERNAL_NETWORK_RANGE=10.0.0.0/8
export GATEWAY_INTERNAL_IP=10.0.0.1
export GATEWAY_EXTERNAL_IP=88.198.185.190
export PIVOTALCF_IP=10.0.0.10
export HA_PROXY_IP=10.0.0.13

# iptables forwarding rules including loopback
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -o eth1 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \
-p tcp --dport 80 -j SNAT --to $GATEWAY_INTERNAL_IP

iptables -t nat -A POSTROUTING -d $HA_PROXY_IP -s $INTERNAL_NETWORK_RANGE \
-p tcp --dport 443 -j SNAT --to $GATEWAY_INTERNAL_IP

iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport 443 -j DNAT \
--to $HA_PROXY_IP
iptables -t nat -A PREROUTING -d $GATEWAY_EXTERNAL_IP -p tcp --dport 80 -j DNAT \
--to $HA_PROXY_IP

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8443 -j DNAT \
--to $PIVOTALCF_IP:443
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT \
--to $HA_PROXY_IP:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j DNAT \
--to $PIVOTALCF_IP:22
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8080 -j DNAT \
--to $PIVOTALCF_IP:80

# DNS iptables rules
iptables -A INPUT -p udp --sport 53 -j ACCEPT
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
iptables -A INPUT -p udp -j DROP
iptables -A OUTPUT -p udp -j DROP

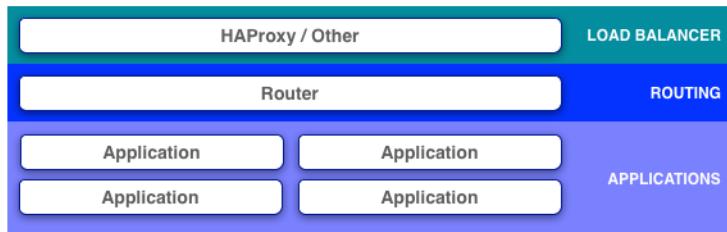
service iptables save
shutdown -r now
```

Step 3: Deploy the Ops Manager VM Template

1. Download PCF from the Pivotal Network at <https://network.pivotal.io/products/pivotal-cf>.
2. Unzip the downloaded file.
3. Deploy the Ops Manager VM template in the private network using the internal IP address. For more information, see the [Getting Started Guide](#). Access the deployed VM at port 8443 using https://GATEWAY_EXTERNAL_IP:8443.

Understanding the Elastic Runtime Network Architecture

The diagram below shows the key [Pivotal Cloud Foundry](#) Elastic Runtime network components.



Load Balancer

Elastic Runtime includes an HAProxy load balancer for terminating SSL. If you do not want to serve SSL certificates for Elastic Runtime on your own load balancer use the HAProxy. If you do choose to manage SSL yourself, omit the HAProxy by setting the number of instances to zero in Ops Manager.

Refer to the [Configuring PCF SSL Termination](#) topic for more information.

Router

The routers in Elastic Runtime are responsible for routing HTTP requests from web clients to application instances in a load balanced fashion. The routers are dynamically configured based on users mapping of applications to location URLs called routes, and updated by the runtime service as application instances are dynamically distributed.

For high availability, the routers are designed to be horizontally scalable. Configure your load balancer to distribute incoming traffic across all router instances.

Refer to the Cloud Foundry [Architecture](#) topic for more information about Cloud Foundry components.

Changing the Quota Plan of an Organization with cf CLI

This page assumes you are using cf CLI v6.

Quota plans are named sets of memory, service, and instance usage quotas. For example, one quota plan might allow up to 10 services, 10 routes, and 2 GB of RAM, while another might offer 100 services, 100 routes, and 10 GB of RAM. Quota plans have user-friendly names, but are identified by unique GUIDs.

Quota plans are not directly associated with user accounts. Instead, every organization has a list of available quota plans, and the account admin assigns a specific quota plan from the list to the organization. Everyone in the organization shares the quotas described by the plan.

Depending on the usage needs of an organization, you may need to change the quota plan. Follow the steps below to change the quota plan of your organization using `cf curl`.

Note: Only an **account** administrator can run `cf curl`.

Step 1: Find the GUID of your organization

To find the GUID of your organization, replace MY-ORGANIZATION with the name of your organization and run this command:

```
CF_TRACE=true cf org MY-ORGANIZATION
```

This command returns a filtered JSON response listing information about your organization.

Data about your organization shows in two sections: “metadata” and “entity.” The “metadata” section shows the organization GUID, while the “entity” section lists the name of the organization.

 **Note:** Because “metadata” is listed before “entity” in the response, the GUID of the organization is shown six lines *above* the name.

Example:

```
$ CF_TRACE=true cf org example-org-name | grep -B7 example-org-name

REQUEST:
GET /v2/organizations?q=name%3Aexample-org-name&inline-relations-depth=1 HTTP/1.1
--
"metadata": {
  "guid": "aaaa1234-1111",
  "url": "/v2/organizations/aaaa1234-1111",
  "created_at": "2014-02-06T02:09:09+00:00",
  "updated_at": null
},
"entity": {
  "name": "example-org-name",
```

The GUID of “example-org-name” is *aaaa1234-1111*.

Step 2: Find the GUID of your current quota plan

To find the current quota plan for your organization, replace MY-ORGANIZATION-GUID with the GUID found in Step 1 and run:

```
cf curl /v2/organizations/MY-ORGANIZATION-GUID -X 'GET' | grep quota_definition_guid
```

This command returns a filtered JSON response showing the quota_definition_guid of your organization.

Example:

```
$ cf curl /v2/organizations/aaaaa1234-1111 -X 'GET' | grep quota_definition_guid
"quota_definition_guid": "bbbb6666-2222",
```

The GUID of the current quota plan is *bbbb6666-2222*.

Step 3: View available quota plans and select the quota plan GUID you want

To view all quota plans available to your organization, run:

```
cf curl /v2/quota_definitions -X 'GET'
```

This command returns a JSON response listing every quota plan available to your organization. Data about each plan shows in two sections: “metadata” and “entity.” The “metadata” section shows the quota plan GUID, while the “entity” section lists the name of the plan and the actual usage quotas.

Note: Because “metadata” is listed before “entity” for each quota plan, the GUID of a plan is shown six lines above the name.

Example:

```
$ cf curl /v2/quota_definitions -X 'GET'

{
  "total_results": 2,
  "total_pages": 1,
  "prev_url": null,
  "next_url": null,
  "resources": [
    {
      "metadata": {
        "guid": "bbbb6666-2222",
        "url": "/v2/quota_definitions/bbbb6666-2222",
        "created_at": "2013-11-19T18:53:48+00:00",
        "updated_at": "2013-11-19T19:34:57+00:00"
      },
      "entity": {
        "name": "small",
        "non_basic_services_allowed": false,
        "total_services": 10,
        "total_routes": 10,
        "memory_limit": 2048,
        "trial_db_allowed": false
      }
    },
    {
      "metadata": {
        "guid": "cccc0000-3333",
        "url": "/v2/quota_definitions/cccc0000-3333",
        "created_at": "2013-11-19T18:53:48+00:00",
        "updated_at": "2013-11-19T19:34:57+00:00"
      },
      "entity": {
        "name": "bigger",
        "non_basic_services_allowed": true,
        "total_services": 100,
        "total_routes": 100,
        "memory_limit": 10240,
        "trial_db_allowed": true
      }
    }
  ]
}
```

In this example, the GUID of the “small” plan is *bbbb6666-2222* and the GUID of the “bigger” plan is *cccc0000-3333*.

Select the quota plans you want to assign to your organization.

In our example, we want to change from the current plan to the “bigger” plan, GUID *cccc0000-3333*.

Step 4: Change the Quota Plan

Changing the quota plan GUID changes the quota plan used by the organization.

To change the quota plan for your organization, run:

```
cf curl /v2/organizations/MY-ORGANIZATION-GUID -X 'PUT' -d "{\"quota_definition_guid\":\"NEW-QUOTA-PLAN-GUID\"}"
```

Example:

```
$ cf curl /v2/organizations/aaaa1234-1111 -X 'PUT' -d "{\"quota_definition_guid\":\"cccc0000-3333\"}"
```

This command changes the quota plan assigned to the organization to the “bigger” plan, GUID *cccc0000-3333*.

Step 5: Verify the change

To verify the change to your quota plan, run the command from Step 2:

```
cf curl /v2/organizations/MY-ORGANIZATION-GUID -X 'GET' | grep quota_definition_guid .
```

This command should return a filtered JSON response showing the new quota_definition_guid" of your organization.

Example:

```
$ cf curl /v2/organizations/aaaa1234-1111 -X 'GET' | grep quota_definition_guid  
"quota_definition_guid": "cccc0000-3333",
```

Understanding the Effects of Single Components on a PCF Upgrade

The **Resource Config** page of Pivotal Elastic Runtime tile in the PCF Ops Manager shows the 24 components that the Ops Manager Director installs. You can specify the number of instances for 11 of the components. We deliver the remaining 13 resources as single components, meaning that they have a preconfigured and unchangeable value of one instance.

In a single-component environment, upgrading can cause the deployment to experience downtime and other limitations because there is no instance redundancy. Although this behavior might be acceptable for a test environment, you should configure the scalable components with editable instance values, such as HAProxy, Router, and DEA, for optimal performance in a production environment.

Note: A full Ops Manager upgrade may take close to two hours, and you will have limited ability to deploy an application during this time.

Summary of Component Limitations

The table lists components in the order that Ops Manager upgrades each component, and includes the following columns:

- **Scalable?**: Indicates whether the component has an editable value or a preconfigured and unchangeable value of one instance.

Note: For components marked with a checkmark in this column, we recommend that you change the preconfigured instance value of 1 to a value that best supports your production environment. For more information about scaling a deployment, refer to the [Scaling Cloud Foundry topic](#).

- **Extended Downtime**: Indicates that the component is unavailable for up to five minutes during an Ops Manager upgrade.
- **Other Limitations and Information**: Provides the following information:
 - Component availability, behavior, and usage during an upgrade
 - Guidance on disabling the component before an upgrade

Note: The table does not include the Run Smoke Tests and Run CF Acceptance Tests errands and the Compilation job. Ops Manager runs the errands after it upgrades the components, and creates compilation VMs as needed during the upgrade process.

Component	Scalable?	Extended Downtime	Other Limitations and Information
HAProxy	✓	✓	
NATS	✓	✓	
etcd		✓	
Health Manager	✓		Ops Manager operators will see a 5 minute delay in app cleanup during an upgrade.
NFS Server		✓	You cannot push, stage, or restart an app when an upgrade affects the NFS Server.
Cloud Controller Database		✓	
Cloud Controller	✓	✓	Your ability to manage an app when an upgrade affects the Cloud Controller depends on the number of instances that you specify for the Cloud Controller and DEA components. If either of these components are single components, you cannot push, stage, or restart an app during the upgrade.
Clock Global			

Cloud Controller Worker	✓	✓	
Router	✓	✓	
Pivotal Ops Metrics Collector	✓		<p>The Pivotal Ops Metrics tool is a JMX extension for Elastic Runtime that you can install. If you install this tool, Ops Manager operators will see a 5 minute delay in metrics collection during an upgrade.</p> <p>You can disable this component before an upgrade to reduce the overall system downtime.</p>
UAA Database			
UAA	✓		If a user has an active authorization token prior to performing an upgrade, the user can still log in using either a UI or the CLI.
Login	✓	✓	
Apps Manager Database			You can disable this component before an upgrade to reduce the overall system downtime.
MySQL Server			
DEA	✓	✓	<p>Your ability to manage an app when an upgrade affects the DEA depends on the number of instances that you specify for the Cloud Controller and DEA components. If either of these components are single components, you cannot push, stage, or restart an app during the upgrade.</p> <p>The Apps Manager app and App Usage Service components for the Apps Manager Database run in a single DEA instance. You cannot use the Apps Manager or the CLI during the upgrade of the DEA.</p>
Doppler Server			Ops Manager operators will see 2-5 minute gaps in logging.
Loggregator Traffic Controller			Ops Manager operators will see 2-5 minute gaps in logging.
Push Apps Manager and Push App Usage Service errands			<p>These errands run scripts that connect the Apps Manager app and the App Usage Service components to the Apps Manager Database.</p> <p>The Apps Manager app and App Usage Service components runs in a single DEA instance and the Apps Manager Database is a single component. If there is an upgrade issue with either Apps Manager Database instance or the DEA instance, the upgrade fails and Ops Manager will not run this errand.</p>

Configuring Single Sign-On

If your user store is exposed as a SAML Identity Provider for single sign-on (SSO), you can set up SSO to allow users to access the [PCF Apps Manager](#) without creating a new account or re-entering credentials.

Configure Pivotal Cloud Foundry as a Service Provider

You must configure [Pivotal Cloud Foundry](#) (PCF) so that your Identity Provider can recognize PCF as a Service Provider. Follow the instructions below:

1. Log in to Ops Manager.
2. Click the **Elastic Runtime** tile. Select **Cloud Controller** on the **Settings** tab and verify your system domain.

Assign Networks Coordinates Pivotal CF Elastic Runtime application lifecycles

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

MySQL Proxy Config

Cloud Controller

External Endpoints

SSO Config

LDAP Config

SMTP Config

Errands

Resource Config

Stemcell

System Domain * system.mydomain.com

Apps Domain * apps.mydomain.com

Cloud Controller DB Encryption Key Secret

Maximum File Upload Size (MB) (min: 1024, max: 2048) * 1024

Disable Custom Buildpacks

Default Quota App Memory (MB) (min: 10240, max: 102400) * 10240

Default Quota Service Instances (min: 0, max: 1000) * 100

Save

3. Select **SSO Config**.

Assign Networks Configure Identity Provider

Assign Availability Zones

System Database Config

File Storage Config

IPs and Ports

MySQL Proxy Config

Cloud Controller

External Endpoints

SSO Config

LDAP Config

Provider Name

Provider Metadata

(OR) Provider Metadata URL

Save

4. Enter a **Provider Name**. Your provider name appears as a link on your Pivotal login page, which you can access at <https://login.YOUR-SYSTEM-DOMAIN>. The image below shows an example login page with the provider name "Your

Provider Name." If you click on this link, you are redirected to your Identity Provider for authentication.

The form shows a 'Welcome!' header, two input fields for 'Email' and 'Password', a large blue 'SIGN IN' button, and a smaller text link 'or Sign in with:' followed by 'Your Provider Name'. Below the provider name are links for 'Create account' and 'Reset password'.

5. Populate either **Provider Metadata** or **(OR) Provider Metadata URL**, depending on whether your Identity Provider exposes a Metadata URL. You can do this by either of the following:
 - Download your Identity Provider metadata and paste this XML into the **Provider Metadata** area. Ensure that the XML declaration tag is present at the beginning of the Metadata XML. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor>
</EntityDescriptor>
```

- If your Identity Provider exposes a Metadata URL, provide the Metadata URL.

Note: You only need to select one of the above configurations. If you configure both, your Identity Provider defaults to the **(OR) Provider Metadata URL**.

6. Click **Save**.

7. Click **Apply Changes** on the Ops Manager home page for the configuration to take effect.

Configure Your Identity Provider

You must configure your Identity Provider to recognize PCF as a Service Provider.

Download the Service Provider Metadata from <https://login.YOUR-SYSTEM-DOMAIN/saml/metadata>. Consult the documentation for your Identity Provider for configuration instructions.

Refer to the table below for information about certain industry-standard Identity Providers and how to integrate them with Pivotal Cloud Foundry:

Solution Name	Integration Guide
CA Single Sign-On aka CA SiteMinder	PDF
Ping Federate	PDF

Note: Some Identity Providers allow uploads of Service Provider Metadata. Other providers require you to manually enter the Service Provider Metadata into a form.

Configuring LDAP

This topic describes connecting [Pivotal Cloud Foundry](#) Elastic Runtime to LDAP by integrating the Cloud Foundry User Account and Authentication server with LDAP.

The Cloud Foundry User Account and Authentication ([UAA](#)) server provides identity management for Elastic Runtime in three ways:

- Issues tokens for use by client applications when they act on behalf of Elastic Runtime users
- Authenticates users with their Elastic Runtime credentials
- Acts as an SSO service using Elastic Runtime or other credentials

You can integrate the UAA with a Lightweight Directory Access Protocol (LDAP) server. Connecting Elastic Runtime to LDAP allows the UAA to authenticate users using LDAP search and bind operations.

See [Onboarding Existing LDAP Users to a PCF Deployment](#) for information about managing user identity with LDAP in PCF.

Configuring the LDAP Endpoint

To integrate the UAA with LDAP, configure Elastic Runtime with your LDAP endpoint information as follows:

1. Log into the Pivotal Cloud Foundry Operations Manager web interface.
2. On the Product Dashboard, select **Pivotal Elastic Runtime**.



3. In the left navigation menu, select **LDAP Config**.
4. Enter the following LDAP endpoint configuration information:
 - **LDAP Server URL:** A URL pointing to the LDAP server. This must include one of the following protocols:
 - `ldap://`: This specifies that the LDAP server uses an unencrypted connection.
 - `ldaps://`: This specifies that the LDAP server uses SSL for an encrypted connection and requires that the LDAP server holds a trusted certificate or that you import a trusted certificate to the JVM truststore.
 - **LDAP Credentials:** The LDAP Distinguished Name (DN) and password for binding to the LDAP Server. Example DN: `cn=administrator,ou=Users,dc=example,dc=com`

Note: We recommend that you provide LDAP credentials that grant read-only permissions on the LDAP Search Base and the LDAP Group Search Base.

- **LDAP Search Base:** The location in the LDAP directory tree from which any LDAP User search begins. The typical LDAP Search Base matches your domain name.

For example, a domain named “cloud.example.com” typically uses the following LDAP User Search Base:
`ou=Users,dc=example,dc=com`

- **LDAP Search Filter:** A string that defines LDAP User search criteria. These search criteria allow LDAP to perform more effective and efficient searches. For example, the standard LDAP search filter `cn=Smith` returns all objects with a common name equal to `Smith`.

In the LDAP search filter string that you use to configure Elastic Runtime, use `{0}` instead of the username. For

example, use `cn={0}` to return all LDAP objects with the same common name as the username.

In addition to `cn`, other attributes commonly searched for and returned are `mail`, `uid` and, in the case of Active Directory, `sAMAccountName`.

- **LDAP Group Search Base:** The location in the LDAP directory tree from which the LDAP Group search begins.

For example, a domain named “cloud.example.com” typically uses the following LDAP Group Search Base:

`ou=Groups,dc=example,dc=com`

Follow the instructions in [Granting Admin Permissions to an LDAP Group](#) in *Creating and Managing Users with the UAA CLI (UAAC)* to map the groups under this search base to Admin roles in Pivotal Cloud Foundry.

You must use the Cloud Foundry command line interface (cf CLI) to map other roles in Pivotal Cloud Foundry to individual users. See [Creating and Managing Users with the cf CLI](#) for more information.

- **LDAP Group Search Filter:** A string that defines LDAP Group search criteria. The standard value is `member={0}`.

Configure an LDAP endpoint for the UAA

LDAP Server URL

LDAP Credentials

Username

Password

LDAP Search Base

LDAP Search Filter

LDAP Group Search Base

LDAP Group Search Filter

5. Click **Save**.
6. Click the **Installation Dashboard** link to return to the Installation Dashboard.
7. On the Installation Dashboard, click **Apply Changes**.

Pending Changes

Revert

UPDATE Pivotal Elastic Runtime

Apply changes

Switching Application Domains

This page assumes you are using cf CLI v6.

This topic describes how to change the domain of an existing [Pivotal Cloud Foundry](#) (PCF) installation, using an example domain change from `myapps.mydomain.com` to `newapps.mydomain.com`.

1. In PCF Ops Manager, select the **Pivotal Elastic Runtime** tile.
2. Select **Cloud Controller** from the menu to see the current **Apps Domain** for your Elastic Runtime deployment. In the following example it is `myapps.mydomain.com`.

3. In the terminal, run `cf login -a YOUR_API_ENDPOINT`. The cf CLI prompts you for your PCF username and password, as well as the org and space you want to access. See [Identifying the API Endpoint for your Elastic Runtime Instance](#) if you don't know your API endpoint.
4. Run `cf domains` to view the domains in the space. If you have more than one shared domain, ensure that the domain you want to change is at the top of the list before you apply the new domain to your Elastic Runtime tile configuration. You can delete and re-create the other shared domains as necessary to push the domain you want to change to the top of the list. If you do this, make sure to [re-map the routes for each domain](#).

```
$ cf domains
Getting domains in org my-org as admin...
name          status
myapps.mydomain.com  shared
```

5. Run `cf routes` to confirm that your apps are assigned to the domain you plan to change.

```
$ cf routes
Getting routes as admin ...
space      host      domain          apps
my-space   myapp    myapps.mydomain.com  myapp
```

6. Run `cf create-shared-domain YOUR_DESIRED_NEW_DOMAIN` to create the new domain you want to use:

```
$ cf create-shared-domain newapps.mydomain.com
Creating shared domain newapps.mydomain.com as admin...
OK
```

7. Run `cf map-route APP_NAME NEW_DOMAIN -n HOST_NAME` to map the new domain to your app. In this example both the `NEW_DOMAIN` and `HOST_NAME` arguments are `myapp`, since this is both the name of the app to which we are mapping a route, and the intended hostname for the URL.

```
$ cf map-route myapp newapps.mydomain.com -n myapp
Creating route myapp.newapps.mydomain.com for org my-org / space my-space as admin...
OK
Adding route myapp.newapps.mydomain.com to app myapp in org my-org / space my-space as admin...
OK
```

8. Repeat the previous step for each app in this space. Afterwards, check Apps Manager to confirm that the route URL has updated correctly for each app:

STATUS	APP	INSTANCES
	myapp myapp.newapps.mydomain...	1

9. Repeat the above steps for each space in your PCF installation except for the System org, beginning with logging into the org and space and ending with confirming the URL update.

Note: Ordinarily the System org contains only PCF apps that perform utility functions for your installation. Pivotal does not recommend pushing apps to this org. However, if you have pushed apps to System, you must also repeat the above steps for these apps.

10. Once you have confirmed that every app in every space has been mapped to the new domain, delete the old domain by running `cf delete-shared-domain OLD_DOMAIN_TO_DELETE`:

```
$ cf delete-shared-domain myapps.mydomain.com
Deleting domain myapps.mydomain.com as admin...
This domain is shared across all orgs.
Deleting it will remove all associated routes, and will make any app with this domain unreachable.
Are you sure you want to delete the domain myapps.mydomain.com?
> yes
OK
```

11. Configure your Elastic Runtime tile to use the new domain, and apply changes. Apps that you push after your update finishes use this new domain.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign Networks Coordinates Pivotal CF Elastic Runtime application

Assign Availability Zones

Root Filesystem

System Database Config

File Storage Config

IPs and Ports

Security Config

MySQL Proxy Config

Cloud Controller

System Domain *

Apps Domain *

Cloud Controller DB Encryption Key

Maximum File Upload Size (MB) (min: 1024, max: 2048) *

Disable Custom Buildpacks

Default Quota App Memory (MB) (min: 10240, max: 102400) *

Diego-SSH Overview

The Diego-SSH package enables SSH access to [Diego Cell](#) containers, allowing developers to access running app instances.

[Diego-SSH on GitHub](#) ↗

Related Topics:

- [Accessing Apps with Diego-SSH](#)
- [Diego-SSH Proxy](#)
- [Migrate Apps to Diego](#)

Introduction

Diego-SSH contains the following components, which are described in more detail below:

- An implementation of an ssh [proxy server](#).
- A lightweight ssh [daemon](#).
- A Cloud Foundry [cli plugin](#).

When these components are deployed and configured correctly, they provide a simple and scalable way to access Diego containers apps and other running long running processes.

Diego-SSH Proxy

The ssh proxy hosts the user-accessible ssh endpoint and is responsible for authentication, policy enforcement, and access controls in the context of Cloud Foundry. After a user has successfully authenticated with the proxy, the proxy will attempt to locate the target container and create an ssh session to a daemon running inside the container. After both sessions have been established, the proxy will manage the communication between the user's ssh client and the container's ssh daemon.

For more information about authentication and daemon discover, see [Diego-SSH Proxy](#)

Diego-SSH Daemon

The ssh daemon is a lightweight implementation that is built around the Go ssh library. It supports command execution, interactive shells, local port forwarding, and scp. The daemon is self-contained and has no dependencies on the container root file system.

The daemon is focused on delivering basic access to application instances in Cloud Foundry. It is intended to run as an unprivileged process and interactive shells and commands will run as the daemon user. The daemon only supports one authorized key is not intended to support multiple users.

The daemon can be made available on a file server and Diego LRPs that want to use it can include a download action to acquire the binary and a run action to start it. Cloud Foundry applications will download the daemon as part of the lifecycle bundle.

Diego-SSH CLI Plugin

The Cloud Foundry CLI plugin adds the following commands:

- `ssh` ssh to an application container instance
- `enable-ssh` enable ssh for the application
- `disable-ssh` disable ssh for the application
- `ssh-enabled` reports whether SSH is enabled on an application container instance

- `allow-space-ssh` allow SSH access for the space
- `disallow-space-ssh` disallow SSH access for the space
- `space-ssh-allowed` reports whether SSH is allowed in a space
- `get-ssh-code` obtain a one-time authorization code that can be used as an ssh password

You can use CF CLI v6.10.0 or higher to install the SSH plugin from the CF-Community repo:

```
$ cf add-plugin-repo CF-Community http://plugins.cloudfoundry.org/
$ cf install-plugin Diego-SSH -r CF-Community
```

Cloud Controller Configuration

The Cloud Controller provides several levels of granularity for enabling or disabling SSH access to CF app instances.

- Each CF app has an `enable_ssh` field that determines whether the Diego SSH server will be invoked alongside the main process in the app and whether users will be authorized to connect to it. Changing the value of `enable_ssh` triggers a new version of the app and hence restarts it on Diego with or without the SSH server running, as desired.
- Each CF space has an `allow_ssh` field that determines whether users are authorized to connect to the SSH server running inside their app instance.
- Cloud Controller has an `allow_app_ssh_access` configuration field that determines whether any user is authorized to connect to the SSH server running inside their app instance.

The Diego-SSH plugin allows users to inspect and alter the app and space fields through plugin commands:

Run `cf enable-ssh APP-NAME` to enable ssh access to the named app.

```
$ cf enable-ssh my-example-app
```

Run `cf disable-ssh APP-NAME` to disable ssh access to the named app.

```
$ cf disable-ssh my-example-app
```

Run `cf ssh-enabled APP-NAME` to see whether SSH access is enabled for that app.

Run `cf allow-space-ssh SPACE-NAME` to allow ssh access to the named space.

Run `cf disallow-space-ssh SPACE-NAME` disallow ssh access to the named space.

`cf space-ssh-allowed SPACE-NAME` reports whether SSH access is allowed for apps in that space. The `allow_app_ssh_access` property must be changed through Cloud Controller's config file. For a BOSH-deployed CF, an operator can specify this config property via the `cc.allow_app_ssh_access` property.

Diego-SSH Proxy

The ssh proxy hosts the user-accessible ssh endpoint and is responsible for authentication, policy enforcement, and access controls in the context of Cloud Foundry. After a user has successfully authenticated with the proxy, the proxy will attempt to locate the target container and create an ssh session to a daemon running inside the container. After both sessions have been established, the proxy will manage the communication between the user's ssh client and the container's ssh daemon.

Proxy Authentication

Clients authenticate with the proxy using a specially formed user name that describes the authentication domain and target container and a password that contains the appropriate credentials for the domain.

The proxy currently supports authentication against a `diego` domain and a `cf` domain. Each authentication domain can be enabled independently via command line arguments.

Diego via custom credentials

For Diego, the user is of the form `diego:PROCESS-GUID/INDEX` and the password must hold the configured credentials.

Client example:

```
$ ssh -p 2222 'diego:my-process-guid/1'@ssh.bosh-lite.com
$ scp -P 2222 -oUser='diego:ssh-process-guid/0' my-local-file.json ssh.bosh-lite.com:my-remote-file.json
```

The credentials checked by the proxy are configurable via the `--diegoCredentials` flag. The password provided by the client to the proxy must match what is present in the flag for successful authentication.

This support is enabled with the `--enableDiegoAuth` flag.

Cloud Foundry via Cloud Controller and UAA

For Cloud Foundry, the user is of the form `cf:APP-GUID/INSTANCE` and the password must be an authorization code that the ssh proxy server can exchange for an authorization token. The SSH proxy must be configured to use an OAuth client id that has been defined in the UAA. The client id used by the proxy must be advertised in the `/v2/info` endpoint under the `app_ssh_oauth_client` key. Please see the [UAA](#) documentation for details on how to allocate an authorization code.

The proxy will contact the Cloud Controller as the user to determine if the policy allows the user to access application containers via SSH.

Client example:

```
$ curl -k -v -H "Authorization: $(cf oauth-token | tail -1)" \
  https://uaa.bosh-lite.com/oauth/authorize \
  --data-urlencode "clientid=$(cf curl /v2/info | jq -r .appsshauthclient)" \
  --data-urlencode 'response_type=code' 2>&1 | \
  grep Location: | \
  cut -f2 -d'?' | \
  cut -f2 -d '=' | \
  pbcopy # paste authorization code when prompted for password
```

or, with the ssh plugin

```
$ cf get-ssh-code | pbcopy # paste authorization code when prompted for password
```

The authorization code can then be used as the password:

```
$ ssh -p 2222 cf:$(cf app app-name --guid)/0@ssh.bosh-lite.com
$ scp -P 2222 -oUser=cf:$(cf app app-name --guid)/0 my-local-file.json ssh.bosh-lite.com:my-remote-file.json
```

Cloud Foundry `cf` client example that uses the [ssh][ssh-plugin] plugin:

```
$ cf install-plugin https://github.com/cloudfoundry-incubator/diego-ssh/releases/download/${version}/ssh-plugin_${platform}-${arch}.zip  
$ cf ssh app-name
```

This support is enabled with the `--enableCFAuth` flag.

Daemon discovery

To be accessible via the SSH proxy, containers must host an ssh daemon, expose it via a mapped port, and advertise the port in a `diego-ssh` route. The proxy will fail end user authentication if the target LRP or a route is not found.

```
"routes": {  
    "diego-ssh": { "container_port": 2222 }  
}
```

The [CC-Bridge][bridge] components of Diego will generate the appropriate LRP definitions for Cloud Foundry applications which reflect the policies that are in effect.

Proxy to Container Authentication

When the proxy attempts to handshake with the SSH daemon inside the target container, it will use the information associated with the `diego-ssh` key in the LRP routes.

container_port (required)

`container_port` indicates which port inside the container the ssh daemon is listening on. The proxy will attempt to connect to host side mapping of this port after authenticating the client.

host_fingerprint (optional)

When present, `host_fingerprint` declares the expected fingerprint of the SSH daemon's host public key. When the fingerprint of the actual target's host key does not match the expected fingerprint, the connection is terminated. The fingerprint should only contain the hex string generated by `ssh-keygen -l`.

user (optional)

`user` declares the user ID to use during authentication with the container's SSH daemon. While it's not a required part of the routing data, it is required for password authentication and may be required for public key authentication.

password (optional)

`password` declares the password to use during password authentication with the container's ssh daemon.

private_key (optional)

`private_key` declares the private key to use when authenticating with the container's SSH daemon. If present, the key must be a PEM encoded RSA or DSA public key.

Example LRP

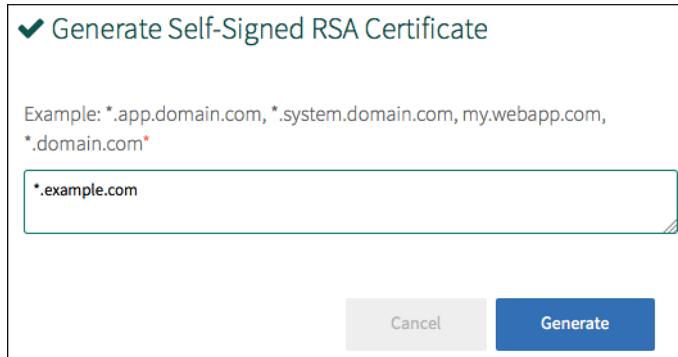
```
{  
  "process_guid": "ssh-process-guid",  
  "domain": "ssh-experiments",  
  "rootfs": "preloaded:cflinuxfs2",  
  "instances": 1,  
  "start_timeout": 30,  
  "setup": {  
    "download": {  
      "artifact": "diego-sshd",  
      "from": "http://file-server.service.cf.internal:8080/v1/static/diego-sshd/diego-sshd.tgz",  
      "to": "/tmp",  
      "cache_key": "diego-sshd"  
    }  
  },  
  "action": {  
    "run": {  
      "path": "/tmp/diego-sshd",  
      "args": [  
        "-address=0.0.0.0:2222",  
        "-authorizedKey=ssh-rsa ..."  
      ],  
      "env": [],  
      "resource_limits": {}  
    }  
  },  
  "ports": [ 2222 ],  
  "routes": {  
    "diego-ssh": {  
      "container_port": 2222,  
      "private_key": "PEM encoded PKCS#1 private key"  
    }  
  }  
}
```

Providing a Certificate for your SSL Termination Point

This topic describes the procedure for providing Elastic Runtime with an SSL certificate, as part of the process of configuring Elastic Runtime for deployment. See [Getting Started with Pivotal Cloud Foundry](#) for help installing PCF on your IaaS of choice.

1. In the Pivotal Cloud Foundry Ops Manager Installation Dashboard, click the **Elastic Runtime** tile.
2. Select **Security Config**.
3. Enter your SSL Certificate, as follows:
 - (Option 1) In a production environment, use a signed **SSL Certificate** from a known certificate authority (CA). Copy and paste the values for **Certificate PEM** and **Private Key PEM** from the signed certificate into the appropriate text fields.
 - (Option 2) In a development or testing environment, you may use a self-signed certificate. To use a self-signed certificate, follow the steps below:
 - Click the **Generate Self-Signed RSA Certificate** link to launch the RSA certificate wizard.
 - Enter your system and app domains in wildcard format. Optionally, also add any custom domains in wildcard format. The example below uses `*.example.com`.

Note: Wildcard DNS records only work for a single domain name component or component fragment. For example, `*.domain.com` works for `apps.domain.com` and `system.domain.com`, but not for `my.apps.domain.com` or `my.system.domain.com`.



- Click **Generate**.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

- Elastic Runtime populates the **SSL Certificate** fields with RSA certificate and private key information.
- Select the **Trust Self-Signed Certificates** checkbox.

1. **(Optional)** Enter a colon-separated list of custom SSL ciphers to pass to HAProxy in the **HAProxy SSL Ciphers** field. If you leave this field empty, PCF uses the default SSL ciphers specified in `cf-release/jobs/haproxy/spec`.
2. Check **Enable cross-container traffic within each DEA** if your installation includes microservices that require cross-container networking, such as [Spring Cloud](#).

Security settings for HAProxy, Router, and DEAs

SSL Termination Certificate *

```
-----BEGIN CERTIFICATE-----
MIIDLTCCAhWgAwIBAgIUCE3jKl2Al0ZD28
36WvkLVE2wwTkwDQYJKoZIhvcNAQEF
BQAwPjELMAKGA1UEBhMCVVMxEDAOBg
NVBAoMBlBpdm90YWwxHTAbBgNVBAM
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEz9fvaybczuZz6kAb+
XgVu3TYEaFOOpAMHDtwJ8QjaTPmKD0j
UhnTQwnrkBjnfO/QUyHn/A9GgcMBLI
5L+3sB71cW2NGwVzIkjWxsNO/03ra
-----
```

[Generate Self-Signed RSA Certificate](#)

Trust Self-Signed Certificates

HAProxy SSL Ciphers

Enable cross-container traffic within each DEA

Save

3. Click **Save**.

Monitoring Instance Usage with the Accounting Report

The Accounting Report displays instance usage information for your PCF installation. To monitor your instance usage information, select **Accounting Report** from the left navigation.

You can see the average and maximum app instances in use per month under **App Instance Statistics**. The Accounting Report calculates these values from the `start`, `stop`, and `scale` app usage events in Cloud Controller.

The maximum is the largest number of app instances in use at any one time. This value is calculated from data generated after 1.5 is installed. The average count of app instances is calculated from all data already in the Usage Service, including the time prior to when 1.5 is installed.

Note: The average and maximum app instances do not include activity in the **system** org.

The screenshot shows the Pivotal CF web interface. The left sidebar has a dark theme with a teal header bar. It includes sections for **ORG** (with a dropdown menu showing "system"), **SPACES** (listing "app-usage-service", "apps-manager", "autoscaling", "notifications-with-ui", and "Marketplace"), and **SYSTEM** (listing "Docs", "Support", and "Tools"). The "Accounting Report" link is highlighted with a teal bar at the bottom of the SYSTEM section. The main content area is titled "Accounting Report" and contains a sub-header "Monthly count of App Instances in use for all orgs (not including the system org)". Below this is a timestamp "As of: 2015-06-02 22:19:26 UTC". A table titled "APP INSTANCE STATISTICS" shows data for the year 2015. The table has three columns: DATE, AVERAGE, and MAXIMUM. It lists two rows: "June (Current)" with Average 16.7 and Maximum 25, and "May" with Average 10.1 and Maximum 12. The footer of the main content area says "Pivotal | © 2015 Pivotal Software Inc. All rights reserved. Pivotal CF v 1.5 (cdf7e3) 6/2/2015".

DATE	AVERAGE	MAXIMUM
June (Current)	16.7	25
May	10.1	12

Enabling IPv6 for Hosted Applications

The procedure described below allows apps deployed to Elastic Runtime to be reached using IPv6 addresses.

Note: Amazon Web Services (AWS) EC2 instances currently do not support IPv6.

Elastic Runtime system components use a separate DNS subdomain from hosted applications. These components currently support only IPv4 DNS resolved addresses. This means that although an IPv6 address can be used for application domains, the system domain must resolve to an IPv4 address.

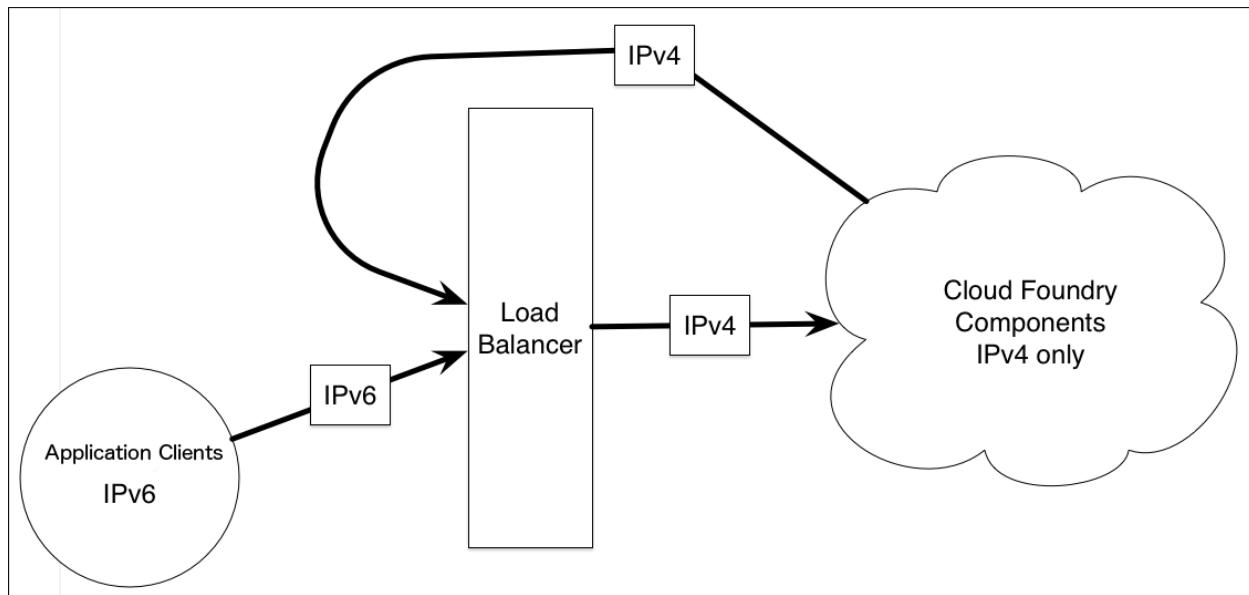
Complete the following steps to enable support for IPv6 application domains:

1. Set up an external load balancer for your Elastic Runtime deployment. See [Using Your Own Load Balancer](#).
2. Configure DNS to resolve application domains to an IPv6 address on your external load balancer.

Note: Your IPv4 interface for the system domain and IPv6 interface for application domain can be configured on the same or different load balancers.

3. Configure the external load balancer to route requests for an IPv6 address to an IPv4 address as follows:
 - If you are using the HAProxy load balancer for SSL termination, route to its IPv4 address.
 - Otherwise, route directly to the IPv4 addresses of the GoRouters.

The following diagram illustrates how a single load balancer can support traffic on both IPv4 and IPv6 addresses for a Elastic Runtime installation.



See [Creating Domains and Routes](#) for more information about domains in Elastic Runtime.

Using Diego in Pivotal Cloud Foundry

This topic describes using Diego in the Pivotal Cloud Foundry (PCF) 1.6 release. Diego is the next-generation container scheduler for [Pivotal Cloud Foundry](#) Elastic Runtime.

In the Pivotal Cloud Foundry (PCF) 1.6 release, Diego replaces the [DEAs](#) and the [Health Manager](#). Diego is installed and enabled in PCF 1.6 by default.

Note: Before you install PCF 1.6, you must uninstall any Diego Beta installations. For more information on upgrading to PCF 1.6, see [Upgrading Operations Manager](#).

Configure Diego in PCF

1. Log in to Pivotal Ops Manager.
2. Click the **Ops Manager Director** tile.

The screenshot shows the 'Pivotal Elastic Runtime' configuration page. At the top, there are four tabs: 'Settings' (selected), 'Status', 'Credentials', and 'Logs'. On the left, a vertical list of configuration items is shown, each with a green checkmark: 'Assign Networks', 'Assign Availability Zones', 'System Database Config', 'File Storage Config', 'IPs and Ports', 'Security Config', 'MySQL Proxy Config', 'Cloud Controller', 'System Logging', 'SSO Config', 'LDAP Config', 'SMTP Config', and 'Diego'. A large grey arrow points from the bottom of the list towards the 'Diego' item. To the right of the list, there is a note: 'Diego is a new replacement for DEAs.' Below this note are two checkboxes: 'Use Diego by default instead of DEAs' (checked) and 'Allow SSH access to apps' (checked). A text input field labeled 'Application Containers Subnet Pool *' contains the value '10.254.0.0/22'. At the bottom right is a blue 'Save' button.

3. Select **Diego**.
4. Select the **Use Diego by default instead of DEAs** checkbox to ensure that all applications pushed to your PCF deployment use the Diego container management system. For new installations, this option is selected by default. If you are upgrading to PCF 1.6, you must select this option to automatically enable Diego for newly pushed applications.

Note: If you do not select this option, application developers must explicitly target Diego when pushing their applications.

5. (Optional) Select the **Allow SSH access to apps** checkbox to allow application developers to `ssh` directly into

their application instances on Diego. See the [Diego-SSH Overview](#) topic for more details.

Deploying Diego for Windows

This topic contains instructions for setting up a Windows cell in a Diego deployment. For more information about Diego, see the [Diego Architecture](#) and [Diego Components](#) topics.

A **cell** is a virtual machine (VM) that stages, hosts, and manages application lifecycles. You can install a Windows cell into your Cloud Foundry (CF) deployment through [Amazon CloudFormation](#) or by [manually](#) connecting directly to a Windows VM.

Install through Amazon CloudFormation

Prerequisites

- A working Diego deployment

Step 1: Download the CloudFormation Template

1. Sign in to [Pivotal Network](#).
2. Select **Elastic Runtime**. From the **DiegoWindows** folder, select and download the CloudFormation template.

Step 2: Upload the Template to Amazon Web Services

Log in to the Amazon Web Services [CloudFormation console](#). When prompted by the CloudFormation wizard, provide the following information:

- **SecurityGroup**: Security group ID to use for the Windows cells
- **BoshUserName**: User name to use to access the BOSH director
- **BoshPassword**: Password to use to access the BOSH director.
- **BoshHost**: BOSH director host
- **ContainerizerPassword**: Password for the containerizer user. Must be alphanumeric and contain at least one capital, one lowercase, and one numeric character.
- **CellName**: The name for your cell
- **VPCID**: The ID of the VPC where you want the cell and subnet to be created
- **NATInstance**: The instance ID of the NAT box. To find this instance ID, search for **NAT** in the CloudFormation dropdown.
- **SubnetCIDR**: The IP range of the Windows cell subnet, for example `10.0.100.0/24`. You should provide a range that does not collide with an existing subnet within the VPC.
- **Keypair**: A keypair. You must have the private key to this keypair to retrieve the Administrator password to the Windows VMs that are created.

The CloudFormation template will configure the Windows cell for the appropriate availability zone based on the provided security group, install the MSI, and register itself with Diego. The CloudFormation template will only succeed if all services are up and running after installation. To debug a failed install, set “Rollback on failure” to `No` under advanced options.

Install Manually

Prerequisites

- A working Diego deployment
- A Windows Server 2012R2 VM instance that is routable to your Diego deployment
 - See [recommended instance types](#) in the GitHub Diego release repo for details.
 - If you are creating a new Windows image, and not using one predefined and supplied by your IaaS, we recommend using this [ISO image](#) as a starting point. You must have an MSDN account to download this ISO image.

- A new subnet for the Windows cell
 - You should place the Windows cell and BOSH in separate subnets to avoid potential conflicts. A BOSH deploy fails if the Windows cell uses an IP address that BOSH needs.

Step 1: Configure the Windows Cell

Follow the instructions below to download and configure the Windows cell.

1. From your Windows cell, log in to the [Pivotal Network](#).
2. Download the `setup.ps1` file from the DiegoWindows folder in [Elastic Runtime](#).
3. Open File Explorer.
4. Right-click on the `setup.ps1` file and select **Run with PowerShell**. The `setup.ps1` script configures Windows features, DNS settings, and the firewall for your Windows cell.

Step 2: Run the Install Script Generator

1. Download `generate.exe` from the DiegoWindows folder in [Elastic Runtime](#).

Note: If you download the file using Internet Explorer, the `.exe` extension will be removed. You must rename the file to add the `.exe` extension.

2. From a command prompt, run `generate.exe` with the following arguments: `$ generate.exe -bosshUrl https://username:password@boshs-director.example.com:25555 -windowsUsername=Administrator -windowsPassword=Password123 -outputDir C:\diego-windows`
 - **bosshUrl:** Find your `bosshUrl` in the [troubleshooting](#) documentation.
 - **windowsUsername:** Used to run Containerizer. The user provided must be a member of the Administrators group. All other services will run as the SYSTEM user.
 - **outputDir:** This specifies the directory that will contain the required certificates and a script to run the installers.

Step 3: Install the MSI

1. Download both `DiegoWindows.msi` and `GardenWindows.msi` from the DiegoWindows folder in [Elastic Runtime](#) to the `outputDir` that you specified above.
2. Run `install.bat` from the `outputDir`.

Step 4: Confirm Successful Deployment

Follow the steps below to deploy a sample .NET application to one of your Windows cells and exercise basic Cloud Foundry functionality to ensure that your deployment is functioning properly.

1. Launch **Task Manager**.
2. Navigate to the Services tab and confirm that the following services are running:

NAME	DESCRIPTION	STATUS
Consul	CF Consul	Running
Containerizer	CF Containerizer	Running
GardenWindows	CF GardenWindows	Running
Metron	CF Metron	Running
Rep	CF Rep	Running

3. Download/clone the [CF Smoke Tests](#) repository.
4. Follow the instructions from the README to run the smoke tests against your environment with the `enable_windows_tests` configuration flag set to `true`.

Administering and Operating Cloud Foundry

For Administrators of a Running Cloud Foundry Deployment

- [Adding Buildpacks to Cloud Foundry](#)
- [Managing Domains and Routes](#)
- [Creating and Managing Users with the cf CLI](#)
- [Creating and Managing Users with the UAA CLI \(UAAC\)](#)
- [Getting Started with the Notifications Service](#)
- [Application Security Groups](#)
- [Feature Flags](#)

For Operators Deploying Cloud Foundry

- [Enabling IPv6 for Hosted Applications](#)
- [Securing Connections from a Load Balancer to the CF Routers](#)

Adding Buildpacks to Cloud Foundry

If your application uses a language or framework that Cloud Foundry system buildpacks do not support, you can [write your own buildpack](#), customize an existing buildpack, or use a [Cloud Foundry Community Buildpack](#) or a [Heroku Third-Party Buildpack](#). You can add the new buildpack to Cloud Foundry, making it available alongside the system buildpacks.

The cf Admin-Only Buildpack Commands

 **Note:** You must be an administrator for your Cloud Foundry org to run the commands discussed in this section.

To add a buildpack, run:

```
$ cf create-buildpack BUILDPACK PATH POSITION [--enable|--disable]
```

The arguments to `cf create-buildpack` do the following:

- **buildpack** specifies what you want to call the buildpack.
- **path** specifies where to find the buildpack. The path can point to a zip file, the URL of a zip file, or a local directory.
- **position** specifies where to place the buildpack in the detection priority list. See [Buildpack Detection](#).
- **enable or disable** specifies whether to allow apps to be pushed with the buildpack. This argument is optional, and defaults to enable. While a buildpack is disabled, app developers cannot push apps using that buildpack.

You can also update and delete buildpacks. For more information, run:

```
cf update-buildpack -h
```

```
cf delete-buildpack -h
```

Confirming that a Buildpack was Added

To confirm that you have successfully added a buildpack, view the available buildpacks by running `cf buildpacks`.

The example below shows the `cf buildpacks` output after the administrator added a python buildpack.

```
$ cf buildpacks
Getting buildpacks...

buildpack      position  enabled  locked   filename
ruby_buildpack  1         true     false    buildpack_ruby_v46-245-g2fc4ad8.zip
nodejs_buildpack 2         true     false    buildpack_nodejs_v8-177-g2b0a5cf.zip
java_buildpack  3         true     false    buildpack_java_v2.1.zip
python_buildpack 4         true     false    buildpack_python_v2.7.6.zip
```

Disabling Custom Buildpacks from Ops Manager

You can disable custom buildpacks using your Ops Manager Elastic Runtime tile. From the **Cloud Controller** tab, check the **Disable Custom Buildpacks** checkbox, as shown in the image below.

The screenshot shows the 'Settings' tab selected in the Pivotal Elastic Runtime dashboard. On the left, a sidebar lists various configuration sections: Assign Networks, Assign Availability Zones, IPs and Ports, MySQL Proxy Config, Cloud Controller (which is currently selected), External Endpoints, SSO Config, LDAP Config, SMTP Config, Lifecycle Errands, Resource Config, and Stemcell. The main panel displays configuration for the selected 'Cloud Controller' section. It includes fields for 'System Domain' (with placeholder 'systemdomain'), 'Apps Domain' (with placeholder 'appdomain'), 'Cloud Controller DB Encryption Key' (set to 'Secret'), 'Maximum File Upload Size (MB)' (set to '1024'), a checked checkbox for 'Disable Custom Buildpacks', 'Default Quota App Memory (MB)' (set to '10240'), and 'Default Quota Service Instances' (set to '100'). A 'Save' button is at the bottom.

By default, the cf CLI gives developers the option of using a custom buildpack when they deploy apps to Elastic Runtime. To do so, they use the `-b` option to provide a custom buildpack URL with the `cf push` command. The **Disable Custom**

Buildpacks checkbox disables the `-b` option.

For more information about custom buildpacks, refer to the [buildpacks](#) section of the PCF documentation.

Managing Domains and Routes

If you are an administrator, you can manage custom shared domains and wildcard routes.

For additional information about managing domains and routes, refer to the following topics:

- [Creating Domains and Routes](#)
- [Domains and Shared Domains](#)

Creating a Shared Custom Domain

You can use a registered domain of your own and associate it with all organizations in your account. Use the `cf create-shared-domain` command to create a shared custom domain available to all organizations in your account.

For example:

```
$ cf create-shared-domain example.com
```

Deleting a Shared Custom Domain

Use the `cf delete-shared-domain` command to delete a shared domain:

```
$ cf delete-shared-domain example.com
```

 **Note:** Deleting a shared domain removes all associated routes, making any application with this domain unreachable.

Creating a Wildcard Route

Use the `cf create-route` command with a **wildcard route** by specifying the host as `*`. The star operator, `*`, signals a wildcard route to match any URL that uses your domain, regardless of the host.

 **Note:** You must surround the `*` with quotation marks when referencing it using the CLI.

For example, the following command created the wildcard route “*.example.org” in the “development” space:

```
$ cf create-route development example.org -n "*"
```

Creating and Managing Users with the cf CLI

This page assumes you are using cf CLI v6.

Note: The procedures described here are not compatible with using LDAP for user identity management. To create and manage user accounts in an LDAP enabled Cloud Foundry deployment, see [Onboarding Existing LDAP Users to a PCF Deployment](#).

Using the cf Command Line Interface (CLI), an administrator can create users and manage user roles. Cloud Foundry uses role-based access control, with each role granting permissions in either an organization or an application space.

For more information, see [Organizations, Spaces, Roles, and Permissions](#).

Note: To manage users, organizations, and roles with the cf CLI, you must log in with **UAA Administrator user credentials**. In Pivotal Operations Manager, refer to **Elastic Runtime > Credentials** for the UAA admin name and password.

Creating and Deleting Users

FUNCTION	COMMAND	EXAMPLE
Create a new user	cf create-user USERNAME PASSWORD	cf create-user Alice pa55w0rd
Delete a user	cf delete-user USERNAME	cf delete-user Alice

Creating Administrator Accounts

To create a new administrator account, use the [UAA CLI](#).

Note: The cf CLI cannot create new administrator accounts.

Org and App Space Roles

A user can have one or more roles. The combination of these roles defines the user's overall permissions in the org and within specific app spaces in that org.

Org Roles

Valid [org roles](#) are OrgManager, BillingManager, and OrgAuditor.

FUNCTION	COMMAND	EXAMPLE
View the organizations belonging to an account	cf orgs	cf orgs
View all users in an organization by role	cf org-users ORGANIZATION_NAME	cf org-users my-example-org
Assign an org role to a user	cf set-org-role USERNAME ORGANIZATION_NAME ROLE	cf set-org-role Alice my-example-org OrgManager
Remove an org role from a user	cf unset-org-role USERNAME ORGANIZATION_NAME ROLE	cf unset-org-role Alice myexample-org OrgManager

App Space Roles

Each app space role applies to a specific app space.

Note: By default, the org manager has app space manager permissions for all spaces within the organization.

Valid [app space roles](#) are SpaceManager, SpaceDeveloper, and SpaceAuditor.

FUNCTION	COMMAND	EXAMPLE
View the spaces in an org	cf spaces	cf spaces
View all users in a space by role	cf space-users ORGANIZATION_NAME SPACE_NAME	cf space-users my-example-org development
Assign a space role to a user	cf set-space-role USERNAME ORGANIZATION_NAME SPACE_NAME ROLE	cf set-space-role Alice my-example-org development SpaceAuditor
Remove a space role from a user	cf unset-space-role USERNAME ORGANIZATION_NAME SPACE_NAME ROLE	cf unset-space-role Alice my-example-org development SpaceAuditor

Creating and Managing Users with the UAA CLI (UAAC)

Using the UAA Command Line Interface (UAAC), an administrator can create users and manage organization and space roles.

For additional details and information, refer to the following topics:

- [UAA Overview](#)
- [UAA Sysadmin Guide ↗](#)
- [Other UAA Documentation ↗](#)

Create an Admin User

1. Install the UAA CLI, `uaac`.

```
$ gem install cf-uaac
```

2. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

```
$ uaac target uaa.example.com
```

3. Record the **uaa:admin:client_secret** from your deployment manifest.

4. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

5. Use `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  access_token: yJhbGciOiJIUzI1NiJ9.e
  token_type: bearer
  expires_in: 43200
  scope: uaa.admin clients.secret scim.read
  jti: 91b3-abcd1233
```

6. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **scim.write**. The value **scim.write** represents sufficient permissions to create accounts.

7. If the admin user lacks permissions to create accounts:

- Use `uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
- Use `uaac token delete` to delete the local token.
- Use `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  ...
  scope: uaa.admin clients.secret scim.read
  ...

$ uaac client update admin --authorities "`uaac client get admin | \
awk '/:/{e=0}/authorities:/{{e=1;if(e==1){$1=""}};print}`` scim.write"

$ uaac token delete
$ uaac token client get admin
```

8. Use `uaac user add NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD --emails NEW-ADMIN-EMAIL` to create an admin user.

```
$ uaac user add Adam -p newAdminSecretPassword --emails newadmin@example.com
```

9. Use `uaac member add GROUP NEW-ADMIN-USERNAME` to add the new admin to the groups `cloud_controller.admin`, `uaa.admin`, `scim.read`, `scim.write`.

```
$ uaac member add cloud_controller.admin Adam
$ uaac member add uaa.admin Adam
$ uaac member add scim.read Adam
$ uaac member add scim.write Adam
```

Grant Admin Permissions to an LDAP Group

To grant all users under an LDAP Group admin permissions:

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

3. Run the following commands to grant all user under the mapped LDAP Group admin permissions:
 - `uaac group map --name scim.read "GROUP-DISTINGUISHED-NAME"`
 - `uaac group map --name cloud_controller.admin "GROUP-DISTINGUISHED-NAME"`

Create Users

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Use `cf login -u NEW-ADMIN-USERNAME -p NEW-ADMIN-PASSWORD` to log in.

```
$ cf login -u Adam -p newAdminSecretPassword
```

3. Use `cf create-user NEW-USER-NAME NEW-USER-PASSWORD` to create a new user.

```
$ cf create-user Charlie aNewPassword
```

Change Passwords

1. Obtain the credentials of an admin client created using UAAC as above, or refer to the `uaa: scim` section of your deployment manifest for the user name and password of an admin user.
2. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.
3. Use `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  access_token: yJhbGciOiJIUzI1NiJ9.e
  token_type: bearer
  expires_in: 43200
  scope: uaa.admin.clients.secret password.read
  jti: 91b3-abcd1233
```

4. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **password.write**. The value **password.write** represents sufficient permissions to change passwords.
5. If the admin user lacks permissions to change passwords:
 - Use `uaac client update admin --authorities "EXISTING-PERMISSIONS"` to add the necessary permissions to the `password.write`" admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
 - Use `uaac token delete` to delete the local token.
 - Use `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  ...
  scope: uaa.admin.clients.secret password.read
  ...

$ uaac client update admin --authorities "`uaac client get admin | \
awk '/:{e=0}/authorities:{e=1;if(e==1){$1=""};print}`` password.write"

$ uaac token delete
$ uaac token client get admin
```

6. Use `uaac password set USER-NAME -p TEMP-PASSWORD` to change an existing user password to a temporary password.

```
$ uaac password set Charlie -p ThisIsATempPassword
```

7. Provide the `TEMP-PASSWORD` to the user. Have the user use `cf target api.YOUR-DOMAIN`, `cf login -u USER-NAME -p TEMP-PASSWORD`, and `cf passwd` to change the temporary password.

```
$ cf target api.example.com
$ cf login -u Charlie -p ThisIsATempPassword
$ cf passwd

Current Password>ThisIsATempPassword

New Password>*****  
Verify Password>*****
Changing password...
```

Retrieve User Email Addresses

Some Cloud Foundry components, like Cloud Controller, only use GUIDs for user identification. You can use the UAA to retrieve the emails of your Cloud Foundry instance users either as a list or for a specific user with that user's GUID.

To retrieve user email addresses:

1. Use `uaac target uaa.YOUR-DOMAIN` to target your UAA server.

```
$ uaac target uaa.example.com
```

2. Record the **uaa:admin:client_secret** from your deployment manifest.

3. Use `uaac token client get admin -s ADMIN-CLIENT-SECRET` to authenticate and obtain an access token for the admin client from the UAA server. UAAC stores the token in `~/.uaac.yml`.

```
$ uaac token client get admin -s MyAdminPassword
```

4. Use `uaac contexts` to display the users and applications authorized by the UAA server, and the permissions granted to each user and application.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  access_token: yJhbGciOiJIUzI1NiJ9.e
  token_type: bearer
  expires_in: 43200
  scope: uaa.admin clients.secret
  jti: 91b3-abcd1233
```

5. In the output from `uaac contexts`, search in the `scope` section of the `client_id: admin` user for **scim.read**. The value **scim.read** represents sufficient permissions to query the UAA server for user information.

6. If the admin user lacks permissions to query the UAA server for user information:

- Use `uaac client update admin --authorities "EXISTING-PERMISSIONS scim.write"` to add the necessary permissions to the admin user account on the UAA server. Replace EXISTING-PERMISSIONS with the current contents of the `scope` section from `uaac contexts`.
- Use `uaac token delete` to delete the local token.
- Use `uaac token client get admin` to obtain an updated access token from the UAA server.

```
$ uaac contexts

[1]*[admin]
  client_id: admin
  ...
  scope: uaa.admin clients.secret
  ...

$ uaac client update admin --authorities "uaa.admin clients.secret scim.read"

$ uaac token delete
$ uaac token client get admin
```

7. Use `uaac users` to list your Cloud Foundry instance users. By default, the `uaac users` command returns information about each user account including GUID, name, permission groups, activity status, and metadata. Use the `--attributes emails` or `-a emails` flag to limit the output of `uaac users` to email addresses.

```
$ uaac users --attributes emails

resources:
  emails:
    value: user1@example.com
    emails:
    value: user2@example.com
    emails:
    value: user3@example.com
```

8. Use `uaac users "id eq GUID" --attributes emails` with the GUID of a specific user to retrieve that user's email address.

```
$ uaac users "id eq 'aabbc11-22a5-87-8056-beaf84'" --attributes emails

resources:
  emails:
    value: user1@example.com
```

Getting Started with the Notifications Service

This guide is intended to walk you through using the Notifications Service.

We'll show you how to set up the service through the following example: Darla is a cloud operator who needs to take the system down for maintenance and wants to notify everybody about her intentions.

Prerequisites

Prior to using the Notifications service, Darla must have done the following:

- Installed Pivotal Cloud Foundry
- Set up an `admin` account on her Cloud Foundry instance
- Installed the Notifications service using the [Notifications with UI errand](#)
- Installed the `cf` and `uaac` command line tools

Create your Client and Get a Token

To interact with the Notifications service, Darla needs certain UAA scopes (authorities). Rather than use her `admin` user account directly, she creates a `notifications-admin` client with the required scopes:

```
$ uaac client add notifications-admin --authorized_grant_types client_credentials --authorities \
  notifications.manage,notifications.write,notification_templates.write,notification_templates.read,critical_notifications
```

It's worth noting that she doesn't need all of these scopes just to send a notification. `notifications.manage` is used to update notifications and assign templates for that notification. `notification_templates.write` allows Darla to custom-make her own template for a notification, and `notification_templates.read` allows her to check which templates are saved in the database. Finally, `notifications.write` is the scope necessary to send a notification to a user, space, everyone in the system, and more.

Now, Darla logs in using her newly created client:

```
$ uaac token client get notifications-admin
```

Stay logged in with this client for the rest of the examples in this guide.

Registering Notifications

Darla cannot send a notification unless she has registered it first. Registering notifications requires the `notifications.manage` scope on her client. For example:

```
$ uaac curl https://notifications.darla.example.com/notifications -X PUT --data '{ "source_name": "Cloud Ops Team",
  "notifications": {
    "system-going-down": { "critical": true, "description": "Cloud going down" },
    "system-up": { "critical": false, "description": "Cloud back up" }
  }
}'
```

Darla has registered two different notifications: `system-going-down` and `system-up`. In addition, she gives the `notifications-admin` client the human-friendly description "Cloud Ops Team." Notice that she has made the `system-going-down` notification `critical`. This means that no users can unsubscribe from that notification. Setting notifications as critical requires the `critical_notifications.write` scope.

Create a Custom Template

The system provides a default template for all notifications, but Darla has decided to forgo this luxury. Darla wants to include her own branding and has opted to create her own custom template using the curl below (this action requires the

```
notification_templates.write scope):  
  
$ uaac curl https://notifications.darla.example.com/templates -X POST --data \  
'{"name":"site-maintenance","subject":"Maintenance: {{.Subject}}","text":"The site has gone down for maintenance. More info  
The site has gone down for maintenance. More information to follow {{.HTML}}"}'
```

A template is made up of a human readable name, a subject, a text representation of the template you are sending (for mail clients that do not support HTML), and an HTML version of the template.

Special attention and care should be paid to the variables that take this form `{{.}}`. These variables interpolate data provided in the send step below into the template before a notification is sent. Data that you can insert into a template during the send step includes `{{.Text}}`, `{{.HTML}}`, and `{{.Subject}}`.

This curl returns a unique template ID that can be used in subsequent calls to refer to your custom template. The result looks similar to this:

```
{"template-id": "E3710280-954B-4147-B7E2-AF5BF62772B5"}
```

Darla can check all of the saved templates by curling:

```
$ uaac curl https://notifications.darla.example.com/templates -X GET
```

To view a list of all templates, you must have the `notifications_templates.read` scope.

Associate a Custom Template with your Notification

Darla now wants to associate her custom template with the `system-going-down` notification. Any notification that does not have a custom template applied, like her `system-up` notification, defaults to a system-provided template.

```
$ uaac curl https://notifications.darla.example.com/clients/notifications-admin/notifications/system-going-down/template \  
-X PUT --data '{"template": "E3710280-954B-4147-B7E2-AF5BF62772B5"}'
```

Here, Darla has associated the `system-going-down` notification belonging to the `notifications-admin` client with the template ID `E3710280-954B-4147-B7E2-AF5BF62772B5`. This is the template id of the template we created in the previous step.

This action requires the `notifications.manage` scope.

Send your Notification to All Users

Darla is ready to send her `system-going-down` notification to all users of the system. She performs this curl and includes some other pertinent information that gets directly inserted into the template:

```
$ uaac curl https://notifications.darla.example.com/everyone -X POST --data \  
'{"kind_id":"system-going-down","text":"The system is going down while we upgrade our storage","html":"<h1>THE SYSTEM IS DOING DOWN</h1>"}'
```

The data included in the post body above gets interpolated into the variables we previously inserted into our created template (they had the special syntax similar to `{{.Text}}`).

Sending a critical notification requires the scope `critical_notifications.write`, whereas sending a non-critical notification requires the scope of `notifications_write`.

Darla could have also chosen to send the above notification to one specific user, an email address, or a particular space.

Application Security Groups

This page assumes you are using cf CLI v6.4 or higher.

Application security groups act as virtual firewalls to control outbound traffic from the applications in your deployment. A security group consists of a list of network egress access rules.

An administrator can assign one or more security groups to a Cloud Foundry deployment or to a specific [space](#) in an [org](#) within a deployment.

Note: A security group assigned to a Cloud Foundry deployment affects all spaces in all orgs within the deployment.

Within a space, Cloud Foundry runs each instance of an application inside a separate application container. When you launch an application for the first time, Cloud Foundry creates a new container for each application instance, then applies any space-specific and deployment-wide security groups to the container. Cloud Foundry determines whether to allow or deny outbound traffic from the container by evaluating the rules defined in these security groups.

Creating Security Groups

A security group consists of a list of operator-defined network egress `allow` rules. These rules define the outgoing traffic allowed to application containers. Each rule contains the following three parts:

- **Protocol:** TCP, UDP, or ICMP
- **Open Port / Port Range:**
 - For TCP and UDP: Either a single port or a range of ports
 - For ICMP: An ICMP type and code
- **Destination:** Destination of the traffic allowed by this rule as an IP address or CIDR block

Run `cf create-security-group SECURITY-GROUP PATH-TO-RULES-FILE` from a command line to create a security group named `SECURITY-GROUP`. `PATH-TO-RULES-FILE` can be an absolute or relative path to a rules file. The rules file must be a JSON-formatted single array containing objects that describe the rules.

Example JSON-formatted rules file:

```
[{"protocol": "tcp", "destination": "10.0.11.0/24", "ports": "1-65535"},  
 {"protocol": "udp", "destination": "10.0.11.0/24", "ports": "1-65535"}]
```

Binding Security Groups

A security group consists of a list of rules. You must bind this list to either your entire deployment or to a space in an org for Cloud Foundry to apply the rules to outgoing traffic.

Note: New security rules apply to new containers as they are created, but not to containers that are already running when the rules are created. To apply new security rules immediately, restart all running apps in the relevant space, or across your whole installation if the new rules are for all spaces.

To apply the rules in a security group to your entire Cloud Foundry deployment, bind the security group to either the **Default Staging** or the **Default Running** security group set. Cloud Foundry applies the rules in every security group in the Default Staging and Default Running sets to all applications in your Cloud Foundry deployment.

Note: If you do not bind a security group to either the **Default Staging** or the **Default Running** set, Cloud Foundry only applies the rules to applications created in the specific space where you bound the security group.

Binding to your Cloud Foundry Deployment

To create a rule to be applied to every space in every org of your Cloud Foundry deployment, bind the security group to

either the **Default Staging** or the **Default Running** security group set.

Cloud Foundry applies the rules in every security group in these sets as follows:

- **Default Staging:** Cloud Foundry applies rules in this set to every application staged anywhere in your CF deployment. To bind a security group to the Default Staging set, run `cf bind-staging-security-group SECURITY-GROUP`.
- **Default Running:** Cloud Foundry applies rules in this set to every application running anywhere in your CF deployment. To bind a security group to the Default Running set, run `cf bind-running-security-group SECURITY-GROUP`.

Binding to a Space

Run `cf bind-security-group SECURITY-GROUP ORG SPACE` to bind a security group to a specific space. Cloud Foundry applies the rules that this security group defines to all application containers in the space. A space may belong to more than one security group.

Network Traffic Rules Evaluation Sequence

Cloud Foundry evaluates security groups and other network traffic rules in a strict priority order. Cloud Foundry returns an `allow`, `deny`, or `reject` result for the first rule that matches the outbound traffic request parameters, and does not evaluate any lower-priority rules.

Cloud Foundry evaluates the network traffic rules for an application in the following order:

1. **Security Groups:** The rules described by the Default Staging set, the Default Running set, and all security groups bound to the space.
2. **Warden ALLOW rules:** Any Warden Server configuration `allow` rules. Set Warden Server configuration rules in the Droplet Execution Agent (DEA) configuration section of your deployment manifest.
3. **Warden DENY rules:** Any Warden Server configuration `deny` rules. Set Warden Server configuration rules in the DEA configuration section of your deployment manifest.
4. **Hard-coded REJECT rule:** Cloud Foundry returns a `reject` result for all outbound traffic from a container if not allowed by a higher-priority rule.

Viewing Security Groups

Run the following commands to view information about existing security groups:

- `cf security-groups`: Displays all security groups in an org
- `cf staging-security-groups`: Displays all security groups in the Default Staging set
- `cf running-security-groups`: Displays all security groups in the Default Running set
- `cf security-group SECURITY-GROUP`: Displays details about the specified security group

Feature Flags

Feature flags are switches that you set using the Cloud Controller API. They allow an administrator to turn on or off functional sections of code, or features, of an application without deploying new code. Use feature flags to enable or disable features available to users.

Feature Flags

There are six feature flags that you can set. They are all enabled by default except `user_org_creation`. When disabled, these features are only available to administrators.

- `user_org_creation`: Any user can create an organization via the API.
- `private_domain_creation`: An organization manager can create private domains for that organization.
- `app_bits_upload`: Space developers can upload app bits.
- `app_scaling`: Space developers can perform scaling operations (i.e. change memory, disk, or instances).
- `route_creation`: Space developers can create routes in a space.
- `service_instance_creation`: Space developers can create service instances in a space.

Feature Flag Commands

Get All Feature Flags

```
GET /v2/config/feature_flags
```

Get a Feature Flag

```
GET /v2/config/feature_flags/FEATURE_FLAG_NAME
```

Set a Feature Flag

```
PUT /v2/config/feature_flags/FEATURE_FLAG_NAME
```

To view the feature flag commands, review the **Feature Flags** section of the [Cloud Foundry API documentation](#).

Page not Found

Visit the [homepage](#).

Page not Found

Visit the [homepage](#).

Using the Apps Manager

This section provides help with using the web-based Apps Manager application for managing users, organizations, spaces, and applications.

Table of Contents

- [Getting Started with the Apps Manager](#)
- [Understanding Apps Manager Permissions](#)
- [Managing Spaces Using the Apps Manager](#)
- [Managing User Accounts in Spaces Using the Apps Manager](#)
- [Managing User Permissions Using the Apps Manager](#)

Getting Started with the Apps Manager

The Apps Manager is a tool to help manage organizations, users, applications, and spaces. Complete the following steps to invite new users to the Apps Manager.

1. Log in to the Apps Manager: [How to log in](#).
2. The Apps Manager displays information for your org and all of the spaces in your org.

3. In the Members tab, click **Invite New Members**.

4. Add an email address, assign a team role or roles, then click **Send Invite** to invite a new user to join your org.

Important Tips

- You can remove users from orgs, but not spaces. Instead, revoke the user's permissions in all spaces to effectively remove the user from the org.
- Not all of the Apps Manager pages are Ajax-enabled. Refresh the page to see the latest information.
- Changes to environment variables, as well as service bindings and unbindings, require a `cf push` to update the application.

Understanding Apps Manager Permissions

In most cases, the actions available to you on the Apps Manager are a limited subset of the commands available through the CLI. However, depending on your user role and the values that the Admin specifies for the [Apps Manager environment variables](#), you might be able to perform certain org and space management actions on the Apps Manager that usually only an Admin can perform with either the CLI or the Apps Manager.

The table below shows the relationship between specific org and space management actions that you can perform and the users that can perform these actions.

Note that:

- Admins can use either the CLI or the Apps Manager to perform these actions.
- Org Managers, like Admins, can perform all actions using the Apps Manager.
- Space Managers, like Admins and Org Managers, can assign and remove users from spaces using the Apps Manager.
- Apps Manager users of any role can create an org and view org and space users.

	CLI	Apps Manager		
Command	Admin	Admin or Org Manager	Space Manager	Org Auditor or Space Developer or Space Auditor
create-org	X	X	X	X
delete-org	X	X		
rename-org	X	X		
org-users	X	X	X	X
set-org-role	X	X		
unset-org-role	X	X		
space-users	X	X	X	X
set-space-role	X	X	X	
unset-space-role	X	X	X	

Managing Spaces Using the Apps Manager

To manage a space in an org, you must have org manager or space manager permissions in that space.

Org managers have space manager permissions by default and can manage user accounts in all spaces associated with the org.

Log in to the Apps Manager: [How to log in.](#)

The screenshot shows the Pivotal CF Apps Manager interface. At the top, it says "My-Org". Below that is a summary bar with "ORG" (My-Org), "QUOTA" (17% of 10 GB Limit), and "Usage Report". Underneath is a navigation bar with "3 Spaces", "1 Domain", and "1 Member". The main area shows three spaces: "development" (highlighted), "staging", and "production". Each space has a card showing "APPS" and "SERVICES" counts. For development, there are 3 apps and 0 services, using 17% of org quota. For staging and production, both have 0 apps and 0 services, using 0% of org quota. At the bottom right is a "Add A Space" button.

Orgs

The current org is highlighted in the Apps Manager navigation. The drop-down menu on the right displays other orgs. Use this menu to switch between orgs.

The screenshot shows the Pivotal CF Apps Manager interface. At the top, it says "Pivotal CF". Below that is a navigation bar with "ORG" and a dropdown menu showing "My-Org" (highlighted) and another option.

Spaces

The Apps Manager navigation also shows the spaces in the current org. The current space is highlighted.

The screenshot shows the Pivotal CF Apps Manager interface. At the top, it says "Pivotal CF". Below that is a navigation bar with "ORG" and a dropdown menu showing "My-Org" and another option. Underneath is a "SPACES" section with a dropdown menu showing "development" (highlighted), "production", "staging", and "Marketplace".

- **Add a Space:** Click the **Add A Space** button.

- **Switch Space:** Select a different space on the Org dashboard or from the navigation.

Edit Space

From the Space page, click the **Edit Space** link.

My-Org > development > Edit Space

SPACE SETTINGS

Current Org: My-Org

Space Name: development

[Delete Space](#)

[Cancel](#) [Save](#)

- **Delete a Space:** Click the **Delete Space** button in the top right corner.
- **Rename a Space:** Change the text in the **Space Name** field, then click **Save**.

Apps List

The Apps List displays information about the **Status**, **Name**, **Route**, number of **Instances**, and amount of **Memory** for each app in the current space.

APPLICATIONS				LEARN MORE
STATUS	APP	INSTANCES	MEMORY	
	hello-world hello-world.ch...	1	128MB	>
	spring-music spring-music.cherr...	1	512MB	>

Services View

The Services view lists services bound to apps in the current space.

SERVICES			ADD SERVICE
SERVICE INSTANCE	SERVICE PLAN	BOUND APPS	
mysql-dev Manage Documentation Support Delete	Pivotal MySQL Dev 100mb	3	

- **Add a Service:** Click the **Add Service** button on the right side of the Services view. Clicking the **Add Service** button takes you to the Marketplace.

Managing User Accounts in Spaces Using the Apps Manager

Note: The procedures described here are not compatible with using LDAP for user identity management. To create and manage user accounts in an LDAP enabled Cloud Foundry deployment, see [Onboarding Existing LDAP Users to a PCF Deployment](#).

To manage user accounts in a space, you must have Space Manager permissions in that space. Org Managers have Space Manager permissions by default and can manage user accounts in all spaces associated with the org.

Space Roles and Permissions

The different user roles are described on the right side of the Teams view.

Space Manager

Space Managers can invite and manage users and enable features for a given space. Assign this role to managers or other users who need to administer the account.

Space Developer

Space Developers can create, delete, and manage applications and services, and have full access to all usage reports and logs. Space Developers can also edit applications, including the number of instances and memory footprint. Assign this role to app developers or other users who need to interact with applications and services.

Space Auditor

Space Auditors have view-only access to all space information, settings, reports, and logs. Assign this role to users who need to view but not edit the application space.

Inviting New Users

1. On the Org dashboard, click the **Members** tab.

The screenshot shows the 'Members' tab of the Org dashboard. At the top, there are three tabs: '1 Space', '1 Domain', and '1 Member'. The '1 Member' tab is selected. Below the tabs, there is a dropdown menu set to 'org'. To the right of the dropdown is a blue 'INVITE NEW MEMBERS' button. The main area displays a table with three columns: 'MEMBER', 'ORG MANAGER', and 'ORG AUDITOR'. A single member named 'admin' is listed. In the 'ORG MANAGER' column, there is a checked checkbox. In the 'ORG AUDITOR' column, there is also a checked checkbox. The entire interface has a light gray background with blue and black text.

2. Click **Invite New Members**. The **Invite New Team Member(s)** form appears.

4 Spaces | 1 Domain | 2 Members | Delete Org

INVITE NEW TEAM MEMBER(S)

Add Email Address(es) Use commas to separate emails

ASSIGN TEAM ROLES

ORG	ORG MANAGER	ORG AUDITOR	ORGANIZATION ROLES
org	<input type="checkbox"/>	<input type="checkbox"/>	ORGANIZATION MANAGER Can invite/manage users, select/change the plan, establish spending limits
<input type="checkbox"/> Select All			ORGANIZATION AUDITOR Read-only access to org info and reports

APP SPACES	SPACE MANAGER	SPACE DEVELOPER	SPACE AUDITOR	APP SPACE ROLES
development	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SPACE MANAGER Can invite/manage users, enable features for a given space
production	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SPACE DEVELOPER Can create, delete, manage applications and services, full access to all usage reports and logs
staging	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SPACE AUDITOR Read-only access to all space information, settings, reports, logs
testing	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Select All				

CANCEL **SEND INVITE**

- In the **Add Email Address(es)** text field, enter the email addresses of the users that you want to invite. Enter multiple email addresses as a comma-delimited list.
- The **Assign Team Roles** view lists the current org and available spaces with checkboxes corresponding to each possible user role. Select the checkboxes that correspond to the permissions that you want to grant to the invited users.
- Click **Send Invite**. The Apps Manager sends an email containing an invitation link to each email address that you specified.

Changing User Permissions

You can also change user permissions for existing users on the account. User permissions are handled on a per-space basis, so you must edit them for each user and for each space that you want to change.

- On the Org dashboard, click the **Members** tab.

1 Space | 1 Domain | 1 Member

org **INVITE NEW MEMBERS**

MEMBER	ORG MANAGER	ORG AUDITOR
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

- Edit the permissions assigned to each user by checking or unchecking the checkboxes under each user role. The Apps Manager saves your changes automatically.

Removing a User

To remove a user from a particular space, revoke that user's permissions for all user roles in that space. The user remains visible in the list unless you remove the user from the org.

Adding Existing LDAP Users to Your PCF Deployment

This topic describes the procedure for adding existing LDAP users to an LDAP-enabled Pivotal Cloud Foundry (PCF) deployment.

Note: You must have admin access to the PCF Ops Manager Installation Dashboard for your deployment to complete the procedure described here.

Step 1: Restrict User Management to Admin Only

Note: After this change, Org Managers are unable to change space and org roles for users.

Follow this procedure to disable **Invitations**, **Create Account**, and **Reset Password** from the Apps Manager console so that users are not allowed to register new accounts. Pivotal recommends this because invitations create duplicate accounts, which can cause authentication problems that are difficult to resolve.

1. Launch Apps Manager at <https://apps.SYSTEM-DOMAIN>.
2. Login with UAA admin user credentials. To find your UAA admin credentials in the PCF Ops Manager Installation Dashboard, click the **Elastic Runtime** tile, select **Credentials**, and find **UAA**.
3. Select your **System** org.
4. Select your **apps_manager** space and click the **apps_manager** app.
5. Select **Environment Variables**.
6. Locate **ENABLE_NON_ADMIN_USER_MANAGEMENT** and set it to **false**.
7. Restart the app to apply the new configuration.

Step 2: Add LDAP Users

Note: Do not create new users in Elastic Runtime via the Cloud Foundry command line interface (CF CLI), by UAAC, or by using invitations in Apps Manager. This will create a user identity in the internal user store separate from the LDAP user identity. Instead, follow the procedure described below.

There are two ways to add existing LDAP users to your PCF deployment:

- In bulk, using the UAA-LDAP Bulk Import Tool. See [the readme](#) for instructions on installing and using the tool.
- Individually, through the CF CLI, as described below:
 1. Each existing LDAP user must log in to Apps Manager or to the CF CLI using their LDAP credentials. Users will not have access to any org or space until these are granted by an Org Manager.
 2. The PCF Admin must log in to the CF CLI and associate the user with the desired org and space roles. See [Org and App Space Roles](#).

(Advanced Option) Integrate with Enterprise Identity Management System

If your organization uses an Enterprise Identity Management System for centralized provisioning and deprovisioning of users, you can use the following APIs to write a connector to manage users and permissions in Cloud Foundry.

Step 1: Create User

1. Create the user in UAA by running the following command. Replace 'EXAMPLE-USERNAME' with the username of the LDAP user you wish to add.

```
$ curl -H "Content-Type: application/json" \
-k /Users -X POST \
-d '{"userName": "EXAMPLE-USERNAME", \
"emails": [{"value": "EXAMPLE-USERNAME@test.com"}], \
"origin": "ldap", "externalId": "cn=EXAMPLE-USERNAME,ou=Users,dc=test,dc=com"}'
```

2. Use the [Users API](#) to create a User record in the Cloud Controller Database with the existing user's LDAP GUID.

```
$ curl "https://api.YOUR-DOMAIN/v2/users" -d '{ \
"guid": "YOUR-USER-LDAP-GUID" \
}' -X POST \
-H "Authorization: bearer YOUR-BEARER-TOKEN" \
-H "Host: YOUR-HOST-URL" \
-H "Content-Type: application/x-www-form-urlencoded" \
-H "Cookie: "
```

Step 2: Provide User Access to Orgs

Associate the user with the appropriate orgs in your Elastic Runtime deployment, using the [Organizations API](#).

Step 3: Associate User with Space or Org Role

Users can be given Space and Org roles using the following API calls:

- [Associate an Auditor with a Space](#).
- [Associate a Developer with a Space](#).
- [Associate a Manager with a Space](#).
- [Associate an Auditor with a Organization](#).
- [Associate a Manager with a Organization](#).

Managing User Permissions Using the Apps Manager

The Apps Manager uses role-based access control, with each role granting permissions in either an organization or an application space. A user can have one or more roles. The combination of these roles defines the user's overall permissions in the org and within specific app spaces in that org.

An administrator can manage user roles in orgs and spaces using the Apps Manager.

To see which permissions each role grants, see [Organizations, Spaces, Roles, and Permissions](#).

Managing Org Roles

Valid [org roles](#) are Organization Manager and Organization Auditor.

To grant or revoke org roles, follow the steps below.

1. In the Apps Manager navigation on the left, the current org is highlighted. Click the drop-down menu to view other orgs belonging to the account.

2. Use the Apps Manager navigation to select an org.

3. Click the **Members** tab.

MEMBER	ORG MANAGER	ORG AUDITOR
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Bob@example.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ClariceP@example.com	<input type="checkbox"/>	<input type="checkbox"/>
EddieG@example.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>
David.Davidson@example.com	<input type="checkbox"/>	<input type="checkbox"/>
Alice@example.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>

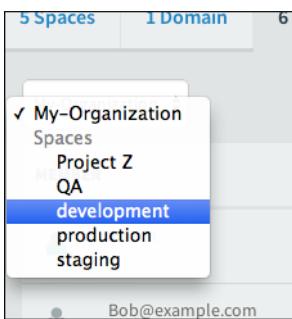
4. The **Members** panel displays all members of the org. Select a checkbox to grant an org role to a user, or deselect a checkbox to revoke a role from a user.

Managing App Space Roles

Valid [app space roles](#) are Space Manager, Space Developer, and Space Auditor.

To grant or revoke app space roles, follow the steps below.

1. In the **Members** tab of an org, click the drop-down menu to view spaces in the org.



2. Use the drop-down menu to select a space.

3. The **Members** panel displays all members of the org. Select a checkbox to grant an app space role to a user, or deselect a checkbox to revoke a role from a user.

MEMBER	SPACE MANAGER	SPACE DEVELOPER	SPACE AUDITOR
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Bob@example.com	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ClariceP@example.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
EddieG@example.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
David.Davidson@example.com	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Alice@example.com	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

APP SPACE ROLES

SPACE MANAGER
Can invite/manage users, enable features for a given space

SPACE DEVELOPER
Can create, delete, manage applications and services, full access to all usage reports and logs

SPACE AUDITOR
Read-only access to all space information, settings, reports, logs

Developer Guide

This guide has instructions for pushing an application to Cloud Foundry and making the application work with any available cloud-based services it uses, such as databases, email, or message servers. The core of this guide is the [Deploy an Application](#) process guide, which provides end-to-end instructions for deploying and running applications on Cloud Foundry, including tips for troubleshooting deployment and application health issues.

Before you can use the instructions in this document, you will need an account on your Cloud Foundry instance.

Preparing Applications for the Cloud

[Considerations for Designing and Running an Application in the Cloud](#)

Deploying and Managing Applications

[Deploy an Application](#)

[Deploy a Large Application](#)

[Creating Domains and Routes](#)

[Domains and Shared Domains](#)

[Deploying with Application Manifests](#)

[Cloud Foundry Environment Variables](#)

[Using Blue-Green Deployment to Reduce Downtime and Risk](#)

[Application Logging in Cloud Foundry](#)

[Troubleshooting Application Deployment and Health](#)

cf Command Line Interface

[Installing the cf Command Line Interface](#)

[Getting Started with the cf CLI](#)

[Using the cf CLI with an HTTP Proxy Server](#)

[Using cf CLI Plugins](#)

[About Starting Applications](#)

[Scaling an Application Using cf scale](#)

Services

[Services Overview](#)

[Managing Service Instances](#)

[User-Provided Service Instances](#)

[Using Log Management Services](#)

[Migrating a Database in Cloud Foundry](#)

Considerations for Designing and Running an Application in the Cloud

Application Design for the Cloud

Applications written in supported application frameworks often run unmodified on Cloud Foundry, if the application design follows a few simple guidelines. Following these guidelines makes an application cloud-friendly, and facilitates deployment to Cloud Foundry and other cloud platforms.

The following guidelines represent best practices for developing modern applications for cloud platforms. For more detailed reading about good app design for the cloud, see [The Twelve-Factor App](#).

Avoid Writing to the Local File System

Applications running on Cloud Foundry should not write files to the local file system for the following reasons:

- **Local file system storage is short-lived.** When an application instance crashes or stops, the resources assigned to that instance are reclaimed by the platform including any local disk changes made since the app started. When the instance is restarted, the application will start with a new disk image. Although your application can write local files while it is running, the files will disappear after the application restarts.
- **Instances of the same application do not share a local file system.** Each application instance runs in its own isolated container. Thus a file written by one instance is not visible to other instances of the same application. If the files are temporary, this should not be a problem. However, if your application needs the data in the files to persist across application restarts, or the data needs to be shared across all running instances of the application, the local file system should not be used. We recommend using a shared data service like a database or blobstore for this purpose.

For example, instead of using the local file system, you can use a Cloud Foundry service such as the MongoDB document database or a relational database like MySQL or Postgres. Another option is to use cloud storage providers such as [Amazon S3](#), [Google Cloud Storage](#), [Dropbox](#), or [Box](#). If your application needs to communicate across different instances of itself, consider a cache like Redis or a messaging-based architecture with RabbitMQ.

Cookies Accessible across Applications

In an environment with shared domains, cookies might be accessible across applications.

Many tracking tools such as Google Analytics and Mixpanel use the highest available domain to set their cookies. For an application using a shared domain such as `example.com`, a cookie set to use the highest domain has a `Domain` attribute of `.example.com` in its HTTP response header. For example, an application at `my-app.example.com` might be able to access the cookies for an application at `your-app.example.com`.

Consider whether you want your applications or tools that use cookies to set and store the cookies at the highest available domain.

HTTP Sessions Not Persisted or Replicated

Cloud Foundry supports session affinity or sticky sessions for incoming HTTP requests to applications if a `jsessionid` cookie is used. If multiple instances of an application are running on Cloud Foundry, all requests from a given client will be routed to the same application instance. This allows application containers and frameworks to store session data specific to each user session.

Cloud Foundry does not persist or replicate HTTP session data. If an instance of an application crashes or is stopped, any data stored for HTTP sessions that were sticky to that instance are lost. When a user session that was sticky to a crashed or stopped instance makes another HTTP request, the request is routed to another instance of the application.

Session data that must be available after an application crashes or stops, or that needs to be shared by all instances of an application, should be stored in a Cloud Foundry service. There are several open source projects that share sessions in a data service.

Port Limitations

Applications running on Elastic Runtime receive requests through the URLs configured for the application. HTTP requests

arrive on ports 80 and 443. Additionally, Elastic Runtime requires a channel for TCP/WebSocket traffic. By default, this is port 443.

Some production load balancers require separate ports for HTTP and TCP/WebSocket traffic. If your load balancer has this requirement, you must do the following in order to access logs from your app with the `cf logs APP_NAME` command:

- Configure a different port for TCP/WebSocket traffic. As of PCF 1.4, you can configure your ports from the Elastic Runtime product tile in Ops Manager.
- Set the value of the `port` sub-key of the `logger_endpoint` key in your manifest.
- Configure your HAProxy or load balancer to receive TCP traffic on the port that you specified.

Ignore Unnecessary Files When Pushing

By default, when you push an application, all files in the application's project directory tree are uploaded to your Cloud Foundry instance, except version control or configuration files with the following file extensions:

- `.cignore`
- `_darcs`
- `.DS_Store`
- `.git`
- `.gitignore`
- `.hg`
- `/manifest.yml`
- `.svn`

If the application directory contains other files (such as `temp` or `log` files), or complete subdirectories that are not required to build and run your application, the best practice is to exclude them using a `.cignore` file. (`.cignore` is similar to git's `.gitignore`, which allows you to exclude files and directories from git tracking.) Especially with a large application, uploading unnecessary files slows down application deployment.

Specify the files or file types you wish to exclude from upload in a text file, named `.cignore`, in the root of your application directory structure. For example, these lines exclude the "tmp" and "log" directories.

```
tmp/  
log/
```

The file types you will want to exclude vary, based on the application frameworks you use. The `.gitignore` templates for common frameworks, available at <https://github.com/github/gitignore>, are a useful starting point.

Run Multiple Instances to Increase Availability

When a DEA is upgraded, the applications running on it are shut down gracefully, then restarted on another DEA. To avoid the risk of an application being unavailable during a Cloud Foundry upgrade processes, you should run more than one instance of the application.

Using Buildpacks

A buildpack consists of bundles of detection and configuration scripts that provide framework and runtime support for your applications. When you deploy an application that needs a buildpack, Cloud Foundry installs the buildpack on the Droplet Execution Agent (DEA) where the application runs.

For more information, see the [Buildpacks](#) topic.

Deploy an Application

This page assumes that you are using cf CLI v6.

 **Note:** See the [buildpacks](#) documentation for complete deployment guides specific to your application language or framework, such as the [Getting Started Deploying Ruby on Rails Apps](#) guide.

Overview of Deployment Process

You deploy an application to Cloud Foundry by running a `push` command from a Cloud Foundry command line interface (CLI). Refer to the [Install cf CLI v6](#) topic for more information. Between the time that you run `push` and the time that the application is available, Cloud Foundry performs the following tasks:

- Uploads and stores application files
- Examines and stores application metadata
- Creates a “droplet” (the Cloud Foundry unit of execution) for the application
- Selects an appropriate droplet execution agent (DEA) to run the droplet
- Starts the application

An application that uses services, such as a database, messaging, or email server, is not fully functional until you provision the service and, if required, bind the service to the application. For more information about services, see the [Services Overview](#) topic.

Step 1: Prepare to Deploy

Before you deploy your application to Cloud Foundry, make sure that:

- Your application is *cloud-ready*. Cloud Foundry behaviors related to file storage, HTTP sessions, and port usage may require modifications to your application.
- All required application resources are uploaded. For example, you may need to include a database driver.
- Extraneous files and artifacts are excluded from upload. You should explicitly exclude extraneous files that reside within your application directory structure, particularly if your application is large.
- An instance of every service that your application needs has been created.
- Your Cloud Foundry instance supports the type of application you are going to deploy, or you have the URL of an externally available buildpack that can stage the application.

For help preparing to deploy your application, see:

- [Considerations for Designing and Running an Application in the Cloud](#)
- [Buildpacks](#)

Step 2: Know Your Credentials and Target

Before you can push your application to Cloud Foundry you need to know:

- The API endpoint for your Cloud Foundry instance. Also known as the target URL, this is [the URL of the Cloud Controller in your Elastic Runtime instance ↗](#).
- Your username and password for your Cloud Foundry instance.
- The organization and space where you want to deploy your application. A Cloud Foundry workspace is organized into organizations, and within them, spaces. As a Cloud Foundry user, you have access to one or more organizations and spaces.

Step 3: (Optional) Configure Domains

Cloud Foundry directs requests to an application using a route, which is a URL made up of a host and a domain.

- The name of an application is the default host for that application.
- Every application is deployed to an application space that belongs to a domain. Every Cloud Foundry instance has a default domain defined. You can specify a non-default, or custom, domain when deploying, provided that the domain is registered and is mapped to the organization which contains the target application space.

For more information about domains, see [Creating Domains and Routes](#).

Step 4: Determine Deployment Options

Before you deploy, you need to decide on the following:

- **Name:** You can use any series of alpha-numeric characters, without spaces, as the name of your application.
- **Instances:** Generally speaking, the more instances you run, the less downtime your application will experience. If your application is still in development, running a single instance can simplify troubleshooting. For any production application, we recommend a minimum of two instances.
- **Memory Limit:** The maximum amount of memory that each instance of your application can consume. If an instance exceeds this limit, Cloud Foundry restarts the instance.

Note: Initially, Cloud Foundry immediately restarts any instances that exceed the memory limit. If an instance repeatedly exceeds the memory limit in a short period of time, Cloud Foundry delays restarting the instance.

- **Start Command:** This is the command that Cloud Foundry uses to start each instance of your application. This start command varies by application framework.
- **Subdomain (host) and Domain:** The route, which is the combination of subdomain and domain, must be globally unique. This is true whether you specify a portion of the route or allow Cloud Foundry to use defaults.
- **Services:** Applications can bind to services such as databases, messaging, and key-value stores. Applications are deployed into application spaces. An application can only bind to a service that has an existing instance in the target application space.

Define Deployment Options

You can define deployment options on the command line, in a manifest file, or both together. See [Deploying with Application Manifests](#) to learn how application settings change from push to push, and how command-line options, manifests, and commands like `cf scale` interact.

When you deploy an application while it is running, Cloud Foundry stops all instances of that application and then deploys. Users who try to run the application get a “404 not found” message while `cf push` runs. Stopping all instances is necessary to prevent two versions of your code from running at the same time. A worst-case example would be deploying an update that involved a database schema migration, because instances running the old code would not work and users could lose data.

Cloud Foundry uploads all application files except version control files with file extensions `.svn`, `.git`, and `.darcs`. To exclude other files from upload, specify them in a `.cfignore` file in the directory where you run the push command. This technique is similar to using a `.gitignore` file. For more information, see the [Ignore Unnecessary Files When Pushing](#) section of the [Considerations for Designing and Running an Application in the Cloud](#) topic.

For more information about the manifest file, see the [Deploying with Application Manifests](#) topic.

Set Environment Variables

Environment variables are key-value pairs defined at the operating system level. These key-value pairs provide a way to configure the applications running on a system. For example, any application can access the **LANG** environment variable to determine which language to use for error messages and instructions, collating sequences, and date formats.

To set an environment variable, add a bash script to the `.profile.d` directory in the source code of your application. Cloud Foundry runs this script when deploying each instance of your application.

Note: You must give your shell scripts a `.sh` extension.

Example `.profile.d/setenv.sh` script:

```
# Set the Java environment path for your applications
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0
export PATH=$PATH:$JAVA_HOME/bin

# Set the default LANG for your applications
export LANG=en_US.UTF-8
```

Step 5: Push the Application

Run the following command to deploy an application without a manifest:

```
cf push APP-NAME
```

If you provide the application name in a manifest, you can reduce the command to `cf push`. See [Deploying with Application Manifests](#).

Since all you have provided is the name of your application, `cf push` sets the number of instances, amount of memory, and other attributes of your application to the default values. You can also use command-line options to specify these and additional attributes.

The following transcript illustrates how Cloud Foundry assigns default values to application when given a `cf push` command.

Note: When deploying your own apps, avoid generic names like `my-app`. Cloud Foundry uses the app name to compose the route to the app, and deployment fails unless the app has a globally unique route.

```
$ cf push my-app
Creating app my-app in org example-org / space development as a.user@example.com...
OK

Creating route my-app.example.com...
OK

Binding my-app.example.com to my-app...
OK

Uploading my-app...
Uploading app: 560.1K, 9 files
OK

Starting app my-app in org example-org / space development as a.user@example.com...
----> Downloaded app package (552K)
OK
----> Using Ruby version: ruby-1.9.3
----> Installing dependencies using Bundler version 1.3.2
      Running: bundle install --without development:test --path vendor/bundle --binstubs vendor/bundle/bin --deployment
      Installing rack (1.5.1)
      Installing rack-protection (1.3.2)
      Installing tilt (1.3.3)
      Installing sinatra (1.3.4)
      Using bundler (1.3.2)
      Updating files in vendor/cache
      Your bundle is complete! It was installed into ./vendor/bundle
      Cleaning up the bundler cache.
----> Uploading droplet (23M)

1 of 1 instances running

App started

Showing health and status for app my-app in org example-org / space development as a.user@example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: my-app.example.com

      state      since          cpu    memory      disk
#0  running   2014-01-24 05:07:18 PM  0.0%  18.5M of 1G  52.5M of 1G
```

Step 6: (Optional) Configure Service Connections

If you bound a service to the application that you deployed, you might need to configure your application with the service URL and credentials. For more information, see the specific documentation for your application framework:

- [Ruby](#)
- [Node.js](#)
- [Spring](#)
- [Grails](#)

Step 7: Troubleshoot Deployment Problems

If your application does not start on Cloud Foundry, first ensure that your application can run locally.

You can troubleshoot your application in the cloud using the cf CLI. See [Troubleshoot Application Deployment and Health](#).

Migrating Apps to Diego

This topic contains instructions for migrating apps originally deployed with Pivotal Cloud Foundry (PCF) versions prior to 1.6 to Diego, the back-end architecture for Elastic Runtime that is new to PCF 1.6.

The procedure described here does not include upgrading PCF to 1.6. See [Upgrading Elastic Runtime]. Your PCF deployment must be running Elastic Runtime v1.6, which includes Diego, before you can migrate your apps.

There are two options for migrating your apps to Diego, the first is simpler but results in a brief downtime interval. The second option avoids downtime by employing Blue-Green deployment.

Simple Migration of Apps (with Downtime)

Step 1: Install Cloud Foundry Command Line Interface (CF CLI) Plugin

Use the CF CLI to install the Diego-Enabler plugin.

```
$ cf add-plugin-repo CF-Community http://plugins.cloudfoundry.org/
$ cf install-plugin Diego-Enabler -r CF-Community
```

Step 2: Enable Diego

Run `cf enable-diego APP-NAME` to enable diego in your app manifest.

```
$ cf enable-diego example-app
Diego support for example-app is set to true
```

Step 3: Push Your App

Run `cf push` to push your newly Diego-enabled app.

```
$ cf push
Using manifest manifest.yml...
Updating app...
```

Migrating Apps with Blue-Green Deployment

- Push your application to PCF with a new name and the route for your application.

```
$ cf push new-app
```

Note: This new application will run on Diego, which is the default on Diego. Make sure to use the same version of your application that is currently deployed so that static assets like CSS and Javascript that may be served from your application are consistent across both versions.

- Once you have confirmed that the new application deployed to Diego is operational, map the route for your application running on the DEAs to your application running on Diego. This mapping will result in a second route with half of your traffic going to the DEA's app and half to Diego's.

```
$ cf map-route new-app my-domain.com -n www
```

- Unmap the route from the application running on the DEAs. This action will send all of the traffic to the application that is running on Diego.

```
$ cf unmap-route old-app my-domain.com -n www
```

- Stop the application that is running on the DEAs. After this point, all of your traffic will be going to your app running

on Diego, and the blue-green migration will be complete. Again, verify that your application is working properly.

```
$ cf stop old-app
```

5. Delete the old application.

```
$ cf delete old-app
```

Application Rollback Procedure

If you encounter any issues, you can rollback to your application on the DEAs by starting it and reversing the route map and unmap commands.

1. Start the old application.

```
$ cf start old-app
```

2. Map the old application to your domain.

```
$ cf map-route old-app my-domain.com -n www
```

3. Unmap the new application to your domain.

```
$ cf unmap-route new-app my-domain.com -n www
```

Deploying a Large Application

This topic describes constraints and recommended settings for deploying applications between 750 MB and 1 GB to Elastic Runtime.

Deployment Considerations and Limitations

Elastic Runtime supports application uploads up to 1 GB.

The deployment process involves uploading, staging, and starting the app. Your app must successfully complete each of these phases within the time limits that your administrator establishes for each phase. The default time limits for the phases are as follows:

- Upload: 15 minutes
- Stage: 15 minutes
- Start: 60 seconds

 **Note:** Your administrator can change these defaults. Check with your administrator for the actual time limits set for app deployment.

To deploy large apps to Elastic Runtime, ensure the following:

- Your network connection speed is sufficient to upload your app within the 15 minute limit. Pivotal recommends a minimum speed of 874 KB/s.

 **Note:** Elastic Runtime provides an authorization token that is valid for a minimum of 20 minutes.

- The total size of the files to upload for your app does not exceed 1 GB.
- You allocate enough memory for all instances of your app. Use either the `-m` flag with `cf push` or set an app memory value in your `manifest.yml` file.
- You allocate enough disk space for all instances of your app. Use either the `-k` flag with `cf push` or set a disk space allocation value in your `manifest.yml` file.
- If you use an app manifest file, `manifest.yml`, be sure to specify adequate values for your app for attributes such as app memory, app start timeout, and disk space allocation.
For more information about using manifests, refer to the [Deploying with Application Manifests](#) topic.
- You push only the files that are necessary for your application.
To meet this requirement, push only the directory for your application, and remove unneeded files or use the `.cfignore` file to [specify excluded files](#).
- You configure Cloud Foundry Command Line Interface (cf CLI) staging, startup, and timeout settings to override settings in the manifest, as necessary.
 - `CF_STAGING_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to stage after Cloud Foundry successfully uploads and packages the app. Value set in minutes.
 - `CF_STARTUP_TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to start. Value set in minutes.
 - `cf push -t TIMEOUT`: Controls the maximum time that the cf CLI waits for an app to start. When you use this flag, the cf CLI ignores any app start timeout value set in the manifest or in the `CF_STARTUP_TIMEOUT` environment variable.
Value set in seconds.

For more information about using the cf CLI to deploy apps, refer to the [Push section](#) of the [Getting Started with the cf CLI](#) topic.

 **Note:** Changing the timeout setting for the cf CLI does not change the timeout limit for Cloud Foundry server-side jobs such as staging or starting applications. Server-side timeouts must be changed in the manifest.

Because of the differences between the Cloud Foundry and cf CLI timeout values, your app might successfully start even though the cf CLI reports `App failed`. Run `cf apps APP_NAME` to review the actual status of your app.

Default Settings and Limitations Summary Table

This table provides summary information of constraints and default settings to consider when you deploy a large app to Elastic Runtime.

Setting	Note
App Package Size	Maximum: 1 GB
Authorization Token Grace Period	Default: 20 minutes, minimum
<code>CF_STAGING_TIMEOUT</code>	cf CLI environment variable Default: 15 minutes
<code>CF_STARTUP_TIMEOUT</code>	cf CLI environment variable Default: 5 minutes
<code>cf push -t TIMEOUT</code>	App start timeout maximum Default: 60 seconds
Disk Space Allocation	Default: 1024 MB
Internet Connection Speed	Recommended Minimum: 874 KB/s

Creating Domains and Routes

This page assumes you are using cf CLI v6.

This topic describes how to create and use domains and routes in Cloud Foundry.

Note: The term **domain** in this topic differs from its common use and is specific to Cloud Foundry.

In Cloud Foundry, you create a domain according to the guidelines in this topic. Cloud Foundry uses these domains in routes to direct requests to applications. A **custom domain** refers to a third party registered domain that you own and can use with your applications in Cloud Foundry. For more information about using custom domains shared across all orgs in your Cloud Foundry account, refer to the [Domains and Shared Domains](#) topic.

Domains are associated with orgs and are not directly bound to applications. Domains can be **shared** or **private**. Shared domains are registered to multiple orgs while private domains, or **owned** domains, are registered to one org. A Cloud Foundry instance defines a default shared domain that your application uses unless you specify a different domain.

A **route** is a path used to access an application online. Each route is directly bound to one or more applications in Cloud Foundry. A route is a URL composed of a domain and an optional host as a prefix. For example, “myappname” is the host and “example.com” is the domain in the route “myappname.example.com.” Here “example.com” could either be a custom domain of your own, or a domain that you have created through Cloud Foundry.

Using a Root Domain

A **root domain** or a **zone apex domain** such as “example.com” consists of a domain without a host. To access an application deployed to a root domain, you must use either custom record types or **subdomain redirection**.

The following section contains instructions for mapping a root domain to your application by configuring ALIAS or ANAME custom record types or by using subdomain redirection.

Configuring an ALIAS or ANAME Record with Your DNS Provider

If your DNS provider supports using an ALIAS or ANAME record, configure your root domain with your DNS provider.

ALIAS and ANAME records follow the pattern shown below:

Record	Name	Target	Note
ALIAS or ANAME	empty or @	foo.example.com.	Refer to your DNS provider documentation to determine whether to use an empty or @ value for the Name entry.

If your DNS provider does not support ANAME or ALIAS records, use subdomain redirection, a technique also known as **domain forwarding**, to add a root domain. If you use this method, note that SSL requests to the root domain result in an error because the SSL certificate generally only matches the subdomain.

Configure the redirect from the root domain to the target “www” subdomain, and configure the “www” subdomain as a CNAME record reference to the target app URL. The table below illustrates this pattern.

Record	Name	Target	Note
URL or Forward	example.org	www.example.org	This method results in a 301 permanent redirect to the subdomain you configure.
CNAME	www	foo.cfapps.io	

Configuring Your Application to Use a Root Domain

The following section contains instructions for creating a domain and using that domain as the route for your application. Since the route is a root domain and does not have a host, both the domain and route in this section are represented with “example.org.”

1. Create the domain in Cloud Foundry and associate it with an organization.
The command below defines the domain “example.org” in the “test-org” organization:

```
$ cf create-domain test-org example.org
```

2. Map the route to your app. The command you use depends on whether the app uses a shared domain.

If your route is not using a shared domain, use the `cf map-route` command as the example shows:

```
$ cf map-route myapp example.org
```

If your route is using a shared domain, use the `-n HOSTNAME` parameter with the `cf map-route` command to specify a unique hostname for the route. The command below does two things:

- Combines the existing route “myapp.example.org” with the hostname “app” to produce the new route “app.myapp.example.org”
- Maps the new route to the application named “myapp”

```
$ cf map-route myapp example.org -n app
```

Creating a Parent Domain

In the multi-level domain “myapp.example.com,” “example.com” is the parent domain of subdomain “myapp.”

Configuring a CNAME Record with Your DNS Provider

Follow the [Configuring a CNAME Record with Your DNS Provider](#) procedure in the Using a Private or Shared Subdomain section.

Configuring Your Application to Use a Private or Shared Parent Domain

Parent domains must meet the following requirements:

- Private Parent Domains:** You can only create a private domain that is parent to a private subdomain.
- Shared Parent Domains:** You can create a shared domain that is parent to either a shared or a private subdomain.

Sharing a Private Domain Between One or More Orgs

As an Org Manager, you can specify a private domain to be used by more than one org. You must have Org Manager permissions for all associated orgs.

- Share a private domain with an org with the `cf share-private-domain` command.

```
$ cf share-private-domain test-org example.com
```

- Unshare a private domain with an org with the `cf share-private-domain` command.

```
$ cf unshare-private-domain test-org example.com
```

Using a Private or Shared Subdomain

Domains in Cloud Foundry can be multi-level and contain subdomains like the “myapp” in the domain “myapp.example.com.” In this case, “example.com” is the parent domain of the subdomain “myapp.” In the domain “test.myapp.example.org,” “test” is a subdomain with the parent domain “myapp.example.org.”

Configuring a CNAME Record with Your DNS Provider

Configure your domain using a CNAME record.

CNAME records follow the pattern illustrated below:

Record	Name	Target	Note
--------	------	--------	------

CNAME	test	foo.example.com.	Refer to your DNS provider documentation to determine whether the trailing <code>.</code> is required.
Wildcard CNAME	sample	*.example.com.	You can use the wildcard in the CNAME record to point all of your subdomains to your parent domain. Each separately configured subdomain has priority over the wildcard configuration.

Configuring Your Application to Use a Private or Shared Subdomain

You can create a new domain by mapping subdomains to an existing domain. Each added subdomain is the child to its following parent domain. The subdomains must meet the following requirements:

- **Private Subdomains:**

- You can map a private subdomain to a private parent domain only if the domains belong to the same org.
- You can map a private subdomain to a shared parent domain.

- **Shared Subdomains:**

- You can only map a shared subdomain to a shared parent domain.
- You cannot map a shared subdomain to a private parent domain.

To configure your application to use a domain with a subdomain, create the domain and map the route to your application as follows:

1. Create the domain in Cloud Foundry and associate it with an org.

The command below defines the domain “myapp.example.org” in the “test-org” organization:

```
$ cf create-domain test-org myapp.example.org
```

2. Map the route to your app with an optional hostname, as the example shows:

```
$ cf map-route myapp myapp.example.org -n test
```

Managing Your Domains

View Domains for an Org

You can view available domains for the targeted organization using the `cf domains` command.

In this example, there are two available domains: a system-wide default “example.com” domain and the custom “example.org” domain.

```
$ cf domains
Getting domains in org my-org as user@example.org... OK

name      status
example.com  shared
example.org   owned
```

Assign Domains and Hosts

You can assign a domain and host to your application using either the command line or the application manifest.

Assign Using the cf CLI

When you run `cf push`, you can optionally assign a domain and host to the application.

- **Domain:** Use the `-d` flag to specify one of the domains available to the targeted org.
- **Host:** Use the `-n` flag to provide a string for the host.

The route Cloud Foundry creates for the application as a result of the following command is `myapp.example.org`.

```
$ cf push myapp -d example.org -n myapp
```

Assign Using the Manifest

When you create or edit the manifest for an application, you can use the `host` and `domain` attributes to define the host and domain components of the application route, respectively. For more information, see [Application Manifests](#).

Delete a Domain

You can delete a domain from Cloud Foundry with the `cf delete-domain` command:

```
$ cf delete-domain example.org
```

Managing Your Routes

List Routes

You can list routes for the current space with `cf routes` command. Notice that the host is separate from the domain segment.

For example:

```
$ cf routes
Getting routes as user@example.org ...

host           domain     apps
myapp          example.org myapp1
myapp2
1test          example.com 1test
sinatra-hello-world example.com sinatra-hello-world
sinatra-to-do   example.com sinatra-to-do
```

Create a Route

Create a route and associate it with a space for later use with the `cf create-route` command. You can use the optional `-n HOSTNAME` parameter to specify a unique hostname for each route that uses the same domain.

For example, this command creates the “myapp.example.org” route in the “development” space:

```
$ cf create-route development example.org -n myapp
```

Assign or Change a Route

Assign or change the route for a particular application with the `cf map-route` command. Specifying the host is optional. If the route does not already exist, this command creates it and then maps it.

For example, the following command maps the route “myapp.example.org” to the “myapp” application:

```
$ cf map-route myapp example.org -n myapp
```

An application bound to a wildcard route is a “fallback” app that a user accesses if no other application routes match what the user typed. For example, if a user types “myap.example.org” in an attempt to access the application bound to “myapp.example.org”, the user accesses the application bound to “*.example.org.”

The command below maps the route “*.example.org” to the “myfallbackapp” application:

```
$ cf map-route myfallbackapp example.org -n *
```

If the application is running when you map a route, restart the application. The new route is not active until the application is restarted.

Sharing a Route between Two Apps

Cloud Foundry allows multiple apps, or versions of the same app, to be mapped to the same route. This enables Blue-Green deployment, the deployment strategy that Pivotal recommends. See [Using Blue-Green Deployment to Reduce Downtime and Risk](#).

Multiple apps should be mapped to the same route only during a Blue-Green deployment. In other situations, routing multiple apps to the same route can cause file load errors and other buggy behavior, as incoming requests are randomly routed to one of the apps on the shared route.

See [Routing Conflicts](#) for more information about troubleshooting this problem.

Remove a Route

You can remove a route from an app using the `cf unmap-route` command. Unmapping a route leaves the route available in the targeted space for later use.

```
$ cf unmap-route myapp example.org -n myapp
```

You can remove a route from a space using the `cf delete-route` command:

```
$ cf delete-route example.org -n myapp
```

The `-n` parameter to specify the host is optional in each of the above commands.

Domains and Shared Domains

This page assumes you are using cf CLI v6.

This page has information about domains and shared domains. For more information, see [Creating Domains and Routes](#). For admin-specific commands, refer to the [Managing Domains and Routes](#) topic.

A domain is an IP resource represented by a domain name like “example.com”. A domain is not directly bound to an application, but acts as part of the route to an application.

When you deploy an application, you supply a hostname. This hostname is prepended to the domain to create the full URL, or route, to the application.

Example: If I use the hostname “demo-time” for my application, and my domain is “example.com”, the route to the application is the full URL “demo-time.example.com”.

A Cloud Foundry instance defines a default shared domain. Unless you specify a different domain, routes to your application are created using this shared domain.

Cloud Foundry also supports custom domains.

Create a Custom Domain

If you want to use a registered domain of your own, use the `create-domain` command to create and associate it with a single organization in your account.

The example uses the `create-domain` command to create the custom domain `example.com` in the “test” organization:

```
$ cf create-domain test example.com
```

View Domains

You can see available domains for the targeted organization using the `cf domains` command. In this example, there are two available domains: a system-wide default `example.com` domain and the custom `example.org` domain.

```
$ cf domains
Getting domains in org my-org as user@example.org@example.com... OK

name      status
example.com  shared
example.org   owned
```

Delete a Domain

Use the `cf delete-domain` command to delete a domain:

```
$ cf delete-domain example.com
```

Deploying with Application Manifests

This page assumes that you are using cf CLI v6.

Application manifests tell `cf push` what to do with applications. This includes everything from how many instances to create and how much memory to allocate to what services applications should use.

A manifest can help you automate deployment, especially of multiple applications at once.

How cf push Finds the Manifest

By default, the `cf push` command deploys an application using a `manifest.yml` file in the current working directory. Use the `-f` option to provide a non-standard manifest location or filename.

To use `cf push` without an `-f` option, you must provide a manifest named `manifest.yml` in the current working directory.

```
$ cf push
Using manifest file /path_to_working_directory/manifest.yml
```

With the `-f` option, a path with no filename means that the filename must be `manifest.yml`.

```
$ cf push -f ./some_directory/some_other_directory/
Using manifest file /path_to_working_directory/some_directory/some_other_directory/manifest.yml
```

If the manifest is named something other than `manifest.yml`, use the `-f` option. You must include both the path and the filename with the `-f` option.

```
$ cf push -f ./some_directory/some_other_directory/alternate_manifest.yml
Using manifest file /path_to_working_directory/some_directory/some_other_directory/alternate_manifest.yml
```

Example Manifest

You can deploy applications without ever using a manifest. The benefits manifests may provide include consistency and reproducibility. When you want applications to be portable between different clouds, manifests may prove especially useful.

Manifests are written in YAML. The manifest below illustrates some YAML conventions, as follows:

- The manifest begins with three dashes.
- The `applications` block begins with a heading followed by a colon.
- The application `name` is preceded by a single dash and one space.
- Subsequent lines in the block are indented two spaces to align with `name`.

```
---
applications:
- name: nifty-gui
  memory: 512M
  host: nifty
```

A minimal manifest requires only an application `name`. To create a valid minimal manifest, remove the `memory` and `host` properties from this example.

Place your `manifest.yml` in the directory that contains the application files that you want to push and run `cf push`. `cf push` searches the current directory for a manifest unless you specify another path with the `-p` option.

Always Provide an Application Name to cf push

`cf push` requires an application name, which you provide either in a manifest or at the command line.

As described in [How cf push Finds the Manifest](#) above, the command `cf push` works when the `manifest.yml` file is in the current working directory.

If you do not use a manifest, the minimal push command looks like this:

```
$ cf push my-app
```

Note: When you provide an application name at the command line, `cf push` uses that application name whether or not there is a different application name in the manifest. If the manifest describes multiple applications, you can push a single application by providing its name at the command line; the cf CLI does not push the others. Use these behaviors for testing.

How cf push Finds the Application

By default, `cf push` recursively pushes the contents of the current working directory. Alternatively, you can provide a path using either a manifest or a command line option.

- If the path is to a directory, `cf push` recursively pushes the contents of that directory instead of the current working directory.
- If the path is to a file, `cf push` pushes only that file.

Note: If you want to push more than a single file, but not the entire contents of a directory, consider using a `.cfignore` file to tell `cf push` what to exclude.

Precedence Between Manifests, Command Line Options, and Most Recent Values

When you push an application for the first time, Cloud Foundry applies default values to any attributes that you do not set in a manifest or `cf push` command line options.

- For example, `cf push my-app` with no manifest might deploy one instance of the app with one gigabyte of memory. In this case the default values for instances and memory are “1” and “1G”, respectively.

Between one push and another, attribute values can change in other ways.

- For example, the `cf scale` command changes the number of instances.

The attribute values on the server at any one time represent the cumulative result of all settings applied up to that point: defaults, attributes in the manifest, `cf push` command line options, and commands like `cf scale`. There is no special name for this resulting set of values on the server. You can think of them as the most recent values.

`cf push` follows rules of precedence when setting attribute values:

- Manifests override most recent values, including defaults.
- Command line options override manifests.

In general, you can think of manifests as just another input to `cf push`, to be combined with command line options and most recent values.

Optional Attributes

This section explains how to describe optional application attributes in manifests. Each of these attributes can also be specified by a command line option. Command line options override the manifest.

The buildpack attribute

If your application requires a custom buildpack, you can use the `buildpack` attribute to specify its URL or name:

```
---  
...  
buildpack: buildpack_URL
```

Note: The `cf buildpacks` command lists the buildpacks that you can refer to by name in a manifest or a command line option.

The command line option that overrides this attribute is `-b`.

The command attribute

Some languages and frameworks require that you provide a custom command to start an application. Refer to the [buildpack](#) documentation to determine if you need to provide a custom start command.

You can provide the custom start command in your application manifest or on the command line.

To specify the custom start command in your application manifest, add it in the `command: START-COMMAND` format as the following example shows:

```
---  
...  
command: bundle exec rake VERBOSE=true
```

On the command line, use the `-c` option to specify the custom start command as the following example shows:

```
$ cf push my-app -c "bundle exec rake VERBOSE=true"
```

Note: The `-c` option with a value of 'null' forces `cf push` to use the buildpack start command. See [About Starting Applications](#) for more information.

The disk quota attribute

Use the `disk_quota` attribute to allocate the disk space for your app instance. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case.

```
---  
...  
disk_quota: 1024M
```

The command line option that overrides this attribute is `-k`.

The domain attribute

Every `cf push` deploys applications to one particular Cloud Foundry instance. Every Cloud Foundry instance may have a shared domain set by an admin. Unless you specify a domain, Cloud Foundry incorporates that shared domain in the route to your application.

You can use the `domain` attribute when you want your application to be served from a domain other than the default shared domain.

```
---  
...  
domain: unique-example.com
```

The command line option that overrides this attribute is `-d`.

The domains attribute

Use the `domains` attribute to provide multiple domains. If you define both `domain` and `domains` attributes, Cloud Foundry creates routes for domains defined in both of these fields.

```
---  
...  
domains:  
- domain-example1.com  
- domain-example2.org
```

The command line option that overrides this attribute is `-d`.

The stack attribute

Use the `stack` attribute to specify which stack to deploy your application to.

To see a list of available stacks, run `cf stacks` from the cf cli.

```
---  
...  
stack: cflinuxfs2
```

The command line option that overrides this attribute is `-s`.

The instances attribute

Use the `instances` attribute to specify the number of app instances that you want to start upon push:

```
---  
...  
instances: 2
```

We recommend that you run at least two instances of any apps for which fault tolerance matters.

The command line option that overrides this attribute is `-i`.

The memory attribute

Use the `memory` attribute to specify the memory limit for all instances of an app. This attribute requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case. For example:

```
---  
...  
memory: 1024M
```

The default memory limit is 1G. You might want to specify a smaller limit to conserve quota space if you know that your app instances do not require 1G of memory.

The command line option that overrides this attribute is `-m`.

The host attribute

Use the `host` attribute to provide a hostname, or subdomain, in the form of a string. This segment of a route helps to ensure that the route is unique. If you do not provide a hostname, the URL for the app takes the form of `APP-NAME.DOMAIN`.

```
---  
...  
host: my-app
```

The command line option that overrides this attribute is `-n`.

The hosts attribute

Use the `hosts` attribute to provide multiple hostnames, or subdomains. Each hostname generates a unique route for the app. `hosts` can be used in conjunction with `host`. If you define both attributes, Cloud Foundry creates routes for hostnames defined in both `host` and `hosts`.

```
...  
...  
hosts:  
- app_host1  
- app_host2
```

The command line option that overrides this attribute is `-n`.

The no-hostname attribute

By default, if you do not provide a hostname, the URL for the app takes the form of `APP-NAME.DOMAIN`. If you want to override this and map the root domain to this app then you can set no-hostname as true.

```
...  
...  
no-hostname: true
```

The command line option that corresponds to this attribute is `--no-hostname`.

The random-route attribute

Use the `random-route` attribute to create a URL that includes the app name and random words. Use this attribute to avoid URL collision when pushing the same app to multiple spaces, or to avoid managing app URLs.

The command line option that corresponds to this attribute is `--random-route`.

```
...  
...  
random-route: true
```

The path attribute

You can use the `path` attribute to tell Cloud Foundry where to find your application. This is generally not necessary when you run `cf push` from the directory where an application is located.

```
...  
...  
path: path_to_application_bits
```

The command line option that overrides this attribute is `-p`.

The timeout attribute

Use the `timeout` attribute to give your application more time to start, up to 180 seconds. The default timeout is 60 seconds. For example:

```
...  
...  
timeout: 80
```

If the app has not started by the timeout limit that you set, Cloud Foundry stops trying to start the app. You might want to increase the timeout length to ensure that very large apps have sufficient time to start.

The command line option that overrides this attribute is `-t`.

The no-route attribute

By default, `cf push` assigns a route to every application. But some applications process data while running in the background, and should not be assigned routes.

You can use the `no-route` attribute with a value of `true` to prevent a route from being created for your application.

```
---  
...  
no-route: true
```

The command line option that corresponds to this attribute is `--no-route`.

If you find that an application which should not have a route does have one:

1. Remove the route using the `cf unmap-route` command.
2. Push the app again with the `no-route: true` attribute in the manifest or the `--no-route` command line option.

For more about these situations, see [Describing Multiple Applications with One Manifest](#) below.

Environment variables

The `env` block consists of a heading, then one or more environment variable/value pairs.

For example:

```
---  
...  
env:  
  RAILS_ENV: production  
  RACK_ENV: production
```

`cf push` deploys the application to a container on the server. The variables belong to the container environment.

While the application is running, Cloud Foundry allows you to operate on environment variables.

- View all variables: `cf env my-app`
- Set an individual variable: `cf set-env my-variable_name my-variable_value`
- Unset an individual variable: `cf unset-env my-variable_name my-variable_value`

Environment variables interact with manifests in the following ways:

- When you deploy an application for the first time, Cloud Foundry reads the variables described in the environment block of the manifest, and adds them to the environment of the container where the application is deployed.
- When you stop and then restart an application, its environment variables persist.

Services

Applications can bind to services such as databases, messaging, and key-value stores.

Applications are deployed into App Spaces. An application can only bind to services instances that exist in the target App Space before the application is deployed.

The `services` block consists of a heading, then one or more service instance names.

Whoever creates the service chooses the service instance names. These names can convey logical information, as in `backend_queue`, describe the nature of the service, as in `mysql_5.x`, or do neither, as in the example below.

```
---  
...  
services:  
  - instance_ABC  
  - instance_XYZ
```

Binding to a service instance is a special case of setting an environment variable, namely `VCAP_SERVICES`. See the [Bind a Service](#) section of the [Binding Applications to Service Instances](#) topic.

Describing Multiple Applications with One Manifest

You can deploy multiple applications with one `cf push` command by describing them in a single manifest. In doing so, you need to pay extra attention to directory structure and path lines in the manifest.

Suppose you want to deploy two applications called respectively spark and flame, and you want Cloud Foundry to create and start spark before flame. You accomplish this by listing spark first in the manifest.

In this situation there are two sets of bits that you want to push. Let's say that they are `spark.rb` in the spark directory and `flame.rb` in the flame directory. One level up, the `fireplace` directory contains the spark and the flame directories along with the `manifest.yml` file. Your plan is to run the cf CLI from the `fireplace` directory, where you know it can find the manifest.

Now that you have changed the directory structure and manifest location, `cf push` can no longer find your applications by its default behavior of looking in the current working directory. How can you ensure that `cf push` finds the bits you want to push?

The answer is to add a path line to each application description to lead `cf push` to the correct bits. Assume that `cf push` is run from the `fireplace` directory.

For `spark`:

```
...
...
path: ./spark/
```

For `flame`:

```
...
...
path: ./flame/
```

The manifest now consists of two applications blocks.

```
...
# this manifest deploys two applications
# apps are in flame and spark directories
# flame and spark are in fireplace
# cf push should be run from fireplace
applications:
- name: spark
  memory: 1G
  instances: 2
  host: flint-99
  domain: example.com
  path: ./spark/
  services:
    - mysql-flint-99
- name: flame
  memory: 1G
  instances: 2
  host: burnin-77
  domain: example.com
  path: ./flame/
  services:
    - redis-burnin-77
```

Follow these general rules when using a multiple-application manifest:

- Name and completely describe your applications in the manifest.
- Use a `no-route` line in the description of any application that provides background services to another application.
- Do not provide an application name with `cf push`.
- Do not use any command line options with `cf push`.

There are only two narrow exceptions:

- If your manifest is not named `manifest.yml` or not in the current working directory, use the `-f` command line option.
- If you want to push a single application rather than all of the applications described in the manifest, provide the

desired application name by running `cf push my-app`.

Minimizing Duplication

In manifests where multiple applications share settings or services, you begin to see content duplicated. While the manifests still work, duplication increases the risk of typographical errors which cause deployment to fail.

The cure for this problem is to “promote” the duplicate content—that is, to move it up above the applications block, where it need appear only once. The promoted content applies to all applications described in the manifest. Note that content *in* the applications block overrides content *above* the applications block, if the two conflict.

The manifest becomes shorter, more readable, and more maintainable.

Notice how much content in the manifest below has been promoted in this way.

```
---  
...  
# all applications use these settings and services  
domain: example.com  
memory: 1G  
instances: 1  
services:  
- clockwork-mysql  
applications:  
- name: springtock  
host: tock09876  
path: ./spring-music/build/libs/spring-music.war  
- name: springtick  
host: tick09875  
path: ./spring-music/build/libs/spring-music.war
```

In the next section we carry this principle further by distributing content across multiple manifests.

Multiple Manifests with Inheritance

A single manifest can describe multiple applications. Another powerful technique is to create multiple manifests with inheritance. Here, manifests have parent-child relationships such that children inherit descriptions from a parent. Children can use inherited descriptions as-is, extend them, or override them.

Content in the child manifest overrides content in the parent manifest, if the two conflict.

This technique helps in these and other scenarios:

- An application has a set of different deployment modes, such as debug, local, and public. Each deployment mode is described in child manifests that extend the settings in a base parent manifest.
- An application is packaged with a basic configuration described by a parent manifest. Users can extend the basic configuration by creating child manifests that add new properties or override those in the parent manifest.

The benefits of multiple manifests with inheritance are similar to those of minimizing duplicated content within single manifests. With inheritance, though, we “promote” content by placing it in the parent manifest.

Every child manifest must contain an “inherit” line that points to the parent manifest. Place the inherit line immediately after the three dashes at the top of the child manifest. For example, every child of a parent manifest called

`base-manifest.yml` begins like this:

```
---  
...  
inherit: base-manifest.yml
```

You do not need to add anything to the parent manifest.

In the simple example below, a parent manifest gives each application minimal resources, while a production child manifest scales them up.

simple-base-manifest.yml

```
---
```

```
path: .
domain: example.com
memory: 256M
instances: 1
services:
- singular-backend

# app-specific configuration
applications:
- name: springtock
  host: 765shower
  path: ./april/build/libs/april-weather.war
- name: wintertick
  host: 321flurry
  path: ./december/target/december-weather.war
```

simple-prod-manifest.yml

```
---
inherit: simple-base-manifest.yml
applications:
- name:springstorm
  memory: 512M
  instances: 1
  host: 765deluge
  path: ./april/build/libs/april-weather.war
- name: winterblast
  memory: 1G
  instances: 2
  host: 321blizzard
  path: ./december/target/december-weather.war
```

Note: Inheritance can add an additional level of complexity to manifest creation and maintenance. Comments that precisely explain how the child manifest extends or overrides the descriptions in the parent manifest can alleviate this complexity.

Cloud Foundry Environment Variables

This page assumes you are using cf CLI v6.

Environment variables are the means by which the Cloud Foundry runtime communicates with a deployed application about its environment. This page describes the environment variables that Droplet Execution Agents (DEAs) and buildpacks set for applications.

For information about setting your own application-specific environment variables, refer to the [Set Environment Variable in a Manifest](#) section in the Application Manifests topic.

View Environment Variables

Use the `cf env` command to view the Cloud Foundry environment variables for your application. `cf env` displays the following environment variables:

- The `VCAP_SERVICES` variables existing in the container environment
- The user-provided variables set using the `cf set-env` command

```
$ cf env my-app
Getting env variables for app my-app in org my-org / space my-space as
admin...
OK

System-Provided:

{
  "VCAP_APPLICATION": {
    "application_id": "fa05c1a9-0fc1-4fb1-bae1-139850dec7a3",
    "application_name": "my-app",
    "application_uris": [
      "my-app.10.244.0.34.xip.io"
    ],
    "application_version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca",
    "limits": {
      "disk": 1024,
      "fds": 16384,
      "mem": 256
    },
    "name": "my-app",
    "space_id": "06450c72-4669-4dc6-8096-45f9777db68a",
    "space_name": "my-space",
    "uris": [
      "my-app.10.244.0.34.xip.io"
    ],
    "users": null,
    "version": "fb8fbcc6-8d58-479e-bcc7-3b4ce5a7f0ca"
  }
}

User-Provided:
MY_DRAIN: http://drain.example.com
MY_ENV_VARIABLE: 100
```

Variables Available to Your Application

The subsections that follow describe the environment variables that Cloud Foundry makes available for your application container.

You can access environment variables programmatically, including variables defined by the buildpack. Refer to the buildpack documentation for [Java](#), [Node.js](#), and [Ruby](#).

HOME

Root folder for the deployed application.

```
HOME=/home/vcap/app
```

MEMORY_LIMIT

The maximum amount of memory that each instance of the application can consume. You specify this value in an application manifest or with the cf CLI when pushing an application. The value is limited by space and org quotas.

If an instance goes over the maximum limit, it will be restarted. If it has to be restarted too often, it will be terminated.

```
MEMORY_LIMIT=512m
```

PORT

The port on the DEA for communication with the application. The DEA allocates a port to the application during staging. For this reason, code that obtains or uses the application port should reference it using `PORT`.

```
PORT=61857
```

PWD

Identifies the present working directory, where the buildpack that processed the application ran.

```
PWD=/home/vcap
```

TMPDIR

Directory location where temporary and staging files are stored.

```
TMPDIR=/home/vcap/tmp
```

USER

The user account under which the DEA runs.

```
USER=vcap
```

VCAP_APP_HOST

The IP address of the DEA host.

```
VCAP_APP_HOST=0.0.0.0
```

VCAP_APPLICATION

This variable contains the associated attributes for a deployed application. Results are returned in JSON format. The table below lists the attributes that are returned.

Attribute	Description
<code>application_id</code>	GUID that identifies the application.
<code>application_name</code> or <code>name</code>	The name assigned to the application when it was pushed.
<code>application_users</code> or <code>users</code>	
<code>application_uris</code> or <code>uris</code>	The URI(s) assigned to the application.
<code>application_version</code> or <code>version</code>	GUID that identifies a version of the application that was pushed. Each time an application is pushed or restarted, this value is updated.
<code>host</code>	IP address of the application instance.
<code>instance_id</code>	GUID that identifies the application instance.
<code>instance_index</code>	Index number of the instance. You can access this value directly with the <code>CF_INSTANCE_INDEX</code> variable.

<code>limits</code>	The memory, disk, and number of files permitted to the instance. Memory and disk limits are supplied when the application is deployed, either on the command line or in the application manifest. The number of files allowed is operator-defined.
<code>port</code>	Port of the application instance. You can access this value directly with the PORT variable.
<code>started_at</code> or <code>start</code>	The last time the application was started.
<code>started_at_timestamp</code>	Timestamp for the last time the application was started.
<code>state_timestamp</code>	The timestamp for the time at which the application achieved its current state.

For example:

```
VCAP_APPLICATION={"instance_id": "fe98dc76ba549876543210abcd1234",
"instance_index": 0, "host": "0.0.0.0", "port": 61857, "started_at": "2013-08-12
00:05:29 +0000", "started_at_timestamp": 1376265929, "start": "2013-08-12 00:05:29
+0000", "state_timestamp": 1376265929, "limits": {"mem": 512, "disk": 1024, "fds": 16384},
"application_version": "ab12cd34-5678-abcd-0123-abcdef987654", "application_name"
:"styx-james", "application_uris": ["styx-james.#{region}.app.cf-app.com"], "version": "ab1
2cd34-5678-abcd-0123-abcdef987654", "name": "my-app", "uris": ["my-app.example.com"]
,"users": null}
```

VCAP_APP_PORT

Deprecated name for the [PORT](#) variable, defined above.

VCAP_SERVICES

For [bindable services](#) Cloud Foundry will add connection details to the `VCAP_SERVICES` environment variable when you restart your application, after binding a service instance to your application.

The results are returned as a JSON document that contains an object for each service for which one or more instances are bound to the application. The service object contains a child object for each service instance of that service that is bound to the application. The attributes that describe a bound service are defined in the table below.

The key for each service in the JSON document is the same as the value of the “label” attribute.

Attribute	Description
<code>name</code>	The name assigned to the service instance by the user
<code>label</code>	The name of the service offering
<code>tags</code>	An array of strings an app can use to identify a service instance
<code>plan</code>	The service plan selected when the service instance was created
<code>credentials</code>	A JSON object containing the service-specific credentials needed to access the service instance.

To see the value of VCAP_SERVICES for an application pushed to Cloud Foundry, see [View Environment Variable Values](#).

The example below shows the value of VCAP_SERVICES for bound instances of several services available in the [Pivotal Web Services](#) Marketplace.

```
VCAP_SERVICES=
{
  "elephantsql": [
    {
      "name": "elephantsql-c6c60",
      "label": "elephantsql",
      "tags": [
        "postgres",
        "postgresql",
        "relational"
      ],
      "plan": "turtle",
      "credentials": {
        "uri": "postgres://seilbmbd:ABcdEF@babar.elephantsql.com:5432/seilbmbd"
      }
    }
  ],
  "sendgrid": [
    {
      "name": "mysendgrid",
      "label": "sendgrid",
      "tags": [
        "smtp"
      ],
      "plan": "free",
      "credentials": {
        "hostname": "smtp.sendgrid.net",
        "username": "QvsXMbJ3rK",
        "password": "HCHMOYluTv"
      }
    }
  ]
}
```

Application Instance-Specific Variables

Each instance of an application can access a set of variables with the `CF_INSTANCE` prefix. These variables provide information for a given application instance, including identification of the host DEA by its IP address.

CF_INSTANCE_ADDR

The `CF_INSTANCE_IP` and `CF_INSTANCE_PORT` of the app instance in the format
`particular-DEA-IP:particular-app-instance-port`.

```
CF_INSTANCE_ADDR=1.2.3.4:5678
```

CF_INSTANCE_INDEX

The index number of the app instance.

```
CF_INSTANCE_INDEX=0
```

CF_INSTANCE_IP

The external IP address of the DEA running the container with the app instance.

```
CF_INSTANCE_IP=1.2.3.4
```

CF_INSTANCE_PORT

The `PORT` of the app instance.

```
CF_INSTANCE_PORT=5678
```

CF_INSTANCE_PORTS

The external and internal ports allocated to the app instance.

`CF_INSTANCE_PORTS=[{external:5678,internal:5678}]`

Environment Variable Groups

Environment variable groups are system-wide variables that enable operators to apply a group of environment variables to all running applications and all staging applications separately.

An environment variable group consists of a single hash of name-value pairs that are later inserted into an application container at runtime or at staging. These values can contain information such as HTTP proxy information. The values for variables set in an environment variable group are case-sensitive.

When creating environment variable groups, consider the following:

- Only the Cloud Foundry operator can set the hash value for each group.
- All authenticated users can get the environment variables assigned to their application.
- All variable changes take effect after the operator restarts or restages the applications.
- Any user-defined variable takes precedence over environment variables provided by these groups.

The table below lists the commands for environment variable groups.

CLI Command	Description
<code>running-environment-variable-group</code> or <code>revg</code>	Retrieves the contents of the running environment variable group
<code>staging-environment-variable-group</code> or <code>sevg</code>	Retrieves the contents of the staging environment variable group
<code>set-staging-environment-variable-group</code> or <code>ssevg</code>	Passes parameters as JSON to create a staging environment variable group
<code>set-running-environment-variable-group</code> or <code>srevg</code>	Passes parameters as JSON to create a running environment variable group

The following examples demonstrate how to retrieve the environment variables:

```
$ cf revg
Retrieving the contents of the running environment variable group as
sampledeveloper@example.com...
OK
Variable Name    Assigned Value
HTTP Proxy      87.226.68.130

$ cf sevg
Retrieving the contents of the staging environment variable group as
sampledeveloper@example.com...
OK
Variable Name    Assigned Value
HTTP Proxy      27.145.145.105
EXAMPLE-GROUP   2001

$ cf apps
Getting apps in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

name    requested state    instances    memory    disk    urls
my-app  started           1/1        256M     1G      my-app.com

$ cf env APP-NAME
Getting env variables for app APP-NAME in org SAMPLE-ORG-NAME / space dev as
sampledeveloper@example.com...
OK

System-Provided:

{
  "VCAP_APPLICATION": {
    "application_name": "APP-NAME",
    "application_uris": [
      "my-app.example.com"
    ],
    "application_version": "7d0d64be-7f6f-406a-9d21-504643147d63",
    "limits": {
      "disk": 1024,
      "fds": 16384,
      "mem": 256
    },
    "name": "APP-NAME",
    "space_id": "37189599-2407-9946-865e-8ebd0e2df89a",
    "space_name": "dev",
    "uris": [
      "my-app.example.com"
    ],
    "users": null,
    "version": "7d0d64be-7f6f-406a-9d21-504643147d63"
  }
}

Running Environment Variable Groups:
HTTP Proxy: 87.226.68.130

Staging Environment Variable Groups:
EXAMPLE-GROUP: 2001
HTTP Proxy: 27.145.145.105
```

The following examples demonstrate how to set environment variables:

```
$ cf ssevg '{"test":"87.226.68.130","test2":"27.145.145.105"}'
Setting the contents of the staging environment variable group as admin...
OK
$ cf sevg
Retrieving the contents of the staging environment variable group as admin...
OK
Variable Name    Assigned Value
test            87.226.68.130
test2           27.145.145.105

$ cf srevg '{"test3":"2001","test4":"2010"}'
Setting the contents of the running environment variable group as admin...
OK
$ cf revg
Retrieving the contents of the running environment variable group as admin...
OK
Variable Name    Assigned Value
test3            2001
test4            2010
```


Accessing Apps with Diego-SSH

Diego-SSH CLI Plugin

The easiest way to start an SSH session into a Cloud-Foundry-deployed app instance is to use the [SSH plugin for the CF CLI](#).

You can use CF CLI v6.10.0 or higher to install the SSH plugin from the CF-Community repo:

```
$ cf add-plugin-repo CF-Community http://plugins.cloudfoundry.org/
$ cf install-plugin Diego-SSH -r CF-Community
```

If your app is running on Diego with SSH enabled and your operator has configured SSH access to be publicly routable, run `cf ssh APP-NAME` in the terminal to start an interactive SSH session in instance index 0 of your app.

```
$ cf ssh my-awesome-app
```

To target a different instance index with `cf ssh`, supply it via the `-i` flag.

The `cf ssh` command also supports executing commands inside the app instance and forwarding local ports to it via the `-L` flag.

CF Authenticator access

It is also possible to connect to your SSH-enabled app instance via other SSH clients. If SSH is configured for your deployment, Cloud Controller's `/v2/info` endpoint contains an `app_ssh_endpoint` field with the domain name and port for the externally available SSH endpoint, and an `app_ssh_host_key_fingerprint` field with the fingerprint of the host key for the deployment. To connect to a particular instance of your app, run

```
ssh -p SSH-PORT cf:APP-GUID/APP-INSTANCE-INDEX@SSH-DOMAIN .
```

The password is a UAA-issued authorization code for a user authorized to modify that app, such as a user with the SpaceDeveloper role in the app's space. Such a code can be obtained via the Diego-SSH plugin's `get-ssh-code` command, or directly from the UAA `/oauth/authorize` endpoint.

Transferring files with `scp` is invoked similarly, although the user name must be specified with the `-o` flag because of the `:` character present in it. For example, to copy a file out of the instance, run

```
ssh -P SSH-PORT -o User=cf:APP-GUID/APP-INSTANCE-INDEX SSH-DOMAIN:REMOTE-FILE-TO-RETRIEVE LOCAL-FILE-DESTINATION
```

More details on the CF Authenticator can be found in the [Diego-SSH README](#).

Using Blue-Green Deployment to Reduce Downtime and Risk

This page assumes you are using cf CLI v6.

Blue-green deployment is a release technique that reduces downtime and risk by running two identical production environments called Blue and Green.

At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.

As you prepare a new release of your software, deployment and the final stage of testing takes place in the environment that is *not* live: in this example, Green. Once you have deployed and fully tested the software in Green, you switch the router so all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

This technique can eliminate downtime due to application deployment. In addition, blue-green deployment reduces risk: if something unexpected happens with your new release on Green, you can immediately roll back to the last version by switching back to Blue.

Note: You can adjust the route mapping pattern to display a static maintenance page during a maintenance window for time-consuming tasks, such as migrating a database. In this scenario, the router switches all incoming requests from Blue to Maintenance to Green.

Blue-Green Deployment with Cloud Foundry Example

For this example, we'll start with a simple application: "demo-time." This app is a web page that displays the words "Blue time" and the date/time on the server.

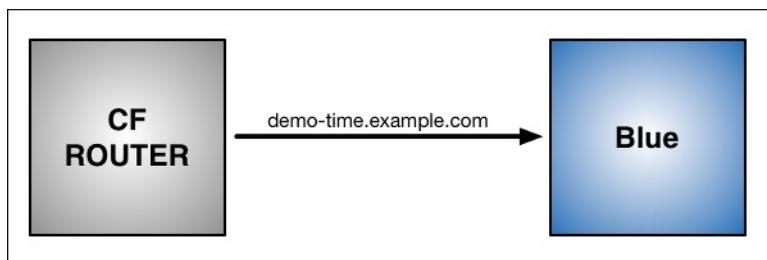
Step 1: Push an App

Use the cf CLI to push the application. Name the application "Blue" with the subdomain "demo-time."

```
$ cf push Blue -n demo-time
```

As shown in the graphic below:

- Blue is now running on Cloud Foundry.
- The CF Router sends all traffic for `demo-time.example.com` traffic to Blue.



Step 2: Update App and Push

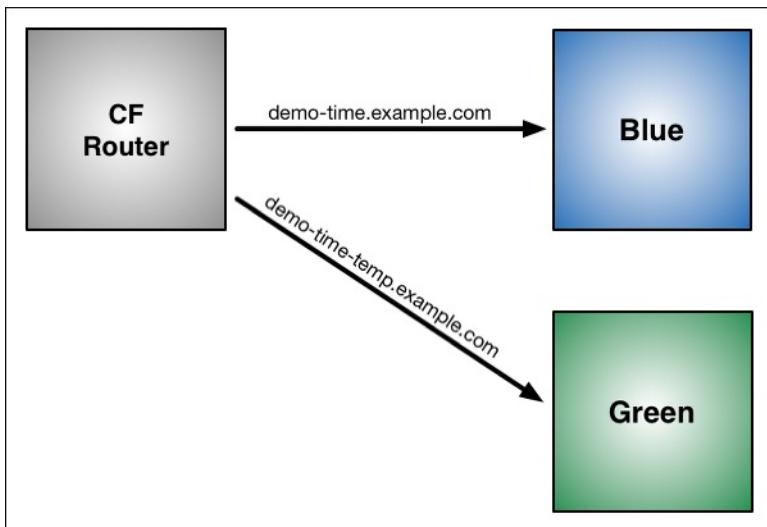
Now make a change to the application. First, replace the word "Blue" on the web page with "Green," then rebuild the source file for the application. Run `cf push` again, but use the name "Green" for the application and provide a different subdomain to create a temporary route:

```
$ cf push Green -n demo-time-temp
```

After this push:

- Two instances of our application are now running on Cloud Foundry: the original Blue and the updated Green.

- The CF Router still sends all traffic for `demo-time.example.com` traffic to Blue. The router now also sends any traffic for `demo-time-temp.example.com` to Green.



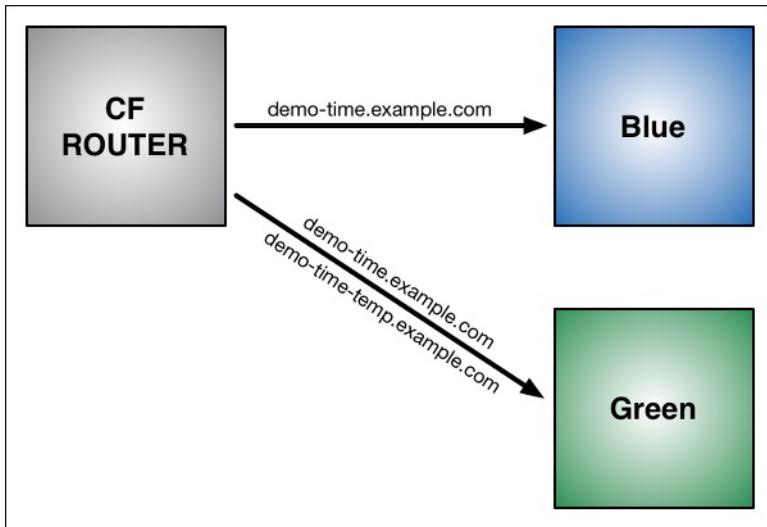
Step 3: Map Original Route to Green

Now that both apps are up and running, switch the router so all incoming requests go to the Green app *and* the Blue app. Do this by mapping the original URL route (`demo-time.example.com`) to the Green application.

```
$ cf map-route Green example.com -n demo-time
Binding demo-time.example.com to Green... OK
```

After the `cf map-route` command :

- The CF Router continues to send traffic for `demo-time-temp.example.com` to Green.
- The CF Router immediately begins to load balance traffic for `demo-time.example.com` between Blue and Green.



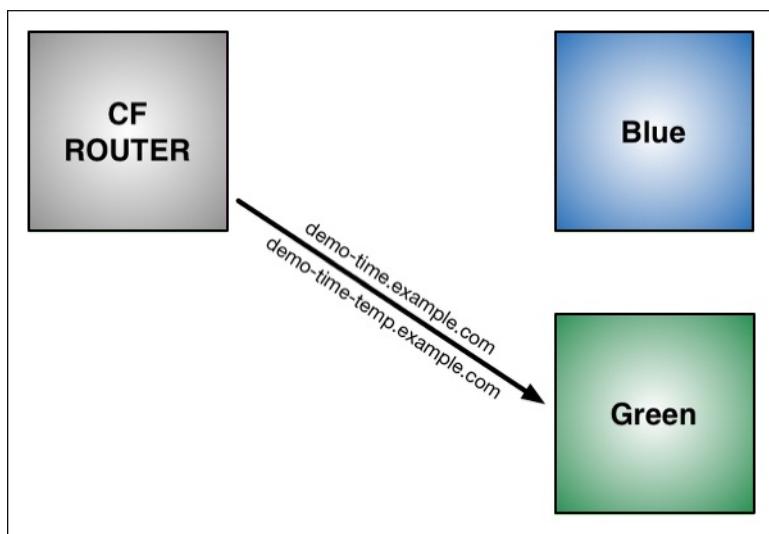
Step 4: Unmap Route to Blue

Once you verify Green is running as expected, stop routing requests to Blue using the `cf unmap-route` command:

```
$ cf unmap-route Blue example.com -n demo-time
Unbinding demo-time.example.com from blue... OK
```

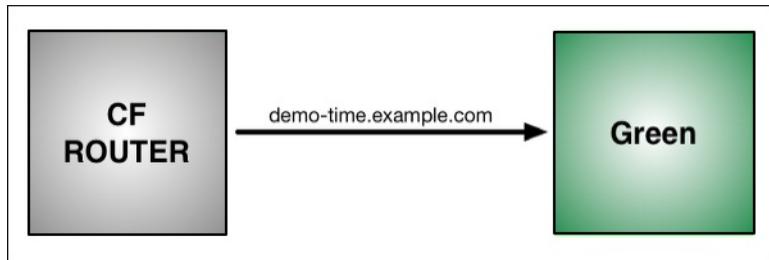
After `cf unmap-route` command:

- The CF Router stops sending traffic to Blue. Instead, it routes all traffic to `demo-time.example.com` to Green:



Step 5: Remove Temporary Route to Green

You can now use `cf unmap-route` to remove the route to `demo-time-temp.example.com`. You can also decommission Blue, or keep it in case you need to roll back your changes.



Troubleshooting Application Deployment and Health

This page assumes you are using cf CLI v6.

Refer to this topic for help diagnosing and resolving common issues when you deploy and run applications on Cloud Foundry.

Common Issues

The following sections describe common issues you might encounter when attempting to deploy and run your application, and possible resolutions.

cf push Times Out

If your deployment times out during the upload or staging phase, you may receive one of the following error messages:

- 504 Gateway Timeout
- Error uploading application
- Timed out waiting for async job JOB-NAME to finish

If this happens, do the following:

- **Check your network speed.** Depending on the size of your application, your `cf push` could be timing out because the upload is taking too long. We recommend an Internet connection speed of at least 768 KB/s (6 Mb/s) for uploads.
- **Make sure you are pushing only needed files.** By default, `cf push` will push all the contents of the current working directory. Make sure you are pushing only the directory for your application. If your application is too large, or if it has many small files, Cloud Foundry may time out during the upload. To reduce the number of files you are pushing, ensure that you push only the directory for your application, and remove unneeded files or use the `.cfignore` file to [specify excluded files](#).
- **Set the CF_STAGING_TIMEOUT and CF_STARTUP_TIMEOUT environment variables.** By default your app has 15 minutes to stage and 5 minutes to start. You can increase these times by setting `CF_STAGING_TIMEOUT` and `CF_STARTUP_TIMEOUT`. Type `cf help` at the command line for more information.
- **If your app contains a large number of files, try pushing the app repeatedly.** Each push uploads a few more files. Eventually, all files have uploaded and the push succeeds. This is less likely to work if your app has many small files.

App Too Large

If your application is too large, you may receive one of the following error messages on `cf push`:

- 413 Request Entity Too Large
- You have exceeded your organization's memory limit

If this happens, do the following:

- **Make sure your org has enough memory for all instances of your app.** You will not be able to use more memory than is allocated for your organization. To view the memory quota for your org, use `cf org ORG_NAME`. Your total memory usage is the sum of the memory used by all applications in all spaces within the org. Each application's memory usage is the memory allocated to it multiplied by the number of instances. To view the memory usage of all the apps in a space, use `cf apps`.
- **Make sure your application is less than 1 GB.** By default, Cloud Foundry deploys all the contents of the current working directory. To reduce the number of files you are pushing, ensure that you push only the directory for your application, and remove unneeded files or use the `.cfignore` file to [specify excluded files](#). The following limits apply:
 - The app files to push cannot exceed 1 GB.
 - The droplet that results from compiling those files cannot exceed 1.5 GB. Droplets are typically a third larger than the pushed files.
 - The combined size of the app files, compiled droplet, and buildpack cache cannot total more than 4 GB of space

during staging.

Unable to Detect a Supported Application Type

If Cloud Foundry cannot [identify an appropriate buildpack](#) for your app, you will see an error message that states

```
Unable to detect a supported application type .
```

You can view what buildpacks are available with the `cf buildpacks` command.

If you see a buildpack that you believe should support your app, refer to the [buildpack documentation](#) for details about how that buildpack detects applications it supports.

If you do not see a buildpack for your app, you may still be able to push your application with a [custom buildpack](#) using `cf push -b` with a path to your buildpack.

App Deploy Fails

Even when the deploy fails, the app might exist on Elastic Runtime. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or the [Apps Manager](#), or you might have to delete the app and redeploy.

Common reasons deploying an app fails include the following:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL or MongoDB service instance. Refer to Step 3: Create and Bind a Service Instance for a RoR Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip App Requires Unique URL.

App Requires Unique URL

Elastic Runtime requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

App Fails to Start

After `cf push` stages the app and uploads the droplet, the app may fail to start, commonly with a pattern of starting and crashing similar to the following example:

```
----> Uploading droplet (23M)
...
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 down
...
0 of 1 instances running, 1 failing
FAILED
Start unsuccessful
```

If this happens, try the following:

Find the reason app is failing and modify your code. Run `cf events APP-NAME` and `cf logs APP-NAME --recent` and look for messages similar to this:

```
2014-04-29T17:52:34.00-0700 app.crash index: 0, reason: CRASHED, exit_description: app instance exited, exit_status: 1
```

These messages may identify a memory or port issue. If they do, take that as a starting point when you re-examine and fix your application code.

- **Make sure your application code uses the `PORT` environment variable.** Your application may be failing because it is listening on the wrong port. Instead of hard coding the port on which your application listens, use the `PORT` environment variable.

For example, this Ruby snippet assigns the port value to the `listen_here` variable:

```
listen_here = ENV['PORT']
```

For more examples specific to your application framework, see the appropriate [buildpacks documentation](#) for your app's language.

- **Make sure your app adheres to the principles of the [Twelve-Factor App](#) and [Prepare to Deploy an Application](#).** These texts explain how to prevent situations where your app builds locally but fails to build in the cloud.

App consumes too much memory, then crashes

An app that `cf push` has uploaded and started can crash later if it uses too much memory.

Make sure your app is not consuming more memory than it should. When you ran `cf push` and `cf scale`, that configured a limit on the amount of memory your app should use. Check your app's actual memory usage. If it exceeds the limit, modify the app to use less memory.

Routing Conflict

Cloud Foundry allows multiple apps, or versions of the same app, to be mapped to the same route. This enables Blue-Green deployment, the deployment strategy that Pivotal recommends. See [Using Blue-Green Deployment to Reduce Downtime and Risk](#).

Multiple apps should be mapped to the same route only during a Blue-Green deployment. In other situations, routing multiple apps to the same route can cause file load errors and other buggy behavior, as incoming requests are randomly routed to one of the apps on the shared route.

If you suspect a routing conflict, run `cf routes` to check the routes in your installation.

If two apps share a route outside of a Blue-Green deploy strategy, choose one app to re-assign to a different route and follow the procedure below:

1. Run `cf unmap-route YOUR-APP-NAME OLD-ROUTE` to remove the existing route from that app.
2. Run `cf map-route YOUR-APP-NAME NEW-ROUTE` to map the app to a new, unique route.

Gathering Diagnostic Information

Use the techniques in this section to gather diagnostic information and troubleshoot app deployment issues.

Examining Environment Variables

`cf push` deploys your application to a container on the server. The environment variables in the container govern your application.

You can set environment variables in a manifest created before you deploy. See [Deploying with Application Manifests](#).

You can also set an environment variable with a `cf set-env` command followed by a `cf push` command. You must run `cf push` for the variable to take effect in the container environment.

Use the `cf env` command to view the environment variables that you have set using the `cf set-env` command and the variables in the container environment:

```
$ cf env my-app
Getting env variables for app my-app in org My-Org / space development as admin...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "p-mysql-n/a": [
      {
        "credentials": {
          "uri": "postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lraa"
        },
        "label": "p-mysql-n/a",
        "name": "p-mysql",
        "syslog_drain_url": "",
        "tags": ["postgres", "postgresql", "relational"]
      }
    ]
  }
}

User-Provided:
my-env-var: 100
my-drain: http://drain.example.com
```

Exploring logs

To explore all logs available in the container, first view the log filenames, then view the log that interests you:

```
$ cf files my-app logs/
Getting files for app my-app in org my-org / space development as a.user@example.com...
OK

staging_task.log          844B
stderr.log                182B
stdout.log                0B

$ cf files my-app logs/stderr.log
Getting files for app my-app in org my-org / space development as a.user@example.com...
OK

[2014-01-27 20:21:58] INFO  WEBrick 1.3.1
[2014-01-27 20:21:58] INFO  ruby 1.9.3 (2013-11-22) [x86_64-linux]
[2014-01-27 20:21:58] INFO  WEBrick::HTTPServer#start: pid=31 port=64391
```

Tracing Cloud Controller REST API Calls

If a command fails or produces unexpected results, re-run it with `CF_TRACE` enabled to view requests and responses between the cf CLI and the Cloud Controller REST API.

For example:

- Re-run `cf push` with `CF_TRACE` enabled:
`CF_TRACE=true cf push APP-NAME`
- Re-run `cf push` while appending API request diagnostics to a log file:
`CF_TRACE=PATH-T0-TRACE.LOG cf push APP-NAME`

These examples enable `CF_TRACE` for a single command only. To enable it for an entire shell session, set the variable first:

```
export CF_TRACE=true
```

```
export CF_TRACE=PATH-T0-TRACE.LOG
```

Note: `CF_TRACE` is a local environment variable that modifies the behavior of the cf CLI. Do not confuse `CF_TRACE` with the [variables in the container environment](#) where your apps run.

cf Troubleshooting Commands

You can investigate app deployment and health using the cf CLI.

Some cf CLI commands may return connection credentials. Remove credentials and other sensitive information from command output before you post the output a public forum.

- `cf apps` : Returns a list of the applications deployed to the current space with deployment options, including the name, current state, number of instances, memory and disk allocations, and URLs of each application.
- `cf app APP-NAME` : Returns the health and status of each instance of a specific application in the current space, including instance ID number, current state, how long it has been running, and how much CPU, memory, and disk it is using.
- `cf env APP-NAME` : Returns environment variables set using `cf set-env` and variables existing in the container environment.
- `cf events APP-NAME` : Returns information about application crashes, including error codes. See <https://github.com/cloudfoundry/errors> for a list of Cloud Foundry errors. Shows that an app instance exited; for more detail, look in the application logs.
- `cf logs APP-NAME --recent` : Dumps recent logs. See [Viewing Logs in the Command Line Interface](#).
- `cf logs APP-NAME` : Returns a real-time stream of the application STDOUT and STDERR. Use **Ctrl-C** (^C) to exit the real-time stream.
- `cf files APP-NAME` : Lists the files in an application directory. Given a path to a file, outputs the contents of that file. Given a path to a subdirectory, lists the files within. Use this to [explore](#) individual logs.

 **Note:** Your application should direct its logs to STDOUT and STDERR. The `cf logs` command also returns messages from any [log4j](#) facility that you configure to send logs to STDOUT.

Identifying the API Endpoint for your Elastic Runtime Instance

The API endpoint, or target URL, for your Elastic Runtime instance is the URL of the Cloud Controller in your Elastic Runtime instance. Find your API endpoint by consulting your cloud operator, from the Apps Manager, or from the command line.

From the Apps Manager

Log in to the Apps Manager for your Elastic Runtime instance, then click **Tools** in the left navigation panel. The **Getting Started** section of the Tools page shows your API endpoint.

GETTING STARTED

```
$ cf help
$ cf login -a https://api.your_endpoint.com
API endpoint: https://api.your_endpoint.com
Username> your_username
Password> your_password
Org> your_org
Space> your_space
$ cf push your_app
```

From the Command Line

From a command line, use the `cf api` command to view your API endpoint.

Example:

```
$ cf api
API endpoint: https://api.example.com (API version: 2.2.0)
```

Installing the cf Command Line Interface

To install the Cloud Foundry command line interface (cf CLI), download it from the Tools page of your Apps Manager instance and follow the instructions for your operating system.

Uninstall cf CLI v5

If you previously used the cf CLI v5 Ruby gem, you must uninstall this gem before installing cf CLI v6.

To uninstall, run `gem uninstall cf`.

 **Note:** To ensure that your Ruby environment manager registers the change, close and reopen your terminal.

Windows Installation

To install the cf CLI on Windows:

1. Unpack the zip file.
2. Double click the `cf CLI` executable.
3. When prompted, click **Install**, then **Close**.

Mac OS X and Linux Installation

1. Open the `.pkg` file.
2. In the installer wizard, click **Continue**.
3. Select an install destination and click **Continue**.
4. When prompted, click **Install**.

 **Note:** If you are using the default Security Settings on your machine, a signing issue appears when you download the `.pkg` file. Refer to the [Apple documentation](#) for instructions for resolving this issue.

Next Steps

To verify your installation, open a terminal window and type `cf`. If your installation was successful, the cf CLI help listing appears.

For more information about cf CLI version 6, see [Getting Started with cf CLI v6](#).

Getting Started with the cf CLI

This guide introduces cf CLI v5 users to what is new in cf CLI v6.

The Cloud Foundry command line interface (cf CLI) v6 is simpler, faster and more powerful than v5. Consider these key differences:

- The cf CLI has been completely re-written in Go, and is more performant than previous versions, which were written in Ruby.
- There are now native installers for all major operating systems.
- Commands behave consistently and logically: required arguments never have flags; optional arguments always have flags.
- Command names are shorter, yet more communicative, and most have single-letter aliases.
- While many cf CLI v5 commands read manifests, only the push command in cf CLI v6 reads manifests.
- While cf CLI v5 used interactive prompts widely, cf CLI v6 uses interactive prompts only for login and optionally for creating user-provided services (you can also create user-provided services non-interactively).

In the guided tour of the cf CLI v6 which follows, we highlight new and improved features and how to use them.

Note: You cannot perform certain tasks in the CLI because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime administrator, the following message displays for these types of commands: `error code: 10003, message: You are not authorized to perform the requested action`

Installation

cf CLI v6 installs with a simple point-and-click, and you no longer need to install Ruby on your system first (or ever). You can use new binaries or new native installers. See [Install cf CLI Version 6](#).

We recommend that you watch our [tools page](#) to learn when updates are released, and download a new binary or a new installer when you want to update.

Localization

The cf CLI translates terminal output into the language that you select.

Note: Localization affects only messages that the cf CLI generates.

Run `cf config --locale YOUR_LANGUAGE` to set the language. The default language is `en_US`. For example:

```
cf config --locale pt_BR
```

The CLI currently supports eight languages:

- German: `de_DE`
- English: `en_US`
- Spanish: `es_ES`
- French: `fr_FR`
- Italian: `it_IT`
- Japanese: `ja_JA`
- Portuguese (Brazil): `pt_BR`
- Chinese (simplified): `zh_Hans`

Login

The `login` command in the cf CLI v6 has expanded functionality. In addition to your username and password, you can provide a target API endpoint, organization, and space.

If not specified on the command line, the cf CLI prompts for:

- **API endpoint:** This is [the URL of the Cloud Controller in your Elastic Runtime instance](#).
- **Username:** Your username.
- **Password:** Your password.
- **Org:** The organization where you want to deploy your application.
- **Space:** The space in the organization where you want to deploy your application.

If you have only one organization and one space, you can omit them because `cf login` targets them automatically.

Usage:

```
$ cf login [-a API_URL] [-u USERNAME] [-p PASSWORD] [-o ORG] [-s SPACE]
```

Alternatively, you can write a script to log in and set your target, using the non-interactive `cf api`, `cf auth`, and `cf target` commands.

Upon successful login, the cf CLI v6 saves a `config.json` file containing your API endpoint, organization, space values, and access token. If you change these settings, the `config.json` file is updated accordingly.

By default, `config.json` is located in your `~/.cf` directory. The new `CF_HOME` environment variable allows you to locate the `config.json` file wherever you like.

Push

In cf CLI v6, `push` is simpler to use and faster.

- `APP`, the name of the application to push, is the only required argument, and the only argument that has no flag. Even `APP` can be omitted when you provide the application name in a manifest.
- Many command line options are now one character long. For example, `-n` is now the flag for hostname or subdomain, replacing `--host`.
- There is no longer an interactive mode.
- You no longer create manifests interactively. See [Deploying with Application Manifests](#).
- You no longer create services with push interactively or in a manifest. See [User-Provided Services](#) to learn about new commands for creating services.
- The `-m` (memory limit) option now requires a unit of measurement: `M`, `MB`, `G`, or `GB`, in upper case or lower case.

The cf CLI v6 has expanded capabilities in the form of four new options.

- `-t` (timeout) allows you to give your application more time to start, up to 180 seconds.
- `--no-manifest` forces the cf CLI to ignore any existing manifest.
- `--no-hostname` makes it possible to specify a route with a domain but no hostname.
- `--no-route` removes existing and suppresses new external routes to your application. Use this flag when pushing worker apps.

Usage:

```
$ cf push APP [-b URL] [-c COMMAND] [-d DOMAIN] [-i NUM_INSTANCES] [-m MEMORY] /  
[-n HOST] [-p PATH] [-s STACK] [--no-hostname] [--no-route] [--no-start]
```

Optional arguments include:

- `-b` — Custom buildpack URL, for example, <https://github.com/heroku/heroku-buildpack-play.git> or <https://github.com/heroku/heroku-buildpack-play.git#stable> to select `stable` branch
- `-c` — Start command for the application.
- `-d` — Domain, for example, example.com.

- `-f` — replaces `--manifest`
- `-i` — Number of instances of the application to run.
- `-m` — Memory limit, for example, 256, 1G, 1024M, and so on.
- `-n` — Hostname, for example, `my-subdomain`.
- `-p` — Path to application directory or archive.
- `-s` — Stack to use.
- `-t` — Timeout to start in seconds.
- `--no-hostname` — Map the root domain to this application (NEW).
- `--no-manifest` — Ignore manifests if they exist.
- `--no-route` — Do not map a route to this application (NEW).
- `--no-start` — Do not start the application after pushing.

 **Note:** The ``-no-route`` option also removes existing routes from previous pushes of this app.

User-Provided Service Instances

The cf CLI v6 has new commands for creating and updating user-provided service instances. You have the choice of three ways to use these commands: interactively, non-interactively, and in conjunction with third-party log management software as described in [RFC 6587](#). When used with third-party logging, the cf CLI sends data formatted according to [RFC 5424](#).

Once created, user-provided service instances can be bound to an application with `cf bind-service`, unbound with `cf unbind-service`, renamed with `cf rename-service`, and deleted with `cf delete-service`.

The cf create-user-provided-service Command

The alias for `create-user-provided-service` is `cups`.

To create a service instance in interactive mode, use the `-p` option with a comma-separated list of parameter names. The cf CLI then prompts you for each parameter in turn.

```
cf cups SERVICE_INSTANCE -p "host, port, dbname, username, password"
```

To create a service instance non-interactively, use the `-p` option with a JSON hash of parameter keys and values.

```
cf cups SERVICE_INSTANCE -p '{"username": "admin", "password": "pa55woRD"}'
```

To create a service instance that drains information to third-party log management software, use the `-l SYSLOG_DRAIN_URL` option.

```
cf cups SERVICE_INSTANCE -l syslog://example.com:514
```

The cf update-user-provided-service Command

The alias for `update-user-provided-service` is `uups`. You can use `cf update-user-provided-service` to update one or more of the attributes for a user-provided service instance. Attributes that you do not supply are not updated.

To update a service instance non-interactively, use the `-p` option with a JSON hash of parameter keys and values.

```
cf uups SERVICE_INSTANCE -p '{"username": "USERNAME", "password": "PASSWORD"}'
```

To update a service instance that drains information to third-party log management software, use the `-l SYSLOG_DRAIN_URL` option.

```
cf uups SERVICE_INSTANCE -l syslog://example.com:514
```

Domains, Routes, Organizations, and Spaces

The relationships between domains, routes, organizations, and spaces have been simplified in cf CLI v6.

- All domains are now mapped to an org.
- Routes are (still) scoped to spaces and apps.

As before, a route is a URL of the form `HOSTNAME.DOMAIN`. If you do not provide a hostname (also known as subdomain), the URL takes the form `APPNAME.DOMAIN`.

The cf CLI v6 has improved separation between management of private domains and management of shared domains. Only administrators can manage shared domains.

In terms of domains, cf CLI v6 is designed to work with both new and older versions of `cf_release`.

 **Note:** The `map-domain` and `unmap-domain` commands no longer exist.

Commands for managing domains:

- `cf create-domain` — Create a domain.
- `cf delete-domain` — Delete a domain.
- `cf create-shared-domain` — Share a domain with all organizations. Admin only.
- `cf delete-shared-domain` — Delete a domain that was shared with all organizations. Admin only.

Commands for managing routes:

- `cf create-route` — Create a route.
- `cf map-route` — Map a route to an application. If the route does not exist, this command creates it and then maps it.
- `cf unmap-route` — Remove a route from an application.
- `cf delete-route` — Delete a route.

Mapping a Route

1. Use `cf create-domain` to create a domain in the desired organization, unless the domain already exists in (or has been shared with) the organization.
2. Use `cf map-route` to map the domain to an application. Use the `-n HOSTNAME` option to specify a unique hostname for each route that uses the same domain.

 **Note:** You can map a single route to multiple applications in the same space. See [Blue-Green Deployment](#) to learn about an important extension of this technique.

Users and Roles

Commands for listing users:

- `cf org-users` — List users in the organization by role.
- `cf space-users` — List users in the space by role.

Commands for managing roles (admin-only):

- `cf set-org-role` — Assign an organization role to a user. The available roles are “OrgManager”, “BillingManager”, and “OrgAuditor”.
- `cf unset-org-role` — Remove an organization role from a user.
- `cf set-space-role` — Assign a space role to a user. The available roles are “SpaceManager”, “SpaceDeveloper”, and “SpaceAuditor”.
- `cf unset-space-role` — Remove a space role from a user.

Consistent Behavior to Help You Work Efficiently

We have focused on building clear principles into the cf CLI v6 to help operators and developers work efficiently and comfortably.

The cf CLI v6 commands behave consistently and logically:

- Required arguments never have flags.
- Optional arguments always have flags.
- When there is more than one required argument, their order matters.
- Optional arguments can be provided in any order.

For example, `cf create-service`, which has three required arguments and you must provide them in order: `SERVICE`, `PLAN`, and `SERVICE_INSTANCE`. On the other hand, `cf push` has one required argument and several optional arguments. You can provide the optional arguments in any order.

Command names have been made more specific and explicit to communicate what they do as clearly as possible. For example:

- `cf map-route` replaces `cf map`.
- `cf marketplace` replaces `cf m`.

One improvement mainly of interest to developers is that `cf curl` is easier to use cf CLI v6.

- `curl` automatically uses the identity with which you are logged in.
- `curl` usage has been simplified to closely resemble the familiar UNIX pattern.

New Aliases and Command Line Help Conventions

The cf CLI v6 command aliases and command line help feature improved consistency and convenience.

The cf CLI v6 introduces single-letter aliases for commonly used commands. For example:

- `cf p` is the alias for `cf push`.
- `cf t` is the alias for `cf target`.

The cf CLI v6 command line help introduces streamlined style conventions:

- User input is capitalized, as in `cf push APP`.
- Optional user input is designated with a flag and brackets, as in `cf create-route SPACE DOMAIN [-n HOSTNAME]`.

Run `cf help` to view a list of all cf CLI commands and a brief description of each. Run `cf COMMAND -h` to view detailed help, including any alias, for any command. For example:

```
$ cf p -h
NAME:
  push - Push a new app or sync changes to an existing app

ALIAS:
  p

USAGE:
  Push a single app (with or without a manifest):
  cf push APP [-b BUILDPACK_NAME] [-c COMMAND] [-d DOMAIN] [-f MANIFEST_PATH]
  [-i NUM_INSTANCES] [-m MEMORY] [-n HOST] [-p PATH] [-s STACK] [-t TIMEOUT]
  [--no-hostname] [--no-manifest] [--no-route] [--no-start]

  Push multiple apps with a manifest:
  cf push [-f MANIFEST_PATH]

OPTIONS:
  -b          Custom buildpack by name (e.g. my-buildpack) or GIT URL
              (e.g. https://github.com/heroku/heroku-buildpack-play.git)
  -c          Startup command, set to null to reset to default start command
  -d          Domain (e.g. example.com)
  -f          Path to manifest
  -i          Number of instances
  -m          Memory limit (e.g. 256M, 1024M, 1G)
  -n          Hostname (e.g. my-subdomain)
  -p          Path of app directory or zip file
  -s          Stack to use
  -t          Start timeout in seconds
  --no-hostname Map the root domain to this app
  --no-manifest Ignore manifest file
  --no-route   Do not map a route to this app
  --no-start   Do not start an app after pushing
```

Using the cf CLI with an HTTP Proxy Server

If you have an HTTP proxy server on your network between a host running the cf CLI and your Cloud Foundry API endpoint, you must set `HTTP_PROXY` with the hostname or IP address of the proxy server.

The `HTTP_PROXY` environment variable holds the hostname or IP address of your proxy server.

`HTTP_PROXY` is a standard environment variable. Like any environment variable, the specific steps you use to set it depends on your operating system.

Format of `HTTP_PROXY`

`HTTP_PROXY` is set with hostname or IP address of the proxy server in URL format: `http_proxy=http://proxy.example.com`

If the proxy server requires a user name and password, include the credentials:

```
http_proxy=http://username:password@proxy.example.com
```

If the proxy server uses a port other than 80, include the port number:

```
http_proxy=http://username:password@proxy.example.com:8080
```

Setting `HTTP_PROXY` in Mac OS or Linux

Set the `HTTP_PROXY` environment variable using the command specific to your shell. For example, in bash, use the `export` command.

Example:

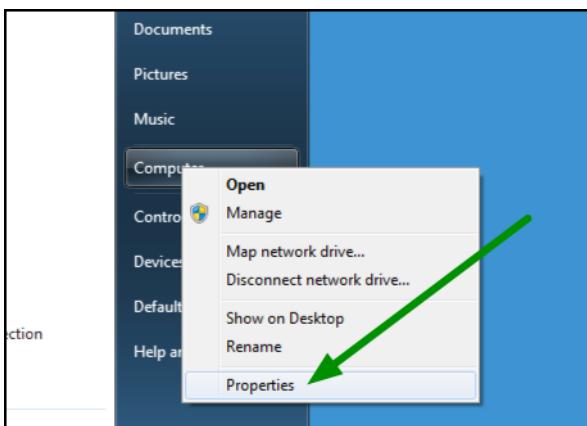
```
$ export HTTP_PROXY=http://my.proxyserver.com:8080
```

To make this change persistent, add the command to the appropriate profile file for the shell. For example, in bash, add a line like the following to your `.bash_profile` or `.bashrc` file:

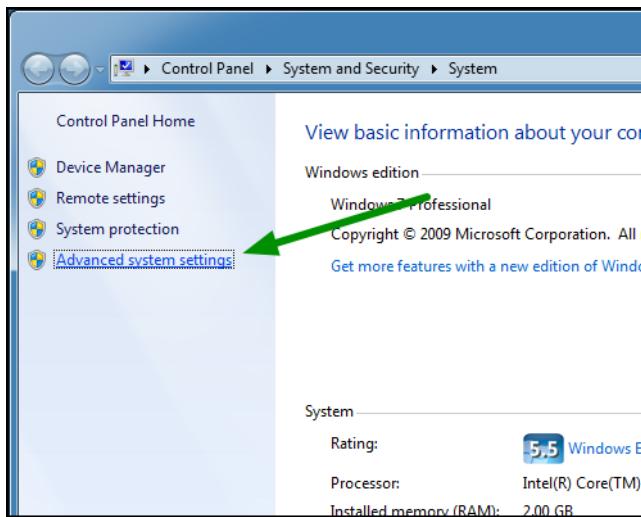
```
http_proxy=http://username:password@hostname:port;
export $HTTP_PROXY
```

Setting `HTTP_PROXY` in Windows

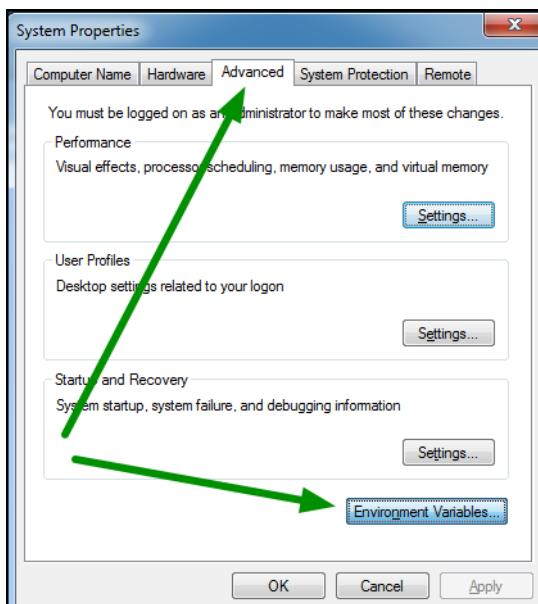
1. Open the Start menu. Right-click **Computer** and select **Properties**.



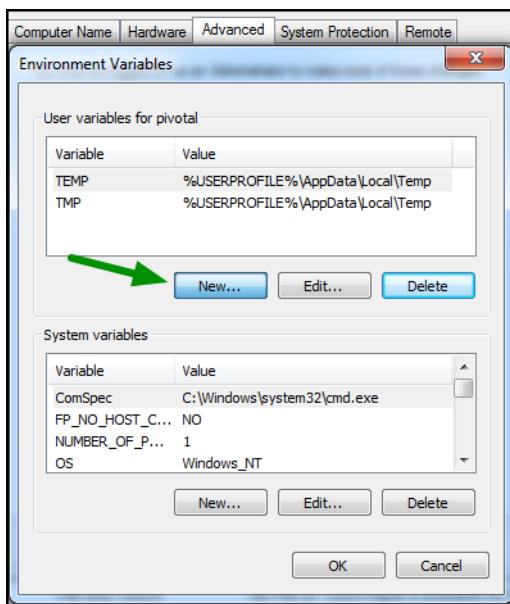
2. In the left pane of the System window, click **Advanced system settings**.



- In the System Properties window, select the **Advanced** tab, then click **Environment Variables**.



- In the Environment Variables window, under User variables, click **New**.



5. In the Variable name field, input `HTTP_PROXY`. In the Variable value field, input your proxy server information.



6. Click **OK**.

Using cf CLI Plugins

The Cloud Foundry Command Line Interface (cf CLI) v.6.7 and higher includes plugin functionality. These plugins enable developers to add custom commands to the cf CLI. You can install and use plugins that Cloud Foundry developers and third-party developers create. You can review the [Cloud Foundry Community CLI Plugin page](#) for a current list of community-supported plugins. You can find information about submitting your own plugin to the community in the [Cloud Foundry CLI plugin repository](#) on GitHub.

The cf CLI identifies a plugin by its binary filename, its developer-defined plugin name, and the commands that the plugin provides. You use the binary filename only to install a plugin. You use the plugin name or a command for any other action.

 **Note:** The cf CLI uses case-sensitive plugin names and commands, but not case-sensitive binary filenames.

Prerequisites

Using plugins requires cf CLI v.6.7 or higher. Refer to the [Installing the cf Command Line Interface](#) topic for information about downloading, installing, and uninstalling the cf CLI.

Changing the Plugin Directory

By default, the cf CLI stores plugins in `$HOME/.cf/plugins` on your workstation. To change the root directory of this path from `$HOME`, set the `CF_PLUGIN_HOME` environment variable. The cf CLI appends `.cf/plugins` to the `CF_PLUGIN_HOME` path that you specify and stores plugins in that location.

For example, if you set `CF_PLUGIN_HOME` to `/my-folder`, cf CLI stores plugins in `/my-folder/.cf/plugins`.

Installing a Plugin

1. Download a binary or the source code for a plugin from a trusted provider.

 **Note:** The cf CLI requires a binary file compiled from source code written in Go. If you download source code, you must compile the code to create a binary.

2. Run `cf install-plugin BINARY_FILENAME` to install a plugin. Replace `BINARY_FILENAME` with the path to and name of your binary file.

 **Note:** You cannot install a plugin that has the same name or that uses the same command as an existing plugin. You must first uninstall the existing plugin.

 **Note:** The cf CLI prohibits you from implementing any plugin that uses a native cf CLI command name or alias. For example, if you attempt to install a third-party plugin that includes the command `cf push`, the cf CLI halts the installation.

Running a Plugin Command

Use the contents of the `cf help PLUGIN` and `PLUGIN COMMANDS` sections to manage plugins and run plugin commands.

1. Run `cf plugins` to list all installed plugins and all commands that the plugins provide.
2. Run `cf PLUGIN_COMMAND` to execute a plugin command.

Uninstalling a Plugin

Use the `PLUGIN_NAME` to remove a plugin, not the `BINARY_FILENAME`.

1. Run `cf plugins` to view the names of all installed plugins.
2. Run `cf uninstall-plugin PLUGIN_NAME` to remove a plugin.

Adding a Plugin Repo

Run `cf add-plugin-repo REPO_NAME URL` to add a plugin repo.

Example:

```
$ cf add-plugin-repo CF-Community http://plugins.cloudfoundry.org/list
OK
http://plugins.cloudfoundry.org/list added as 'CF-Community'
```

Listing Available Plugin Repos

Run `cf list-plugin-repos` to view your available plugin repos.

Example:

```
$ cf list-plugin-repos
OK

Repo Name      Url
CF-Community   http://plugins.cloudfoundry.org/
```

Listing All Plugins by Repo

Run `cf repo-plugins` to show all plugins from all available repos.

Troubleshooting

The cf CLI provides the following error messages to help you troubleshoot installation and usage issues. Third-party plugins can provide their own error messages.

Permission Denied

If you receive a `permission denied` error message, you lack required permissions to the plugin. You must have `read` and `execute` permissions to the plugin binary file.

Plugin Command Collision

Plugin names and commands must be unique. The CLI displays an error message if you attempt to install a plugin with a non-unique name or command.

If the plugin has the same name or command as a currently installed plugin, you must first uninstall the existing plugin to install the new plugin.

If the plugin has a command with the same name as a native cf CLI command or alias, you cannot install the plugin.

About Starting Applications

This page assumes you are using cf CLI v6.

cf push starts your application with a command from one of three sources:

1. The -c command-line option, for example:

```
$ cf push my-app -c "node my-app.js"
```

2. The command attribute in the application manifest, for example:

```
command: node my-app.js
```

3. The buildpack, which provides a start command appropriate for a particular type of application.

The source that the cf CLI uses depends on factors explained below.

How cf push Determines its Default Start Command

The first time you deploy an application, cf push uses the buildpack start command by default. After that, cf push defaults to whatever start command was used for the previous push.

To override these defaults, provide the -c option, or the command attribute in the manifest. When you provide start commands *both* at the command line and in the manifest, cf push ignores the command in the manifest.

Forcing cf push to use the Buildpack Start Command

To force the cf CLI to use the buildpack start command, specify a start command of null.

You can specify a null start command in one of two ways.

1. Using the -c command-line option:

```
$ cf push my-app -c "null"
```

2. Using the command attribute in the application manifest:

```
command: null
```

This can be helpful after you have deployed while providing a start command at the command line or the manifest. At this point, a command that you provided, rather than the buildpack start command, has become the default start command. In this situation, if you decide to deploy using the buildpack start command, the null command makes that easy.

Start Commands when Migrating a Database

Start commands are used in special ways when you migrate a database as part of an application deployment. See [Migrating a Database in Cloud Foundry](#).

Scaling an Application Using cf scale

Factors such as user load, or the number and nature of tasks performed by an application, can change the disk space and memory the application uses. For many applications, increasing the available disk space or memory can improve overall performance. Similarly, running additional instances of an application can allow the application to handle increases in user load and concurrent requests. These adjustments are called **scaling** an application.

Use `cf scale` to scale your application up or down to meet changes in traffic or demand.

Scaling Horizontally

Horizontally scaling an application creates or destroys instances of your application.

Incoming requests to your application are automatically load balanced across all instances of your application, and each instance handles tasks in parallel with every other instance. Adding more instances allows your application to handle increased traffic and demand.

Use `cf scale APP -i INSTANCES` to horizontally scale your application. Cloud Foundry will increase or decrease the number of instances of your application to match `INSTANCES`.

```
$ cf scale myApp -i 5
```

Scaling Vertically

Vertically scaling an application changes the disk space limit or memory limit that Cloud Foundry applies to all instances of the application.

Use `cf scale APP -k DISK` to change the disk space limit applied to all instances of your application. `DISK` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -k 512M
```

Use `cf scale APP -m MEMORY` to change the memory limit applied to all instances of your application. `MEMORY` must be an integer followed by either an **M**, for megabytes, or **G**, for gigabytes.

```
$ cf scale myApp -m 1G
```

Services Overview

This documentation is intended for end users of Cloud Foundry and covers provisioning of service instances and integrating them with applications that have been pushed to Cloud Foundry. If you are interested in building Services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

Services and Service Instances

Cloud Foundry offers a marketplace of services, from which users can provision reserved resources on-demand. Examples of resources services provide include databases on a shared or dedicated server, or accounts on a SaaS application. These resources are known as Service Instances and the systems that deliver and operate these resources are known as Services. Think of a service as a factory that delivers service instances.

For documentation on provisioning service instances and other lifecycle operations, see [Managing Service Instances](#).

 **Note:** For a service to be available in the marketplace, it must be integrated with Cloud Foundry by way of APIs. If you are interested in building Services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

User-Provided Service Instances

Cloud Foundry enables users to integrate services that are not available in the marketplace with their applications using a feature called [User-Provided Service Instances](#) (UPSI).

Service Instance Credentials

Cloud Foundry enables users to automatically provision credentials needed to reach a service instance. These credentials can be managed automatically for use by applications on Cloud Foundry, or managed manually for use by external and local clients.

 **Note:** Not all services support automated generation of credentials. Some services support credentials through application binding only.

Application Binding

Service instance credentials can be delivered automatically to applications running on Cloud Foundry in an environment variable. For more information, see [Binding Applications to Service Instances](#).

For details on binding specific to your application development framework, refer to the Service Binding section in the documentation for [your framework's buildpack](#).

Service Keys

Credentials managed manually are known as **Service Keys**. For more information, see [Managing Service Keys](#).

Streaming Applications Logs to Log Management Services

To learn how your application logs can be streamed to third-party log management services, see [Log Management Services](#).

User-provided service instances can be used to drain applications logs to a service not available in the marketplace. This is also known as setting up a syslog drain. We've documented instructions for a few providers in the [Service-Specific Instructions for Streaming Application Logs](#) topic.

Database Migrations

If your application relies on a relational database, you will need to apply schema changes periodically. For guidance on how to do database migrations on Cloud Foundry-managed services, see [Migrating a Database in Cloud Foundry](#).

Managing Service Instances with the CLI

This page assumes you are using cf CLI v6.

This topic describes lifecycle operations for service instances, including creating, updating, and deleting. For an overview of services, and documentation on other service management operations, see [Using Services](#). If you are interested in building Services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

List Marketplace Services

After targeting and logging into Cloud Foundry, you can view what services are available to your targeted organization with the command `cf marketplace`. Available services may differ between organizations and between Cloud Foundry marketplaces.

```
$ cf marketplace
Getting services from marketplace in org my-org / space test as me@example.com...
OK

service          plans           description
p-mysql         100mb, 1gb      A DBaaS
p-riakcs        developer       An S3-compatible object store
```

Creating Service Instances

You can create a service instance with the command: `cf create-service SERVICE PLAN SERVICE_INSTANCE`

- `SERVICE` The service you choose.
- `PLAN` Service plans are a way for providers to offer varying levels of resources or features for the same service.
- `SERVICE_INSTANCE` A name you provide for your service instance. This is an alias for the instance which is meaningful to you. Use any series of alpha-numeric characters, hyphens (-), and underscores (_). You can rename the instance at any time.

```
$ cf create-service rabbitmq small-plan my_rabbitmq
Creating service my_rabbitmq in org console / space development as user@example.com...
OK
```

Note: User Provided Service Instances provide a way for developers to bind applications with services that are not available in their Cloud Foundry marketplace.

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the provision request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf create-service my-db-service small-plan my-db -c '{"storage_gb":4}'
Creating service my-db in org console / space development as user@example.com...
OK
```

```
$ cf create-service my-db-service small-plan my-db -c /tmp/config.json
Creating service my-db in org console / space development as user@example.com...
OK
```

Instance Tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the [VCAP_SERVICES Environment Variable](#). These tags provide developers with a more generic way for applications to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including a comma-separated list of tags with the `-t` flag.

```
$ cf create-service my-db-service small-plan my-db -t "prod, workers"  
Creating service my-db in org console / space development as user@example.com...  
OK
```

List Service Instances

You can list the service instances in your targeted space with the command `cf services`. The output includes any bound apps, along with the state of the last requested operation for the service instance.

```
$ cf services  
Getting services in org my-org / space test as user@example.com...  
OK  


| name     | service  | plan      | bound apps | last operation   |
|----------|----------|-----------|------------|------------------|
| mybucket | p-riakcs | developer | myapp      | create succeeded |
| mydb     | p-mysql  | 100mb     |            | create succeeded |


```

Get Details for a Particular Service Instance

Details include dashboard urls, if applicable, and operation start and last updated timestamps.

```
$ cf service mydb  
Service instance: mydb  
Service: p-mysql  
Plan: 100mb  
Description: MySQL databases on demand  
Documentation url:  
Dashboard: https://p-mysql.example.com/manage/instances/cc4eab9d-aff4-4beb-bc46-123f2a02dcf1  
  
Last Operation  
Status: create succeeded  
Message:  
Started: 2015-05-08T22:59:07Z  
Updated: 2015-05-18T22:01:26Z
```

Rename a Service Instance

You can change the name given to a service instance. Keep in mind that upon restarting any bound applications, the name of the instance will change in the [VCAP_SERVICES](#) environment variable. If your application depends on the instance name for discovering credentials, changing the name could break your applications use of the service instance.

```
$ cf rename-service mydb mydb1  
Renaming service mydb to mydb1 in org my-org / space test as me@example.com...  
OK
```

Update a Service Instance

Upgrade/Downgrade Service Plan

Changing a plan requires cf CLI v6.7+ and cf-release v192+

By updating the service plan for an instance, users can effectively upgrade and downgrade their service instance to other service plans. Though the platform and CLI now support this feature, services must expressly implement support for it so not all services will. Further, a service might support updating between some plans but not others (e.g., a service might support updating a plan where only a logical change is required, but not where data migration is

necessary). In either case, users can expect to see a meaningful error when plan update is not supported.

```
$ cf update-service mydb -p new-plan  
Updating service instance mydb as me@example.com...  
OK
```

Arbitrary Parameters

Arbitrary parameters require cf CLI v6.12.1+

Some services support additional configuration parameters with the update request. These parameters are passed in a valid JSON object containing service-specific configuration parameters, provided either in-line or in a file. For a list of supported configuration parameters, see documentation for the particular service offering.

```
$ cf update-service mydb -c '{"storage_gb":4}'  
Updating service instance mydb as me@example.com...
```

```
$ cf update-service mydb -c /tmp/config.json  
Updating service instance mydb as me@example.com...
```

Instance Tags

Instance tags require cf CLI v6.12.1+

Some services provide a list of tags that Cloud Foundry delivers in the [VCAP_SERVICES Environment Variable](#). These tags provide developers with a more generic way for applications to parse `VCAP_SERVICES` for credentials. Developers may provide their own tags when creating a service instance by including a comma-separated list of tags with the `-t` flag.

```
$ cf update-service my-db -t "staging, web"  
Updating service my-db in org console / space development as user@example.com...  
OK
```

Delete a Service Instance

Deleting a service instance deprovisions the service instance and deletes *all data* associated with the service instance.

```
$ cf delete-service mydb  
Are you sure you want to delete the service mydb ? y  
Deleting service mydb in org my-org / space test as me@example.com...  
OK
```

Managing Service Keys

This page assumes you are using cf CLI v6.

This topic describes generation of credentials for service instances for use by external or local clients. For an overview of services, and documentation on other service management operations, see the [Services Overview](#) topic. If you are interested in building Services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

 **Note:** Not all services support automated generation of credentials. Some services support credentials through application binding only.

Create a Service Key

Generates credentials for a service instance.

```
$ cf create-service-key myservice mykey  
Creating service key mykey for service instance myservice as me@example.com...  
OK
```

List Service Keys for a Service Instance

```
$ cf service-keys myservice  
Getting service keys for service instance myservice as me@example.com...  
  
name  
mykey1  
mykey2
```

Get Credentials for a Service Key

```
$ cf service-key myservice mykey  
Getting key mykey for service instance myservice as me@example.com...  
  
{  
  uri: foo://user2:pass2@example.com/mydb,  
  servicename: mydb  
}
```

`--guid` can be provided to display the API guid for the service key.

```
$ cf service-key --guid myservice mykey  
Getting key mykey for service instance myservice as me@example.com...  
  
e3696fcb-7a8f-437f-8692-436558e45c7b  
  
OK
```

Delete a Service Instance

```
$ cf delete-service-key myservice mykey  
  
Are you sure you want to delete the service key mykey ? y  
Deleting service key mykey for service instance myservice as me@example.com...  
  
OK
```

Add option `-f` to force deletion without confirmation.

```
$ cf delete-service-key -f myservice mykey
Deleting service key mykey for service instance myservice as me@example.com...
OK
```

User-Provided Service Instances

This page assumes you are using cf CLI v6.

User-provided service instances allow you to represent external assets like databases, messaging services, and key-value stores in Cloud Foundry.

For example, suppose a developer obtains credentials, a URL, and a port number for communicating with an Oracle database that is managed outside of Cloud Foundry. Then the developer creates a user-provided service instance to represent this external resource and provisions the service instance with all the parameters required for communicating with the real Oracle database. Finally, she binds the user-provided service instance to her application.

The user-provided service instance allows the developer to:

- Avoid hard-coding credentials for a service instance into her application.
- Treat the user-provided service instance exactly like any other service instance.

Although the external asset in this example was a real Oracle database, it could have been a simple placeholder used for testing.

User provided services are outside of Cloud Foundry, so you never create or otherwise manage the service itself through Cloud Foundry; you only create and manage service *instances*.

The `cf services` command lists all the service instances (both user-provided and managed) in your target space.

In contrast to the user-provided kind, a managed service is one that a service broker advertises in a catalog.

cf CLI Commands for Working with User-Provided Service Instances

cf CLI commands for creating and updating user-provided service instances can be used in three ways: interactively, non-interactively, and in conjunction with third-party log management software as described in [RFC 6587](#). When used with third-party logging, the cf CLI sends data formatted according to [RFC 5424](#).

Once created, user-provided service instances can be bound to an application with `cf bind-service`, unbound with `cf unbind-service`, renamed with `cf rename-service`, and deleted with `cf delete-service`.

The cf create-user-provided-service Command

The alias for `create-user-provided-service` is `cups`.

To create a service instance in interactive mode, use the `-p` option with a comma-separated list of parameter names. The cf CLI then prompts you for each parameter in turn.

```
cf cups SERVICE_INSTANCE -p "host, port, dbname, username, password"
```

To create a service instance non-interactively, use the `-p` option with a JSON hash of parameter keys and values.

```
cf cups SERVICE_INSTANCE -p '{"username": "admin", "password": "pa55woRD"}'
```

To create a service instance that drains information to third-party log management software, use the `-l SYSLOG_DRAIN_URL` option.

```
cf cups SERVICE_INSTANCE -l syslog://example.com:514
```

The cf update-user-provided-service Command

The alias for `update-user-provided-service` is `uups`. You can use `cf update-user-provided-service` to update one or more of the attributes for a user-provided service instance. Attributes that you do not supply are not updated.

To update a service instance non-interactively, use the `-p` option with a JSON hash of parameter keys and values.

```
cf uups SERVICE_INSTANCE -p '{"username": "USERNAME", "password": "PASSWORD"}'
```

To update a service instance that drains information to third-party log management software, use the `-l SYSLOG_DRAIN_URL` option.

```
cf uups SERVICE_INSTANCE -l syslog://example.com:514
```

 **Note:** 514 is the default syslog port. Ensure that this value is correct for your environment.

Binding User-Provided Service Instances to Applications

There are two ways to bind user-provided service instances to applications:

- With a manifest, *when* you push the application.
- With the `cf bind-service` command, *after* you push the application.

In either case, you must push your app to a space where the desired user-provided instances already exist.

For example, if you want to bind a service instance called `test-mysql-01` to an app, the services block in the manifest should look like this:

```
services:  
- test-mysql-01
```

This excerpt from the `cf push` command and response demonstrates that Cloud Foundry reads the manifest and binds the service instance to the app, which is called `msgapp`.

```
$ cf push  
Using manifest file /Users/a.user/test-apps/msgapp/manifest.yml  
...  
Binding service test-mysql-01 to msgapp in org my-org / space development as a.user@example.com  
OK
```

Once the binding has taken effect, it is described in the [VCAP_SERVICES](#) environment variable.

For more information about manifests, see [Deploying with Application Manifests](#).

For an example of the `cf bind-service` command, see the next section.

Example: Creating a User-Provided Service Instance and Binding it to an App

Suppose we want to create and bind an instance of a publish-subscribe messaging service to an app called `msgapp`. Our Cloud Foundry username is `a.user`, and we plan to call the instance `pubsub`. We know the URL and port for the service.

1. We begin with the `cf create-user-provided-service` command in interactive mode, using its alias, `cups`.

```
$ cf cups pubsub -p "host, port, binary, username, password"  
host> pubsub01.example.com  
port> 1234  
url> pubsub-example.com  
username> pubsubuser  
password> p@$w0rd  
Creating user provided service pubsub in org my-org / space development as a.user@example.com...  
OK
```

2. When we list available services, we see that the new service is not yet bound to any apps.

```
$ cf services
Getting services in org my-org / space development as a.user@example.com...
OK

name          service      plan   bound apps
pubsub        user-provided
```

3. Now we bind the service to our app.

```
$ cf bind-service msgapp pubsub
Binding service pubsub to app msgapp in org my-org / space development as a.user@example.com...
OK
TIP: Use 'cf push' to ensure your env variable changes take effect
```

4. For the binding to take effect, we must push our app a second time.

Note: You can skip this step if you bind a syslog drain service instance to a deployed app. Syslog drain service instances bound to deployed apps automatically start after a short delay.

```
$ cf push msgapp
Updating app msgapp in org my-org / space development as a.user@example.com...
OK
...
```

5. To verify that the service instance is now recorded in `VCAP_SERVICES`, examine the output of `cf env`:

```
$ cf env msgapp
Getting env variables for app msgapp in org My-Org / space development as a.user@example.com...

System-Provided:
{
  "VCAP_SERVICES": {
    "user-provided": [
      {
        "name": "pubsub",
        "label": "user-provided",
        "tags": [],
        "credentials": {
          "binary": "pubsub.rb",
          "host": "pubsub01.example.com",
          "password": "pg29w0rd",
          "port": "1234",
          "username": "pubsubuser"
        },
        "syslog_drain_url": ""
      }
    ]
  }
}
```

Configuring Play Framework Service Connections

Cloud Foundry provides support for connecting a Play Framework application to services such as MySQL, and Postgres. In many cases, a Play Framework application running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry will detect service connections in a Play Framework application and configure them to use the credentials provided in the Cloud Foundry environment. Auto-configuration will only happen if there is a single service of any of the supported types - MySQL or Postgres.

Migrating a Database in Cloud Foundry

This page assumes you are using cf CLI v6.

Application development and maintenance often requires changing a database schema, known as migrating the database. This topic describes three ways to migrate a database on Cloud Foundry.

Migrate Once

This method executes SQL commands directly on the database, bypassing Cloud Foundry. This is the fastest option for a single migration. However, this method is less efficient for multiple migrations because it requires manually accessing the database every time.

Note: Use this method if you expect your database migration to take longer than the timeout that `cf push` applies to your application. The timeout defaults to 60 seconds, but you can extend it up to 180 seconds with the `-t` command line option.

- Run `cf env` and obtain your database credentials by searching in the `VCAP_SERVICES` environment variable:

```
$ cf env db-app
Getting env variables for app my-db-app in org My-Org / space development as admin...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "example-db-n/a": [
      {
        "name": "test-777",
        "label": "example-db-n",
        "tags": ["mysql", "relational"],
        "plan": "basic",
        "credentials": {
          "jdbcUrl": "jdbc:mysql://aa11:2b@cdb-05.example.net:3306/ad_01",
          "uri": "mysql://aa11:2b@cdb-05.example.net:34/ad_01?reconnect=true",
          "name": "ad_01",
          "hostname": "cdb-05.example.net",
          "port": "1234",
          "username": "aa11",
          "password": "2b"
        }
      }
    ]
  }
}
```

- Connect to the database using your database credentials.
- Migrate the database using SQL commands.
- Update the application using `cf push`.

Migrate Occasionally

This method requires you to:

- Create a schema migration command or script.
- Run the migration command when deploying a single instance of the application.
- Re-deploy the application with the original start command and number of instances.

This method is efficient for occasional use because you can re-use the schema migration command or script.

- Create a schema migration command or SQL script to migrate the database. For example:
`rake db:migrate`
- Deploy a single instance of your application with the database migration command as the start command. For example:

```
cf push APP -c 'rake db:migrate' -i 1
```

Note: After this step the database has been migrated but the application itself has not started, because the normal start command is not used.

3. Deploy your application again with the normal start command and desired number of instances. For example:

```
cf push APP -c 'null' -i 4
```

Note: This example assumes that the normal start command for your application is the one provided by the buildpack, which the `-c 'null'` option forces Cloud Foundry to use.

Migrate Frequently

This method uses an idempotent script to partially automate migrations. The script runs on the first application instance only.

This option takes the most effort to implement, but becomes more efficient with frequent migrations.

1. Create a script that:

- Examines the `instance_index` of the `VCAP_APPLICATION` environment variable. The first deployed instance of an application always has an `instance_index` of "0". For example, this code uses Ruby to extract the `instance_index` from `VCAP_APPLICATION`:
`instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"]`
- Determines whether or not the `instance_index` is "0".
- If and only if the `instance_index` is "0", runs a script or uses an existing command to migrate the database. The script or command must be idempotent.

2. Create a manifest that provides:

- The application name
- The `command` attribute with a value of the schema migration script chained with a start command.

Example partial manifest:

```
---  
applications:  
- name: my-rails-app  
  command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rails s -p $PORT -e $RAILS_ENV
```

3. Update the application using `cf push`.

For an example of the migrate frequently method used with Rails, see [Running Rake Tasks](#).

Using Log Management Services

This page assumes you are using cf CLI v6.

This topic describes how to drain logs from Cloud Foundry to a third party log management service.

Cloud Foundry aggregates logs for all instances of your applications as well as for requests made to your applications through internal components of Cloud Foundry. For example, when the Cloud Foundry Router forwards a request to an application, the Router records that event in the log stream for that app. Run the following command to access the log stream for an app in the terminal:

```
cf logs YOUR-APP-NAME
```

If you want to persist more than the limited amount of logging information that Cloud Foundry can buffer, drain these logs to a log management service.

For more information about the systems responsible for log aggregation and streaming in Cloud Foundry, see [Application Logging in Cloud Foundry](#).

Using Services from the Cloud Foundry Marketplace

The Cloud Foundry marketplace offers several log management services. To use one of these services, create an instance of the service and bind it to your application. See the following topics for more information:

- [Service-Specific Instructions for Streaming Application Logs](#) to learn how to stream application logs to specific services including Papertrail, Splunk, and SumoLogic.
- [Managing Service Instances](#) to learn about creating and binding service instances.

 **Note:** Not all marketplace services support syslog drains. Some services implement an integration with Cloud Foundry that enables automated streaming of application syslogs. If you are interested in building Services for Cloud Foundry and making them available to end users, see the [Custom Services](#) documentation.

Using Other Services

If a compatible log management service is not available in your Cloud Foundry marketplace, use the following procedure to stream your logs to a service of your choice.

 **Note:** Only use this procedure if the service that you want to use is not covered in the [Service-Specific Instructions for Streaming Application Logs](#) topic.

Step 1: Configure the Log Management Service

Complete the following steps to set up a communication channel between the log management service and your Cloud Foundry deployment:

1. Obtain the external IP addresses that your Cloud Foundry administrator assigns to outbound traffic.
2. Provide these IP addresses to the log management service. The specific steps to configure a third-party log management service depend on the service. See the [Service-Specific Instructions for Streaming Application Logs](#) topic for more information.
3. Whitelist these IP addresses to ensure unrestricted log routing to your log management service.
4. Record the syslog URL provided by the third-party service. Third-party services typically provide a syslog URL to use as an endpoint for incoming log data. You use this syslog URL in Step 2: Create a User-provided Service Instance.

Cloud Foundry uses the syslog URL to route messages to the service. The syslog URL has a scheme of `syslog`, `syslog-tls`, or `https`, and can include a port number. For example:
`syslog://logs.example.com:1234`

Note: Elastic Runtime does not support using `syslog-tls` with self-signed certificates. If you are running your own syslog server and want to use `syslog-tls`, you must have an SSL certificate signed by a well-known certificate authority.

Step 2: Create a User-provided Service Instance

Create a user-provided service instance using the `cf CLI` `create-user-provided-service` command with the `-l` flag and the syslog URL that you obtained in Step 1: Configure the Log Management Service. The `-l` flag configures the syslog drain.

```
$ cf create-user-provided-service SERVICE-INSTANCE -l SYSLOG-URL
```

Refer to [User-Provided Service Instances](#) for more information.

Step 3: Bind the Service Instance

You have two options for binding the service instance to an application:

- Run `cf push` with a manifest. The services block in the manifest must specify the service instance that you want to bind.
- Run `cf bind-service`. After a short delay, logs begin to flow automatically.

Refer to [Binding User-Provided Service Instances to Applications](#) for more information.

Step 4: Verify Logs are Draining

To verify that logs are draining correctly to a third-party log management service:

1. Take actions that produce log messages, such as making requests of your app.
2. Compare the logs displayed in the CLI against those displayed by the log management service.

For example, if your application serves web pages, you can send HTTP requests to the application. In Cloud Foundry, these generate Router log messages, which you can view in the CLI. Your third-party log management service should display corresponding messages.

Note: For security reasons, Cloud Foundry applications do not respond to `ping`. Therefore, you cannot use `ping` to generate log entries.

Service-Specific Instructions for Streaming Application Logs

This topic provides instructions for configuring some third-party log management services.

Once you have configured a service, refer to the [Third-Party Log Management Services](#) topic for instructions on binding your application to the service.

Papertrail

From your Papertrail account:

1. Click **Add System**.

The screenshot shows the Papertrail dashboard with a header bar containing 'Dashboard', 'Add Systems', and 'Create Group'. Below the header is a message: 'Let's aggregate some logs. Add your first system in about 45 seconds, or take a tour.' A 'Create New System' button is visible at the bottom right of the main area.

2. Click the **Alternatives** link.

The screenshot shows the 'Setup Systems' page. It displays the text 'Your systems will log to logs.papertrailapp.com:11699.' and a link '» Other log methods: Use an app hosting service or old syslog daemon? Alternatives'.

3. Select **I use Heroku**, enter a name, and click **Save**.

The screenshot shows the 'Choose your situation' step. It has three options:

- A My syslogd only uses the default port**: Describes GNU syslogd and embedded OSes.
- B I use Heroku**: Selected, describing Heroku's log setup.
- C My system's hostname changes**: Describes hosts changing IP addresses.

On the right, it says 'Let's create a destination for all logs from this app's dynos.' and 'What should we call it?' with 'CloudFoundry' entered. A 'Save →' button is at the bottom.

4. Record the URL with port that is displayed after creating the system.

The screenshot shows a message box: 'CloudFoundry will log to logs.papertrailapp.com:36129.'

5. Create the log drain service in Cloud Foundry.

```
$ cf cups my-logs -l syslog://logs.papertrailapp.com:PORT
```

6. Bind the service to an app. Restage or push the app using either the `cf restage APPLICATION-NAME` or `cf push APPLICATION-NAME` command. After a short delay, logs begin to flow automatically.

7. Once Papertrail starts receiving log entries, the view automatically updates to the logs viewing page.

All Systems → Dashboard Events Account Help Me

```

Skiping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} INFO: No beans of type org.springframework.data.mongodb.MongoDbFactory found
in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} INFO: No beans of type
org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean found
in application context. Skipping auto-reconfiguration.
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:36 PM
org.cloudfoundry.reconfiguration.AbstractServiceConfigurer configure
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} INFO: Class
org.springframework.amp.ConnectionFactory not
in classpath. Skipping auto-reconfiguration for it
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,680 INFO RequestMappingHandlerMapping:181 - Mapped
"/{albums}/{id}",methods={DELETE},params=[],headers=[],consumes=[],produces[],custom={} onto public void
org.cloudfoundry.samples.music.web.controllers.AlbumController.deleteById(java.lang.String)
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,681 INFO RequestMappingHandlerMapping:181 - Mapped
"/{id}",methods={PUT},params=[],headers=[],consumes=[],produces[],custom={} onto public void
org.cloudfoundry.samples.music.domain.Album
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,682 INFO RequestMappingHandlerMapping:181 - Mapped
"/{id}",methods={POST},params=[],headers=[],consumes=[],produces[],custom={} onto public void
org.cloudfoundry.samples.music.domain.Album
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,683 INFO RequestMappingHandlerMapping:181 - Mapped
"/{info}",methods={GET},params=[],headers[],consumes[],produces[],custom={} onto public
java.lang.Iterable<org.cloudfoundry.samples.music.web.controllers.AlbumController.albums>
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,684 INFO RequestMappingHandlerMapping:181 - Mapped
"/{info}",methods={PUT},params=[],headers[],consumes[],produces[],custom={} onto public
org.cloudfoundry.samples.music.domain.Album
org.cloudfoundry.samples.music.web.controllers.AlbumController.update(jav
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,685 INFO RequestMappingHandlerMapping:181 - Mapped
"/{info}",methods={DELETE},params=[],headers[],consumes[],produces[],custom={} onto public
org.cloudfoundry.samples.music.domain.ApplicationInfo
org.cloudfoundry.samples.music.web.controllers.InfoController.info
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,686 INFO RequestMappingHandlerMapping:181 - Mapped
"/{env}",methods={PUT},params=[],headers[],consumes[],produces[],custom={} onto public
java.util.List<org.springframework.cloud.service.ServiceInfo>
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,684 INFO RequestMappingHandlerMapping:181 - Mapped
"/{env}",methods={POST},params=[],headers[],consumes[],produces[],custom={} onto public
java.util.Map<java.lang.String, java.lang.String>
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,704 INFO SimpleUrlHandlerMapping:362 - Root mapping to handler of
type [class org.springframework.web.servlet.mvc.ParameterizableViewController]
Mar 05 14:57:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:36,726 INFO SimpleUrlHandlerMapping:315 - Mapped URL path [/assets/**]
onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
Mar 05 14:57:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:37,000 INFO SimpleUrlHandlerMapping:315 - Mapped URL path [/**] onto
handler of type [class org.springframework.web.servlet.DispatcherServlet]
Mar 05 14:57:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} 22:57:37,040 INFO DispatcherServlet:488 - FrameworkServlet 'appServlet':
initialization completed in 989 ms
Mar 05 14:57:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:37 PM org.apache.coyote.AbstractProtocol start
Mar 05 14:57:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:37 PM org.apache.catalina.startup.Catalina start
Mar 05 14:57:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Mar 05, 2014 10:57:37 PM org.apache.catalina.core.StandardServer start
Mar 05 14:59:36 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Tried to stop app that never received a start event
Mar 05 14:59:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{App/1} Stopped app instance (Index 1) with guid 40dc96a-c093-4bbb-8718-186f147f3e73
Mar 05 14:59:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{DEA} Stopped app instance (Index 1) with guid 40dc96a-c093-4bbb-8718-186f147f3e73
Mar 05 14:59:37 CloudFoundry AppId:960-c093-4bbb-8718-186f147f3e73/{DEA} Stopped app instance (Index 1) with guid 40dc96a-c093-4bbb-8718-186f147f3e73

```

Example "access denied" (1.2.3.4 OR redis) → Search ⌂ ⌂ Contrast ⌂ PAUSE

https://papertrailapp.com/groups/553803/events?centered_on_id=378003402845892611&q=program%3A4aa&c060-c093-4bbb-8718-186f147f3e73%27INSD

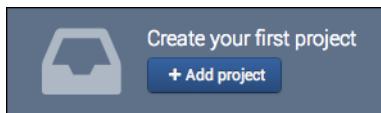
Splunk

See [Streaming Application Logs to Splunk](#) for details.

Splunk Storm

From your Splunk Storm account:

1. Click **Add project**.



2. Enter the project details.

Add project

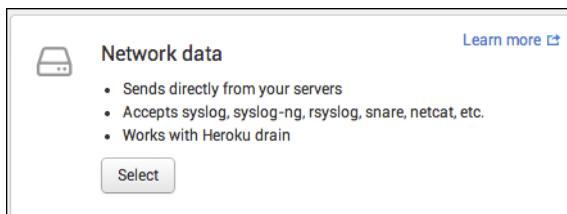
* Project name

* Project time zone

The Storm interface will use this time zone. If data you send to this project does not have a time zone already, it will be assigned this timezone by default [Learn More](#)

[Cancel](#) [Continue](#)

3. Create a new **input** for **Network data**.



4. Manually enter the external IP addresses your Cloud Foundry administrator assigns to outbound traffic. If you are using Pivotal Web Services, refer to the [Externally Visible IP Addresses](#) topic.

The screenshot shows a 'Network Data' configuration screen. At the top, a note says: 'Storm can receive data from any network port. For example, Storm can accept remote data from syslog, rsyslog, syslog-ng, snare, netcat or any other application that transmits via TCP or UDP. Learn more' with a link icon. Below this is a blue button labeled 'Authorize your IP address'. Underneath are two options: 'Automatically' (selected) and 'Manually'.

5. Note the host and port provided for TCP input.

The screenshot shows the 'Authorized network inputs' section. It includes a note about receiving data from network ports via syslog, rsyslog, syslog-ng, snare, netcat, or other applications via TCP or UDP. It lists two sections: '1. Authorize your IP address' (with 'Authorize automatically' and 'Authorize manually' options) and '2. Send data to these ports for this project only' (listing 'TCP' and 'UDP' ports). The TCP port listed is 'tcp k22g-wt6r.data.splunkstorm.com:15486' and the UDP port is 'udp k22g-wt6r.data.splunkstorm.com:15486'.

6. Create the log drain service in Cloud Foundry using the displayed TCP host and port.

```
$ cf cups my-logs -l syslog://HOST:PORT
```

7. Bind the service to an app. Restage or push the app using either the `cf restage APPLICATION-NAME` or `cf push APPLICATION-NAME` command. After a short delay, logs begin to flow automatically.

8. Wait for some events to appear, then click **Data Summary**.

The screenshot shows the 'Data Summary' interface. It displays a search bar with 'What to Search' containing the text '266 Events INDEXED', a timestamp range from 'a minute ago' (EARLIEST EVENT) to 'a few seconds ago' (LATEST EVENT), and a 'Data Summary' button.

9. Click the **loggregator** link to view all incoming log entries from Cloud Foundry.

The screenshot shows a table in the 'Data Summary' interface. The table has columns for Host, Count, and Last Update. There is one entry for 'loggregator' with a count of 26 and a last update time of '3/7/14 3:26:01.000 PM'.

Host	Count	Last Update
loggregator	26	3/7/14 3:26:01.000 PM

SumoLogic

Note: SumoLogic uses HTTPS for communication. HTTPS is supported in Cloud Foundry v158 and above.

From your SumoLogic account:

1. Click the **Add Collector** link.

The screenshot shows the 'Manage Collectors and Sources' interface. At the top, there are links for 'Upgrade Collectors', 'Add Collector', and 'Access Keys'. The 'Add Collector' link is highlighted.

2. Choose **Hosted Collector** and fill in the details.



Add Collector

Name *	Cloud Foundry
Description	(empty)
Category	(empty)
Unless overwritten by Source metadata, the Collector will set the Source category of all messages to this value.	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

3. In the new collector's row of the collectors view, click the **Add Source** link.

Manage Collectors and Sources						
Upgrade Collectors Add Collector Access Keys						
Show: All Collectors Running Collectors Stopped Collectors Expand: All None						
Name	Type	Status	Source Category	Sources	Last Hour	Messages
▼ Cloud Foundry	Hosted	Green		1	None	Add Source Edit Delete

4. Select **HTTP** source and fill in the details.

Select a type of Source:

Amazon S3	HTTP
Collects logs from an Amazon S3 bucket.	HTTP receiver that collects logs sent to a specific address.
Name * <input type="text" value="CloudFoundry"/> Maximum name length is 128 characters	
Description <input type="text"/>	
Source Host <input type="text"/> Host name for the system from which the log files are being collected, e.g. LDAP_Server	
Source Category <input type="text"/> Log category metadata to use later for querying, e.g. OS_Security	
▶ Advanced ▶ Filters	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

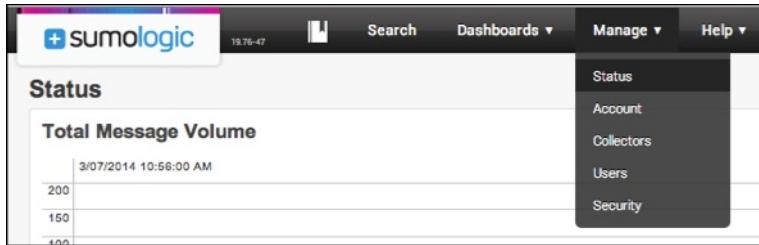
5. Once the source is created, a URL should be displayed. You can also view the URL by clicking the **Show URL** link beside the created source.

Manage Collectors and Sources						
Upgrade Collectors Add Collector Access Keys						
Show: All Collectors Running Collectors Stopped Collectors Expand: All None						
Name	Type	Status	Source Category	Sources	Last Hour	Messages
▼ Cloud Foundry	Hosted	Green		1	None	Add Source Edit Delete
CloudFoundry	HTTP	Green				Show URL Edit Delete

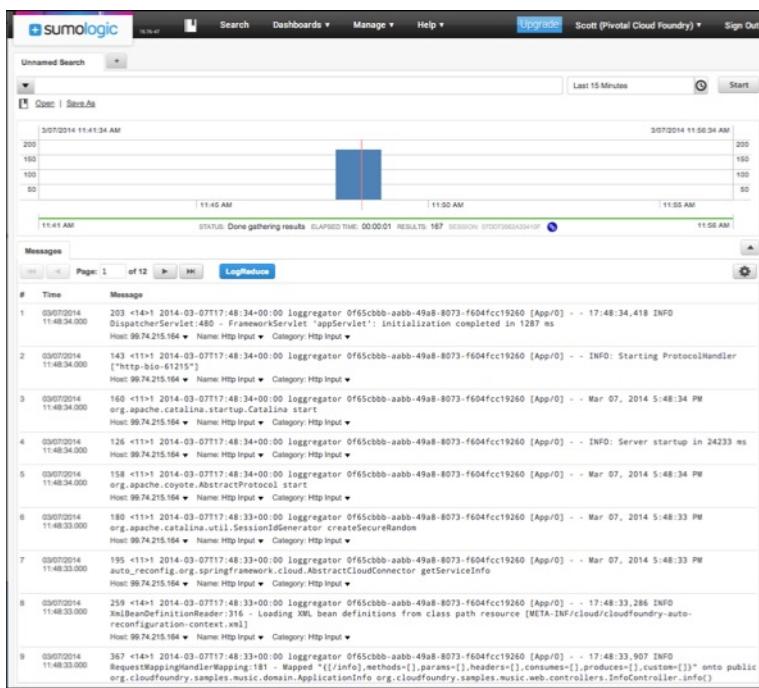
6. Create the log drain service in Cloud Foundry using the displayed URL.

```
$ cf cups my-logs -l HTTP-SOURCE-URL
```

7. Bind the service to an app. Restage or push the app using either the `cf restage APPLICATION-NAME` or `cf push APPLICATION-NAME` command. After a short delay, logs begin to flow automatically.
8. In the SumoLogic dashboard, click **Manage**, then click **Status** to see a view of log messages received over time.



9. In the SumoLogic dashboard, click on **Search**. Place the cursor in the search box, then press **Enter** to submit an empty search query.



Logentries is Not Supported

Cloud Foundry distributes log messages over multiple servers in order to handle load. Currently, we do not recommend using Logentries as it does not support multiple syslog sources.

Streaming Application Logs to Splunk

This page assumes you are using cf CLI v6.

To integrate Cloud Foundry with Splunk Enterprise, complete the following process.

1. Create a Cloud Foundry Syslog Drain for Splunk

In Cloud Foundry, create a syslog drain user-provided service instance as described in [Using Third-Party Log Management Services](#).

Choose one or more applications whose logs you want to drain to Splunk through the service.

Bind each app to the service instance and restart the app.

Note the GUID for each app, the IP address of the Loggregator host, and the port number for the service. Locate the port number in the syslog URL. For example:

```
syslog://logs.example.com:1234
```

2. Prepare Splunk for Cloud Foundry

For detailed information about the following tasks, see the [Splunk documentation](#).

Install the RFC5424 Syslog Technology Add-On

The Cloud Foundry Loggregator component formats logs according to the Syslog Protocol defined in [RFC 5424](#). Splunk does not parse log fields according to this protocol. To allow Splunk to correctly parse RFC 5424 log fields, install the Splunk [RFC5424 Syslog Technical Add-On](#).

Patch the RFC5424 Syslog Technology Add-On

1. SSH into the Splunk VM
2. Replace `/opt/splunk/etc/apps/rfc5424/default/transforms.conf` with a new `transforms.conf` file that consists of the following text:

```
[rfc5424 host]
DEST_KEY = MetaData:Host
REGEX = <\d+>\d{1}\s{1}\S+\s{1}(\S+)
FORMAT = host:$1

[rfc5424_header]
REGEX = <(\d+)>\d{1}\s{1}\S+\s{1}\S+\s{1}(\S+)\s{1}(\S+)\s{1}(\S+)
FORMAT = privel::$1 appname::$2 procid::$3 msgid::$4
MV_ADD = true
```

3. Restart Splunk

Create a TCP Syslog Data Input

Create a TCP Syslog Data Input in Splunk, with the following settings:

- **TCP port** is the port number you assigned to your log drain service
- **Set sourcetype** is `Manual`
- **Source type** is `rfc5424_syslog` (type this value into text field)
- **Index** is the index you created for your log drain service

Your Cloud Foundry syslog drain service is now integrated with Splunk.

3. Verify that Integration was Successful

Use Splunk to execute a query of the form:

```
sourcetype=rfc5424_syslog index=<the_index_you_created> appname=<app_guid>
```

To view logs from all apps at once, you can omit the `appname` field.

Verify that results rows contain the three Cloud Foundry-specific fields:

- **appname** — the GUID for the Cloud Foundry application
- **host** — the IP address of the Loggregator host
- **procid** — the Cloud Foundry component emitting the log

If the Cloud Foundry-specific fields appear in the log search results, integration is successful.

If logs from an app are missing, make sure that:

- The app is bound to the service and was restarted after binding
- The service port number matches the TCP port number in Splunk

Buildpacks

Buildpacks provide framework and runtime support for your applications. Buildpacks typically examine user-provided artifacts to determine what dependencies to download and how to configure applications to communicate with bound services.

When you push an application, Cloud Foundry automatically [detects](#) which buildpack is required and installs it on the Droplet Execution Agent (DEA) where the application needs to run.

Cloud Foundry deployments often have limited access to dependencies. This occurs when the deploy is behind a firewall, or when administrators want to use local mirrors and proxies. In these circumstances, Cloud Foundry provides a [Buildpack Packager ↗](#) application.

See also the [Supported Binary Dependencies](#) topic.

System Buildpacks

This table describes Cloud Foundry system buildpacks. Each buildpack row lists supported languages and frameworks and includes a GitHub repo link. Specific buildpack names also link to additional documentation.

Name	Other Supported Languages and Frameworks	GitHub Repo
Java	Grails, Play, Spring, or any other JVM-based language or framework	Java source ↗
Node.js	Node or JavaScript	Node.js source ↗
Ruby	Ruby, Rack, Rails, or Sinatra	Ruby source ↗
Binary	NA	Binary source ↗
Go	NA	Go source ↗
PHP	NA	PHP source ↗
Python	NA	Python source ↗
Staticfile	HTML, CSS, or JavaScript	Staticfile source ↗

Other Buildpacks

If your application uses a language or framework that Cloud Foundry buildpacks do not support, you can try the following:

- Customize an existing buildpack

 **Note:** A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. To merge upstream patches to your custom buildpack, use the approach that Github recommends for [syncing a fork ↗](#).

- [Write](#) your own buildpack
- Use a [Cloud Foundry Community Buildpack ↗](#)
- Use a [Heroku Third-Party Buildpack ↗](#)

Buildpack Detection

When you push an app, Cloud Foundry determines which buildpack to use. Each buildpack has a position in the detection priority list. Cloud Foundry first checks whether the buildpack in position 1 is right for the app. If the position 1 buildpack is not appropriate, Cloud Foundry moves on to the position 2 buildpack. This goes on until Cloud Foundry finds the correct buildpack. Otherwise, `cf push` fails with an `Unable to detect a supported application type` error.

Each buildpack contains a detect script. Cloud Foundry determines whether a buildpack is appropriate for an app by running that buildpack's detect script against the app.

By default, the three systems buildpacks occupy the first three positions on the detection priority list. Since running detect scripts takes time, an administrator can decrease the average time required to push apps by making sure that the most frequently used buildpacks occupy the lowest positions on the list.

Custom Buildpacks

Buildpacks are a convenient way of packaging framework and/or runtime support for your application.

For example, by default Cloud Foundry does not support Haskell. Using a buildpack for Haskell allows you to add support for it at the deployment stage. When you push an application written using Haskell, the required buildpack is automatically installed on the Cloud Foundry Droplet Execution Agent (DEA) where the application will run.

Note: A common development practice for custom buildpacks is to fork existing buildpacks and sync subsequent patches from upstream. To merge upstream patches to your custom buildpack, use the approach that Github recommends for [syncing a fork ↗](#).

Custom Buildpacks

The structure of a buildpack is straightforward. A buildpack repository contains three main scripts, situated in a folder named `bin`.

bin/detect

The `detect` script is used to determine whether or not to apply the buildpack to an application. The script is called with one argument, the build directory for the application. It returns an exit code of `0` if the application can be supported by the buildpack. If the script does return `0`, it should also print a framework name to STDOUT.

Shown below is an example `detect` script written in Ruby that checks for a Ruby application based on the existence of a `Gemfile`:

```
#!/usr/bin/env ruby

gemfile_path = File.join ARGV[0], "Gemfile"

if File.exist?(gemfile_path)
  puts "Ruby"
  exit 0
else
  exit 1
end
```

Note that cf CLI displays the returned STDOUT “framework name” when printing the details of an application. Buildpack authors are therefore encouraged to include additional details in this returned framework name, such as buildpack versioning information, and a detailed list of configured frameworks and their associated versions. The following is an example of the detailed information returned by the Java buildpack:

```
java-buildpack=v3.0-https://github.com/cloudfoundry/java-buildpack.git#3bd15e1 open-jdk-jre=1.8.0_45 spring-auto-reconfiguration=1.7.0_RELEASE
```

bin/compile

The `compile` script builds the droplet that will be run by the DEA and will therefore contain all the components necessary to run the application.

The script is run with two arguments, the build directory for the application and the cache directory, which is a location the buildpack can use to store assets during the build process.

During execution of this script all output sent to STDOUT will be relayed via the cf CLI back to the end user. The generally accepted pattern for this is to break out this functionality into a ‘language_pack’. A good example of this can be seen at [https://github.com/cloudfoundry/heroku-buildpack-ruby/blob/master/lib/language_pack/ruby.rb ↗](https://github.com/cloudfoundry/heroku-buildpack-ruby/blob/master/lib/language_pack/ruby.rb)

A simple `compile` script is shown below:

```
#!/usr/bin/env ruby

#sync output
$stdout.sync = true

build_path = ARGV[0]
cache_path = ARGV[1]

install_ruby

private

def install_ruby
  puts "Installing Ruby"

  # !!! build tasks go here !!!
  # download ruby to cache_path
  # install ruby
end
```

bin/release

The `release` script provides feedback metadata back to Cloud Foundry indicating how the application should be executed. The script is run with one argument, the build location of the application. The script must generate a YAML file in the following format:

```
config_vars:
  name: value
default_process_types:
  web: commandLine
```

Where `config_vars` is an optional set of environment variables that will be defined in the environment in which the application is executed. `default_process_types` indicates the type of application being run and the command line used to start it. At this time only `web` type of applications are supported.

The following example shows what a Rack application's `release` script might return:

```
config_vars:
  RACK_ENV: production
default_process_types:
  web: bundle exec rackup config.ru -p $PORT
```

Packaging Custom Buildpacks

Partially or completely disconnected environments

Cloud Foundry buildpacks work with limited or no internet connectivity. A Cloud Foundry operator can enable the same support in your custom buildpack by using the [Buildpack Packager](#) application.

Using buildpack-packager

1. Create a manifest.yml in your buildpack.
2. Run the packager in cached mode: `buildpack-packager cached`

The packager will add (almost) everything in your buildpack directory into a zip file. It will exclude anything marked for exclusion in your manifest.

In cached mode, the packager will download and add dependencies as described in the manifest.

Option Flags

```
--force-download
```

By default, buildpack-packager stores the dependencies that it downloads while building a cached buildpack in a local cache at `~/.buildpack-packager`. This is in order to avoid re-downloading them when repackaging similar buildpacks. Running `buildpack-packager cached` with the `--force-download` option will force the packager to download dependencies from the s3 host and ignore the local cache. When packaging an uncached buildpack, `--force-download` does nothing.

```
-use-custom-manifest
```

If you would like to include a different manifest file in your packaged buildpack, you may call buildpack-packager with the `--use-custom-manifest PATH/TO/MANIFEST.YML` option. buildpack-packager will generate a buildpack with the specified manifest. If you are building a cached buildpack, buildpack-packager will vendor dependencies from the specified manifest as well.

You can find more documentation at the [buildpack-packager Github repo ↗](#).

Uploading the buildpack

Once you have packaged your buildpack using `buildpack-packager`, you can upload the resulting `.zip` file to your instance of Cloud Foundry:

```
cf create-buildpack BUILDPACK_NAME BUILDPACK_ZIP_PATH POSITION
```

You can find more documentation at the [buildpack admin guide ↗](#).

Deploying With a Custom Buildpack

Once a custom buildpack has been created and pushed to a public git repository, the git URL can be passed via the cf CLI when pushing an application.

For example, for a buildpack that has been pushed to Github:

```
$ cf push my-new-app -b git://github.com/johndoe/my-buildpack.git
```

Alternatively, you can use a private git repository, with https and username/password authentication, as follows:

```
$ cf push my-new-app -b https://username:password@github.com/johndoe/my-buildpack.git
```

By default, Cloud Foundry uses the master branch of the buildpack's git repository. You can specify a different branch using the git url as shown in the following example:

```
$ cf push my-new-app -b https://username:password@github.com/johndoe/my-buildpack.git#my-branch-name
```

The application will then be deployed to Cloud Foundry, and the buildpack will be cloned from the repository and applied to the application (provided that the `detect` script returns `0`).

Packaging Dependencies for Offline Buildpacks

This topic describes the dependency storage options available to developers creating custom offline buildpacks.

Package dependencies in the buildpack

The simplest way to package dependencies in a custom buildpack is to keep the dependencies in your buildpack source. However, this is strongly discouraged. Keeping the dependencies in your source consumes unnecessary space.

To avoid keeping the dependencies in source control, load the dependencies into your buildpack and provide a script for the operator to create a zipfile of the buildpack.

For example, the operator might complete the following process:

```
# Clones your buildpack
$ git clone http://your.git/repo
$ cd SomeBuildPackName

# Creates a zipfile using your script
$ ./SomeScriptName
----> downloading-dependencies.... done
----> creating zipfile: ZippedBuildPackName.zip

# Adds the buildpack zipfile to the Cloud Foundry instance
$ cf create-buildpack SomeBuildPackName ZippedBuildPackName.zip 1
```

Pros

- Least complicated process for operators
- Least complicated maintenance process for buildpack developers

Cons

- Cloud Foundry admin buildpack uploads are limited to 1 GB, so the dependencies might not fit
- Security and functional patches to dependencies require updating the buildpack

Package selected dependencies in the buildpack

This is a variant of the [package dependencies in the buildpack](#) method described above. In this variation, the administrator edits a configuration file such as `dependencies.yml` to include a limited subset of the buildpack dependencies, then packages and uploads the buildpack.

 **Note:** This approach is strongly discouraged. Please see the Cons section below for more information.

The administrator completes the following steps:

```
# Clones your buildpack
$ git clone http://your.git/repo
$ cd

$ # Selects dependencies
$ vi dependencies.yml # Or copy in a preferred config

$ # Builds a package using your script
$ ./package
----> downloading-dependencies.... done
----> creating zipfile: cobol_buildpack.zip

$ # Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
```

Pros

- Possible to avoid the Cloud Foundry admin buildpack upload size limit in one of two ways:
 - If the administrator chooses a limited subset of dependencies
 - If the administrator maintains different packages for different dependency sets

Cons

- More complex for buildpack maintainers
- Security updates to dependencies require updating the buildpack
- Proliferation of buildpacks that require maintenance:
 - For each configuration, there is an update required for each security patch
 - Culling orphan configurations may be difficult or impossible
 - Administrators need to track configurations and merge them with updates to the buildpack
 - May result in with a different config for each app

Rely on a local mirror

In this method, the administrator provides a compatible file store of dependencies. When running the buildpack, the administrator specifies the location of the file store. The buildpack should handle missing dependencies gracefully.

The administrator completes the following process:

```
# Clones your buildpack
$ git clone http://your.git/repo
$ cd

# Builds a package using your script
$ ./package https://our.internal.site/dependency/repo
----> creating zipfile: cobol_buildpack.zip

# Adds the buildpack to the Cloud Foundry instance
$ cf create-buildpack cobol-buildpack cobol_buildpack.zip 1

$ # Pushes an app using your buildpack
$ cd ~/my_app
$ cf push my-cobol-webapp -b cobol-buildpack
----> deploying app
----> downloading dependencies:
https://our.internal.site/dependency/repo/dep1.tgz.... done
https://our.internal.site/dependency/repo/dep2.tgz.... WARNING: dependency not found!
```

Pros

- Avoids the Cloud Foundry admin buildpack upload size limit
- Leaves the administrator completely in control of providing dependencies
- Security and functional patches for dependencies can be maintained separately on the mirror given the following conditions:
 - The buildpack is designed to use newer semantically versioned dependencies
 - Buildpack behavior does not change with the newer functional changes

Cons

- The administrator needs to set up and maintain a mirror
- The additional config option presents a maintenance burden

Binary Buildpack

Use the binary buildpack for running arbitrary binary web servers.

For more information, see the [buildpack documentation](#).

Usage

Unlike most other Cloud Foundry buildpacks, you must specify the binary buildpack in order to use it in staging your binary file. On a command line, use `cf push APP-NAME` with the `-b` option to specify the buildpack.

For example:

```
$ cf push my_app -b https://github.com/cloudfoundry/binary-buildpack.git
```

You can provide Cloud Foundry with the shell command to execute your binary in the following two ways:

- **Procfile:** In the root directory of your app, add a `Procfile` that specifies a `web` task:

```
web: ./app
```

- **Command line:** Use `cf push APP-NAME` with the `-c` option:

```
$ cf push my_app -c './app' -b binary-buildpack
```

Compiling your Binary

Cloud Foundry expects your binary to bind to the port specified by the `PORT` environment variable.

The following example in [Go](#) binds a binary to the `PORT` environment variable:

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %s", "world!")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(": "+os.Getenv("PORT"), nil)
}
```

Your binary should run without any additional runtime dependencies on the cflinuxfs2 or lucid64 root filesystem (rootfs). Any such dependencies should be statically linked to the binary.

To boot a docker container running the cflinuxfs2 filesystem, run the following command:

```
$ docker run -it cloudfoundry/cflinuxfs2 bash
```

To boot a docker container running the lucid64 filesystem, run the following command:

```
$ docker run -it cloudfoundry/lucid64 bash
```

To compile the above Go application on the rootfs, golang must be installed. `apt-get install golang` and `go build app.go` will produce an `app` binary.

When deploying your binary to Cloud Foundry, use `cf push` with the `-s` option to specify the root filesystem it should

run against.

```
$ cf push my_app -s (cflinuxfs2|lucid64)
```

To run docker on Mac OS X, we recommend [boot2docker](#).

Supported Binary Dependencies

Each buildpack only supports the stable patches for each dependency listed in the buildpack's `manifest.yml` and also in its GitHub releases page. For example, see the [php-buildpack releases page](#).

If you try to use a binary that is not currently supported, staging your app fails with the following error message:

```
Could not get translated url, exited with: DEPENDENCY_MISSING_IN_MANIFEST:  
...  
!  
!     exit  
!  
Staging failed: Buildpack compilation step failed
```

Java Buildpack

Use the Java buildpack with applications written in Grails, Play, Spring or any other JVM-based language or framework.

See the following topics:

- [Getting Started Deploying Grails Apps](#)
- [Getting Started Deploying Ratpack Apps](#)
- [Getting Started Deploying Spring Apps](#)
- [Tips for Java Developers](#)
- [Configure Service Connections for Grails](#)
- [Configure Service Connections for Play](#)
- [Configure Service Connections for Spring](#)
- [Cloud Foundry Eclipse Plugin](#)
- [Cloud Foundry Java Client Library](#)
- [Build Tool Integration](#)

The source for the buildpack is [here](#).

Buildpack Logging and Application Logging

The buildpack only runs during the staging process, and therefore only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

The Java buildpack source documentation states:

- The Java buildpack logs all messages, regardless of severity to `<app dir>/java-buildpack.log`. It also logs messages to `$stderr`, filtered by a configured severity.
- If the buildpack fails with an exception, the exception message is logged with a log level of `ERROR` whereas the exception stack trace is logged with a log level of `DEBUG` to prevent users from seeing stack traces by default.

Once staging is done, the buildpack stops logging. Application logging is a separate concern.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. See [Application Logging in Cloud Foundry](#).

Getting Started Deploying Grails Apps

This guide is intended to walk you through deploying a Grails app to Elastic Runtime. If you experience a problem following the steps below, check the [Known Issues](#) topic or refer to the [Troubleshooting Application Deployment and Health](#) topic.

Sample App Step

If you want to go through this tutorial using the sample app, run

```
git clone https://github.com/cloudfoundry-samples/pong_matcher_grails.git
```

 to clone the `pong_matcher_grails` app from GitHub, and follow the instructions in the Sample App Step sections.

 **Note:** Ensure that your Grails app runs locally before continuing with this procedure.

Deploy a Grails Application

This section describes how to deploy a Grails application to Elastic Runtime.

Prerequisites

- A Grails app that runs locally on your workstation
- Intermediate to advanced Grails knowledge
- The [cf Command Line Interface \(CLI\)](#)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

 **Note:** You can develop Grails applications in Groovy, Java 7 or 8, or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7. Refer to Step 8: Configure the Deployment Manifest.

Step 1: Declare App Dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle, Grails, and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	<code>build.gradle</code>	Gradle User Guide
Grails	<code>BuildConfig.groovy</code>	Grails: Configuration - Reference Documentation
Maven	<code>pom.xml</code>	Apache Maven Project Documentation

Sample App Step

You can skip this step. The `pong_matcher_grails/app/grails-app/conf/BuildConfig.groovy` file contains the dependencies for the `pong_matcher_grails` sample app, as the example below shows.

```
dependencies {  
    // specify dependencies here under either 'build', 'compile', 'runtime', 'test' or 'provided' scopes e.g.  
    // runtime 'mysql:mysql-connector-java:5.1.29'  
    // runtime 'org.postgresql:postgresql:9.3-1101-jdbc41'  
    test "org.grails:grails-datastore-test-support:1.0-grails-2.4"  
    runtime 'mysql:mysql-connector-java:5.1.33'  
}
```

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory

allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP_NAME` command to see memory utilization.

Sample App Step

You can skip this step. In the `manifest.yml` of the `pong_matcher_grails` sample app, the `memory` sub-block of the `applications` block allocates 1 GB to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step

You can skip this step. The `pong_matcher_grails` sample app declares a MySQL JDBC driver in the `pong_matcher_grails/app/grails-app/conf/DataSource.groovy` file because the app uses ClearDB, which is a database-as-service for MySQL-powered apps. The example below shows this declaration.

```
dataSource {  
    pooled = true  
    jmxExport = true  
    driverClassName = "com.mysql.jdbc.Driver"  
    dialect = org.hibernate.dialect.MySQL5InnoDBDialect  
    uri = new URI(System.env.DATABASE_URL ?: "mysql://foo:bar@localhost")  
    username = uri.userInfo ? uri.userInfo.split(":")[0] : ""  
    password = uri.userInfo ? uri.userInfo.split(":")[1] : ""  
    url = "jdbc:mysql://" + uri.host + uri.path  
  
    properties {  
        dbProperties {  
            autoReconnect = true  
        }  
    }  
}
```

Step 4: (Optional) Configure a Procfile

A *Procfile* enables you to declare required runtime processes, called *process types*, for your web app. Process managers in a server use the process types to run and manage the workload.

When you deploy, Elastic Runtime determines if a Procfile exists and uses the Procfile to configure your app with the specified settings.

In a Procfile, you declare one process type per line and use the following syntax, as shown in the example below:

- `PROCESS_TYPE` is the command name in the format of an alphanumeric string.
- `COMMAND` is the command line to launch the process.

Example process type syntax:

```
PROCESS_TYPE: COMMAND
```

You create a Procfile and add a command line for a `web` process type. Store the Procfile in the root directory of your app. The example shows a process type that starts the launch script created by the build process.

Example Procfile:

```
web: build/install/MY-PROJECT-NAME/bin/MY-PROJECT-NAME]
```

Sample App Step

You can skip this step. The `pong_matcher_grails` app does not require a Procfile.

Step 5: Create and Bind a Service Instance for a Grails Application

This section describes using the CLI to configure a ClearDB managed service instance for an app. You can use either the CLI or the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the [Services Overview](#) topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `cleardb`, database-as-a-service for MySQL-powered apps and `elephantsql` PostgreSQL as a Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service      plans      description
...
cleardb    spark, boost, amp      Highly available MySQL for your apps
...
elephantsql  turtle, panda, elephant  PostgreSQL as a Service
...
```

Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

Sample App Step

Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `mysql` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql
Creating service mysql in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, Elastic Runtime writes information about the service instance to the `VCAP_SERVICES` app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the [Apps Manager](#).

Sample App Step

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---  
applications:  
...  
services:  
- mysql
```

Step 6: Configure the Deployment Manifest

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the [Deploying with Application Manifests](#) topic for more information.

Sample App Step

You can skip this step. The `manifest.yml` file for the `pong_matcher_grails` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and Target the API Endpoint

Run `cf login -a API_ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is [the URL of the Cloud Controller in your Elastic Runtime instance](#).

Sample App Step

You must do this step to run the sample app.

Step 8: Deploy the Application

 **Note:** You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

 **Note:** You must deploy the `.war` artifact for a Grails app, and you must include the path to the `.war` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the [Grails section](#) in the Tips for Java Developers topic.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `example.com`. For example: `cf push my-app` creates the URL `my-app.example.com`.

The URL for your app must be unique from other apps that Elastic Runtime hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

1. Change to the `app` directory, and run `./grails war` to build the app.
2. Run `cf push pong_matcher_grails -n HOST_NAME` to push the app.

Example: `cf push pong_matcher_grails -n my-grails-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_grails` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and follows the instructions in the manifest to start two instances of the app, allocating 1 GB of memory between the instances. After the instances start, the output displays their health and status.

```
$ cf push pong_matcher_grails -n my-grails-app
Using manifest file /Users/example/workspace/pong_matcher_grails/app/manifest.yml

Creating app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route my-grails-app.cfapps.io...
OK

Binding my-grails-app.cfapps.io to pong_matcher_grails...
OK

Uploading pong_matcher_grails...
Uploading app files from: /Users/example/workspace/pong_matcher_grails/app/target/pong_matcher_grails-0.1.war
Uploading 4.8M, 704 files
OK
Binding service mysql to app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK
----> Downloaded app package (38M)
----> Java Buildpack Version: v2.5 | https://github.com/cloudfoundry/java-buildpack.git#840500e
----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz
      Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto
      Modifying /WEB-INF/web.xml for Auto Reconfiguration
----> Downloading Tomcat Instance 8.0.14 from https://download.run.pivotal.io/tomcat/tomcat-8.0.14.tar.gz (0.4s)
      Expanding Tomcat to .java-buildpack/tomcat (0.1s)
----> Downloading Tomcat Lifecycle Support 2.4.0 RELEASE from https://download.run.pivotal.io/tomcat-lifecycle-support/tom
----> Downloading Tomcat Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-logging-support/tomcat-
----> Downloading Tomcat Access Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-access-logging-s

----> Uploading droplet (83M)

0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
0 of 2 instances running, 2 starting
2 of 2 instances running

App started

Showing health and status for app pong_matcher_grails in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 2/2
usage: 1G x 2 instances
urls: my-grails-app.cfapps.io

      state      since          cpu    memory      disk
#0  running   2014-11-10 05:07:33 PM  0.0%  686.4M of 1G  153.6M of 1G
#1  running   2014-11-10 05:07:36 PM  0.0%  677.2M of 1G  153.6M of 1G
```

Step 9: Test Your Deployed App

You've deployed an app to Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the [Manage Your Application with the cf CLI](#) section for more information. See also [Using the Apps Manager](#).

Sample App Step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE_APP_URL`, substituting the URL for your app for `SAMPLE_APP_URL`.

2. To clear the database from any previous tests, run:

```
curl -v -X DELETE $HOST/all
```

You should get a response of 200.

3. To request a match as "andrew", run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'
```

You should again get a response of 200.

4. To request a match as a different player, run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'
```

5. To check the status of the first match request, run:

```
curl -v -X GET $HOST/match_requests/firstrequest
```

The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d ' { "match_id": "MATCH_ID", "winner": "andrew", "loser": "navratilova" }'
```

You should receive a 201 Created response.

Alternative Methods for Pushing Apps

Integrate a Plugin for Your Build Tool

Elastic Runtime provides plugins for Maven and Gradle. You can deploy and manage your apps using Maven or Gradle command-line syntax and configure security credentials.

For more information, refer to the [Build Tool Integration](#) topic.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf Command Line Interface (CLI) topics, especially the [Getting Started with cf CLI v6](#) topic.

 **Note:** You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only an Elastic Runtime administrator can run. If you are not an Elastic Runtime administrator, the following message displays for these types of commands:

```
error code: 10003, message: You are not authorized to perform the requested action
```

For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the [Understanding Apps Manager Permissions](#) topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the [Troubleshooting Application Deployment and Health](#) topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on Elastic Runtime. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or the [Apps Manager](#), or you might have to delete the app and redeploy.

App Requires a Unique URL

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Getting Started Deploying Ratpack Apps

This guide is intended to walk you through deploying a Ratpack app to Elastic Runtime. If you experience a problem following the steps below, check the [Known Issues](#) topic or refer to the [Troubleshooting Application Deployment and Health](#) topic.

Sample App Step

If you want to go through this tutorial using the sample app, run

```
git clone https://github.com/cloudfoundry-samples/pong_matcher_groovy.git
```

 to clone the `pong_matcher_groovy` app from GitHub, and follow the instructions in the Sample App Step sections.

 **Note:** Ensure that your Ratpack app runs locally before continuing with this procedure.

Deploy a Ratpack Application

This section describes how to deploy a Ratpack application to Elastic Runtime.

Prerequisites

- A Ratpack app that runs locally on your workstation
- Intermediate to advanced Ratpack knowledge
- The [cf Command Line Interface \(CLI\)](#)
- JDK 1.7 or 1.8 for Java 7 or 8 configured on your workstation

 **Note:** You can develop Ratpack applications in Java 7 or 8 or any JVM language. The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to JDK 1.7. Refer to Step 8: Configure the Deployment Manifest.

Step 1: Declare App Dependencies

Declare all the dependency tasks for your app in the build script of your chosen build tool. The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	<code>build.gradle</code>	Gradle User Guide
Maven	<code>pom.xml</code>	Apache Maven Project Documentation

Sample App Step

You can skip this step. The `build.gradle` file contains the dependencies for the `pong_matcher_groovy` sample app, as the example below shows.

```
dependencies {  
    // SpringLoaded enables runtime hot reloading.  
    // It is not part of the app runtime and is not shipped in the distribution.  
    springloaded "org.springframework:springloaded:1.2.0.RELEASE"  
  
    // Default SLF4J binding. Note that this is a blocking implementation.  
    // See here for a non blocking appender http://logging.apache.org/log4j/2.x/manual/async.html  
    runtime 'org.slf4j:slf4j-simple:1.7.7'  
  
    compile group: 'redis.clients', name: ' jedis', version: '2.5.2', transitive: true  
  
    testCompile "org.spockframework:spock-core:0.7-groovy-2.0"  
}
```

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP_NAME` command to see memory utilization.

Sample App Step

You can skip this step. In the `manifest.yml` of the `pong_matcher_groovy` sample app, the `memory` sub-block of the `applications` block allocates 512 MB to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step

You can skip this step. The `pong_matcher_groovy` sample app does not require a JDBC driver.

Step 4: (Optional) Configure a Procfile

A *Procfile* enables you to declare required runtime processes, called *process types*, for your web app. Process managers in a server use the process types to run and manage the workload.

When you deploy, Elastic Runtime determines if a Procfile exists and uses the Procfile to configure your app with the specified settings.

In a Procfile, you declare one process type per line and use the following syntax, as shown in the example below:

- `PROCESS_TYPE` is the command name in the format of an alphanumeric string.
- `COMMAND` is the command line to launch the process.

Example process type syntax:

```
PROCESS_TYPE: COMMAND
```

You create a Procfile and add a command line for a `web` process type. Store the Procfile in the root directory of your app. The example shows a process type that starts the launch script created by the build process.

Example Procfile:

```
web: build/install/MY-PROJECT-NAME/bin/MY-PROJECT-NAME
```

Sample App Step

You can skip this step. The `pong_matcher_groovy` app does not require a Procfile.

Step 5: Create and Bind a Service Instance for a Ratpack Application

This section describes using the CLI to configure a Redis managed service instance for an app. You can use either the CLI or the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the [Services Overview](#) topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans available to you.

The example shows two of the available managed database-as-a-service providers and their offered plans: `elephantsql` PostgreSQL as a Service and `rediscloud` Enterprise-Class Redis for Developers.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service      plans      description
...
elephantsql  turtle, panda, elephant    PostgreSQL as a Service
...
rediscloud   25mb, 100mb, 1gb, 10gb, 50gb Enterprise-Class Redis for Developers
...
```

Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

Sample App Step

Run `cf create-service rediscloud 25mb baby-redis`. This creates a service instance named `baby-redis` that uses the `rediscloud` service and the `25mb` plan, as the example below shows.

```
$ cf create-service rediscloud 25mb baby-redis
Creating service baby-redis in org Cloud-Apps / space development as clouduser@example.com...
OK
```

Bind a Service Instance

When you bind an app to a service instance, Elastic Runtime writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` sub-block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App step.

You can also bind a service using the [Apps Manager](#).

Sample App Step

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---
applications:
---
```

```
services:  
- baby-redis
```

Step 6: Configure the Deployment Manifest

You can specify deployment options in the `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the [Deploying with Application Manifests](#) topic for more information.

Sample App Step

You can skip this step. The `manifest.yml` file for the `pong_matcher_groovy` sample app does not require any additional configuration to deploy the app.

Step 7: Log in and Target the API Endpoint

Run `cf login -a API_ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is [the URL of the Cloud Controller in your Elastic Runtime instance ↗](#).

Sample App Step

You must do this step to run the sample app.

Step 8: Deploy the Application

 **Note:** You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.distZip` to deploy your application.

 **Note:** You must deploy the `.distZip` artifact for a Ratpack app, and you must include the path to the `.distZip` file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. For more information, refer to the [Tips for Java Developers](#) topic.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `example.com`. For example: `cf push my-app` creates the URL `my-app.example.com`.

The URL for your app must be unique from other apps that Elastic Runtime hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

1. Change to the `app` directory, and run `./gradlew distZip` to build the app.
2. Run `cf push pong_matcher_groovy -n HOST_NAME` to push the app.

Example: `cf push pong_matcher_groovy -n groovy-ratpack-app`

Note: You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_groovy` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `baby-redis` service and follows the instructions in the manifest to start one instance of the app with 512 MB. After the app starts, the output displays the health and status of the app.

```
$ cf push pong_matcher_groovy -n groovy-ratpack-app
Using manifest file /Users/example/workspace/pong_matcher_groovy/app/manifest.yml

Creating app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route groovy-ratpack-app.cfapps.io...
OK

Binding groovy-ratpack-app.cfapps.io to pong_matcher_groovy...
OK

Uploading pong_matcher_groovy...
Uploading app files from: /Users/example/workspace/pong_matcher_groovy/app/build/distributions/app.zip
Uploading 138.2K, 18 files
OK
Binding service baby-redis to app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK
----> Downloaded app package (12M)
Cloning into '/tmp/buildpacks/java-buildpack'...
----> Java Buildpack Version: 9e096be | https://github.com/cloudfoundry/java-buildpack#9e096be
      Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.3s)
----> Uploading droplet (49M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_groovy in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 512M x 1 instances
urls: groovy-ratpack-app.cfapps.io

      state     since            cpu    memory      disk
#0   running   2014-10-28 04:48:58 PM  0.0%  193.5M of 512M  111.7M of 1G
```

Step 9: Test Your Deployed App

You've deployed an app to Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the [Manage Your Application with the cf CLI](#) section for more information. See also [Using the Apps Manager](#).

Sample App Step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE_APP_URL`, substituting the URL for your app for `SAMPLE_APP_URL`.

2. To clear the database from any previous tests, run:

```
curl -v -X DELETE $HOST/all
```

You should get a response of 200.

3. To request a match as "andrew", run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'
```

You should again get a response of `200`.

4. To request a match as a different player, run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'
```

5. To check the status of the first match request, run:

```
curl -v -X GET $HOST/match_requests/firstrequest
```

The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d '{ "match_id": "MATCH_ID", "winner": "andrew", "loser": "navratilova" }'
```

You should receive a `201 Created` response.

Alternative Methods for Pushing Apps

Integrate a Plugin for Your Build Tool

Elastic Runtime provides plugins for Maven and Gradle. You can deploy and manage your apps using Maven or Gradle command-line syntax and configure security credentials.

For more information, refer to the [Build Tool Integration](#) topic.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf Command Line Interface (CLI) topics, especially the [Getting Started with cf CLI v6](#) topic.

Note: You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only an Elastic Runtime administrator can run. If you are not an Elastic Runtime administrator, the following message displays for these types of commands:
`error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the [Understanding Apps Manager Permissions](#) topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the [Troubleshooting Application Deployment and Health](#) topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on Elastic Runtime. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or the [Apps Manager](#), or you might have to delete the app and redeploy.

App Requires a Unique URL

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long

URL, depending on the number of words that the app name includes.

Getting Started Deploying Spring Apps

This guide is intended to walk you through deploying a Spring app to Elastic Runtime. You can choose whether to push a sample app, your own app, or both.

If at any time you experience a problem following the steps below, try checking the [Known Issues](#) topic, or refer to the [Troubleshooting Application Deployment and Health](#) topic for more tips.

Sample App Step

If you want to go through this tutorial using the sample app, run

```
git clone https://github.com/cloudfoundry-samples/pong_matcher_spring
```

 to clone the `pong_matcher_spring` app from GitHub, and follow the instructions in the Sample App Step sections.

 **Note:** Ensure that your Spring app runs locally before continuing with this procedure.

Deploy a Spring Application

This section describes how to deploy your Spring application to Elastic Runtime.

Prerequisites

- A Spring app that runs locally on your workstation
- Intermediate to advanced Spring knowledge
- The [cf Command Line Interface \(CLI\)](#)
- JDK 1.6, 1.7, or 1.8 for Java 6, 7, or 8 configured on your workstation

 **Note:** The Cloud Foundry Java buildpack uses JDK 1.8, but you can modify the buildpack and the manifest for your app to compile to an earlier version. For more information, refer to the [Custom Buildpacks](#) topic.

Step 1: Declare App Dependencies

Be sure to declare all the dependency tasks for your app in the build script of your chosen build tool.

The [Spring Getting Started Guides](#) demonstrate features and functionality you can add to your app, such as consuming RESTful services or integrating data. These guides contain Gradle and Maven build script examples with dependencies. You can copy the code for the dependencies into your build script.

The table lists build script information for Gradle and Maven and provides documentation links for each build tool.

Build Tool	Build Script	Documentation
Gradle	<code>build.gradle</code>	Gradle User Guide
Maven	<code>pom.xml</code>	Apache Maven Project Documentation

Sample App Step

You can skip this step. The `pom.xml` file contains the dependencies for the `pong_matcher_spring` sample app, as the example below shows.

```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.jayway.jsonpath</groupId>
        <artifactId>json-path</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Step 2: Allocate Sufficient Memory

Use the `cf push -m` command to specify the amount of memory that should be allocated to the application. Memory allocated this way is done in preset amounts of `64M`, `128M`, `256M`, `512M`, `1G`, or `2G`. For example:

```
$ cf push -m 128M
```

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Sample App Step

You can skip this step. The Cloud Foundry Java buildpack uses settings declared in the sample app to allocate 1 GB of memory to the app.

Step 3: Provide a JDBC Driver

The Java buildpack does not bundle a JDBC driver with your application. If your application accesses a SQL RDBMS, you must do the following:

- Include the appropriate driver in your application.
- Create a dependency task for the driver in the build script for your build tool or IDE.

Sample App Step

You can skip this step. In the `pong_matcher_spring` sample app, the `src/main/resources/application.yml` file declares the JDBC driver, and the `pom.xml` file includes the JDBC driver as a dependency.

Step 4: Configure Service Connections for a Spring App

Elastic Runtime provides extensive support for creating and binding a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. For more information about creating and binding a service connection for your app, refer to the [Configure Service Connections for Spring](#) topic.

Sample App Step: Create a Service Instance

Run `cf create-service cleardb spark mysql`. This creates a service instance named `mysql` that uses the `cleardb` service and the `mysql` plan, as the example below shows.

```
$ cf create-service cleardb spark mysql  
Creating service mysql in org Cloud-Apps / space development as a.user@example.com....  
OK
```

Sample App Step: Bind a Service Instance

You can skip this step because the service instance is already bound. Open the `manifest.yml` file in a text editor to view the bound service instance information. Locate the file in the app root directory and search for the `services` sub-block in the `applications` block, as the example below shows.

```
---  
applications:  
...  
services:  
- mysql
```

Step 5: Configure the Deployment Manifest

You can specify deployment options in a manifest file `manifest.yml` that the `cf push` command uses when deploying your app.

Refer to the [Deploying with Application Manifests](#) topic for more information.

Sample App Step

You can skip this step. The `manifest.yml` file for the `pong_matcher_spring` sample app does not require any additional configuration to deploy the app.

Step 6: Log in and Target the API Endpoint

Run `cf login -a API-ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is [the URL of the Cloud Controller in your Elastic Runtime instance ↗](#).

Sample App Step

You must do this step to run the sample app.

Step 7: Deploy Your Application

Note: You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP-NAME -p PATH-TO-FILE.war` to deploy your application.

Note: Most Spring apps include an artifact, such as a `.jar`, `.war`, or `.zip` file. You must include the path to this file in the `cf push` command using the `-p` option if you do not declare the path in the `applications` block of the manifest file. The example shows how to specify a path to the `.war` file for a Spring app. Refer to the [Tips for Java Developers](#) topic for CLI examples for specific build tools, frameworks, and languages that create an app with an artifact.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `example.com`. For example: `cf push my-app` creates the URL `my-app.example.com`.

The URL for your app must be unique from other apps that Elastic Runtime hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app
- `--random-route` to create a URL that includes the app name and random words
- `cf help push` to view other options for this command

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

1. Run `brew install maven`.
2. Change to the `app` directory, and run `mvn package` to build the app.
3. Run `cf push pong_matcher_spring -n HOSTNAME` to push the app.

Example: `cf push pong_matcher_spring -n my-spring-app`

 **Note:** You do not have to include the `-p` flag when you deploy the sample app. The sample app manifest declares the path to the archive that `cf push` uses to upload the app files.

The example below shows the terminal output of deploying the `pong_matcher_spring` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `mysql` service and starts one instance of the app with 1 GB of memory. After the app starts, the output displays the health and status of the app.

```
$ cf push pong_matcher_spring -n spring1119
Using manifest file /Users/example/workspace/pong_matcher_spring/manifest.yml

Creating app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

Creating route spring1119.cfapps.io...
OK

Binding spring1119.cfapps.io to pong_matcher_spring...
OK

Uploading pong_matcher_spring...
Uploading app files from: /Users/example/workspace/pong_matcher_spring/target/pong-matcher-spring-1.0.0.BUILD-SNAPSHOT.jar
Uploading 797.5K, 116 files
OK
Binding service mysql to app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

Starting app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK
----> Downloaded app package (25M)
----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz
      Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
----> Downloading Spring Auto Reconfiguration 1.5.0 RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto
----> Uploading droplet (63M)

0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_spring in org Cloud-Apps / space development as a.user@example.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: spring1119.cfapps.io

      state      since          cpu    memory      disk
#0   running   2014-11-19 12:29:27 PM  0.0%  553.6M of 1G  127.4M of 1G
```

Step 8: Test Your Deployed App

You've deployed an app to Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime account. For example, you can edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the [Manage Your Application with the cf CLI](#) section for more information. See also [Using the Apps Manager](#).

Sample App Step

To test the sample app, do the following:

1. To export the test host, run `export HOST=SAMPLE-APP-URL`, substituting the URL for your app for `SAMPLE-APP-URL`.

2. To clear the database from any previous tests, run:

```
curl -v -X DELETE $HOST/all
```

You should get a response of 200.

3. To request a match as "andrew", run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/firstrequest -d '{"player": "andrew"}'
```

You should again get a response of 200.

4. To request a match as a different player, run:

```
curl -v -H "Content-Type: application/json" -X PUT $HOST/match_requests/secondrequest -d '{"player": "navratilova"}'
```

5. To check the status of the first match request, run:

```
curl -v -X GET $HOST/match_requests/firstrequest
```

The last line of the output shows the `match_id`.

6. Replace `MATCH_ID` with the `match_id` value from the previous step in the following command:

```
curl -v -H "Content-Type: application/json" -X POST $HOST/results -d '{ "match_id": "MATCH_ID", "winner": "andrew", "loser": "navratilova" }'
```

You should receive a `201 Created` response.

Alternative Methods for Pushing Apps

Alternative Method 1: Integrate a Plugin for Your Build Tool

Elastic Runtime provides plugins for Maven and Gradle. You can deploy and manage your apps using Maven or Gradle command-line syntax and configure security credentials.

For more information, refer to the [Build Tool Integration](#) topic.

Alternative Method 2: Integrate the Cloud Foundry Eclipse Plugin for STS

Elastic Runtime provides an Eclipse plugin extension that enables you to deploy and manage Spring applications on a Cloud Foundry instance from the Spring Tool Suite (STS), version 3.0.0 and later. For more information, refer to the [Cloud Foundry Eclipse Plugin](#) topic. You must follow the instructions in the [Install to STS from the IDE Extensions Tab](#) and [Create a Cloud Foundry Server](#) sections before you can deploy and manage your apps with the plugin.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf Command Line Interface (CLI) topics, especially the [Getting Started with cf CLI v6](#) topic.

 **Note:** You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime administrator, the following message displays for these types of commands:

```
error code: 10003, message: You are not authorized to perform the requested action
```

For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the [Understanding Apps Manager Permissions](#) topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the [Troubleshooting Application Deployment and Health](#) topic to learn more about troubleshooting.

App Deploy Fails

Even when the deploy fails, the app might exist on Elastic Runtime. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or the [Apps Manager](#), or you might have to delete the app and redeploy.

App Requires a Content-Type

If you specify a `Content-Encoding` header of `gzip` but do not specify a `Content-Type` within your application, Elastic Runtime might send a `Content-Type` of `application/x-gzip` to the browser. This scenario might cause the deploy to fail if it conflicts with the actual encoded content of your app. To avoid this issue, be sure to explicitly set `Content-Type` within your app.

App Requires a Unique URL

Elastic Runtime requires that each app that you deploy have a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime cannot successfully deploy the app. You can fix this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Tips for Java Developers

This page assumes you are using cf CLI v6.

Cloud Foundry can deploy a number of different JVM-based artifact types. For a more detailed explanation of what it supports, see the [Java Buildpack documentation](#).

Java Client Library

The Cloud Foundry Client Library provides a Java API for interacting with a Cloud Foundry instance. This library, `cloudfoundry-client-lib`, is used by the Cloud Foundry Maven plugin, the Cloud Foundry Gradle plugin, the [Cloud Foundry STS integration](#), and other Java-based tools.

For information about using this library, see the [Java Cloud Foundry Library](#) page.

Grails

Grails packages applications into WAR files for deployment into a Servlet container. To build the WAR file and deploy it, run the following:

```
$ grails prod war  
$ cf push my-application; -p target/my-application-version.war
```

Groovy

Groovy applications based on both [Ratpack](#) and a simple collection of files are supported.

Ratpack

Ratpack packages applications into two different styles; Cloud Foundry supports the `distZip` style. To build the ZIP and deploy it, run the following:

```
$ gradle distZip  
$ cf push my-application -p build/distributions/my-application.zip
```

Raw Groovy

Groovy applications that are made up of a [single entry point](#) plus any supporting files can be run without any other work. To deploy them, run the following:

```
$ cf push my-application
```

Java Main

Java applications with a `main()` method can be run provided that they are packaged as [self-executable JARs](#).

Note: If your application is not web-enabled, you must suppress route creation to avoid a “failed to start accepting connections” error. To suppress route creation, add `no-route: true` to the application manifest or use the `--no-route` flag with the `cf push` command.

For more information about the `no-route` attribute, see the [Deploying with Application Manifests](#) topic.

Maven

A Maven build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
$ mvn package  
$ cf push my-application -p target/my-application-version.jar
```

Gradle

A Gradle build can create a self-executable JAR. To build and deploy the JAR, run the following:

```
$ gradle build  
$ cf push my-application -p build/libs/my-application-version.jar
```

Play Framework

The [Play Framework](#) packages applications into two different styles. Cloud Foundry supports both the `staged` and `dist` styles. To build the `dist` style and deploy it, run the following:

```
$ play dist  
$ cf push my-application -p target/universal/my-application-version.zip
```

Spring Boot CLI

[Spring Boot](#) can run applications [comprised entirely of POGOs](#). To deploy then, run the following:

```
$ spring grab *.groovy  
$ cf push my-application
```

Servlet

Java applications can be packaged as Servlet applications.

Maven

A Maven build can create a Servlet WAR. To build and deploy the WAR, run the following:

```
$ mvn package  
$ cf push my-application -p target/my-application-version.war
```

Gradle

A Gradle build can create a Servlet WAR. To build and deploy the JAR, run the following:

```
$ gradle build  
$ cf push my-application -p build/libs/my-application-version.war
```

Binding to Services

Information about binding apps to services can be found on the following pages:

- [Service Bindings for Grails Applications](#)
- [Service Bindings for Play Framework Applications](#)
- [Service Bindings for Spring Applications](#)

Java Buildpack

For detailed information about using, configuring, and extending the Cloud Foundry Java buildpack, see

<https://github.com/cloudfoundry/java-buildpack>.

Design

The Java Buildpack is designed to convert artifacts that run on the JVM into executable applications. It does this by identifying one of the supported artifact types (Grails, Groovy, Java, Play Framework, Spring Boot, and Servlet) and downloading all additional dependencies needed to run. The collection of services bound to the application is also analyzed and any dependencies related to those services are also downloaded.

As an example, pushing a WAR file that is bound to a PostgreSQL database and New Relic for performance monitoring would result in the following:

```
Initialized empty Git repository in /tmp/buildpacks/java-buildpack/.git/
--> Java Buildpack source: https://github.com/cloudfoundry/java-buildpack#0928916a2dd78e9faf9469c558046eef09f60e5d
--> Downloading Open Jdk JRE 1.7.0_51 from
    http://.../openjdk/lucid/x86_64/openjdk-1.7.0_51.tar.gz (0.0s)
    Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.9s)
--> Downloading New Relic Agent 3.4.1 from
    http://.../new-relic/new-relic-3.4.1.jar (0.4s)
--> Downloading Postgresql JDBC 9.3.1100 from
    http://.../postgresql-jdbc/postgresql-jdbc-9.3.1100.jar (0.0s)
--> Downloading Spring Auto Reconfiguration 0.8.7 from
    http://.../auto-reconfiguration/auto-reconfiguration-0.8.7.jar (0.0s)
    Modifying /WEB-INF/web.xml for Auto Reconfiguration
--> Downloading Tomcat 7.0.50 from
    http://.../tomcat/tomcat-7.0.50.tar.gz (0.0s)
    Expanding Tomcat to .java-buildpack/tomcat (0.1s)
--> Downloading Buildpack Tomcat Support 1.1.1 from
    http://.../tomcat-buildpack-support/tomcat-buildpack-support-1.1.1.jar (0.1s)
--> Uploading droplet (57M)
```

Configuration

In most cases, the buildpack should work without any configuration. If you are new to Cloud Foundry, we recommend that you make your first attempts without modifying the buildpack configuration. If the buildpack requires some configuration, use [a fork of the buildpack](#).

Java and Grails Best Practices

Provide JDBC driver

The Java buildpack does not bundle a JDBC driver with your application. If your application will access a SQL RDBMS, include the appropriate driver in your application.

Allocate Sufficient Memory

If you do not allocate sufficient memory to a Java application when you deploy it, it may fail to start, or Elastic Runtime may terminate it. You must allocate enough memory to allow for the following:

- Java heap
- Metaspace, if using Java 8
- PermGen, if using Java 7 or earlier
- Thread stacks
- JVM overhead

The `config/open_jdk_jre.yml` file of the [Cloud Foundry Java buildpack](#) contains default memory size and weighting settings for the JRE. See the [Open JDK JRE README](#) on GitHub for an explanation of JRE memory sizes and weightings and how the Java buildpack calculates and allocates memory to the JRE for your app.

To configure memory-related JRE options for your app, you create a custom buildpack and specify this buildpack in your deployment manifest. For more information about configuring custom buildpacks and manifests, refer to the [Custom Buildpacks](#) and [Deploying with Application Manifests](#) topics.

When your app is running, you can use the `cf app APP-NAME` command to see memory utilization.

Troubleshoot Failed Upload

If your application fails to upload when you push it to Cloud Foundry, it may be for one of the following reasons:

- WAR is too large: An upload may fail due to the size of the WAR file. Cloud Foundry testing indicates WAR files as large as 250 MB upload successfully. If a WAR file larger than that fails to upload, it may be a result of the file size.
- Connection issues: Application uploads can fail if you have a slow Internet connection, or if you upload from a location that is very remote from the target Cloud Foundry instance. If an application upload takes a long time, your authorization token can expire before the upload completes. A workaround is to copy the WAR to a server that is closer to the Cloud Foundry instance, and push it from there.
- Out-of-date cf CLI client: Upload of a large WAR is faster and hence less likely to fail if you are using a recent version of the cf CLI. If you are using an older version of the cf CLI client to upload a large WAR, and having problems, try updating to the latest version of the cf CLI.
- Incorrect WAR targeting: By default, `cf push` uploads everything in the current directory. For a Java application, a plain `cf push` push will upload source code and other unnecessary files, in addition to the WAR. When you push a Java application, specify the path to the WAR:

```
$ cf push MY-APP -p PATH/TO/WAR-FILE
```

You can determine whether or not the path was specified for a previously pushed application by looking at the application deployment manifest, `manifest.yml`. If the `path` attribute specifies the current directory, the manifest will include a line like this:

```
path: .
```

To re-push just the WAR, either:

- Delete `manifest.yml` and push again, specifying the location of the WAR using the `-p` flag, or
- Edit the `path` argument in `manifest.yml` to point to the WAR, and re-push the application.

Debugging Java Apps on Cloud Foundry

Because of the way that Cloud Foundry deploys your applications and isolates them, it is not possible to connect to your application with the remote Java debugger. Instead, instruct the application to connect to the Java debugger on your local machine.

Here are the instructions for setting up remote debugging when using BOSH Lite or a CloudFoundry installation.

1. Open your project in [Eclipse](#).
2. Right-click on your project, go to **Debug as** and pick **Debug Configurations**.
3. Create a new **Remote Java Application**.
4. Make sure your project is selected, pick **Standard (Socket Listen)** from the **Connection Type** drop down and set a port. Make sure this port is open if you are running a firewall.
5. Click **Debug**.

The debugger should now be running. If you switch to the Debug perspective, you should see your application listed in the Debug panel and it should say `Waiting for vm to connect at port`.

Next, push your application to Cloud Foundry and instruct Cloud Foundry to connect to the debugger running on your local machine using the following instructions:

1. Edit your `manifest.yml` file. Set the instances count to 1. If you set this greater than one, multiple applications try to connect to your debugger.
2. Also in `manifest.yml`, add the `env` section and create a variable called `JAVA_OPTS`.
3. Add the remote debugger configuration to the `JAVA_OPTS` variable:
`-agentlib:jdwp=transport=dt_socket,address=YOUR-IP-ADDRESS:YOUR-PORT`.

4. Save the `manifest.yml` file.

5. Run `cf push`.

Upon completion, you should see that your application has started and is now connected to the debugger running in your IDE. You can now add breakpoints and interrogate the application just as you would if it were running locally.

Slow Starting Java or Grails Apps

Some Java and Grails applications do not start quickly, and the DEA health check for an application can fail if an application starts too slowly.

The current Java buildpack implementation sets the Tomcat `bindOnInit` property to `false`. This prevents Tomcat from listening for HTTP requests until an application has fully deployed.

If your application does not start quickly, the DEA health check may fail because it checks the health of the application before the application can accept requests. By default, the DEA health check fails after a timeout threshold of 60 seconds.

To resolve this issue, use `cf push APP-NAME -t TIMEOUT-THRESHOLD` with the `-t TIMEOUT-THRESHOLD` option to increase the timeout threshold. Specify `TIMEOUT-THRESHOLD` in seconds.

```
$ cf push my-app -t 180
```

Note: The timeout threshold cannot exceed 180 seconds. Specifying a timeout threshold greater than 180 seconds results in the following error:

```
Server error, status code: 400, error code: 100001, message: The app is invalid: health_check_timeout maximum_exceeded
```

If your Java or Grails application requires more than 180 seconds to start, you can fork the Java buildpack and change the value of `bindOnInit` to `true` in `resources/tomcat/conf/server.xml`. This change allows Tomcat to listen for HTTP requests before your application has fully deployed.

Note: Changing the value of `bindOnInit` to `true` allows the DEA health check of your application to pass even before your application has fully deployed. In this state, Cloud Foundry might pass requests to your application before your application can serve them.

Extension

The Java Buildpack is also designed to be easily extended. It creates abstractions for [three types of components](#) (containers, frameworks, and JREs) in order to allow users to easily add functionality. In addition to these abstractions, there are a number of [utility classes](#) for simplifying typical buildpack behaviors.

As an example, the New Relic framework looks like the following:

```
class NewRelicAgent < JavaBuildpack::Component::VersionedDependencyComponent

# @macro base_component_compile
def compile
  FileUtils.mkdir_p logs_dir

  download_jar
  @droplet.copy_resources
end

# @macro base_component_release
def release
  @droplet.java_opts
    .add_javaagent(@droplet.sandbox + jar_name)
    .add_system_property('newrelic.home', @droplet.sandbox)
    .add_system_property('newrelic.config.license_key', license_key)
    .add_system_property('newrelic.config.app_name', "#{application_name}")
    .add_system_property('newrelic.config.log_file_path', logs_dir)
end

protected

# @macro versioned_dependency_component_supports
def supports?
  @application.services.one_service? FILTER, 'licenseKey'
end

private

FILTER = /newrelic/.freeze

def application_name
  @application.details['application_name']
end

def license_key
  @application.services.find_service(FILTER)['credentials']['licenseKey']
end

def logs_dir
  @droplet.sandbox + 'logs'
end

end
```

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` as follows:

```
System.getenv("VCAP_SERVICES");
```

The environment variables you can access are those [defined by the DEA](#).

Configure Service Connections for Grails

Cloud Foundry provides extensive support for connecting a Grails application to services such as MySQL, Postgres, MongoDB, Redis, and RabbitMQ. In many cases, a Grails application running on Cloud Foundry can automatically detect and configure connections to services. For more advanced cases, you can control service connection parameters yourself.

Auto-Configuration

Grails provides plugins for accessing SQL (using [Hibernate](#)), [MongoDB](#), and [Redis](#) services. If you install any of these plugins and configure them in your `Config.groovy` or `DataSource.groovy` file, Cloud Foundry re-configures the plugin with connection information when your app starts.

If you were using all three types of services, your configuration might look like this:

```
environments {
  production {
    dataSource {
      url = 'jdbc:mysql://localhost/db?useUnicode=true&characterEncoding=utf8'
      dialect = org.hibernate.dialect.MySQLInnoDBDialect
      driverClassName = 'com.mysql.jdbc.Driver'
      username = 'user'
      password = "password"
    }
    grails {
      mongo {
        host = 'localhost'
        port = 27107
        databaseName = "foo"
        username = 'user'
        password = 'password'
      }
      redis {
        host = 'localhost'
        port = 6379
        password = 'password'
        timeout = 2000
      }
    }
  }
}
```

The `url`, `host`, `port`, `databaseName`, `username`, and `password` fields in this configuration will be overridden by the Cloud Foundry auto-reconfiguration if it detects that the application is running in a Cloud Foundry environment. If you want to test the application locally against your own services, you can put real values in these fields. If the application will only be run against Cloud Foundry services, you can put placeholder values as shown here, but the fields must exist in the configuration.

Manual Configuration

If you do not want to use auto-configuration, you can configure the Cloud Foundry service connections manually.

Follow the steps below to manually configure a service connection.

1. Add the `spring-cloud` library to the `dependencies` section of your `BuildConfig.groovy` file.

```
repositories {
  grailsHome()
  mavenCentral()
  grailsCentral()
  mavenRepo "http://repo.spring.io/milestone"
}

dependencies {
  compile "org.springframework.cloud:spring-cloud-cloudfoundry-connector:1.0.0.RELEASE"
  compile "org.springframework.cloud:spring-cloud-spring-service-connector:1.0.0.RELEASE"
}
```

Adding the `spring-cloud` library allows you to disable auto-configuration and use the `spring-cloud` API in your

`DataSource.groovy` file to set the connection parameters.

2. Add the following to your `grails-app/conf/spring/resources.groovy` file to disable auto-configuration:

```
beans = {
    cloudFactory(org.springframework.cloud.CloudFactory)
}
```

3. Add the following `imports` to your `DataSource.groovy` file to allow `spring-cloud` API commands:

```
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException
```

4. Add the following code to your `DataSource.groovy` file to enable Cloud Foundry's `getCloud` method to function locally or in other environments outside of a cloud.

```
def cloud = null
try {
    cloud = new CloudFactory().cloud
} catch (CloudException) {}
```

5. Use code like the following to access the cloud object:

```
def dbInfo = cloud?.getServiceInfo('myapp-mysql')
url = dbInfo?.jdbcUrl
username = dbInfo?.userName
password = dbInfo?.password
```

The example `DataSource.groovy` file below contains the following:

- The `imports` that allow `spring-cloud` API commands
- The code that enables the `getCloud` method to function locally or in other environments outside of a cloud
- Code to access the cloud object for SQL, MongoDB, and Redis services

```
import org.springframework.cloud.CloudFactory
import org.springframework.cloud.CloudException

def cloud = null
try {
    cloud = new CloudFactory().cloud
} catch (CloudException) {}

dataSource {
    pooled = true
    dbCreate = 'update'
    driverClassName = 'com.mysql.jdbc.Driver'
}

environments {
    production {
        dataSource {
            def dbInfo = cloud?.getServiceInfo('myapp-mysql')
            url = dbInfo?.jdbcUrl
            username = dbInfo?.userName
            password = dbInfo?.password
        }
        grails {
            mongo {
                def mongoInfo = cloud?.getServiceInfo('myapp-mongodb')
                host = mongoInfo?.host
                port = mongoInfo?.port
                databaseName = mongoInfo?.database
                username = mongoInfo?.userName
                password = mongoInfo?.password
            }
            redis {
                def redisInfo = cloud?.getServiceInfo('myapp-redis')
                host = redisInfo?.host
                port = redisInfo?.port
                password = redisInfo?.password
            }
        }
    }
    development {
        dataSource {
            url = 'jdbc:mysql://localhost:5432/myapp'
            username = 'sa'
            password = ''
        }
        grails {
            mongo {
                host = 'localhost'
                port = 27107
                databaseName = 'foo'
                username = 'user'
                password = 'password'
            }
            redis {
                host = 'localhost'
                port = 6379
                password = 'password'
            }
        }
    }
}
```

Configure Service Connections for Play Framework

Cloud Foundry provides support for connecting a Play Framework application to services such as MySQL and Postgres. In many cases, a Play Framework application running on Cloud Foundry can automatically detect and configure connections to services.

Auto-Configuration

By default, Cloud Foundry detects service connections in a Play Framework application and configures them to use the credentials provided in the Cloud Foundry environment. Note that auto-configuration happens only if there is a single service of either of the supported types—MySQL or Postgres.

Configure Service Connections for Spring

Cloud Foundry provides extensive support for connecting a Spring application to services such as MySQL, PostgreSQL, MongoDB, Redis, and RabbitMQ. In many cases, Cloud Foundry can automatically configure a Spring application without any code changes. For more advanced cases, you can control service connection parameters yourself.

Auto-Reconfiguration

If your Spring application requires services such as a relational database or messaging system, you might be able to deploy your application to Cloud Foundry without changing any code. In this case, Cloud Foundry automatically reconfigures the relevant bean definitions to bind them to cloud services.

Cloud Foundry auto-reconfigures applications only if the following items are true for your application:

- Only one service instance of a given service type is bound to the application. In this context, MySQL and PostgreSQL are considered the same service type, relational database, so if both a MySQL and a PostgreSQL service are bound to the application, auto-reconfiguration will not occur.
- Only one bean of a matching type is in the Spring application context. For example, you can have only one bean of type `javax.sql.DataSource`.

With auto-reconfiguration, Cloud Foundry creates the database or connection factory bean itself, using its own values for properties such as host, port, username and so on. For example, if you have a single `javax.sql.DataSource` bean in your application context that Cloud Foundry auto-reconfigures and binds to its own database service, Cloud Foundry does not use the username, password and driver URL you originally specified. Instead, it uses its own internal values. This is transparent to the application, which really only cares about having a relational database to which it can write data but does not really care what the specific properties are that created the database. Also note that if you have customized the configuration of a service, such as the pool size or connection properties, Cloud Foundry auto-reconfiguration ignores the customizations.

For more information about auto-reconfiguration of specific services types, see the [Service-Specific Details](#) section.

Manual Configuration

Use manual configuration if you have multiple services of a given type or you want to have more control over the configuration than auto-reconfiguration provides.

To use manual configuration, include the `spring-cloud` library in your list of application dependencies. Update your application Maven `pom.xml` or Gradle `build.gradle` file to include dependencies on the `org.springframework.cloud:spring-cloud-spring-service-connector` and `org.springframework.cloud:spring-cloud-cloudfoundry-connector` artifacts.

For example, if you use Maven to build your application, the following `pom.xml` snippet shows how to add this dependency.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-spring-service-connector</artifactId>
    <version>1.1.0.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-cloudfoundry-connector</artifactId>
    <version>1.1.0.RELEASE</version>
  </dependency>
</dependencies>
```

You also need to update your application build file to include the Spring Framework Milestone repository. The following `pom.xml` snippet shows how to do this for Maven:

```
<repositories>
  <repository>
    <id>repository.springsource.milestone</id>
    <name>SpringSource Milestone Repository</name>
    <url>http://repo.springsource.org/milestone</url>
  </repository>
  ...
</repositories>
```

Java Configuration

Typical use of Java config involves extending the `AbstractCloudConfig` class and adding methods with the `@Bean` annotation to create beans for services. Apps migrating from [auto-reconfiguration](#) might first try [Scanning for Services](#) until they need more explicit control. Java config also offers a way to expose application and service properties. Use this for debugging or to create service connectors using a lower-level access.

Creating Service Beans

In the following example, the configuration creates a `DataSource` bean connecting to the only relational database service bound to the app. It also creates a `MongoDbFactory` bean, again, connecting to the only MongoDB service bound to the app. Check Javadoc for `AbstractCloudConfig` for ways to connect to other services.

```
class CloudConfig extends AbstractCloudConfig {
  @Bean
  public DataSource inventoryDataSource() {
    return connectionFactory().dataSource();
  }
  ... more beans to obtain service connectors
}
```

The bean names match the method names unless you specify an explicit value to the annotation such as `@Bean("inventory-service")`, following Spring's Java configuration standards.

If you have more than one service of a type bound to the app or want to have an explicit control over the services to which a bean is bound, you can pass the service names to methods such as `dataSource()` and `mongoDbFactory()` as follows:

```
class CloudConfig extends AbstractCloudConfig {

  @Bean
  public DataSource inventoryDataSource() {
    return connectionFactory().dataSource("inventory-db-service");
  }

  @Bean
  public MongoDbFactory documentMongoDbFactory() {
    return connectionFactory().mongoDbFactory("document-service");
  }

  ... more beans to obtain service connectors
}
```

Method such as `dataSource()` come in an additional overloaded variant that offer specifying configuration options such as the pooling parameters. See Javadoc for more details.

Connecting to Generic Services

Java config supports access to generic services through the `service()` method. Generic services do not have a directly mapped method. This is typical for a newly introduced service or when connecting to a private service in private PaaS. The generic `service()` method follows the same pattern as the `dataSource()`, except it allows supplying the connector type as an additional parameters.

Scanning for Services

You can scan for each bound service using the `@ServiceScan` annotation as shown below. This is conceptually similar to the `@ComponentScan` annotation in Spring:

```
@Configuration  
@ServiceScan  
class CloudConfig {  
}
```

Here, one bean of the appropriate type (`DataSource` for a relational database service, for example) is created. Each created bean will have the `id` matching the corresponding service name. You can then inject such beans using auto-wiring:

```
@Autowired DataSource inventoryDb;
```

If the app is bound to more than one services of a type, you can use the `@Qualifier` annotation supplying it the name of the service as in the following code:

```
@Autowired @Qualifier("inventory-db") DataSource inventoryDb;  
@Autowired @Qualifier("shipping-db") DataSource shippingDb;
```

Accessing Service Properties

You can expose raw properties for all services and the app through a bean as follows:

```
class CloudPropertiesConfig extends AbstractCloudConfig {  
  
    @Bean  
    public Properties cloudProperties() {  
        return properties();  
    }  
}
```

Cloud Profile

Spring Framework versions 3.1 and above support bean definition profiles as a way to conditionalize the application configuration so that only specific bean definitions are activated when a certain condition is true. Setting up such profiles makes your application portable to many different environments so that you do not have to manually change the configuration when you deploy it to, for example, your local environment and then to Cloud Foundry.

See the Spring Framework documentation for additional information about using Spring bean definition profiles.

When you deploy a Spring application to Cloud Foundry, Cloud Foundry automatically enables the `cloud` profile.

Note: Cloud Foundry auto-reconfiguration requires the Spring application to be version 3.1 or later and include the Spring context JAR. If you are using an earlier version, update your framework or use a custom buildpack.

Profiles in Java Configuration

The `@Profile` annotation can be placed on `@Configuration` classes in a Spring application to set conditions under which configuration classes are invoked. By using the `default` and `cloud` profiles to determine whether the application is running on Cloud Foundry or not, your Java configuration can support both local and cloud deployments using Java configuration classes like these:

```

public class Configuration {
    @Configuration
    @Profile("cloud")
    static class CloudConfiguration {

        @Bean
        public DataSource dataSource() {
            CloudFactory cloudFactory = new CloudFactory();
            Cloud cloud = cloudFactory.getCloud();
            String serviceID = cloud.getServiceID();
            return cloud.getServiceConnector(serviceID, DataSource.class, null);
        }
    }

    @Configuration
    @Profile("default")
    static class LocalConfiguration {

        @Bean
        public DataSource dataSource() {
            BasicDataSource dataSource = new BasicDataSource();
            dataSource.setUrl("jdbc:postgresql://localhost/db");
            dataSource.setDriverClassName("org.postgresql.Driver");
            dataSource.setUsername("postgres");
            dataSource.setPassword("postgres");
            return dataSource;
        }
    }
}

```

Property Placeholders

Cloud Foundry exposes a number of application and service properties directly into its deployed applications. The properties exposed by Cloud Foundry include basic information about the application, such as its name and the cloud provider, and detailed connection information for all services currently bound to the application.

Service properties generally take one of the following forms:

```

cloud.services.{service-name}.connection.{property}
cloud.services.{service-name}.{property}

```

In this form, `{service-name}` refers to the name you gave the service when you bound it to your application at deploy time, and `{property}` is a field in the credentials section of the `VCAP_SERVICES` environment variable.

For example, assume that you created a Postgres service called `my-postgres` and then bound it to your application. Assume also that this service exposes credentials in `VCAP_SERVICES` as discrete fields. Cloud Foundry exposes the following properties about this service:

```

cloud.services.my-postgres.connection.hostname
cloud.services.my-postgres.connection.name
cloud.services.my-postgres.connection.password
cloud.services.my-postgres.connection.port
cloud.services.my-postgres.connection.username
cloud.services.my-postgres.plan
cloud.services.my-postgres.type

```

If the service exposed the credentials as a single `uri` field, then the following properties would be set up:

```

cloud.services.my-postgres.connection.uri
cloud.services.my-postgres.plan
cloud.services.my-postgres.type

```

For convenience, if you have bound just one service of a given type to your application, Cloud Foundry creates an alias based on the service type instead of the service name. For example, if only one MySQL service is bound to an application, the properties takes the form `cloud.services.mysql.connection.{property}`. Cloud Foundry uses the following aliases in this case:

- `mysql`
- `postgresql`

- `mongodb`
- `redis`
- `rabbitmq`

A Spring application can take advantage of these Cloud Foundry properties using the property placeholder mechanism. For example, assume that you have bound a MySQL service called `spring-mysql` to your application. Your application requires a `c3p0` connection pool instead of the connection pool provided by Cloud Foundry, but you want to use the same connection properties defined by Cloud Foundry for the MySQL service - in particular the username, password and JDBC URL.

The following table lists all the application properties that Cloud Foundry exposes to deployed applications.

Property	Description
<code>cloud.application.name</code>	The name provided when the application was pushed to Cloud Foundry.
<code>cloud.provider.url</code>	The URL of the cloud hosting the application, such as <code>cloudfoundry.com</code> .

The service properties that are exposed for each type of service are listed in the [Service-Specific Details](#) section.

Service-Specific Details

The following sections describe Spring auto-reconfiguration and manual configuration for the services supported by Cloud Foundry.

MySQL and Postgres

Auto-Reconfiguration

Auto-reconfiguration occurs if Cloud Foundry detects a `javax.sql.DataSource` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```
<bean class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close" id="dataSource">
    <property name="driverClassName" value="org.h2.Driver" />
    <property name="url" value="jdbc:h2:mem:" />
    <property name="username" value="sa" />
    <property name="password" value="" />
</bean>
```

The relational database that Cloud Foundry actually uses depends on the service instance you explicitly bind to your application when you deploy it: MySQL or Postgres. Cloud Foundry creates either a commons DBCP or Tomcat datasource depending on which datasource implementation it finds on the classpath.

Cloud Foundry internally generates values for the following properties: `driverClassName`, `url`, `username`, `password`, `validationQuery`.

Manual Configuration in Java

To configure a database service in Java configuration, create a `@Configuration` class with a `@Bean` method to return a `javax.sql.DataSource` bean. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class DataSourceConfig {
    @Bean
    public DataSource dataSource() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        String serviceID = cloud.getServiceID();
        return cloud.getServiceConnector(serviceID, DataSource.class, null);
    }
}
```

MongoDB

Auto-Reconfiguration

You must use Spring Data MongoDB 1.0 M4 or later for auto-reconfiguration to work.

Auto-reconfiguration occurs if Cloud Foundry detects an `org.springframework.data.document.mongodb.MongoDbFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-reconfigure:

```
<mongo:db-factory
  id="mongoDbFactory"
  dbname="pwdtest"
  host="127.0.0.1"
  port="1234"
  username="test_user"
  password="test_pass" />
```

Cloud Foundry creates a `SimpleMongoDbFactory` with its own values for the following properties: `host`, `port`, `username`, `password`, `dbname`.

Manual Configuration in Java

To configure a MongoDB service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.mongodb.MongoDbFactory` bean from Spring Data MongoDB. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class MongoConfig {

    @Bean
    public MongoDbFactory mongoDbFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        MongoServiceInfo serviceInfo = (MongoServiceInfo) cloud.getServiceInfo("my-mongodb");
        String serviceID = serviceInfo.getId();
        return cloud.getServiceConnector(serviceID, DataSource.class, null);
    }

    @Bean
    public MongoTemplate mongoTemplate() {
        return new MongoTemplate(mongoDbFactory());
    }
}
```

Redis

Auto-Configuration

You must be using Spring Data Redis 1.0 M4 or later for auto-configuration to work.

Auto-configuration occurs if Cloud Foundry detects a `org.springframework.data.redis.connection.RedisConnectionFactory` bean in the Spring application context. The following snippet of a Spring XML application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```
<bean id="redis"
  class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"
  p:hostName="localhost" p:port="6379" />
```

Cloud Foundry creates a `JedisConnectionFactory` with its own values for the following properties: `host`, `port`, `password`. This means that you must package the Jedis JAR in your application. Cloud Foundry does not currently support the JRedis and RJC implementations.

Manual Configuration in Java

To configure a Redis service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.data.redis.connection.RedisConnectionFactory` bean from Spring Data Redis. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class RedisConfig {

    @Bean
    public RedisConnectionFactory redisConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        RedisServiceInfo serviceInfo = (RedisServiceInfo) cloud.getServiceInfo("my-redis");
        String serviceID = serviceInfo.getID();
        return cloud.getServiceConnector(serviceID, DataSource.class, null);
    }

    @Bean
    public RedisTemplate redisTemplate() {
        return new StringRedisTemplate(redisConnectionFactory());
    }

}
```

RabbitMQ

Auto-Configuration

You must be using Spring AMQP 1.0 or later for auto-configuration to work. Spring AMQP provides publishing, multi-threaded consumer generation, and message conversion. It also facilitates management of AMQP resources while promoting dependency injection and declarative configuration.

Auto-configuration occurs if Cloud Foundry detects an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean in the Spring application context. The following snippet of a Spring application context file shows an example of defining this type of bean which Cloud Foundry will detect and potentially auto-configure:

```
<rabbit:connection-factory
    id="rabbitConnectionFactory"
    host="localhost"
    password="testpwd"
    port="1238"
    username="testuser"
    virtual-host="virthost" />
```

Cloud Foundry creates an `org.springframework.amqp.rabbit.connection.CachingConnectionFactory` with its own values for the following properties: `host`, `virtual-host`, `port`, `username`, `password`.

Manual Configuration in Java

To configure a RabbitMQ service in Java configuration, create a `@Configuration` class with a `@Bean` method to return an `org.springframework.amqp.rabbit.connection.ConnectionFactory` bean from the Spring AMQP library. The bean can be created by helper classes in the `spring-cloud` library, as shown here:

```
@Configuration
public class RabbitConfig {
    @Bean
    public ConnectionFactory rabbitConnectionFactory() {
        CloudFactory cloudFactory = new CloudFactory();
        Cloud cloud = cloudFactory.getCloud();
        AmqpServiceInfo serviceInfo = (AmqpServiceInfo) cloud.getServiceInfo("my-rabbit");
        String serviceID = serviceInfo.getID();
        return cloud.getServiceConnector(serviceID, ConnectionFactory.class, null);
    }

    @Bean
    public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        return new RabbitTemplate(connectionFactory);
    }
}
```

Cloud Foundry Eclipse Plugin

The Cloud Foundry Eclipse Plugin is an extension that enables Cloud Foundry users to deploy and manage Java and Spring applications on a Cloud Foundry instance from Eclipse or Spring Tool Suite (STS).

The plugin supports Eclipse v3.8 and v4.3 (a Java EE version is recommended), and STS 3.0.0 and later.

This page has instructions for installing and using v1.7.2 of the plugin.

You can use the plugin to perform the following actions:

- Deploy applications from an Eclipse or STS workspace to a running Cloud Foundry instance. The Cloud Foundry Eclipse plugin supports the following application types:
 - Spring Boot
 - Spring
 - Java Web
 - Java standalone
 - Grails
- Create, bind, and unbind services.
- View and manage deployed applications and services.
- Start and stop applications.

v1.7.2 of this plugin provides the following updates and changes:

- Cloud Foundry Eclipse is now enabled for NLS and Internationalization.
- A “New Service Binding” wizard that allows service instances to be bound to applications. This wizard serves as an alternative to binding through the existing drag-and-drop feature.
- Improvements in loggregator streaming to the console.
- Improvements in deploying Spring Boot and Getting Started projects with templates.

Install Cloud Foundry Eclipse Plugin

If you have a previous version of the Cloud Foundry Eclipse Plugin installed, uninstall it before installing the new version. To uninstall the plugin:

1. Choose **About Eclipse** (or **About Spring Tool Suite**) from the Eclipse (or **Spring Tool Suite**) menu and click **Installation Details**.
2. In **Installation Details**, select the previous version of the plugin and click **Uninstall**.

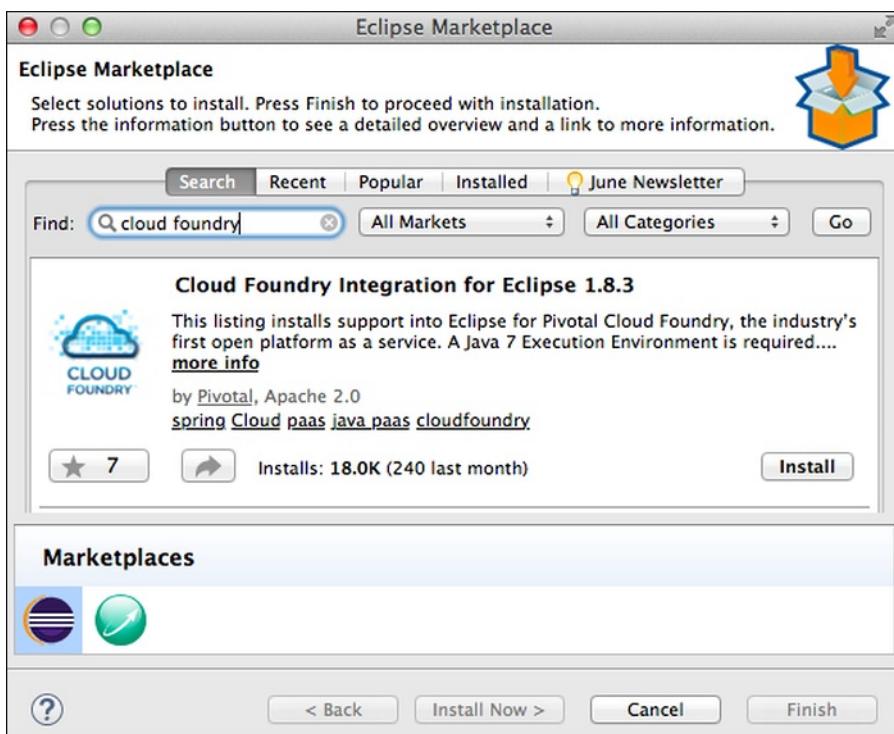
Follow the installation instructions appropriate for your environment:

- [Install to Eclipse from Marketplace](#)
- [Install to STS from IDE Extensions Tab](#)
- [Install from a Local Repository](#)

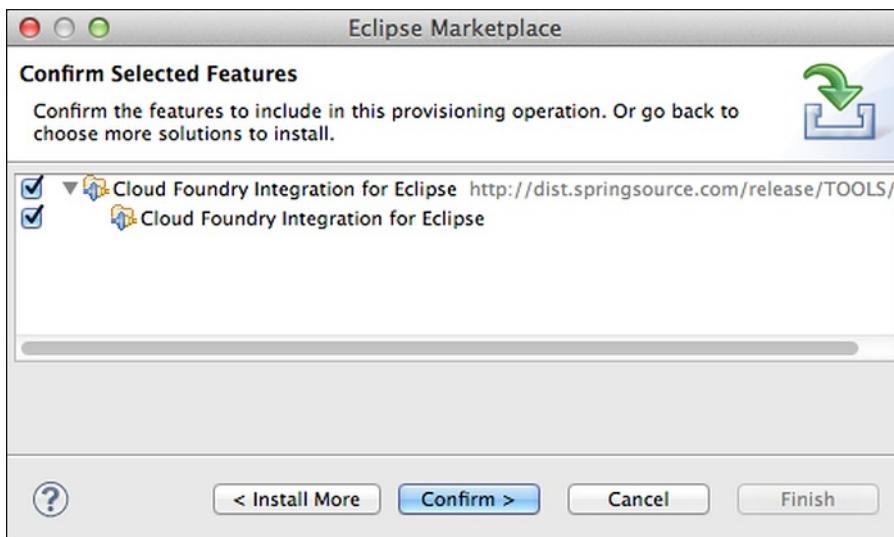
Install to Eclipse from Marketplace

Follow the instructions below to install the Cloud Foundry Eclipse Plugin to Eclipse from the Eclipse Marketplace.

1. Start Eclipse.
2. From the Eclipse **Help** menu, select **Eclipse Marketplace**.
3. In the Eclipse Marketplace window, enter “Cloud Foundry” in the **Find** field. Click **Go**.
4. In the search results, next to the listing for Cloud Foundry Integration, click **Install**.



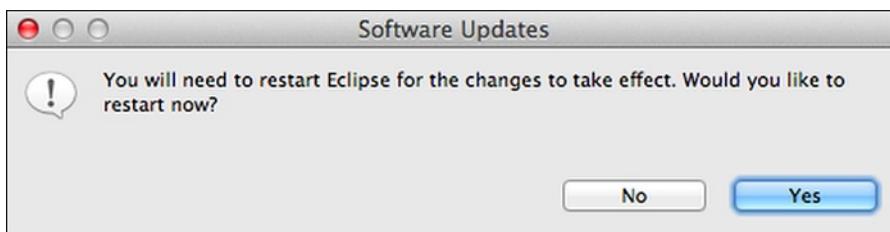
5. In the **Confirm Selected Features** window, click **Confirm**.



6. The **Review Licenses** window appears. Select "I accept the terms of the license agreement" and click **Finish**.



7. The **Software Updates** window appears. Click **Yes** to restart Eclipse.



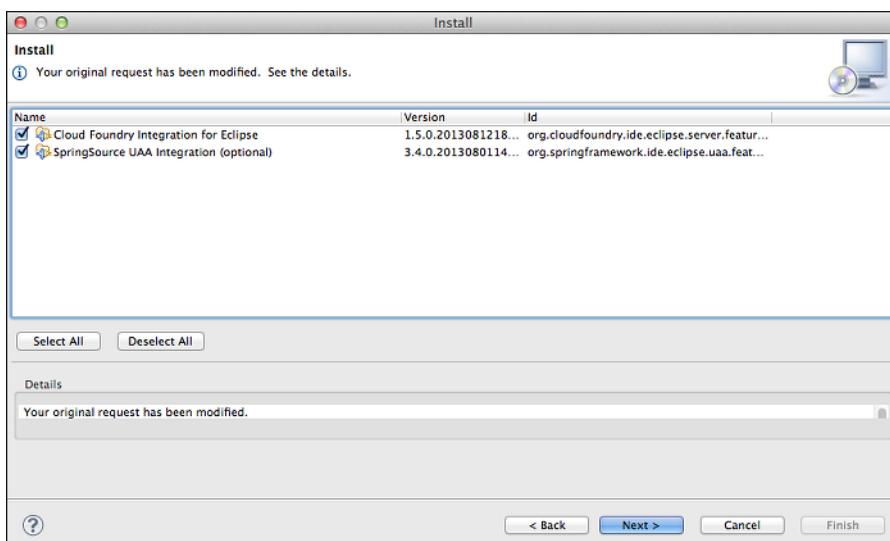
Install to STS from IDE Extensions Tab

Follow these instructions to install the Cloud Foundry Eclipse Plugin to Spring Tool Suite (STS) from the **IDE Extensions** tab.

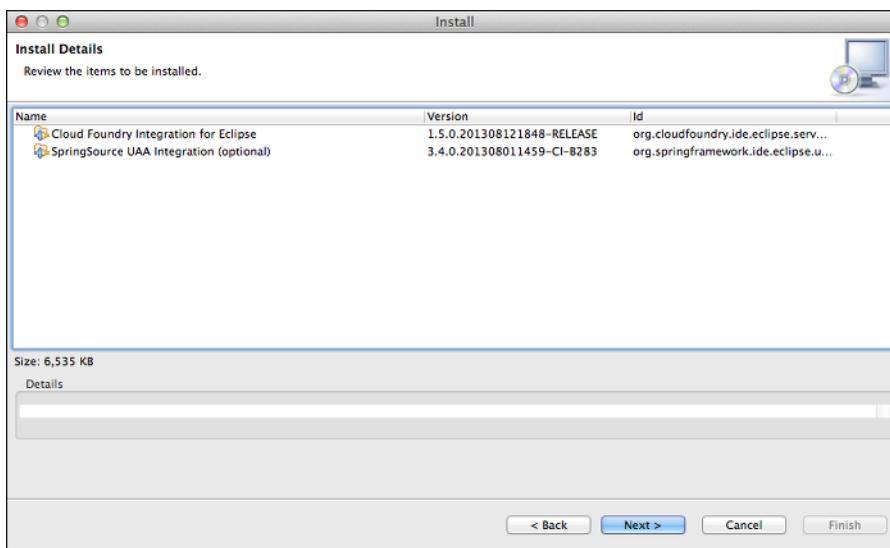
1. Start STS.
2. On the STS Dashboard, click **IDE Extensions**.

3. Enter "Cloud Foundry" in the **Find** field.
4. Select **Cloud Foundry Integration for Eclipse** and click **Install**.

5. In the **Install** window, click **Next**.



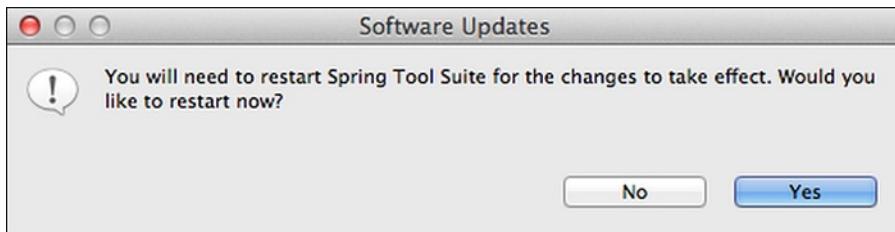
- In the **Install Details** window, click **Next**.



- The **Review Licenses** window appears. Select "I accept the terms of the license agreement" and click **Finish**.



- The **Software Updates** window appears. Click **Yes** to restart Spring Tool Suite.



Install a Release Version Offline

If you need to install a release version of Cloud Foundry Eclipse Plugin in offline mode, you can download a release update site zip file and transfer it to the offline environment.

To install a Release Version offline, follow the steps below on a computer running Eclipse or Spring Tool Suite (STS).

1. Browse to <https://github.com/cloudfoundry/eclipse-integration-cloudfoundry/blob/master/updatesites.md> and download a release update site zip file.
2. In Eclipse or STS, select **Install New Software** from the **Help** menu.
3. In the **Available Software** window, to the right of the **Work with** field, click **Add**.
4. In the **Add Repository** window, enter `Cloud Foundry Integration` or a name of your choice for the repository. Click **Archive**.
5. In the **Open** window, browse to the location of the update site zip file and click **Open**.
6. In the **Add Repository** window, click **OK**.
7. In the **Available Software** window, select **Core/Cloud Foundry Integration** and, optionally, **Resources/Cloud Foundry Integration**. Click **Next**.
8. In the **Review Licenses** window, select “I accept the terms of the license agreement” and click **Finish**.

Install from a Local Build

If you need to install the Cloud Foundry Eclipse Plugin from a local build, rather than from a release version, you can download and build the source, create a repository and copy it to the target machine, then install from the copied repository.

1. Obtain the plugin source from GitHub in one of the following ways:
 - Download archived source code for released versions of the plugin from <https://github.com/SpringSource/eclipse-integration-cloudfoundry/releases>
 - Clone the project repository:

```
$ git clone https://github.com/SpringSource/eclipse-integration-cloudfoundry
```
2. Unzip the downloaded archive. In a terminal, run the following command:

```
$ mvn -Pe37 package
```
3. Copy the `org.cloudfoundry.ide.eclipse.server.site/target/site` directory to the machine where you want to install the plugin.
4. On the machine where you want to install the plugin, launch Eclipse or Spring Tool Suite (STS).
5. Select **Install New Software** from the **Help** menu.
6. In the **Available Software** window, to the right of the **Work with** field, click **Add**.
7. In the **Add Repository** window, enter `Cloud Foundry Integration` or a name of your choice for the repository. Click **Local**.
8. In the **Open** window, browse to the `org.cloudfoundry.ide.eclipse.server.site/target/site` directory. Click **Open**.

9. In the **Add Repository** window, click **OK**.
10. In the **Available Software** window, select **Core/Cloud Foundry Integration** and, optionally, **Resources/Cloud Foundry Integration**. Click **Next**.
11. In the **Review Licenses** window, select “I accept the terms of the license agreement” and click **Finish**.

About the Plugin User Interface

The sections below describe the Cloud Foundry Eclipse plugin user interface. If you do not see the tabs described below, select the Pivotal Cloud Foundry server in the **Servers** view. To expose the Servers view, ensure that you are using the Java perspective, then select **Window > Show View > Other > Server > Servers**.

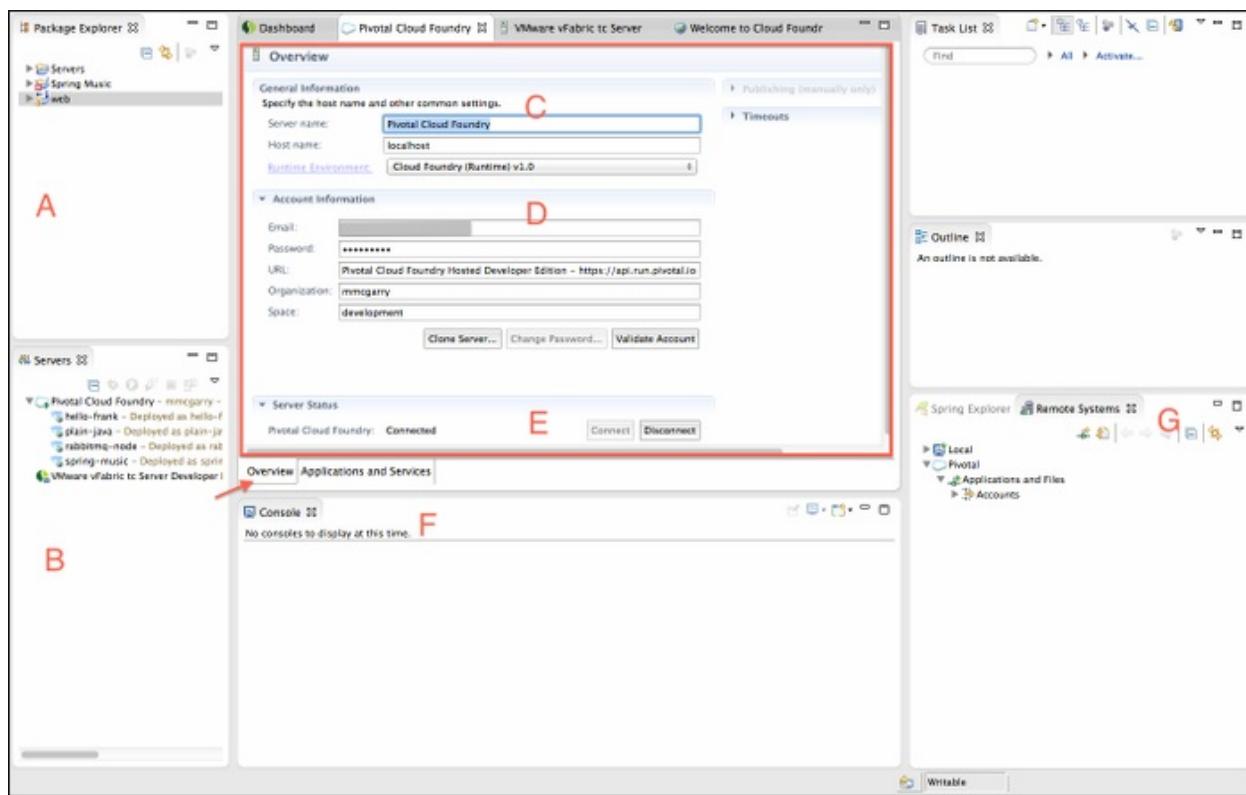
The Cloud Foundry editor, outlined in red in the screenshot below, is the primary plugin user interface. Some workflows involve interacting with standard elements of the Eclipse user interface, such as the **Project Explorer** and the **Console** and **Servers** views.

Note that the Cloud Foundry editor allows you to work with a single Cloud Foundry space. Each space is represented by a distinct server instance in the **Servers** view (B). Multiple editors, each targeting a different space, can be open simultaneously. However, only one editor targeting a particular Cloud Foundry server instance can be open at a time.

Overview Tab

The follow panes and views are present when the **Overview** tab is selected:

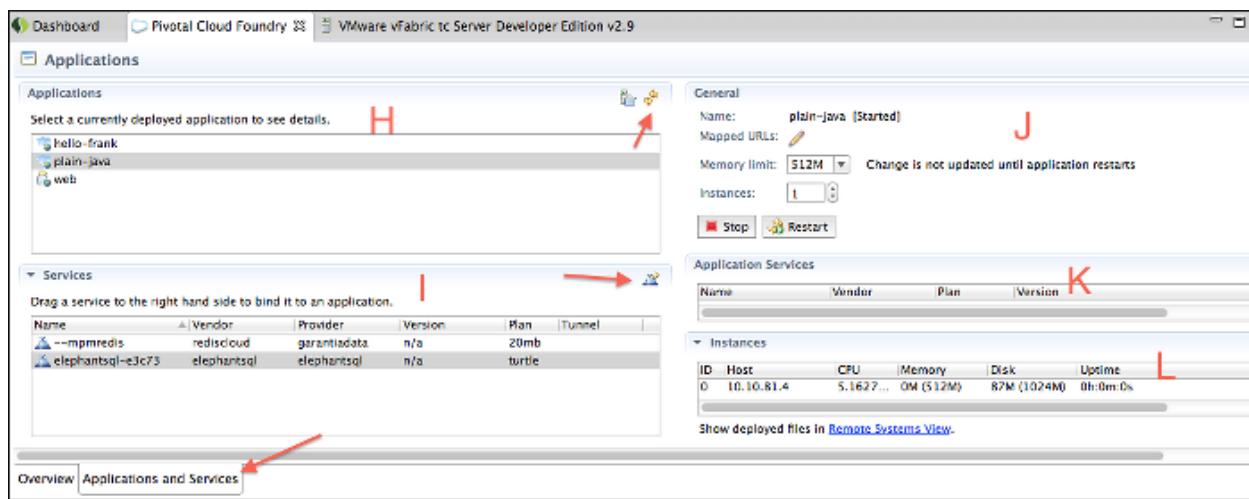
- A — The **Package Explorer** view lists the projects in the current workspace.
- B – The **Servers** view lists server instances configured in the current workspace. A server of type **Pivotal Cloud Foundry** represents a targeted space in a Cloud Foundry instance.
- C – The **General Information** pane.
- D – The **Account Information** pane lists your Cloud Foundry credentials and the target organization and space. The pane includes these controls:
 - **Clone Server** — Use to create additional Pivotal Cloud Foundry server instances. You must configure a server instance for each Cloud Foundry space that you wish to target. For more information, see [Create Additional Server Instances](#).
 - **Change Password** — Use to change your Cloud Foundry password.
 - **Validate Account** — Use to verify your currently configured Cloud Foundry credentials.
- E – The **Server Status** pane shows whether or not you are connected to the target Cloud Foundry space, and the **Disconnect** and **Connect** controls.
- F – The **Console** view displays status messages when you perform an action such as deploying an application.
- G – The **Remote Systems** view allows you to view the contents of a file that is part of a deployed application. For more information, see [View an Application File](#).



Applications and Services Tab

The following panes are present when the **Applications and Services** tab is selected:

- H — The **Applications** pane lists the applications deployed to the target space.
- I — The **Services** pane lists the services provisioned in the targeted space.
- J — The **General** pane displays the following information for the application currently selected in the **Applications** pane:
 - **Name**
 - **Mapped URLs** — Lists URLs mapped to the application. You can click a URL to open a browser to the application within Eclipse or STS, and click the pencil icon to add or remove mapped URLs. See [Manage Application URLs](#).
 - **Memory Limit** — The amount of memory allocated to the application. You can use the pull-down to change the memory limit.
 - **Instances** — The number of instances of the application that are deployed. You can use the pull-down to change the number of instances.
 - **Start, Stop, Restart, Update and Restart** — The controls that appear depend on the current state of the application. The **Update and Restart** command will attempt an incremental push of only those application resources that have changed. It will not perform a full application push. See [Push Application Changes](#) below.
- K — The **Services** pane lists services that are bound to the application currently selected in the **Applications** pane. The icon in the upper right corner of the pane allows you to create a service, as described in [Create a Service](#).

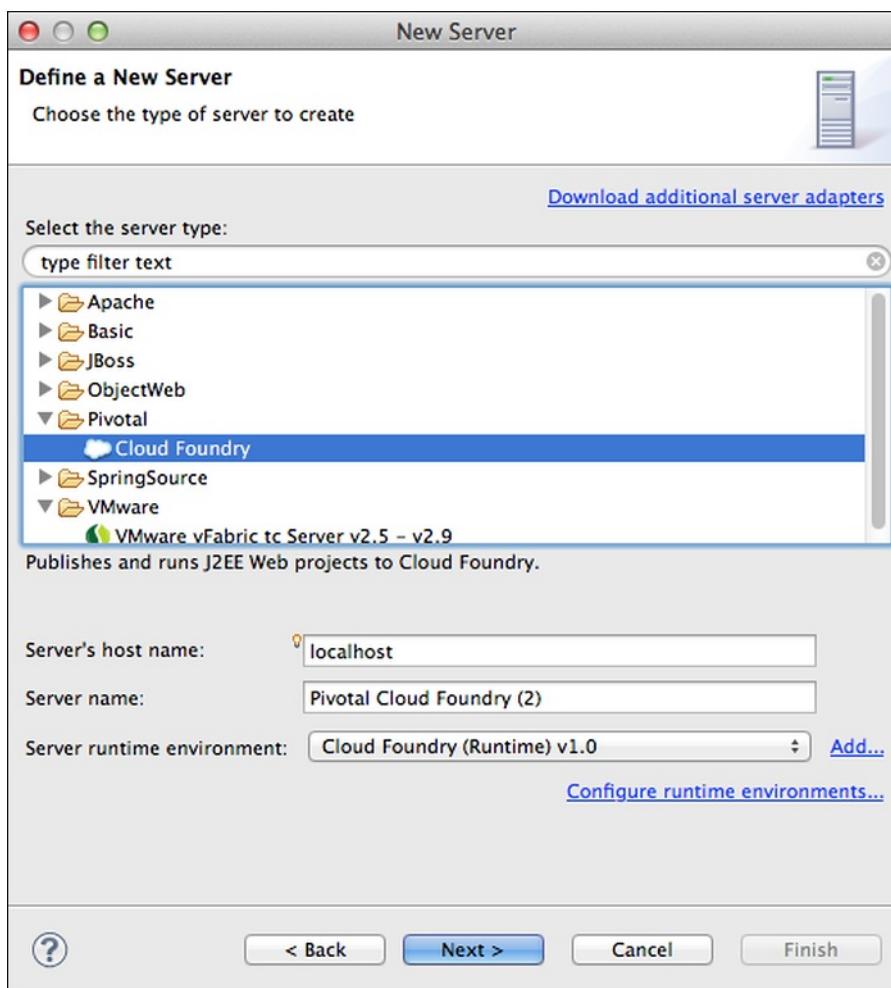


Create a Cloud Foundry Server

This section contains instructions for configuring a server resource that will represent a target Cloud Foundry space. You will create a server for each space in Cloud Foundry to which you will deploy applications. Once you create your first Cloud Foundry service instances using the instructions below, you can create additional instances using the [Clone Server](#) feature.

1. Right-click the **Servers** view and select **New > Server**.
2. In the **Define a New Server** window, expand the **Pivotal** folder, select **Cloud Foundry**, and click **Next**.

Note: Do not modify default values for **Server host name** or **Server Runtime Environment**. These fields are not used

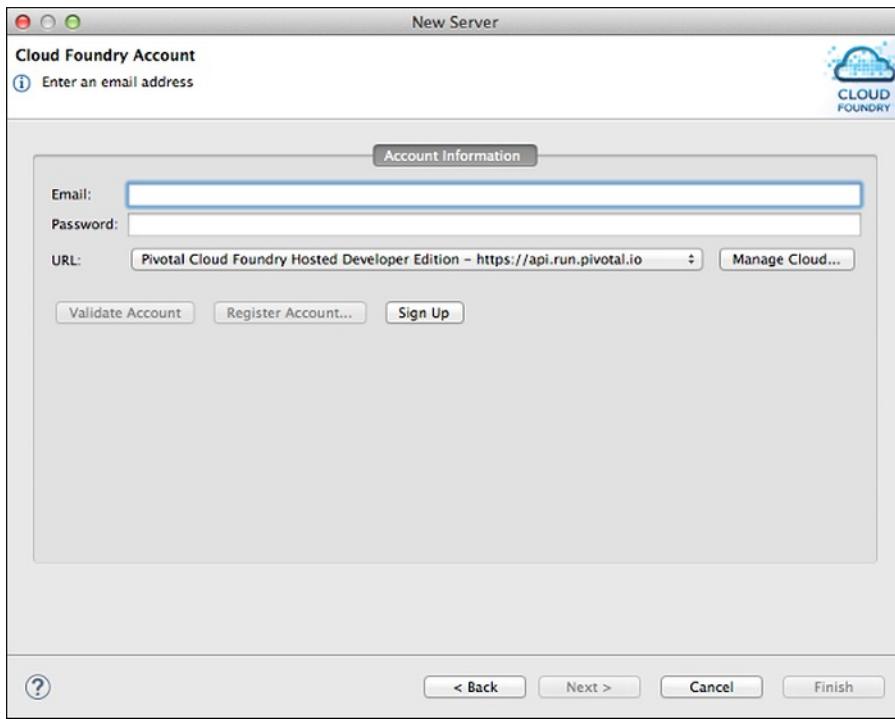


- In the **Cloud Foundry Account** window, if you already have a Pivotal Cloud Foundry Hosted Developer Edition account, enter your email account and password credentials and click **Validate Account**.

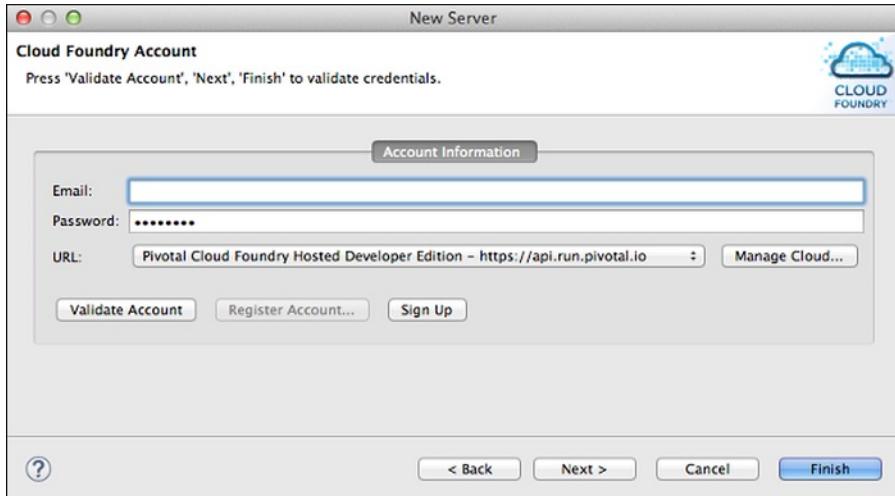
Note: By default, the **URL** field points to the Pivotal Cloud Foundry Hosted Developer Edition URL of <https://api.run.pivotal.io>. If you have a Pivotal Elastic Runtime account, refer to the [Logging into the Apps Manager](#) topic to determine your Pivotal Elastic Runtime URL. Click **Manage Cloud...** to add this URL to your Cloud Foundry account. Validate the account and continue through the wizard.

If you do not have a Cloud Foundry account and want to register a new Pivotal Cloud Foundry Hosted Developer Edition account, click **Sign Up**. After you create the account, you can complete this procedure.

Note: The **Register Account** button is inactive.

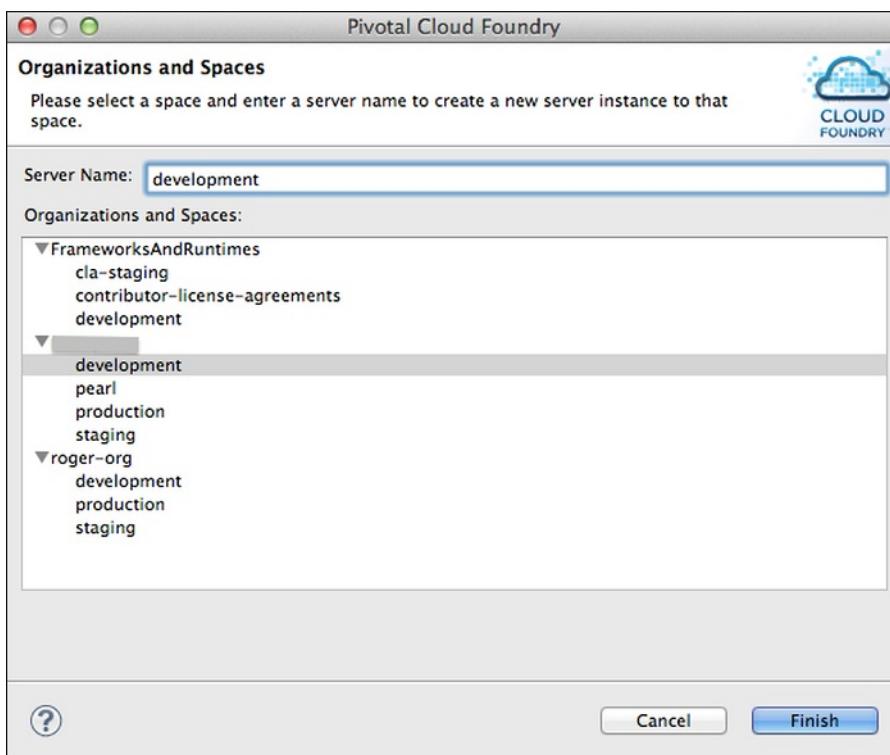


4. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials were valid. Click **Next**.



5. In the **Organizations and Spaces** window, select the space that you want to target, and click **Finish**.

Note: If you do not select a space, the server will be configured to connect to the default space, which is the first encountered in a list of your spaces.



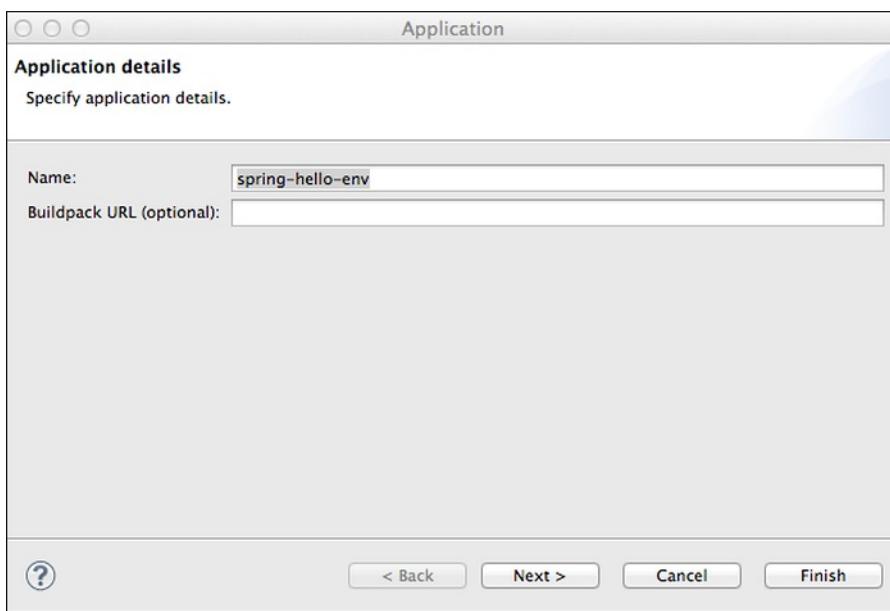
- Once you have successfully configured the Pivotal Cloud Foundry server, it will appear in the **Servers** view of the Eclipse or STS user interface. To familiarize yourself with the plugin user interface, see [About the Plugin User Interface](#). Following this step, proceed to [Deploy an Application](#).

Deploy an Application

To deploy an application to Cloud Foundry using the plugin:

- To initiate deployment either:
 - Drag the application from the **Package Explorer** view onto the Pivotal Cloud Foundry server in the **Servers** view, or
 - Right-click the Pivotal Cloud Foundry server in the **Servers** view, select **Add and Remove** from the server context menu, and move the application from the **Available** to the **Configured** column.
- In the **Application Details** window:
 - By default, the **Name** field is populated with the application project name. You can enter a different name. The name is assigned to the deployed application, but does not rename the project.
 - If you want to use an external buildpack to stage the application, enter the URL of the buildpack.

You can deploy the application without further configuration by clicking **Finish**. Note that because the application default values may take a second or two to load, the **Finish** button might not be enabled immediately. A progress indicator will indicate when the application default values have been loaded, and the "Finish" button will be enabled. Click **Next** to continue.



3. In the **Launch Deployment** window:

Host — By default, contains the name of the application. You can enter a different value if desired. If you push the same application to multiple spaces in the same organization, you must assign a unique **Host** to each.

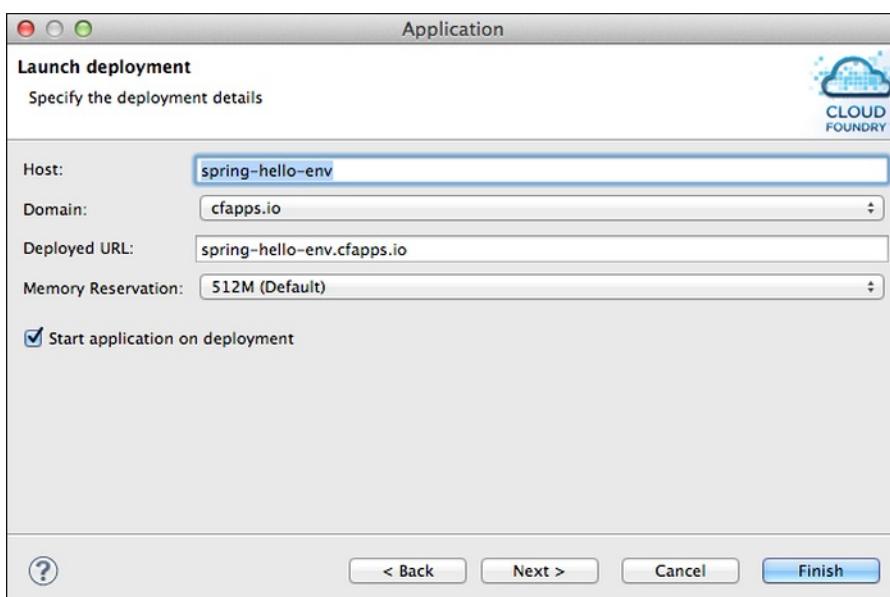
Domain — Contains the default domain. If you have mapped custom domains to the target space, they appear in the pull-down list.

Note: This version of the Cloud Foundry Eclipse plugin does not provide a mechanism for mapping a custom domain to a space. You must use the `cf map domain` command to do so.

Deployed URL — By default, contains the value of the **Host** and **Domain** fields, separated by a period (.) character.

Memory Reservation — Select the amount of memory to allocate to the application from the pull-down list.

Start application on deployment — If you do not want the application to be started on deployment, uncheck the box.



4. The **Services Selection** window lists services provisioned in the target space. Checkmark the services, if any, that you want to bind to the application, and click **Finish**. You can bind services to the application after deployment, as described in [Bind and Unbind Services](#).

Name	Vendor	Provider	Version	Plan	Tunnel
<input checked="" type="checkbox"/> --mpmredis	rediscloud	garantiadata	n/a	20mb	
<input checked="" type="checkbox"/> elephantsql-e3c73	elephantsql	elephantsql	n/a	turtle	

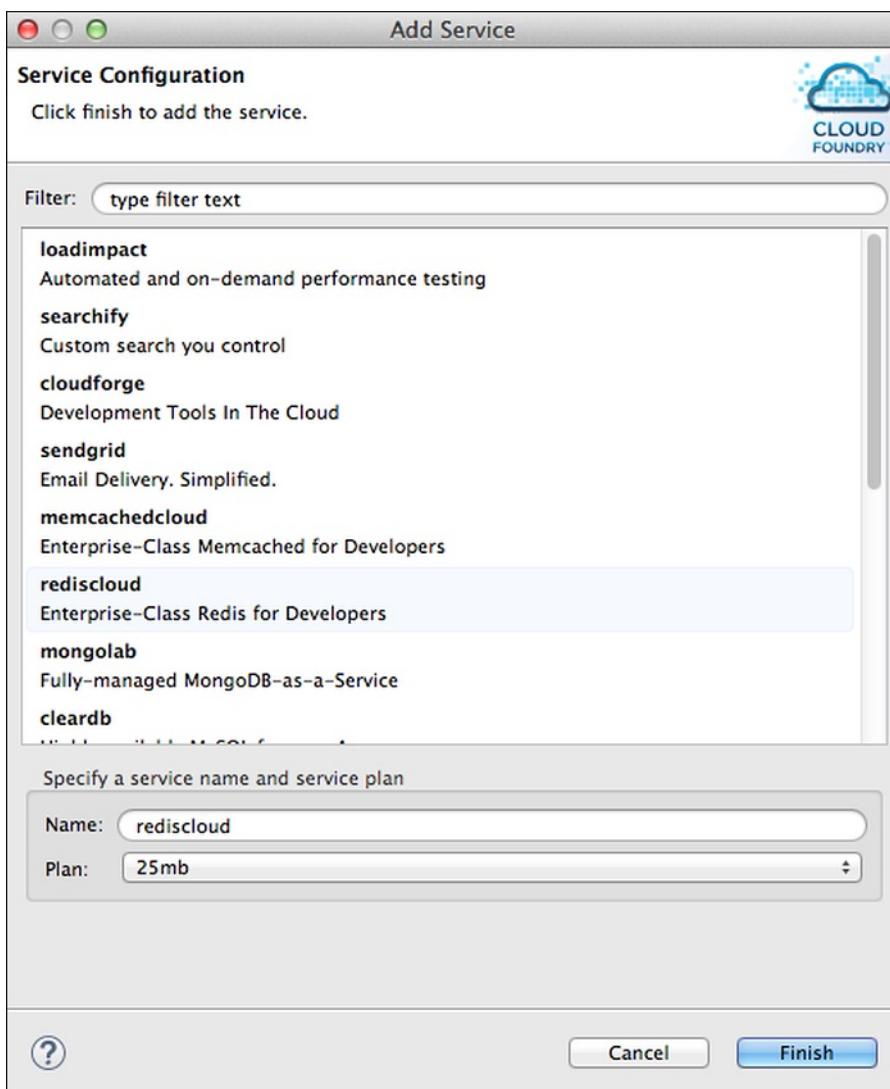
As the deployment proceeds, progress messages appear in the **Console** view. When deployment is complete, the application is listed in the **Applications** pane.

Create a Service

Before you can bind a service to an application, you must create it.

To create a service:

1. Select the **Applications and Services** tab.
2. Click the icon in the upper right corner of the **Services** pane.
3. In the **Service Configuration** window, enter a text pattern to **Filter** for a service. Matches are made against both service name and description.
4. Select a service from the **Service List**. The list automatically updates based on the filter text.
5. Enter a **Name** for the service and select a service **Plan** from the drop-down list.



6. Click **Finish**. The new service appears in the **Services** pane.

Bind and Unbind Services

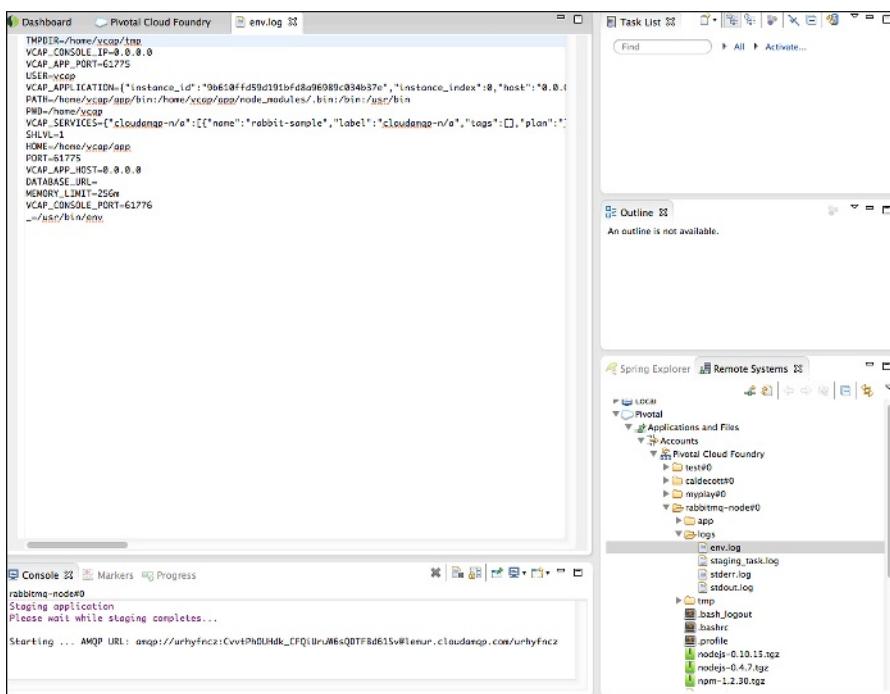
You can bind a service to an application when you deploy it. To bind a service to an application that is already deployed, drag the service from the **Services** pane to the **Application Services** pane. (See the area labelled “G” in the screenshot in the [Applications and Services](#) above.)

To unbind a service, right-click the service in the **Application Services** pane, and select **Unbind from Application**.

View an Application File

You can view the contents of a file in a deployed application by selecting it in the **Remote Systems View**. (See the areas labelled “I” and “J” in the screenshot in the [Applications and Services Tab](#) above.)

1. If the **Remote Systems View** is not visible:
 - Select the **Applications and Services** tab.
 - Select the application of interest from the **Applications** pane.
 - In the **Instances** pane, click the **Remote Systems View** link.
2. In the **Remote Systems View**, browse to the application and application file of interest, and double-click the file. A new tab appears in the editor area with the contents of the selected file.



Undeploy an Application

To undeploy an application, right click the application in either the **Servers** or the **Applications** pane and click **Remove**.

Scale an Application

You can change the memory allocation for an application and the number of instances deployed in the **General** pane when the **Applications and Services** tab is selected. Use the **Memory Limit** and **Instances** selector lists.

Although the scaling can be performed while the application is running, if the scaling has not taken effect, restart the application. If necessary, the application statistics can be manually refreshed by clicking the **Refresh** button in the top, right corner of the “Applications” pane, labelled “H” in the screenshot in [Applications and Services Tab](#).

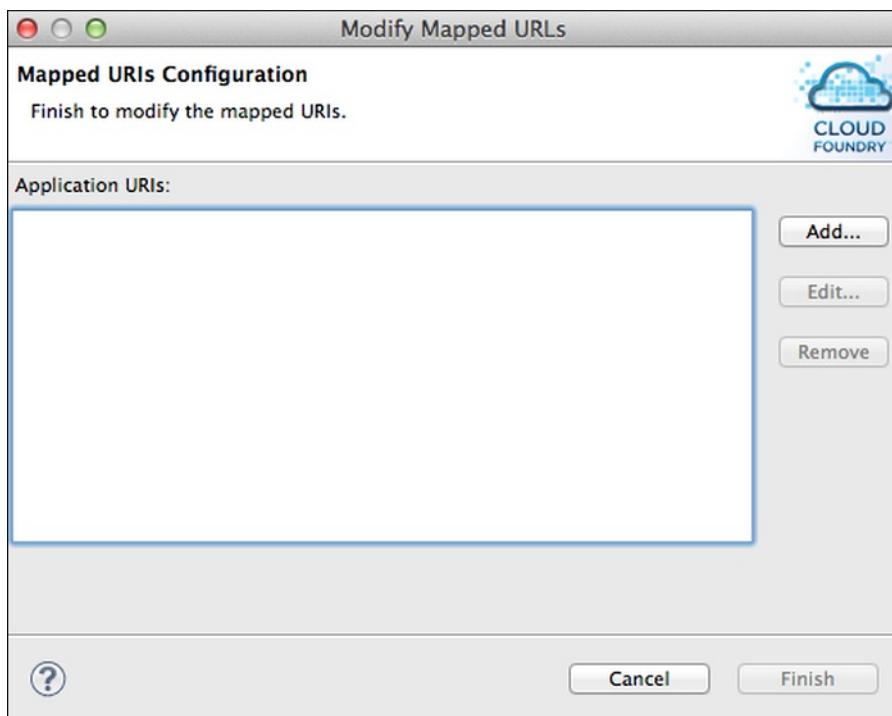
Push Application Changes

The Cloud Foundry editor supports these application operations:

- **Start and Stop** — When you **Start** an application, the plugin pushes all application files to the Cloud Foundry instance before starting the application, regardless of whether there are changes to the files or not.
- **Restart** — When you **Restart** a deployed application, the plugin does not push any resources to the Cloud Foundry instance.
- **Update and Restart** — When you run this command, the plugin pushes only the changes that were made to the application since last update, not the entire application. This is useful for performing incremental updates to large applications.

Manage Application URLs

You add, edit, and remove URLs mapped to the currently selected application in the **General** pane when the **Applications and Services** tab is selected. Click the pencil icon to display the **Mapped URIs Configuration** window.



Information in the Console View

When you start, restart, or update and restart an application, application output will generally be streamed to the **Console** view (labelled “F” in the screenshot in [Overview Tab](#)). The information shown in the **Console** view for a running application instance includes staging information, and the application’s `std.out` and `std.error` logs.

If multiple instances of the application are running, only the output of the first instance appears in the **Console** view. To view the output of another running instance, or to refresh the output that is currently displayed:

1. In the **Applications and Services** tab, select the deployed application in the *Applications* pane.
2. Click **Refresh** on the top right corner of the **Applications** pane.
3. In the **Instances** pane, wait for the application instances to be updated.
4. Once non-zero health is shown for an application instance, right-click on that instance to open the context menu and select **Show Console**.

Clone a Cloud Foundry Server Instance

Each space in Cloud Foundry to which you want to deploy applications must be represented by a Cloud Foundry server instance in the **Servers** view. After you have created a Cloud Foundry server instance, as described in [Create a Cloud Foundry Server](#), you can clone it to create another.

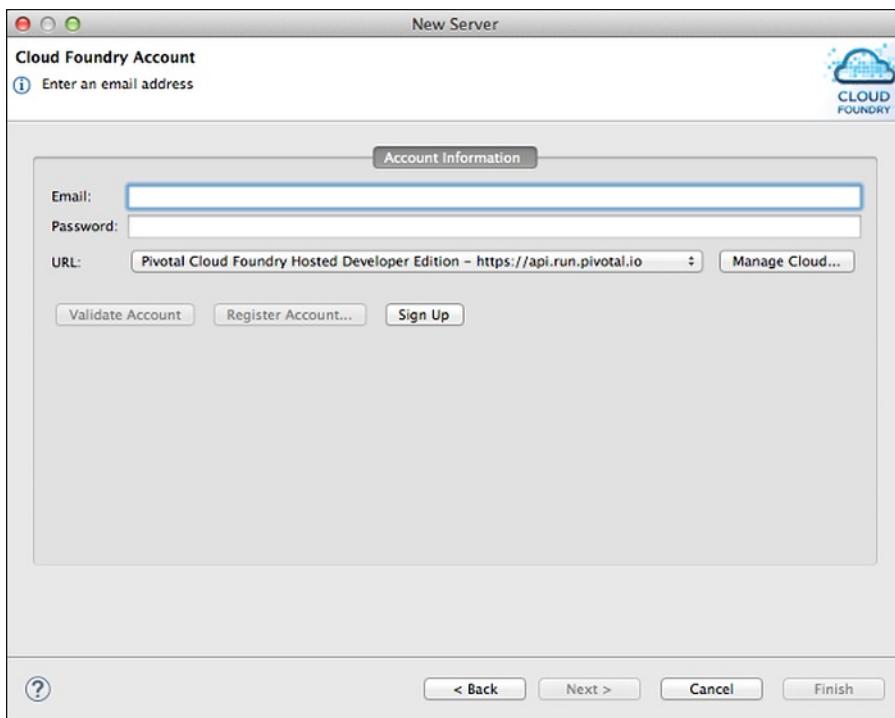
Follow the step below to clone a server:

1. Perform one of the following actions:
 - In the Cloud Foundry server instance editor “Overview” tab, click **Clone Server**.
 - Right-click a Cloud Foundry server instance in the **Servers** view, and select **Clone Server** from the context menu.
2. In the **Organizations and Spaces** window, select the space that you want to target.
3. The name field will be filled with the name of the space that you selected. If desired, edit the server name before clicking finish **Finish**.

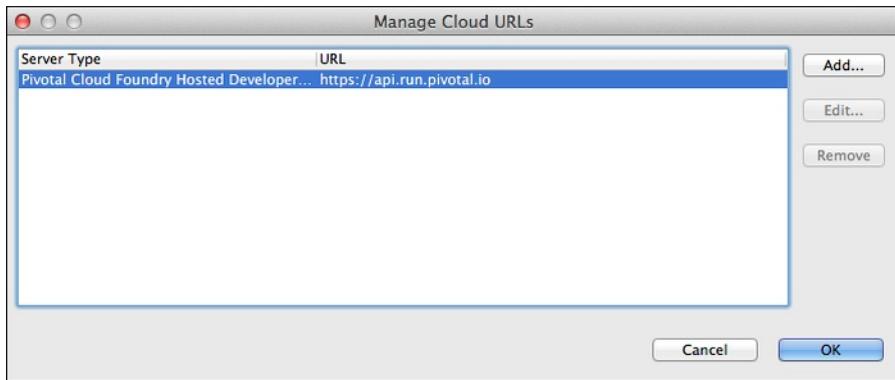
Add a Cloud Foundry Instance URL

You can configure the plugin to work with any Cloud Foundry instances to which you have access. To do so:

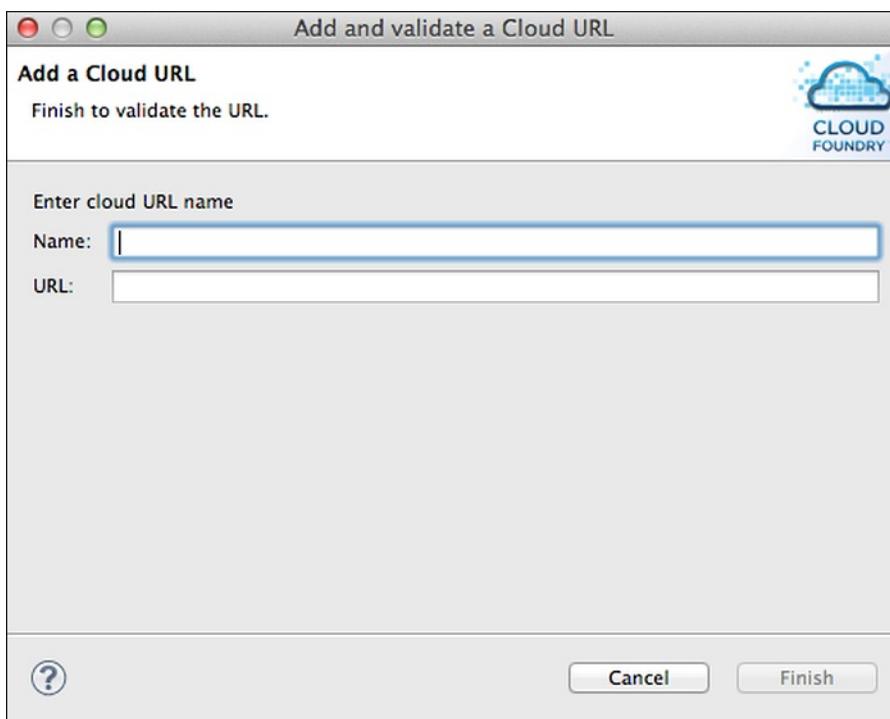
1. Perform steps 1 and 2 of [Create a Cloud Foundry Server](#).
2. In the **Cloud Foundry Account** window, enter the email account and password that you use to log on to the target instance, then click **Manage Cloud URLs**



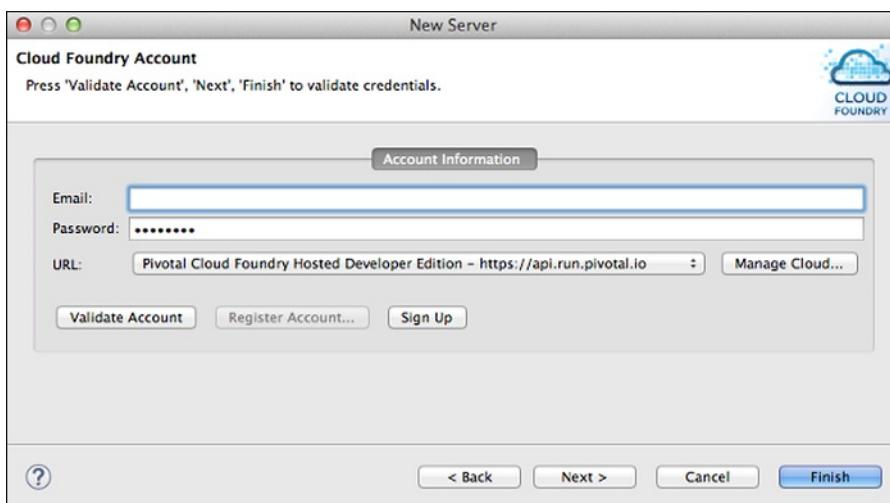
3. In the **Manage Cloud URLs** window, click **Add**.



4. In the **Add a Cloud URL** window, enter the name and URL of the target cloud instance and click **Finish**.

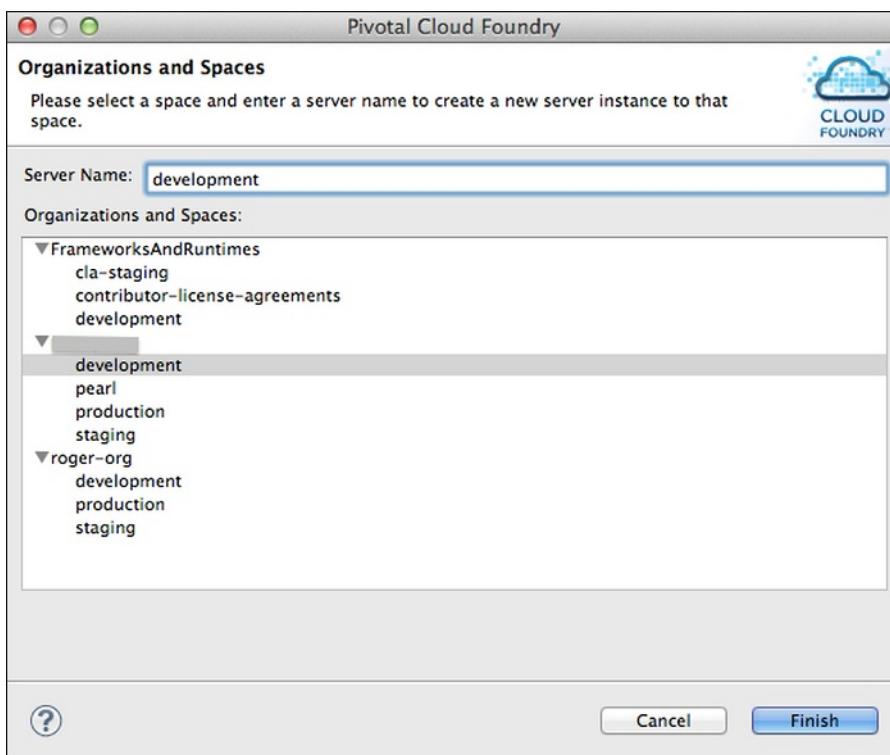


5. The new cloud instance should appear in the list on the **Manage Cloud URLs** window. Click **OK** to proceed.
6. In the **Cloud Foundry Account** window, click **Validate Account**.
7. The **Cloud Foundry Account** window is refreshed and displays a message indicating whether or not your credentials were valid. Click **Next**.



8. In the **Organizations and Spaces** window, select the space that you want to target, and click **Finish**.

Note: If you do not select a space, the server will be configured to connect to the default space, which is the first encountered in a list of your spaces.



- Once you have successfully configured the Pivotal Cloud Foundry server, it will appear in the **Servers** view of the Eclipse or STS user interface. To familiarize yourself with the plugin user interface, see [About the Plugin User Interface](#). Following this step, proceed to [Deploy an Application](#).

Cloud Foundry Java Client Library

Introduction

This is a guide to using the Cloud Foundry Java Client Library to manage an account on a Cloud Foundry instance.

Getting the Library

The Cloud Foundry Java Client Library is available in Maven Central. The library can be added as a dependency to Maven or Gradle project using the following information.

Maven

The `cloudfoundry-client-lib` dependency can be added to your `pom.xml` as follows:

```
<dependencies>
  <dependency>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cloudfoundry-client-lib</artifactId>
    <version>1.1.2</version>
  </dependency>
</dependencies>
```

Gradle

To use the Java client library in a Gradle project, add the `cloudfoundry-client-lib` dependency to your `build.gradle` file:

```
repositories {
  mavenCentral()
}

dependencies {
  compile 'org.cloudfoundry:cloudfoundry-client-lib:1.1.2'
}
```

Sample Code

The following is a very simple sample application that connects to a Cloud Foundry instance, logs in, and displays some information about the Cloud Foundry account. When running the program, provide the Cloud Foundry target (e.g. <https://api.run.pivotl.io>) along with a valid user name and password as command-line parameters.

```
import org.cloudfoundry.client.lib.CloudCredentials;
import org.cloudfoundry.client.lib.CloudFoundryClient;
import org.cloudfoundry.client.lib.domain.CloudApplication;
import org.cloudfoundry.client.lib.domain.CloudService;
import org.cloudfoundry.client.lib.domain.CloudSpace;

import java.net.MalformedURLException;
import java.net.URI;
import java.net.URL;

public final class JavaSample {

    public static void main(String[] args) {
        String target = args[0];
        String user = args[1];
        String password = args[2];

        CloudCredentials credentials = new CloudCredentials(user, password);
        CloudFoundryClient client = new CloudFoundryClient(credentials, getTargetURL(target));
        client.login();

        System.out.printf("%nSpaces:%n");
        for (CloudSpace space : client.getSpaces()) {
            System.out.printf(" %s\t(%s)%n", space.getName(), space.getOrganization().getName());
        }

        System.out.printf("%nApplications:%n");
        for (CloudApplication application : client.getApplications()) {
            System.out.printf(" %s%n", application.getName());
        }

        System.out.printf("%nServices%n");
        for (CloudService service : client.getServices()) {
            System.out.printf(" %s\t(%s)%n", service.getName(), service.getLabel());
        }
    }

    private static URL getTargetURL(String target) {
        try {
            return URI.create(target).toURL();
        } catch (MalformedURLException e) {
            throw new RuntimeException("The target URL is not valid: " + e.getMessage());
        }
    }
}
```

For more details on the Cloud Foundry Java Client Library, view the [source](#) on GitHub. The [domain package](#) shows the objects that can be queried and inspected.

The source for the Cloud Foundry [Maven Plugin](#) is also a good example of using the Java client library.

Build Tool Integration

This page assumes you are using cf CLI v6 and version 1.1.2 of either the Cloud Foundry Maven plugin or the Cloud Foundry Gradle plugin.

Maven Plugin

The Cloud Foundry Maven plugin allows you to deploy and manage applications with Maven goals. This plugin provides Maven users with access to the core functionality of the Cloud Foundry cf command-line tool.

Basic Configuration

To install the Cloud Foundry Maven plugin, add the `cf-maven-plugin` to the `<plugins>` section of the `pom.xml` file:

```
<plugins>
  <plugin>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cf-maven-plugin</artifactId>
    <version>1.1.2</version>
  </plugin>
</plugins>
```

This minimal configuration is sufficient to execute many of the Maven goals provided by the plugin, as long as you provide all other necessary configuration information through command-line parameters.

Additional Configuration

Instead of relying on command-line parameters, you can include additional configuration information in the `pom.xml` by nesting a `<configuration>` section within the `cf-maven-plugin` section.

Example:

```
<plugins>
  <plugin>
    <groupId>org.cloudfoundry</groupId>
    <artifactId>cf-maven-plugin</artifactId>
    <version>1.1.2</version>
    <configuration>
      <target>http://api.run.pivotl.io</target>
      <org>mycloudfoundry-org</org>
      <space>development</space>
      <appname>my-app</appname>
      <url>my-app.example.com</url>
      <memory>512</memory>
      <instances>2</instances>
      <env>
        <ENV-VAR-NAME>env-var-value</ENV-VAR-NAME>
      </env>
      <services>
        <service>
          <name>my-rabbitmq</name>
          <label>rabbitmq</label>
          <provider>rabbitmq</provider>
          <version>n/a</version>
          <plan>small_plan</plan>
        </service>
      </services>
    </configuration>
  </plugin>
</plugins>
```

After adding and configuring the plugin you can build and push the application to Cloud Foundry with the following command:

```
$ mvn clean package cf:push
```

Security Credentials

While you can include Cloud Foundry security credentials in the `pom.xml` file, a more secure method is to store the credentials in the Maven `settings.xml` file, using the server XML configuration element (<http://maven.apache.org/settings.html#Servers>). The default location for this configuration file is `~/.m2/settings.xml`.

To implement this:

1. Add a server to the servers section of the `settings.xml` file. Include the Cloud Foundry security credentials (username and password) and an `ID` tag. The `pom.xml` references this ID to access the security credentials.

```
<settings>
    ...
    <servers>
        ...
        <server>
            <id>cloud-foundry-credentials</id>
            <username>my-name@email.com</username>
            <password>s3cr3t</password>
        </server>
        ...
    </servers>
    ...
</settings>
```

2. Add a server configuration element referencing the ID to the `pom.xml` file:

```
<plugins>
    <plugin>
        <groupId>org.cloudfoundry</groupId>
        <artifactId>cf-maven-plugin</artifactId>
        <version>1.1.2</version>
        <configuration>
            <server>cloud-foundry-credentials</server>
            ...
        </configuration>
    </plugin>
</plugins>
```

Command-Line Usage

Key functionality available with the Cloud Foundry Maven plugin:

Maven Goal	Cloud Foundry Command	Syntax
cf:login	login -u USERNAME	\$ mvn cf:login
cf:logout	logout	\$ mvn cf:logout
cf:app	app APPNAME	\$ mvn cf:app [-Dcf.appname=APPNAME]
cf:apps	apps	\$ mvn cf:apps
cf:target	api	\$ mvn cf:target
cf:push	push	\$ mvn cf:push [-Dcf.appname=APPNAME] [-Dcf.path=PATH] [-Dcf.url=URL] [-Dcf.no-start=BOOLEAN]
cf:start	start APPNAME	\$ mvn cf:start [-Dcf.appname=APPNAME]
cf:stop	stop APPNAME	\$ mvn cf:stop [-Dcf.appname=APPNAME]
cf:restart	restart APPNAME	\$ mvn cf:stop [-Dcf.appname=APPNAME]
cf:delete	delete APPNAME	\$ mvn cf:delete [-Dcf.appname=APPNAME]
cf:scale	scale APPNAME -i INSTANCES	\$ mvn cf:scale [-Dcf.appname=APPNAME] [-Dcf.instances=INTEGER]
cf:env	env APPNAME	\$ mvn cf:env [-Dcf.appname=APPNAME]
cf:services	services	\$ mvn cf:services
cf:create-services	create-service SERVICE PLAN SERVICE_INSTANCE	\$ mvn cf:create-services
cf:delete-services	delete-service SERVICE_INSTANCE	\$ mvn cf:delete-service

cf:bind-services	bind-service APPNAME SERVICE_INSTANCE	\$ mvn cf:bind-services
cf:unbind-services	unbind-service APPNAME SERVICE_INSTANCE	\$ mvn cf:unbind-services

Gradle Plugin

The Cloud Foundry Gradle plugin allows you to deploy and manage applications with Gradle tasks. This plugin provides Gradle users with access to the core functionality of the Cloud Foundry cf command-line tool.

Basic Configuration

To install the Cloud Foundry Gradle plugin, add the `cf-gradle-plugin` as a dependency in the `buildscript` section of the `build.gradle` file:

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'org.cloudfoundry:cf-gradle-plugin:1.1.2'
        ...
    }
}

apply plugin: 'cloudfoundry'
```

This minimal configuration is sufficient to execute many of the Gradle tasks provided by the plugin, as long as you provide all other necessary configuration information through command-line parameters

Additional Configuration

Instead of relying on command-line parameters, you can add additional configuration information to `build.gradle` in a `cloudfoundry` configuration section:

```
cloudfoundry {
    target = "https://api.run.pivotal.io"
    space = "deployment"
    file = file("path/to/my/file.war")
    uri = "my-app.example.com"
    memory = 512
    instances = 1
    env = [ "key": "value" ]
    serviceInfos {
        "my_rabbitmq" {
            label = "rabbitmq"
            plan = "small_plan"
            bind = true
        }
    }
}
```

After adding and configuring the plugin you can build and push the application to Cloud Foundry with the following command:

```
$ gradle clean assemble cfPush
```

Security Credentials

While you can include Cloud Foundry security credentials in the `build.gradle` file, a more secure method is to store the credentials in a `gradle.properties` file. This file can be placed in either the project directory or in the `~/.gradle` directory.

To implement this, add `cfUsername` and `cfPassword` with the Cloud Foundry security credentials parameters to the `gradle.properties` file as follows:

```
cfUsername=user@example.com  
cfPassword=examplePassword
```

(Note there are no quotes around either the username or password.)

Command-Line Usage

Key functionality available with the Cloud Foundry Gradle plugin:

Gradle Task	Cloud Foundry Command	Syntax
cfLogin	login -u USERNAME	\$ gradle cfLogin
cfLogout	logout	\$ gradle cfLogout
cfApp	app APPNAME	\$ gradle cfApp [-PcfApplication=APPNAME]
cfApps	apps	\$ gradle cfApps
cfTarget	api	\$ gradle cfTarget
cfPush	push	\$ gradle cfPush [-PcfApplication=APPNAME] [-PcfUri=URL] [-PcfStartApp=BOOLEAN]
cfStart	start APPNAME	\$ gradle cfStart [-PcfApplication=APPNAME]
cfStop	stop APPNAME	\$ gradle cfStop [-PcfApplication=APPNAME]
cfRestart	restart APPNAME	\$ gradle cfStop [-PcfApplication=APPNAME]
cfDelete	delete APPNAME	\$ gradle cfDelete [-PcfApplication=APPNAME]
cfScale	scale APPNAME -i INSTANCES	\$ gradle cfScale [-PcfApplication=APPNAME] [-PcfInstances=INTEGER]
cfEnv	env APPNAME	\$ gradle cfEnv [-PcfApplication=APPNAME]
cfServices	services	\$ gradle cfServices
cfCreateService	create-service SERVICE PLAN SERVICE_INSTANCE	\$ gradle cfCreateServices
cfDeleteServices	delete-service SERVICE_INSTANCE	\$ gradle cfDeleteServices
cfBind	bind-service APPNAME SERVICE_INSTANCE	\$ gradle cfBind
cfUnbind	unbind-service APPNAME SERVICE_INSTANCE	\$ gradle cfUnbind

Node.js Buildpack

Use the Node.js buildpack with Node or JavaScript applications.

See the following topics:

- [Tips for Node.js Developers](#)
- [Environment Variables Defined by the Node Buildpack](#)
- [Configure Service Connections for Node](#)

The source for the buildpack is [here](#).

Buildpack Logging and Application Logging

The buildpack only runs during the staging process, and therefore only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

Once staging is done, the buildpack stops logging. Application logging is a separate concern.

Your application must write to STDOUT or STDERR for its logs to be included in the Loggregator stream. See [Application Logging in Cloud Foundry](#).

Tips for Node.js Applications

This page assumes you are using cf CLI v6.

This topic provides Node-specific information to supplement the general guidelines in the [Deploy an Application](#) topic.

Application Package File

Cloud Foundry expects a `package.json` in your Node.js application. You can specify the version of Node.js you want to use in the `engine` node of your `package.json` file. As of April, 2015, and build pack version 1.3, Cloud Foundry uses 0.12.2 as the default. See the GitHub [Node.js buildpack page](#) for current information.

Example `package.json` file:

```
{  
  "name": "first",  
  "version": "0.0.1",  
  "author": "Demo",  
  "dependencies": {  
    "express": "3.4.8",  
    "consolidate": "0.10.0",  
    "express": "3.4.8",  
    "swig": "1.3.2"  
  },  
  "engines": {  
    "node": "0.12.2",  
    "npm": "2.7.4"  
  }  
}
```

Application Port

You must use the PORT environment variable to determine which port your application should listen on. In order to also run your application locally, you may want to make port 3000 the default:

```
app.listen(process.env.VCAP_APP_PORT || 3000);
```

Application Start Command

Node.js applications require a start command. You can specify a Node.js application's web start command in a Procfile or in the application deployment manifest.

You will be asked if you want to save your configuration the first time you deploy. This will save a `manifest.yml` in your application with the settings you entered during the initial push. Edit the `manifest.yml` file and create a start command as follows:

```
---  
applications:  
- name: my-app  
  command: node my-app.js  
... the rest of your settings ...
```

Alternately, specify the start command with `cf push -c`.

```
$ cf push my-app -c "node my-app.js"
```

Application Bundling

You do not need to run `npm install` before deploying your application. Cloud Foundry will run it for you when your

application is pushed. If you would prefer to run `npm install` and create a `node_modules` folder inside of your application, this is also supported.

Solving Discovery Problems

If Cloud Foundry does not automatically detect that your application is a Node.js application, you can override the auto-detection by specifying the Node.js buildpack.

Add the buildpack into your `manifest.yml` and re-run `cf push` with your manifest:

```
---  
applications:  
- name: my-app  
  buildpack: https://github.com/cloudfoundry/nodejs-buildpack  
... the rest of your settings ...
```

Alternately, specify the buildpack on the command line with `cf push -b`:

```
$ cf push my-app -b https://github.com/cloudfoundry/nodejs-buildpack
```

Binding Services

Refer to [Configure Service Connections for Node.js](#).

About the Node.js Buildpack

For information about using and extending the Node.js buildpack in Cloud Foundry, see the [nodejs-buildpack repo](#).

You can find current information about this buildpack on the Node.js buildpack [release page](#) in GitHub.

The buildpack uses a default Node.js version of `0.12.7`. To specify the versions of Node.js and npm an application requires, edit the application's `package.json`, as described in "node.js and npm versions" in the [nodejs-buildpack repo](#).

Environment Variables

You can access environments variable programmatically.

For example, you can obtain `VCAP_SERVICES` like this:

```
process.env.VCAP_SERVICES
```

Environment variables available to you include both those [defined by the DEA](#) and those defined by the Node.js buildpack, as described below.

BUILD_DIR

Directory into which Node.js is copied each time a Node.js application is run.

CACHE_DIR

Directory that Node.js uses for caching.

PATH

The system path used by Node.js.

```
PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin
```


Environment Variables Defined by the Node Buildpack

When you use the Node buildpack, you get three Node-specific environment variables in addition to the regular [Cloud Foundry environment variables](#).

- `BUILD_DIR` — The directory into which Node.js is copied each time a Node.js application is run.
- `CACHE_DIR` — The directory that Node.js uses for caching.
- `PATH` — The system path used by Node.js:
`PATH=/home/vcap/app/bin:/home/vcap/app/node_modules/.bin:/bin:/usr/bin`

Configure Service Connections for Node.js

This page assumes that you are using cf CLI v6.

This guide is for developers who wish to bind a data source to a Node.js application deployed and running on Cloud Foundry.

Parse VCAP_SERVICES for Credentials

You must parse the VCAP_SERVICES environment variable in your code to get the required connection details such as host address, port, user name, and password.

For example, if you are using PostgreSQL, your VCAP_SERVICES environment variable might look something like this:

```
{  
  "mypostgres": [{  
    "name": "myinstance",  
    "credentials": {  
      "uri": "postgres://myusername:mypassword@host.example.com:5432/serviceinstance"  
    }  
  }]  
}
```

This example JSON is simplified; yours may contain additional properties.

Parse with cfenv

The `cfenv` package provides access to Cloud Foundry application environment settings by parsing all the relevant environment. The settings are returned as JavaScript objects. `cfenv` provides reasonable defaults when running locally, as well as when running as a Cloud Foundry application.

- <https://www.npmjs.org/package/cfenv>

Manual Parsing

First, parse the VCAP_SERVICES environment variable.

For example:

```
var vcap_services = JSON.parse(process.env.VCAP_SERVICES)
```

Then pull out the credential information required to connect to your service. Each service packages requires different information. If you are working with Postgres, for example, you will need a `uri` to connect. You can assign the value of the `uri` to a variable as follows:

```
var uri = vcap_services.mypostgres[0].credentials.uri
```

Once assigned, you can use your credentials as you would normally in your program to connect to your database.

Connecting to a Service

You must include the appropriate package for the type of services your application uses. For example:

- Rabbit MQ via the [amqp](#) module
- Mongo via the [mongodb](#) and [mongoose](#) modules
- MySQL via the [mysql](#) module
- Postgres via the [pg](#) module
- Redis via the [redis](#) module

Add the Dependency to package.json

Edit `package.json` and add the intended module to the `dependencies` section. Normally, only one would be necessary, but for the sake of the example we will add all of them:

```
{  
  "name": "hello-node",  
  "version": "0.0.1",  
  "dependencies": {  
    "express": "*",  
    "mongodb": "*",  
    "mongoose": "*",  
    "mysql": "*",  
    "pg": "*",  
    "redis": "*",  
    "amqp": "*"  
  },  
  "engines": {  
    "node": "0.8.x"  
  }  
}
```

You must run `npm shrinkwrap` to regenerate your `npm-shrinkwrap.json` file after you edit `package.json`.

Ruby Buildpack

Use the Ruby buildpack with Ruby, Rack, Rails or Sinatra applications.

See the following topics:

- [Getting Started Deploying Ruby Apps](#)
- [Getting Started Deploying Ruby on Rails Apps](#)
- [Deploy a Sample Ruby on Rails App](#)
- [Configure a Production Server for Ruby Apps](#)
- [Configure Rake Tasks for Deployed Apps](#)
- [Tips for Ruby Developers](#)
- [Environment Variables Defined by the Ruby Buildpack](#)
- [Configure Service Connections for Ruby](#)

The source for the buildpack is [here ↗](#).

Buildpack Logging and Application Logging

The buildpack only runs during the staging process, and only logs what is important to staging, such as what is being downloaded, what the configuration is, and work that the buildpack does on your application.

The buildpack stops logging when the staging process finishes. Application logging is a separate concern.

If you are deploying a Ruby, Rack, or Sinatra application, your application must write to STDOUT or STDERR to include its logs in the Loggregator stream. See [Application Logging in Cloud Foundry](#).

If you are deploying a Rails application, the buildpack may or may not automatically install the necessary plugin or gem for logging, depending on the Rails version of the application:

- Rails 2.x: The buildpack automatically installs the `rails_log_stdout` plugin into the application.
- Rails 3.x: The buildpack automatically installs the `rails_12factor` gem if it is not present and issues a warning message.
- Rails 4.x: The buildpack only issues a warning message that the `rails_12factor` gem is not present, but does not install the gem.

You must add the `rails_12factor` gem to your `Gemfile` to quiet the warning message.

For more information about the `rails_log_stdout` plugin, refer to the [Github README ↗](#).

For more information about the `rails_12factor` gem, refer to the [Github README ↗](#).

Getting Started Deploying Ruby Apps

This guide is intended to walk you through deploying a Ruby app to Elastic Runtime. If you experience a problem following the steps below, check the [Known Issues](#) topic, or refer to the [Troubleshooting Application Deployment and Health](#) topic.

Sample App Step

If you want to go through this tutorial using the sample app, run

```
git clone https://github.com/cloudfoundry-samples/pong_matcher_ruby.git
```

 to clone the `pong_matcher_ruby` app from GitHub, and follow the instructions in the Sample App Step sections.

 **Note:** Ensure that your Ruby app runs locally before continuing with this procedure.

Deploy a Ruby Application

This section describes how to deploy a Ruby application to Elastic Runtime, and uses output from a sample app to show specific steps of the deployment process.

Prerequisites

- A Ruby 2.x application that runs locally on your workstation
- [Bundler](#) configured on your workstation
- Basic to intermediate Ruby knowledge
- The [cf Command Line Interface \(CLI\)](#) installed on your workstation

Step 1: Create and Bind a Service Instance for a Ruby Application

This section describes using the CLI to configure a Redis Cloud managed service instance for an app. You can use either the CLI or the [Apps Manager](#) to perform this task.

Elastic Runtime supports two types of service instances:

- Managed services integrate with Elastic Runtime through service brokers that offer services and plans and manage the service calls between Elastic Runtime and a service provider.
- User-provided service instances enable you to connect your application to pre-provisioned external service instances.

For more information about creating and using service instances, refer to the [Services Overview](#) topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans that are available to you.

The example shows three of the available managed database-as-a-service providers and the plans that they offer:
`cleardb MySQL`, `elephantsql PostgreSQL` as a Service, and `mongolab MongoDB`-as-a-Service.

```
$ cf marketplace
Getting services from marketplace in org Cloud-Apps / space development as clouduser@example.com...
OK

service      plans  description
...
cleardb      spark, boost, amp, shock  Highly available MySQL for your Apps
...
elephantsql  turtle, panda, hippo, elephant  PostgreSQL as a Service
...
mongolab     sandbox   Fully-managed MongoDB-as-a-Service
...
```

Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

Sample App Step

Run `cf create-service rediscloud 25mb redis`. This creates a service instance named `redis` that uses the `rediscloud` service and the `25mb` plan, as the example below shows.

```
$ cf create-service rediscloud 25mb redis
Creating service redis in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Bind a Service Instance

When you bind an app to a service instance, Elastic Runtime writes information about the service instance to the `VCAP_SERVICES` app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm if they support this functionality.

You can bind a service to an application with the command `cf bind-service APPLICATION SERVICE_INSTANCE`.

Alternately, you can configure the deployment manifest file by adding a `services` block to the `applications` block and specifying the service instance. For more information and an example on service binding using a manifest, see the Sample App Step.

You can also bind a service using the [Apps Manager](#).

Sample App Step

You can skip this step. The manifest for the sample app contains a `services` sub-block in the `applications` block, as the example below shows. This binds the `redis` service instance that you created in the previous step.

```
services:
  - redis
```

Step 2: Configure Deployment Options

Configure the Deployment Manifest

You can specify app deployment options in a manifest that the `cf push` command uses. For more information about application manifests and supported attributes, refer to the [Deploying with Application Manifests](#) topic.

Configure a Production Server

Elastic Runtime uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, Elastic Runtime can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the [Configure a Production Server](#) topic for help configuring a server other than WEBrick.

Sample App Step

You can skip this step. The `manifest.yml` file for `pong_matcher_ruby` does not require any additional configuration to deploy the app.

Step 3: Log in and Target the API Endpoint

Run `cf login -a API_ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is [the URL of the Cloud Controller in your Elastic Runtime instance](#).

Sample App Step

You must do this step to run the sample app.

Step 4: Deploy an App

 **Note:** You must use the cf CLI to deploy apps.

From the root directory of your application, run `cf push APP_NAME` to deploy your application.

`cf push APP_NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP_NAME and DOMAIN is specified by your administrator. Your DOMAIN is `example.com`. For example: `cf push my-app` creates the URL `my-app.example.com`.

The URL for your app must be unique from other apps that Elastic Runtime hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words.
- `cf help push` to view other options for this command.

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP_NAME`.

Once your app deploys, browse to your app URL. Search for the `urls` field in the `App started` block in the output of the `cf push` command. Use the URL to access your app online.

Sample App Step

Run `cf push pong_matcher_ruby -n HOST_NAME`.

Example: `cf push pong_matcher_ruby -n pongmatch-ex12`

The example below shows the terminal output of deploying the `pong_matcher_ruby` app. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `redis` service and follows the instructions in the manifest to start one instance of the app with 256M. After the app starts, the output displays the health and status of the app.

 **Note:** The `pong_matcher_ruby` app does not include a web interface. To interact with the `pong_matcher_ruby` app, see the interaction instructions on GitHub: https://github.com/cloudfoundry-samples/pong_matcher_ruby.

```
$ cf push pong_matcher_ruby -n pongmatch-ex12
Using manifest file /Users/clouduser/workspace/pong_matcher_ruby/manifest.yml

Creating app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

Creating route pongmatch-ex12.example.com
      Binding pongmatch-ex12.example.com to pong_matcher_ruby...
OK

Uploading pong_matcher_ruby...
Uploading app files from: /Users/clouduserl/workspace/pong_matcher_ruby
Uploading 8.8K, 12 files
OK
Binding service redis to app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

Starting app pong_matcher_ruby in org cf-Cloud-Apps / space development as clouduser@example.com...
OK
...
0 of 1 instances running, 1 starting
1 of 1 instances running

App started

Showing health and status for app pong_matcher_ruby in org Cloud-Apps / space development as clouduser@example.com...
OK

requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: pongmatch-ex12.cfapps.io

     state      since          cpu    memory      disk
#0   running   2014-12-09 10:04:40 AM  0.0%  35.2M of 256M  45.8M of 1G
```

Step 5: Test a Deployed App

You've deployed an app to Elastic Runtime!

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime account. For example, you could edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

See the [Manage Your Application with the cf CLI](#) section for more information. See also [Using the Apps Manager](#).

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf Command Line Interface (CLI) topics, especially the [Getting Started with cf CLI v6](#) topic.

Note: You cannot perform certain tasks in the CLI or the [Apps Manager](#) because these are commands that only a Elastic Runtime administrator can run. If you are not a Elastic Runtime administrator, the following message displays for these types of commands:

`error code: 10003, message: You are not authorized to perform the requested action` For more information about specific Admin commands you can perform with the Apps Manager, depending on your user role, refer to the [Understanding Apps Manager Permissions](#) topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the [Troubleshooting Application Deployment and Health](#) topic to learn more about troubleshooting.

App Deploy Fails

Even when deploying an app fails, the app might exist on Elastic Runtime. Run `cf apps` to review the apps in the currently targeted org and space. You might be able to correct the issue using the CLI or the [Apps Manager](#), or you might have to delete the app and redeploy.

Common reasons deploying an app fails include:

- You did not successfully create and bind a needed service instance to the app, such as a PostgreSQL or MongoDB service instance. Refer to Step 2: Create and Bind a Service Instance for a Ruby Application.
- You did not successfully create a unique URL for the app. Refer to the troubleshooting tip **App Requires Unique URL**.

App Requires Unique URL

Elastic Runtime requires that each app that you deploy has a unique URL. Otherwise, the new app URL collides with an existing app URL and Elastic Runtime cannot successfully deploy the app. You can resolve this issue by running `cf push` with either of the following flags to create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words. Using this option might create a long URL, depending on the number of words that the app name includes.

Getting Started Deploying Ruby on Rails Apps

This guide walks you through deploying a Ruby on Rails (RoR) app to Elastic Runtime. To deploy a sample RoR app, refer to the [Deploy a Sample Ruby on Rails App](#) topic.

 **Note:** Ensure that your RoR app runs locally before continuing with this procedure.

Prerequisites

- A Rails 4.x app that runs locally
- [Bundler](#) configured on your workstation
- Intermediate to advanced RoR knowledge
- The [cf Command Line Interface \(CLI\)](#)

Step 1: Create and Bind a Service Instance for a RoR Application

This section describes using the CLI to configure a PostgreSQL managed service instance for an app. For more information about creating and using service instances, refer to the [Services Overview](#) topic.

Create a Service Instance

Run `cf marketplace` to view managed and user-provided services and plans available to you.

Run `cf create-service SERVICE PLAN SERVICE_INSTANCE` to create a service instance for your app. Choose a SERVICE and PLAN from the list, and provide a unique name for the SERVICE_INSTANCE.

Bind a Service Instance

When you bind an app to a service instance, Elastic Runtime writes information about the service instance to the VCAP_SERVICES app environment variable. The app can use this information to integrate with the service instance.

Most services support bindable service instances. Refer to your service provider's documentation to confirm whether they support this functionality.

To bind a service to an application, run `cf bind-service APPLICATION SERVICE_INSTANCE`.

Step 2: Configure Deployment Options

Configure the Deployment Manifest

You can specify app deployment options in a manifest that the `cf push` command uses. For more information about application manifests and supported attributes, refer to the [Deploying with Application Manifests](#) topic.

Configure a Production Server

Elastic Runtime uses the default standard Ruby web server library, WEBrick, for Ruby and RoR apps. However, Elastic Runtime can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn. If your app requires a more robust web server, refer to the [Configure a Production Server](#) topic for help configuring a server other than WEBrick.

Step 3: Log in and Target the API Endpoint

Run `cf login -a API_ENDPOINT`, enter your login credentials, and select a space and org. The API endpoint is [the URL of](#)

the Cloud Controller in your Elastic Runtime instance [.](#)

Step 4: Deploy Your App

From the root directory of your application, run `cf push APP-NAME --random-route` to deploy your application.

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. Your DOMAIN is `example.com`. For example: `cf push my-app` creates the URL `my-app.example.com`.

The URL for your app must be unique from other apps that Elastic Runtime hosts or the push will fail. Use the following options to help create a unique URL:

- `-n` to assign a different HOST name for the app.
- `--random-route` to create a URL that includes the app name and random words.
- `cf help push` to view other options for this command.

To view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

In the terminal output of the `cf push` command, the `urls` field of the `App started` block contains the app URL. This is the `HOST_NAME` you specified with the `-n` flag plus the domain `example.com`. Once your app deploys, use this URL to access the app online.

Next Steps

You've deployed an app to Elastic Runtime! Consult the sections below for information about what to do next.

Test a Deployed App

Use the cf CLI or the [Apps Manager](#) to review information and administer your app and your Elastic Runtime account. For example, you could edit the `manifest.yml` to increase the number of app instances from 1 to 3, and redeploy the app with a new app name and host name.

Manage Your Application with the cf CLI

Run `cf help` to view a complete list of commands, grouped by task categories, and run `cf help COMMAND` for detailed information about a specific command. For more information about using the cf CLI, refer to the cf Command Line Interface (CLI) topics, especially the [Getting Started with cf CLI v6](#) topic.

Troubleshooting

If your application fails to start, verify that the application starts in your local environment. Refer to the [Troubleshooting Application Deployment and Health](#) topic to learn more about troubleshooting.

Deploy a Sample Ruby on Rails Application

This topic walks the reader through the process of deploying a sample Ruby on Rails app to Elastic Runtime.

The following subjects are covered:

- Using the CF CLI to target a CF deployment.
- App manifests.
- Deploying an app.
- How app URL endpoints are generated in CF.
- Adding a service instance (A postgres database, in this example).

Prerequisites

- A working CF deployment. See [Deploying CF Using Bosh Lite](#).
- [Cloud Foundry CLI](#)
- Username and password with **Space Developer** permissions. See [Roles and Permissions](#) for more information.

Step 1: Clone the App

Run the following in the terminal to clone the `rails_sample_app` from GitHub:

```
git clone https://github.com/cloudfoundry-samples/rails_sample_app
```

Change into the `rails_sample_app` directory and examine its contents. The release contains a very simple Ruby on Rails app with the single addition of the application manifest `manifest.yml`, which allows CF to deploy the app. For more about application manifests, see [Deploying with Application Manifests](#).

```
~/workspace/rails_sample_app $ ls
Gemfile README.markdown app bin config db files manifest.yml script vendor
Gemfile.lock   Rakefile autotest bundler_stubs config.ru doc lib public spec
```

Step 2: Log in and Target the API Endpoint

Run the following to log in and target the API endpoint of your Cloud Foundry deployment. You will be prompted for login credentials, and to select a space and org.

For `YOUR-API-ENDPOINT`, enter the IP of the BOSH director for your Cloud Foundry deployment. The IP must be entered in the following format: `http://api.IP-ADDRESS`

```
cf login -a YOUR-API-ENDPOINT
```

For more information about Spaces and Orgs, see [Orgs, Spaces, Roles, and Permissions](#).

Step 3: Create a Service Instance for our Application Database

In Cloud Foundry, a database is considered a type of service. To add a database for our sample application we will create an instance of [elephantsql](#), a third party database as a service provider.

Run the following terminal command to create a postgresql service instance for our app.

```
$ cf create-service elephantsql turtle rails-postgres
Creating service rails-postgres in org Cloud-Apps / space development as clouduser@example.com....
OK
```

Our service instance is named `rails-postgres`. It uses the `elephantsql` service and the `turtle` plan.

The manifest for the sample app contains a `services` sub-block in the `applications` block, as the example below shows. This binds the `rails-postgres` service instance you created in the previous step.

```
---  
applications:  
- name: rails-sample  
  memory: 256M  
  instances: 1  
  path: .  
  command: bundle exec rake db:migrate && bundle exec rails s -p $PORT  
  services:  
    - rails-postgres
```

Step 4: Deploy the App

From the root directory of your application, run the following to deploy the app:

```
cf push rails_sample_app
```

`cf push APP-NAME` creates a URL route to your application in the form HOST.DOMAIN, where HOST is your APP-NAME and DOMAIN is specified by your administrator. For example, if your domain is `example.com` then running `cf push my-app` creates the URL `my-app.example.com`.

If you want to view log activity while the app deploys, launch a new terminal window and run `cf logs APP-NAME`.

The example below shows the terminal output of deploying the `rails_sample_app`. `cf push` uses the instructions in the manifest file to create the app, create and bind the route, and upload the app. It then binds the app to the `rails-postgres` service and follows the instructions in the manifest to start one instance of the app with 256M. After the app starts, the output displays the health and status of the app.

```
$ cf push rails_sample_app  
Using manifest file ~/workspace/rails_sample_app/manifest.yml  
  
Updating app rails_sample_app in org Cloud-Apps / space development as clouduser@example.com...  
OK  
  
Using route my-app.example.com  
Uploading rails_sample_app...  
Uploading app files from: ~/workspace/rails_sample_app  
Uploading 445.7K, 217 files  
OK  
Binding service rails-postgres to app rails_sample_app in org Cloud-Apps / space development as clouduser@example.com  
OK  
  
Starting app rails_sample_app in org Cloud-Apps / space development as clouduser@example.com...  
OK  
  
...  
  
0 of 1 instances running, 1 starting  
1 of 1 instances running  
  
App started  
  
Showing health and status for app rails_sample_app in org Cloud-Apps / space development as clouduser@example.com...  
OK  
  
requested state: started  
instances: 1/1  
usage: 256M x 1 instances  
urls: my-app.example.com  
  
     state      since          cpu    memory       disk  
#0  running   2014-08-25 03:32:10 PM  0.0%  68.4M of 256M  73.4M of 1G
```

Step 5: Verifying the app

You can verify the sample app you have pushed by browsing to the URL generated in the output of the previous step. In this example, the url is `urls: my-app.example.com`.

You've now pushed an app to Elastic Runtime! For help deploying your own app, visit the **Deploy Your App: Guides by Framework** section of the help.

Configure a Production Server for Ruby Apps

Elastic Runtime uses the default standard Ruby web server library WEBrick for Ruby and RoR apps. However, Elastic Runtime can support a more robust production web server, such as Phusion Passenger, Puma, Thin, or Unicorn.

To instruct Elastic Runtime to use a web server other than WEBrick, you configure a text file called a *Procfile*. A Procfile enables you to declare required runtime processes, called *process types*, for your web app. Process managers in a server use the process types to run and manage the workload.

When you deploy, Elastic Runtime determines if a Procfile exists and uses the Procfile to configure your app with the specified production web server settings.

In a Procfile, you declare one process type per line and use the following syntax, as shown in the example below:

- `PROCESS_TYPE` is the command name in the format of an alphanumeric string. Specifically for RoR web apps, you can declare `web` and `worker` process types.
- `COMMAND` is the command line to launch the process.

Example process type syntax:

```
PROCESS_TYPE: COMMAND
```

To set a different production web server for your app:

1. Add the gem for the web server to your Gemfile.
2. In the `config` directory of your app, create a new configuration file or modify an existing file.
Refer to your web server documentation for how to configure this file.

Example Puma config file:

```
# config/puma.rb
threads 8,32
workers 3

on_worker_boot do
  # things workers do
end
```

3. In the root directory of your app, create a Procfile and add a command line for a `web` process type that points to your web server.
The example below shows a command that starts a Puma web server and specifies the app runtime environment, TCP port, and paths to the server state information and configuration files. Refer to your web server documentation for how to configure the specific command for a process type.

Example Procfile:

```
web: bundle exec puma -e $RAILS_ENV -p 1234 -S ~/puma -C config/puma.rb
```

Configure Rake Tasks for Deployed Apps

For Elastic Runtime to automatically invoke a Rake task while a Ruby or Ruby on Rails app is deployed, you must:

- Include the Rake task in your app.
- Configure the application start command using the `command` attribute in the application manifest.

The following is an example of how to invoke a Rake database migration task at application startup.

1. Create a file with the Rake task name and the extension `.rake`, and store it in the `lib/tasks` directory of your application.
2. Add the following code to your rake file:

```
namespace :cf do
  desc "Only run on the first application instance"
  task :on_first_instance do
    instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"] rescue nil
    exit(0) unless instance_index == 0
  end
end
```

This Rake task limits an idempotent command to the first instance of a deployed application.

3. Add the task to the `manifest.yml` file with the `command` attribute, referencing the idempotent command `rake db:migrate` chained with a start command.

```
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && rails s -p $PORT
```

Tips for Ruby Developers

This page assumes you are using cf CLI v6.

This page has information specific to deploying Rack, Rails, or Sinatra applications.

Application Bundling

You must run [Bundler](#) to create a `Gemfile` and a `Gemfile.lock`. These files must be in your application before you push to Cloud Foundry.

Rack Config File

For Rack and Sinatra, you must have a `config.ru` file. For example:

```
require './hello_world'  
run HelloWorld.new
```

Asset Precompilation

Cloud Foundry supports the Rails asset pipeline. If you do not precompile assets before deploying your application, Cloud Foundry will precompile them when staging the application. Precompiling before deploying reduces the time it takes to stage an application.

Use the following command to precompile assets before deployment:

```
rake assets:precompile
```

Note that the Rake precompile task reinitializes the Rails application. This could pose a problem if initialization requires service connections or environment checks that are unavailable during staging. To prevent reinitialization during precompilation, add the following line to `application.rb`:

```
config.assets.initialize_on_precompile = false
```

If the `assets:precompile` task fails, Cloud Foundry uses live compilation mode, the alternative to asset precompilation. In this mode, assets are compiled when they are loaded for the first time. You can force live compilation by adding the following line to `application.rb`.

```
Rails.application.config.assets.compile = true
```

Running Rake Tasks

Cloud Foundry does not provide a mechanism for running a Rake task on a deployed application. If you need to run a Rake task that must be performed in the Cloud Foundry environment, rather than locally before deploying or redeploying, you can configure the command that Cloud Foundry uses to start the application to invoke the Rake task.

An application's start command is configured in the application's manifest file, `manifest.yml`, using the `command` attribute.

If you have previously deployed the application, the application manifest should already exist. There are two ways to create a manifest. You can manually create the file and save it in the application's root directory before you deploy the application for the first time. If you do not manually create the manifest file, the cf CLI will prompt you to supply deployment settings when you first push the application, and will create and save the manifest file for you, with the settings you specified interactively. For more information about application manifests, and supported attributes, see [Deploying with Application Manifests](#).

Example: Invoking a Rake database migration task at application startup

The following is an example of the “migrate frequently” method described in the [Migrating a Database in Cloud Foundry](#) topic.

1. Create a Rake task to limit an idempotent command to the first instance of a deployed application:

```
namespace :cf do
  desc "Only run on the first application instance"
  task :on_first_instance do
    instance_index = JSON.parse(ENV["VCAP_APPLICATION"])["instance_index"] rescue nil
    exit(0) unless instance_index == 0
  end
end
```

2. Add the task to the `manifest.yml` file, referencing the idempotent command `rake db:migrate` with the `command` attribute.

```
---
applications:
- name: my-rails-app
  command: bundle exec rake cf:on_first_instance db:migrate && bundle exec rails s -p $PORT -e $RAILS_ENV
```

3. Update the application using `cf push`.

Rails 3 Worker Tasks

This section shows you how to create and deploy an example Rails application that uses a worker library to defer a task that a separate application executes.

The guide also describes how to scale the resources available to the worker application.

 **Note:** Most worker tasks do not serve external requests. Use the `--no-route` flag with the `cf push` command, or `no-route: true` in the application manifest, to suppress route creation and remove existing routes.

Choose a Worker Task Library

You must choose a worker task library. The table below summarizes the three main libraries available for Ruby / Rails:

Library	Description
Delayed::Job	A direct extraction from Shopify where the job table is responsible for a multitude of core tasks.
Resque	A Redis-backed library for creating background jobs, placing those jobs on multiple queues, and processing them later.
Sidekiq	Uses threads to handle many messages at the same time in the same process. It does not require Rails but will integrate tightly with Rails 3 to make background message processing dead simple. This library is also Redis-backed and is actually somewhat compatible with Resque messaging.

For other alternatives, see https://www.ruby-toolbox.com/categories/Background_Jobs.

Create an Example Application

For the purposes of the example application, we will use Sidekiq.

First, create a Rails application with an arbitrary model called “Things”:

```
$ rails create rails-sidekiq
$ cd rails-sidekiq
$ rails g model Thing title:string description:string
```

Add `sidekiq` and `uuidtools` to the Gemfile:

```
source 'https://rubygems.org'

gem 'rails', '3.2.9'
gem 'mysql2'

group :assets do
  gem 'sass-rails',    '~> 3.2.3'
  gem 'coffee-rails', '~> 3.2.1'
  gem 'uglifier',   '>= 1.0.3'
end

gem 'jquery-rails'
gem 'sidekiq'
gem 'uuidtools'
```

Install the bundle.

```
$ bundle install
```

Create a worker (in app/workers) for Sidekiq to carry out its tasks:

```
$ touch app/workers/thing_worker.rb
```

```
class ThingWorker
  include Sidekiq::Worker

  def perform(count)

    count.times do

      thing_uuid = UUIDTools::UUID.random_create.to_s
      Thing.create :title =>"New Thing #{thing_uuid}", :description =>
      "Description for thing #{thing_uuid}"
    end
  end
end
```

This worker will create n number of things, where n is the value passed to the worker.

Create a controller for “Things”:

```
$ rails g controller Thing
```

```
class ThingController < ApplicationController

  def new
    ThingWorker.perform_async(2)
    redirect_to '/thing'
  end

  def index
    @things = Thing.all
  end

end
```

Add a view to inspect our collection of “Things”:

```
$ mkdir app/views/things
$ touch app/views/things/index.html.erb
```

```
nil
```

Deploy the Application

This application needs to be deployed twice for it to work, once as a Rails web application and once as a standalone Ruby application. The easiest way to do this is to keep separate Cloud Foundry manifests for each application type:

Web Manifest: Save this as `web-manifest.yml`:

```
---  
applications:  
- name: sidekiq  
memory: 256M  
instances: 1  
host: sidekiq  
domain: ${target-base}  
path: .  
services:  
- sidekiq-mysql:  
- sidekiq-redis:
```

Worker Manifest: Save this as `worker-manifest.yml`:

```
---  
applications:  
- name: sidekiq-worker  
memory: 256M  
instances: 1  
path: .  
command: bundle exec sidekiq  
no-route: true  
services:  
- sidekiq-redis:  
- sidekiq-mysql:
```

Since the url “sidekiq.cloudfoundry.com” is probably already taken, change it in `web-manifest.yml` first, then push the application with both manifest files:

```
$ cf push -f web-manifest.yml  
$ cf push -f worker-manifest.yml
```

If the cf CLI asks for a URL for the worker application, select “none”.

Test the Application

Test the application by visiting the new action on the “Thing” controller at the assigned url. In this example, the URL would be <http://sidekiq.cloudfoundry.com/thing/new>.

This will create a new Sidekiq job which will be queued in Redis, then picked up by the worker application. The browser is then redirected to </thing> which will show the collection of “Things”.

Scale Workers

Use the `cf scale` command to change the number of Sidekiq workers.

Example:

```
$ cf scale sidekiq-worker -i 2
```

Use `rails_serve_static_assets` on Rails 4

By default Rails 4 returns a 404 if an asset is not handled via an external proxy such as Nginx. The `rails_serve_static_assets` gem enables your Rails server to deliver static assets directly, instead of returning a 404. You can use this capability to populate an edge cache CDN or serve files directly from your web application. The `rails_serve_static_assets` gem enables this behavior by setting the `config.serve_static_assets` option to `true`, so you do not need to configure it manually.

Additional Ruby Buildpack Information

For information about using and extending the Ruby buildpack in Cloud Foundry, see the [ruby-buildpack GitHub repo](#).

You can find current information about this buildpack on the Ruby buildpack [release page](#) in GitHub.

The buildpack uses a default Ruby version of 2.2.2. To override this value for your app, add a Ruby declaration in the Gemfile. This also applies to using a JRuby interpreter.

Environment Variables

You can access environments variable programmatically. For example, you can obtain VCAP_SERVICES as follows:

```
ENV['VCAP_SERVICES']
```

Environment variables available to you include both those [defined by the DEA](#) and those defined by the Ruby buildpack, as described below.

BUNDLE_BIN_PATH

Location where Bundler installs binaries.

```
BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle
```

BUNDLE_GEMFILE

Path to application's Gemfile.

```
BUNDLE_GEMFILE:/home/vcap/app/Gemfile
```

BUNDLE_WITHOUT

The BUNDLE_WITHOUT environment variable causes Cloud Foundry to skip installation of gems in excluded groups.

BUNDLE_WITHOUT is particularly useful for Rails applications, where there are typically “assets” and “development” gem groups containing gems that are not needed when the app runs in production

For information about using this variable, see <http://blog.cloudfoundry.com/2012/10/02/polishing-cloud-foundrys-ruby-gem-support>.

```
BUNDLE_WITHOUT=assets
```

DATABASE_URL

The Ruby buildpack looks at the database_uri for bound services to see if they match known database types. If there are known relational database services bound to the application, the buildpack sets up the DATABASE_URL environment variable with the first one in the list.

If your application depends on DATABASE_URL being set to the connection string for your service, and Cloud Foundry does not set it, you can set this variable manually.

```
$ cf set-env my_app_name DATABASE_URL mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab
```

GEM_HOME

Location where gems are installed.

```
GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1
```

GEM_PATH

Location where gems can be found.

```
GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:
```

RACK_ENV

This variable specifies the Rack deployment environment: development, deployment, or none. This governs what middleware is loaded to run the application.

```
RACK_ENV=production
```

RAILS_ENV

This variable specifies the Rails deployment environment: development, test, or production. This controls which of the environment-specific configuration files will govern how the application will be executed.

```
RAILS_ENV=production
```

RUBYOPT

This Ruby environment variable defines command-line options passed to Ruby interpreter.

```
RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup
```

Environment Variables Defined by the Ruby Buildpack

When you use the Ruby buildpack, you get three Ruby-specific environment variables in addition to the regular [Cloud Foundry environment variables](#).

- `BUNDLE_BIN_PATH` — Location where Bundler installs binaries.
`BUNDLE_BIN_PATH:/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/bin/bundle`
- `BUNDLE_GEMFILE` — Path to application's gemfile.
`BUNDLE_GEMFILE:/home/vcap/app/Gemfile`
- `BUNDLE_WITHOUT` — This variable causes Cloud Foundry to skip installation of gems in excluded groups. Useful for Rails applications, where "assets" and "development" gem groups typically contain gems that are not needed when the app runs in production.
See [this](#) blog post for more information.
`BUNDLE_WITHOUT=assets`
- `DATABASE_URL` — The Ruby buildpack looks at the `database_uri` for bound services to see if they match known database types. If there are known relational database services bound to the application, the buildpack sets up the `DATABASE_URL` environment variable with the first one in the list.
If your application depends on `DATABASE_URL` being set to the connection string for your service, and Cloud Foundry does not set it, you can set this variable manually.
`$ cf set-env my_app_name DATABASE_URL mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab`
- `GEM_HOME` — Location where gems are installed.
`GEM_HOME:/home/vcap/app/vendor/bundle/ruby/1.9.1`
- `GEM_PATH` — Location where gems can be found.
`GEM_PATH=/home/vcap/app/vendor/bundle/ruby/1.9.1:`
- `RACK_ENV` — This variable specifies the Rack deployment environment: development, deployment, or none. This governs what middleware is loaded to run the application.
`RACK_ENV=production`
- `RAILS_ENV` — This variable specifies the Rails deployment environment: development, test, or production. This controls which of the environment-specific configuration files will govern how the application will be executed.
`RAILS_ENV=production`
- `RUBYOPT` — This Ruby environment variable defines command-line options passed to Ruby interpreter.
`RUBYOPT: -I/home/vcap/app/vendor/bundle/ruby/1.9.1/gems/bundler-1.3.2/lib -rbundler/setup`

Configure Service Connections for Ruby

This page assumes that you are using cf CLI v6.

After you create a service instance and bind it to an application, you must configure the application to connect to the service.

Query VCAP_SERVICES with cf-app-utils

The `cf-app-utils` gem allows your application to search for credentials from the `VCAP_SERVICES` environment variable by name, tag, or label.

- [cf-app-utils-ruby ↗](#)

VCAP_SERVICES defines DATABASE_URL

At runtime, the Ruby buildpack creates a `DATABASE_URL` environment variable for every Ruby application based on the `VCAP_SERVICES` environment variable.

Example `VCAP_SERVICES`:

```
VCAP_SERVICES =
{
  "elephantsql": [
    {
      "name": "elephantsql-c6c60",
      "label": "elephantsql",
      "credentials": {
        "uri": "postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampled"
      }
    }
  ]
}
```

Based on this `VCAP_SERVICES`, the Ruby buildpack creates the following `DATABASE_URL` environment variable:

```
DATABASE_URL = postgres://exampleuser:examplepass@babar.elephantsql.com:5432/exampled
```

The Ruby buildpack uses the structure of the `VCAP_SERVICES` environment variable to populate `DATABASE_URL`. Any service containing a JSON object with the following form will be recognized by Cloud Foundry as a candidate for `DATABASE_URL`:

```
{
  "some-service": [
    {
      "credentials": {
        "uri": "<some database URL>"
      }
    }
  ]
}
```

Cloud Foundry uses the first candidate found to populate `DATABASE_URL`.

Older Rails Applications Have Auto-Configured database.yml

On Rails versions 4 or before, the Ruby buildpack replaces your `database.yml` with one based on the `DATABASE_URL` variable.

 **Note:** On Rails versions 4 or before, Ruby buildpack ignores the contents of any `database.yml` that you provide and overwrites it during staging.

Configuring Non-Rails Applications

Non-Rails applications can also access the `DATABASE_URL` variable.

If you have more than one service with credentials, only the first will be populated into `DATABASE_URL`. To access other credentials, you can inspect the `VCAP_SERVICES` environment variable.

```
vcap_services = JSON.parse(ENV['VCAP_SERVICES'])
```

Use the hash key for the service to obtain the connection credentials from `VCAP_SERVICES`.

- For services that use the [v2 format](#), the hash key is the name of the service.
- For services that use the [v1 format](#), the hash key is formed by combining the service provider and version, in the format PROVIDER-VERSION.

For example, the service provider “p-mysql” with version “n/a” forms the hash key `p-mysql-n/a`.

Seed or Migrate Database

Before you can use your database the first time, you must create and populate or migrate it. For more information, see [Migrating a Database in Cloud Foundry](#).

Troubleshooting

To aid in troubleshooting issues connecting to your service, you can examine the environment variables and log messages Cloud Foundry records for your application.

View Environment Variables

Use the `cf env` command to view the Cloud Foundry environment variables for your application. `cf env` displays the following environment variables:

- The `VCAP_SERVICES` variables existing in the container environment
- The user-provided variables set using the `cf set-env` command

```
$ cf env my-app
Getting env variables for app my-app in org My-Org / space development as admin...
OK

System-Provided:
{
  "VCAP_SERVICES": {
    "p-mysql-n/a": [
      {
        "credentials": {
          "uri": "postgres://lrra:e6B-X@p-mysqlprovider.example.com:5432/lraa"
        },
        "label": "p-mysql-n/a",
        "name": "p-mysql",
        "syslog_drain_url": "",
        "tags": ["postgres", "postgresql", "relational"]
      }
    ]
  }
}

User-Provided:
my-env-var: 100
my-drain: http://drain.example.com
```

View Logs

Use the `cf logs` command to view the Cloud Foundry log messages for your application. You can direct current logging to standard output, or you can dump the most recent logs to standard output.

Run `cf logs APPNAME` to direct current logging to standard output:

```
$ cf logs my-app
Connected, tailing logs for app my-app in org My-Org / space development as admin...
1:27:19.72 [App/0] OUT [CONTAINER]  org.apache.coyote.http11.Http11Protocol  INFO Starting ProtocolHandler ["http-bio-6]
1:27:19.77 [App/0] OUT [CONTAINER]  org.apache.catalina.startup.Catalina      INFO Server startup in 10427 ms
```

Run `cf logs APPNAME --recent` to dump the most recent logs to standard output:

```
$ cf logs my-app --recent
Connected, dumping recent logs for app my-app in org My-Org / space development as admin...
1:27:15.93 [App/0] OUT 15,935  INFO EmbeddedDatabaseFactory:124 - Creating embedded database 'SkyNet'
1:27:16.31 [App/0] OUT 16,318  INFO LocalEntityManagerFactory:287 - Building TM container EntityManagerFactory for unit
1:27:16.50 [App/0] OUT 16,505  INFO Version:37 - HCANN001: Hibernate Commons Annotations {4.0.1.Final}
1:27:16.51 [App/0] OUT 16,517  INFO Version:41 - HHH412: Hibernate Core {4.1.9.Final}
1:27:16.95 [App/0] OUT 16,957  INFO SkyNet-Online:73 - HHH268: Transaction strategy: org.hibernate.internal.TransactionF
1:27:16.96 [App/0] OUT 16,963  INFO IninitiateTerminatorT1000Deployment:48 - HHH000397: Using TranslatorFactory
1:27:17.02 [App/0] OUT 17,026  INFO Version:27 - HV001: Hibernate Validator 4.3.0.Final
```

If you encounter the error, “A fatal error has occurred. Please see the Bundler troubleshooting documentation,” update your version of bundler and run `bundle install`.

```
$ gem update bundler
$ gem update --system
$ bundle install
```

Services

The documentation in this section is intended for developers and operators interested in creating Managed Services for Cloud Foundry. Managed Services are defined as having been integrated with Cloud Foundry via APIs, and enable end users to provision reserved resources and credentials on demand. For documentation targeted at end users, such as how to provision services and integrate them with applications, see [Services Overview](#).

To develop Managed Services for Cloud Foundry, you'll need a Cloud Foundry instance to test your service broker with as you are developing it. You must have admin access to your CF instance to manage service brokers and the services marketplace catalog. For local development, we recommend using [BOSH Lite](#) to deploy your own local instance of Cloud Foundry.

Table of Contents

- [Overview](#)
- [Service Broker API](#)
- [Managing Service Brokers](#)
- [Access Control](#)
- [Catalog Metadata](#)
- [Binding Credentials](#)
- [Dashboard Single Sign-On](#)
- [Example Service Brokers](#)
- [Application Log Streaming](#)
- [Supporting Multiple Cloud Foundry Instances](#)

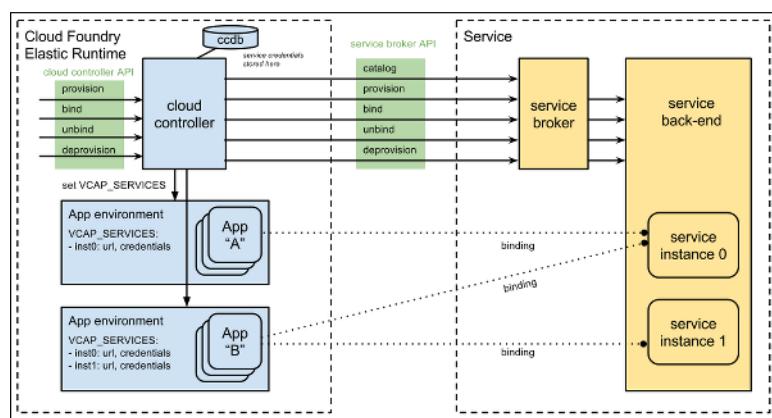
Overview

Architecture & Terminology

Services are integrated with Cloud Foundry by implementing a documented API for which the cloud controller is the client; we call this the Service Broker API. This should not be confused with the cloud controller API, often used to refer to the version of Cloud Foundry itself; when one refers to “Cloud Foundry v2” they are referring to the version of the cloud controller API. The services API is versioned independently of the cloud controller API.

Service Broker is the term we use to refer to a component of the service which implements the service broker API. This component was formerly referred to as a Service Gateway, however as traffic between applications and services does not flow through the broker we found the term gateway caused confusion. Although gateway still appears in old code, we use the term broker in conversation, in new code, and in documentation.

Service brokers advertise a catalog of service offerings and service plans, as well as interpreting calls for provision (create), bind, unbind, and deprovision (delete). What a broker does with each call can vary between services; in general, ‘provision’ reserves resources on a service and ‘bind’ delivers information to an application necessary for accessing the resource. We call the reserved resource a Service Instance. What a service instance represents can vary by service; it could be a single database on a multi-tenant server, a dedicated cluster, or even just an account on a web application.



Implementation & Deployment

How a service is implemented is up to the service provider/developer. Cloud Foundry only requires that the service provider implement the service broker API. A broker can be implemented as a separate application, or by adding the required http endpoints to an existing service.

Because Cloud Foundry only requires that a service implements the broker API in order to be available to Cloud Foundry end users, many deployment models are possible. The following are examples of valid deployment models.

- Entire service packaged and deployed by BOSH alongside Cloud Foundry
- Broker packaged and deployed by BOSH alongside Cloud Foundry, rest of the service deployed and maintained by other means
- Broker (and optionally service) pushed as an application to Cloud Foundry user space
- Entire service, including broker, deployed and maintained outside of Cloud Foundry by other means

Service Broker API v2.7

Document Changelog

[v2 API Change Log](#)

Versions

Two major versions of the Service Broker API are currently supported by Cloud Foundry, v1 and v2. As v1 is deprecated and support will be removed in the next major version of Cloud Foundry, it is recommended that all new brokers implement v2 and that current brokers are upgraded.

Current Version	Past Versions
2.7	2.6 2.5 2.4 2.3 2.2 2.1 2.0 v1 (unversioned)

Changes

Change Policy

- Existing endpoints and fields will not be removed or renamed.
- New optional endpoints, or new HTTP methods for existing endpoints, may be added to enable support for new features.
- New fields may be added to existing request/response messages. These fields must be optional and should be ignored by clients and servers that do not understand them.

Changes Since v2.6

1. [Asynchronous operations](#) for provisioning, updating, and deprovisioning are no longer experimental.
2. Added support for querying `last_operation` status at new endpoint `GET /v2/service_instances/:guid/last_operation`

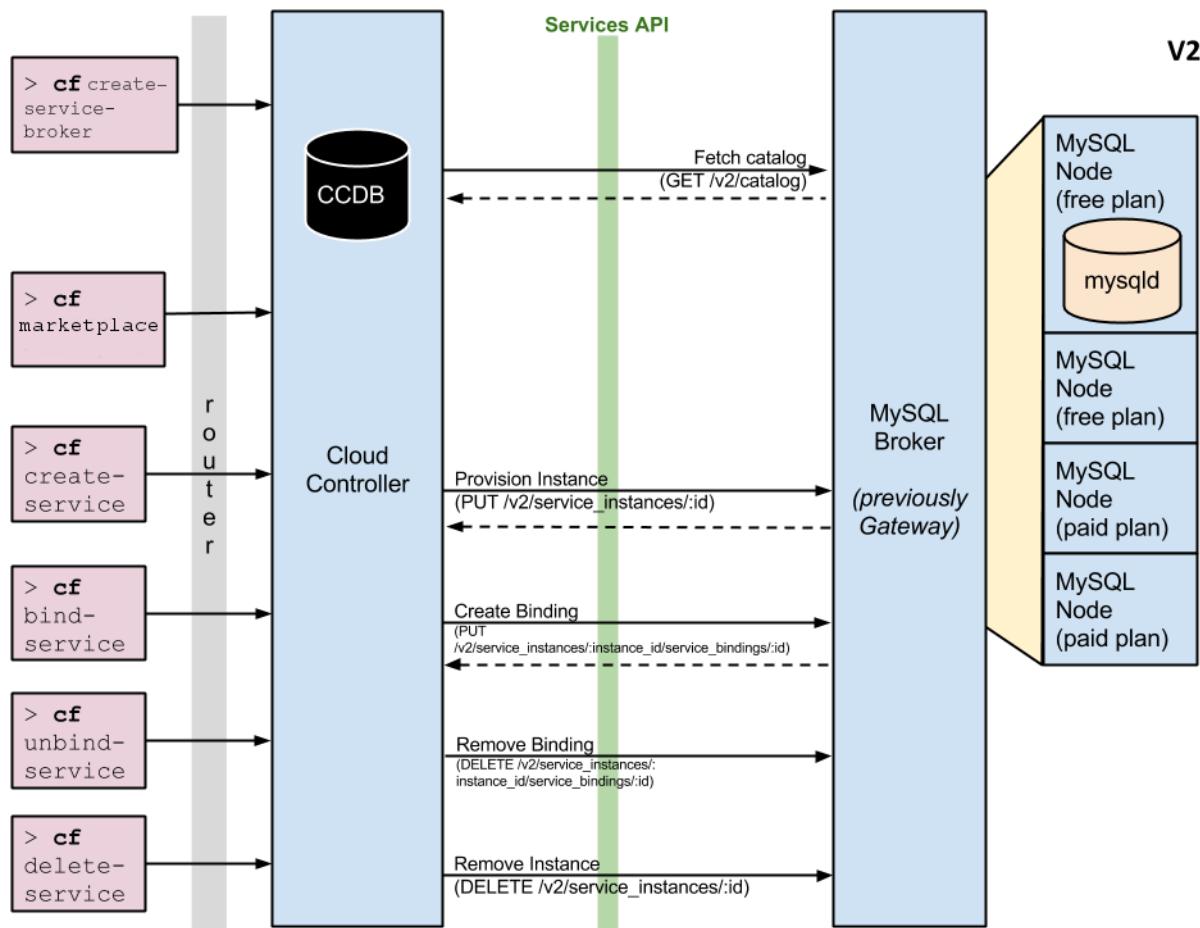
Dependencies

v2.7 of the services API has been supported since:

- [Final build 218](#) of [cf-release](#)
- v2.37.0 of the Cloud Controller API
- CLI [v6.12.1](#)

API Overview

The Cloud Foundry services API defines the contract between the Cloud Controller and the service broker. The broker is expected to implement several HTTP (or HTTPS) endpoints underneath a URI prefix. One or more services can be provided by a single broker, and load balancing enables horizontal scalability of redundant brokers. Multiple Cloud Foundry instances can be supported by a single broker using different URL prefixes and credentials.



API Version Header

Requests from the Cloud Controller to the broker contain a header that defines the version number of the Broker API that Cloud Controller will use. This header will be useful in future minor revisions of the API to allow brokers to reject requests from Cloud Controllers that they do not understand. While minor API revisions will always be additive, it is possible that brokers will come to depend on a feature that was added after 2.0, so they may use this header to reject the request. Error messages from the broker in this situation should inform the operator of what the required and actual version numbers are so that an operator can go upgrade Cloud Controller and resolve the issue. A broker should respond with a `412 Precondition Failed` message when rejecting a request.

The version numbers are in the format `MAJOR.MINOR`, using semantic versioning such that 2.9 comes before 2.10. An example of this header as of publication time is:

```
X-Broker-Api-Version: 2.7
```

Authentication

Cloud Controller (final release v145+) authenticates with the Broker using HTTP basic authentication (the `Authorization:` header) on every request and will reject any broker registrations that do not contain a username and password. The broker is responsible for checking the username and password and returning a `401 Unauthorized` message if credentials are invalid. Cloud Controller supports connecting to a broker using SSL if additional security is desired.

Catalog Management

The first endpoint that a broker must implement is the service catalog. Cloud Controller will initially fetch this endpoint from all brokers and make adjustments to the user-facing service catalog stored in the Cloud Controller database. If the catalog fails to initially load or validate, Cloud Controller will not allow the operator to add the new broker and will give a meaningful error message. Cloud Controller will also update the catalog whenever a broker is updated, so you can use `update-service-broker` with no changes to force a catalog refresh.

When Cloud Controller fetches a catalog from a broker, it will compare the broker's id for services and plans with the `unique_id` values for services and plan in the Cloud Controller database. If a service or plan in the broker catalog has an id that is not present amongst the `unique_id` values in the database, a new record will be added to the database. If services or plans in the database are found with `unique_id`'s that match the broker catalog's id, Cloud Controller will update the records to match the broker's catalog.

If the database has plans which are not found in the broker catalog, and there are no associated service instances, Cloud Controller will remove these plans from the database. Cloud Controller will then delete services that do not have associated plans from the database. If the database has plans which are not found in the broker catalog, and there **are** provisioned instances, the plan will be marked "inactive" and will no longer be visible in the marketplace catalog or be provisionable.

Request

Route

```
GET /v2/catalog
```

cURL

```
$ curl -H "X-Broker-API-Version: 2.7" http://username:password@broker-url/v2/catalog
```

Response

Status Code	Description
200 OK	The expected response body is below

Body

CLI and web clients have different needs with regard to service and plan names. A CLI-friendly string is all lowercase, with no spaces. Keep it short – imagine a user having to type it as an argument for a longer command. A web-friendly display name is camel-cased with spaces and punctuation supported.

Response field	Type	Description
services*	array-of-objects	Schema of service objects defined below:
id*	string	An identifier used to correlate this service in future requests to the catalog. This must be unique within Cloud Foundry, using a GUID is recommended.
name*	string	The CLI-friendly name of the service that will appear in the catalog. All lowercase, no spaces.
description*	string	A short description of the service that will appear in the catalog.
bindable*	boolean	Whether the service can be bound to applications.
tags	array-of-strings	Tags provide a flexible mechanism to expose a classification, attribute, or base technology of a service, enabling equivalent services to be swapped out without changes to dependent logic in applications, buildpacks, or other services. E.g. mysql, relational, redis, key-value, caching, messaging, amqp.
metadata	object	A list of metadata for a service offering. For more information, see Service Metadata .
requires	array-of-strings	A list of permissions that the user would have to give the service, if they provision it. The only permission currently supported is syslog_drain; for more info see Application Log Streaming .
plan_updateable	boolean	Whether the service supports upgrade/downgrade for some plans. Please note that the misspelling of the attribute <code>plan_updatable</code> to <code>plan_updateable</code> was done by mistake. We have opted to keep that misspelling instead of fixing it and thus breaking backward compatibility.
plans*	array-of-objects	A list of plans for this service, schema defined below:

Response field	Type	Description
id*	string	An identifier used to correlate this plan in future requests to the catalog. This must be unique within Cloud Foundry, using a GUID is recommended.
name*	string	The CLI-friendly name of the plan that will appear in the catalog. All lowercase, no spaces.
description*	string	A short description of the service that will appear in the catalog.
metadata	object	A list of metadata for a service plan. For more information, see Service Metadata .
free	boolean	This field allows the plan to be limited by the non_basic_services_allowed field in a Cloud Foundry Quota, see Quota Plans . Default: true
dashboard_client	object	Contains the data necessary to activate the Dashboard SSO feature for this service
id	string	The id of the OAuth2 client that the service intends to use. The name may be taken, in which case the API will return an error to the operator
secret	string	A secret for the dashboard client
redirect_uri	string	A domain for the service dashboard that will be whitelisted by the UAA to enable SSO

* Fields with an asterisk are required.

```
{
  "services": [
    {
      "id": "service-guid-here",
      "name": "mysql",
      "description": "A MySQL-compatible relational database",
      "bindable": true,
      "plans": [
        {
          "id": "plan1-guid-here",
          "name": "small",
          "description": "A small shared database with 100mb storage quota and 10 connections"
        },
        {
          "id": "plan2-guid-here",
          "name": "large",
          "description": "A large dedicated database with 10GB storage quota, 512MB of RAM, and 100 connections",
          "free": false
        }
      ],
      "dashboard_client": {
        "id": "client-id-1",
        "secret": "secret-1",
        "redirect_uri": "https://dashboard.service.com"
      }
    }
  ]
}
```

Adding a Broker to Cloud Foundry

Once you've implemented the first endpoint `GET /v2/catalog` above, you'll want to [register the broker with CF](#) to make your services and plans available to end users.

Asynchronous Operations

Previously, Cloud Foundry only supported synchronous integration with service brokers. Brokers must return a valid response within 60 seconds and if the response is `201 CREATED`, users expect a service instance to be usable. This limits the services brokers can offer to those that can be provisioned in 60 seconds; brokers could return a success prematurely, but this leaves users wondering why their service instance is not usable and when it will be.

With support for Asynchronous Operations, brokers still must respond within 60 seconds but may now return a `202 ACCEPTED`, indicating that the requested operation has been accepted but is not complete. This triggers Cloud Foundry to poll a new endpoint `/v2/service_instances/:guid/last_operation` until the broker indicates that the requested operation has succeeded or failed. During the intervening time, end users are able to discover the state of the requested operation using Cloud Foundry API clients such as the CLI.

For an operation to be executed asynchronously, all three components (CF API client, CF, and broker) must support the feature. The parameter `accepts_incomplete=true` must be passed in a request by the CF API client, triggering CF to include the same parameter in a request to the broker. The broker can then choose to execute the request synchronously or asynchronously.

If the broker executes the request asynchronously, the response must use the status code `202 ACCEPTED`; the response body should be the same as if the broker were serving the request synchronously.

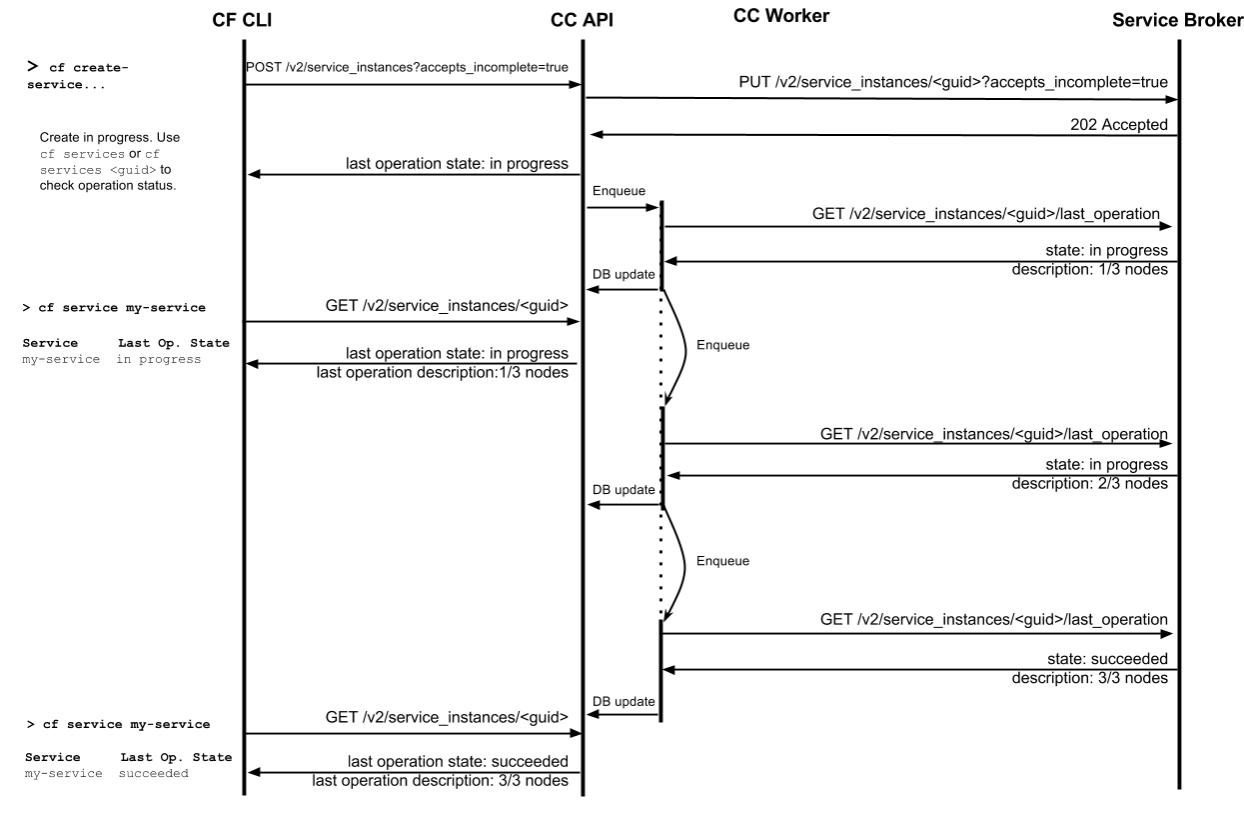
Note: Asynchronous Operations are currently supported only for provision, update, and deprovision. Bind and unbind will be added once the feature is considered stable.

If the `accepts_incomplete=true` parameter is not included, and the broker can not fulfill the request synchronously (guaranteeing that the operation is complete on response), then the broker should reject the request with the status code `422 UNPROCESSABLE ENTITY` and the following body:

```
{
  "error": "AsyncRequired",
  "description": "This service plan requires client support for asynchronous service operations."
}
```

To execute a request synchronously, the broker need only return the usual status codes; `201 CREATED` for create, and `200 OK` for update and delete.

Sequence Diagram



Blocking Operations

The Cloud Controller ensures that service brokers do not receive requests for an instance while an asynchronous operation is in progress. For example, if a broker is in the process of provisioning an instance asynchronously, the Cloud Controller will not allow any update, bind, unbind, or deprovision requests to be made through the platform. A user who attempts to perform one of these actions while an operation is already in progress will get an HTTP 400 with error message “Another operation for this service instance is in progress.”

When to use Asynchronous Service Operations

Service brokers should respond to all Cloud Controller requests within 60 seconds. Brokers that can guarantee completion of the requested operation with the response may return the synchronous response (e.g. `201 CREATED` for a provision request). Brokers that can not guarantee completion of the operation with the response should implement support for asynchronous provisioning. Support for asynchronous or synchronous responses may vary by service.

offering, even by service plan.

Polling Last Operation (async only)

When a broker returns status code `202 ACCEPTED` for [provision](#), [update](#), or [deprovision](#), Cloud Foundry will begin to poll the `/v2/service_instances/:guid/last_operation` endpoint to obtain the state of the last requested operation. The broker response must contain the field `state` and an optional field `description`.

Valid values for `state` are `in progress`, `succeeded`, and `failed`. Cloud Foundry will poll the `last_operation` endpoint as long as the broker returns `"state": "in progress"`. Returning `"state": "succeeded"` or `"state": "failed"` will cause Cloud Foundry to cease polling. The value provided for `description` will be passed through to the CF API client and can be used to provide additional detail for users about the state of the operation.

Request

Route

```
GET /v2/service_instances/:instance_id/last_operation
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/last_operation
```

Response

Status Code	Description
200 OK	The expected response body is below.
410 GONE	Appropriate only for asynchronous delete requests. Cloud Foundry will consider this response a success and remove the resource from its database. The expected response body is <code>{}</code> . Returning this while Cloud Foundry is polling for create or update operations will be interpreted as an invalid response and Cloud Foundry will continue polling.

Responses with any other status code will be interpreted as an error or invalid response; Cloud Foundry will continue polling until the broker returns a valid response or the [maximum polling duration](#) is reached. Brokers may use the `description` field to expose user-facing error messages about the operation state; for more info see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are valid.

Response field	Type	Description
state*	string	Valid values are <code>in progress</code> , <code>succeeded</code> , and <code>failed</code> . While <code>"state": "in progress"</code> , Cloud Foundry will continue polling. A response with <code>"state": "succeeded"</code> or <code>"state": "failed"</code> will cause Cloud Foundry to cease polling.
description	string	Optional field. A user-facing message displayed to the Cloud Foundry API client. Can be used to tell the user details about the status of the operation.

```
{
  "state": "in progress",
  "description": "Creating service (10% complete)."
}
```

Polling Interval

When a broker responds asynchronously to a request from Cloud Foundry containing the `accepts_incomplete=true`

parameter, Cloud Foundry will poll the broker for the operation state at a configured interval. The Cloud Foundry operator can configure this interval in the BOSH deployment manifest using the property `properties.cc.broker_client_default_async_poll_interval_seconds` (defaults to 60 seconds). The maximum supported polling interval is 86400 seconds (24 hours).

Maximum Polling Duration

When a broker responds asynchronously to a request from Cloud Foundry containing the `accepts_incomplete=true` parameter, Cloud Foundry will poll the broker for the operation state until the broker response includes `"state": "succeeded"` or `"state": "failed"`, or until a maximum polling duration is reached. If the max polling duration is reached, Cloud Foundry will cease polling and the operation state will be considered `failed`. The Cloud Foundry operator can configure this max polling duration in the BOSH deployment manifest using the property `properties.cc.broker_client_max_async_poll_duration_minutes` (defaults to 10080 minutes or 1 week).

Additional Resources

- An example broker that implements this feature can be found at [Example Service Brokers](#).
- A demo video of the CLI user experience using the above broker can be found [here ↗](#).

Provisioning

When the broker receives a provision request from Cloud Controller, it should synchronously take whatever action is necessary to create a new service resource for the developer. The result of provisioning varies by service type, although there are a few common actions that work for many services. For a MySQL service, provisioning could result in:

- An empty dedicated `mysqld` process running on its own VM.
- An empty dedicated `mysqld` process running in a lightweight container on a shared VM.
- An empty dedicated `mysqld` process running on a shared VM.
- An empty dedicated database, on an existing shared running `mysqld`.
- A database with business schema already there.
- A copy of a full database, for example a QA database that is a copy of the production database.

For non-data services, provisioning could just mean getting an account on an existing system.

Request

Route

```
PUT /v2/service_instances/:instance_id
```

Note: The `:instance_id` of a service instance is provided by the Cloud Controller. This ID will be used for future requests (bind and deprovision), so the broker must use it to correlate the resource it creates.

Body

Request field	Type	Description
organization_guid*	string	The Cloud Controller GUID of the organization under which the service is to be provisioned. Although most brokers will not use this field, it could be helpful in determining data placement or applying custom business rules.
plan_id*	string	The ID of the plan within the above service (from the catalog endpoint) that the user would like provisioned. Because plans have identifiers unique to a broker, this is enough information to determine what to provision.
service_id*	string	The ID of the service within the catalog above. While not strictly necessary, some brokers might make use of this ID.
space_guid*	string	Similar to organization_guid, but for the space.

Request field	Type	Description
parameters	JSON object	Cloud Foundry API clients can provide a JSON object of configuration parameters with their request and this value will be passed through to the service broker. Brokers are responsible for validation.
accepts_incomplete	boolean	A value of true indicates that both the Cloud Controller and the requesting client support asynchronous provisioning. If this parameter is not included in the request, and the broker can only provision an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.

```
{
  "organization_guid": "org-guid-here",
  "plan_id": "plan-guid-here",
  "service_id": "service-guid-here",
  "space_guid": "space-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "value"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id -d '{ "organization_guid": "org-guid-here", "plan_id": "plan-guid-here", "service_id": "service-guid-here", "space_guid": "space-guid-here", "parameters": { "parameter1": 1, "parameter2": "value" } }' -X PUT -H "X-Broker-API-Version: 2.7" -H "Content-Type: application/json"
```

In this case, `:instance_id` refers to the service instance id generated by Cloud Controller

Response

Status Code	Description
201 Created	Service instance has been created. The expected response body is below.
200 OK	May be returned if the service instance already exists and the requested parameters are identical to the existing service instance. The expected response body is below.
202 Accepted	Service instance creation is in progress. This triggers Cloud Controller to poll the Service Instance Last Operation Endpoint for operation status.
409 Conflict	Should be returned if the requested service instance already exists. The expected response body is " <code>{}</code> "
422 Unprocessable Entity	Should be returned if the broker only supports asynchronous provisioning for the requested plan and the request did not include <code>?accepts_incomplete=true</code> . The expected response body is: <code>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</code> as described below.

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see [Broker Errors](#).

Response field	Type	Description
----------------	------	-------------

Response field	Type	Description
dashboard_url	string	The URL of a web-based management user interface for the service instance; we refer to this as a service dashboard. The URL should contain enough information for the dashboard to identify the resource being accessed ("9189kdfsk0vfnku" in the example below). For information on how users can authenticate with service dashboards via SSO, see Dashboard Single Sign-On .

```
{
  "dashboard_url": "http://example-dashboard.com/9189kdfsk0vfnku"
}
```

Updating a Service Instance

Brokers that implement this endpoint can enable users to modify attributes of an existing service instance. The first attribute Cloud Foundry supports users modifying is the service plan. This effectively enables users to upgrade or downgrade their service instance to other plans. To see how users make these requests, see [Managing Services](#).

To enable this functionality, a broker declares support for each service by including `plan_updateable: true` in its [catalog endpoint](#). If this optional field is not included, Cloud Foundry will return a meaningful error to users for any plan change request, and will not make an API call to the broker. If this field is included and configured as true, Cloud Foundry will make API calls to the broker for all plan change requests, and it is up to the broker to validate whether a particular permutation of plan change is supported. Not all permutations of plan changes are expected to be supported. For example, a service may support upgrading from plan "shared small" to "shared large" but not to plan "dedicated". If a particular plan change is not supported, the broker should return a meaningful error message in response.

Request

Route

```
PATCH /v2/service_instances/:instance_id
```

Note: `:instance_id` is the global unique ID of a previously-provisioned service instance.

Body

Request Field	Type	Description
service_id*	string	The ID of the service within the catalog above. While not strictly necessary, some brokers might make use of this ID.
plan_id	string	ID of the new plan from the catalog.
parameters	JSON object	Cloud Foundry API clients can provide a JSON object of configuration parameters with their request and this value will be passed through to the service broker. Brokers are responsible for validation.
previous_values	object	Information about the instance prior to the update.
previous_values.plan_id	string	ID of the plan prior to the update.
previous_values.service_id	string	ID of the service for the instance.
previous_values.organization_id	string	ID of the organization containing the instance.
previous_values.space_id	string	ID of the space containing the instance..
accepts_incomplete	boolean	A value of true indicates that both the Cloud Controller and the requesting client support asynchronous update. If this parameter is not included in the request, and the broker can only update an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.

```
{
  "service_id": "service-guid-here",
  "plan_id": "plan-guid-here",
  "parameters": {
    "parameter1": 1,
    "parameter2": "value"
  },
  "previous_values": {
    "plan_id": "old-plan-guid-here",
    "service_id": "service-guid-here",
    "organization_id": "org-guid-here",
    "space_id": "space-guid-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id -d '{\n  "service_id": "service-guid-here",\n  "plan_id": "plan-guid-here",\n  "parameters": {\n    "parameter1": 1,\n    "parameter2": "value"\n  },\n  "previous_values": {\n    "plan_id": "old-plan-guid-here",\n    "service_id": "service-guid-here",\n    "organization_id": "org-guid-here",\n    "space_id": "space-guid-here"\n  }\n}' -X PATCH -H "X-Broker-API-Version: 2.7" -H "Content-Type: application/json"
```

Response

Status Code	Description
200 OK	New plan is effective. The expected response body is `{}`.
202 Accepted	Service instance update is in progress. This triggers Cloud Controller to poll the Service Instance Last Operation Endpoint for operation status.
422 Unprocessable entity	May be returned if the particular plan change requested is not supported or if the request can not currently be fulfilled due to the state of the instance (eg. instance utilization is over the quota of the requested plan). Broker should include a user-facing message in the body; for details see Broker Errors . Additionally, a 422 can also be returned if the broker only supports asynchronous update for the requested plan and the request did not include <code>?accepts_incomplete=true</code> . The expected response body is: <pre>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</pre>

Responses with any other status code will be interpreted as a failure. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For a success response, the expected response body is `{}`.

Binding

 **Note:** Not all services must be bindable — some deliver value just from being provisioned. Brokers that offer services that are not bindable should declare them as such using `bindable: false` in the [Catalog](#catalog-mgmt). Brokers that do not offer any bindable services do not need to implement the endpoint for bind requests.

When the broker receives a bind request from the Cloud Controller, it should return information which helps an application to utilize the provisioned resource. This information is generically referred to as `credentials`. Applications

should be issued unique credentials whenever possible, so one application's access can be revoked without affecting other bound applications. For more information on credentials, see [Binding Credentials](#).

In addition, the bind operation can enable streaming of application logs from Cloud Foundry to a consuming service instance. For details, see [Application Log Streaming](#).

Request

Route

```
PUT /v2/service_instances/:instance_id/service_bindings/:binding_id
```

Note: The `:binding_id` of a service binding is provided by the Cloud Controller. `:instance_id` is the ID of a previously-provisioned service instance; `:binding_id` will be used for future unbind requests, so the broker must use it to correlate the resource it creates.

Body

Request Field	Type	Description
service_id*	string	ID of the service from the catalog. While not strictly necessary, some brokers might make use of this ID.
plan_id*	string	ID of the plan from the catalog. While not strictly necessary, some brokers might make use of this ID.
app_guid	string	GUID of the application that you want to bind your service to. Will be included when users bind applications to service instances.
parameters	JSON object	Cloud Foundry API clients can provide a JSON object of configuration parameters with their request and this value will be passed through to the service broker. Brokers are responsible for validation.

```
{
  "plan_id": "plan-guid-here",
  "service_id": "service-guid-here",
  "app_guid": "app-guid-here",
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}
```

cURL

```
$ curl http://username:password@broker-url/v2/service_instances/:instance_id/service_bindings/:binding_id -d '{
  "plan_id": "plan-guid-here",
  "service_id": "service-guid-here",
  "app_guid": "app-guid-here",
  "parameters": {
    "parameter1-name-here": 1,
    "parameter2-name-here": "parameter2-value-here"
  }
}' -X PUT
```

In this case, `:instance_id` refers to the id of an existing service instance in a previous provisioning, while `:binding_id` is service binding id generated by Cloud Controller.

Response

Status Code	Description
201 Created	Binding has been created. The expected response body is below.
200 OK	May be returned if the binding already exists and the requested parameters are identical to the existing binding. The expected response body is below.

Status Code	Description
409 Conflict	Should be returned if the requested binding already exists. The expected response body is `{}`, though the description field can be used to return a user-facing error message, as described in Broker Errors .
422 Unprocessable Entity	Should be returned if the broker requires that <code>app_guid</code> be included in the request body. The expected response body is: <code>{ "error": "RequiresApp", "description": "This service supports generation of credentials through binding an application only." }</code>

Responses with any other status code will be interpreted as a failure and an unbind request will be sent to the broker to prevent an orphan being created on the broker. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For success responses, the following fields are supported. Others will be ignored. For error responses, see [Broker Errors](#).

Response Field	Type	Description
credentials	object	A free-form hash of credentials that the bound application can use to access the service. For more information, see Binding Credentials .
syslog_drain_url	string	A URL to which Cloud Foundry should drain logs for the bound application. <code>requires:syslog_drain</code> must be declared in the catalog endpoint or Cloud Foundry will consider the response invalid. For details, see Application Log Streaming .

```
{
  "credentials": {
    "uri": "mysql://mysqluser:pass@mysqlhost:3306/dbname",
    "username": "mysqluser",
    "password": "pass",
    "host": "mysqlhost",
    "port": 3306,
    "database": "dbname"
  }
}
```

Unbinding

Note: Brokers that do not provide any bindable services do not need to implement the endpoint for unbind requests.

When a broker receives an unbind request from Cloud Controller, it should delete any resources it created in bind. Usually this means that an application immediately cannot access the resource.

Request

Route

```
DELETE /v2/service_instances/:instance_id/service_bindings/:binding_id
```

The `:binding_id` in the URL is the identifier of a previously created binding (the same `:binding_id` passed in the bind request). The request has no body, because DELETE requests generally do not have bodies.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
--------------------	------	-------------

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog. While not strictly necessary, some brokers might make use of this ID.
plan_id*	string	ID of the plan from the catalog. While not strictly necessary, some brokers might make use of this ID.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id/
  service_bindings/:binding_id?service_id=service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.7"
```

Response

Status Code	Description
200 OK	Binding was deleted. The expected response body is `{}`
410 Gone	Should be returned if the binding does not exist. The expected response body is `{}`

Responses with any other status code will be interpreted as a failure and the binding will remain in the Cloud Controller database. Brokers can include a user-facing message in the `:description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object (`{}`). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For a success response, the expected response body is `{}`.

Deprovisioning

When a broker receives a deprovision request from Cloud Controller, it should delete any resources it created during the provision. Usually this means that all resources are immediately reclaimed for future provisions.

Request

Route

```
DELETE /v2/service_instances/:instance_id
```

The `:instance_id` in the URL is the identifier of a previously provisioned instance (the same `:instance_id` passed in the provision request). The request has no body, because DELETE requests generally do not have bodies.

Parameters

The request provides these query string parameters as useful hints for brokers.

Query-String Field	Type	Description
service_id*	string	ID of the service from the catalog. While not strictly necessary, some brokers might make use of this ID.
plan_id*	string	ID of the plan from the catalog. While not strictly necessary, some brokers might make use of this ID.
accepts_incomplete	boolean	A value of true indicates that both the Cloud Controller and the requesting client support asynchronous deprovisioning. If this parameter is not included in the request, and the broker can only deprovision an instance of the requested plan asynchronously, the broker should reject the request with a 422 as described below.

cURL

```
$ curl 'http://username:password@broker-url/v2/service_instances/:instance_id?service_id=
  service-id-here&plan_id=plan-id-here' -X DELETE -H "X-Broker-API-Version: 2.7"
```

Response

Status Code	Description
200 OK	Service instance was deleted. The expected response body is “{}”
202 Accepted	Service instance deletion is in progress. This triggers Cloud Controller to poll the Service Instance Last Operation Endpoint for operation status.
410 Gone	Should be returned if the service instance does not exist. The expected response body is “{}”
422 Unprocessable Entity	Should be returned if the broker only supports asynchronous deprovisioning for the requested plan and the request did not include <code>?accepts_incomplete=true</code> . The expected response body is: <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <pre>{ "error": "AsyncRequired", "description": "This service plan requires client support for asynchronous service operations." }</pre> </div> , as described below.

Responses with any other status code will be interpreted as a failure and the service instance will remain in the Cloud Controller database. Brokers can include a user-facing message in the `description` field; for details see [Broker Errors](#).

Body

All response bodies must be a valid JSON Object ({}). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For a success response, the expected response body is {}.

Broker Errors

Response

Broker failures beyond the scope of the well-defined HTTP response codes listed above (like 410 on delete) should return an appropriate HTTP response code (chosen to accurately reflect the nature of the failure) and a body containing a valid JSON Object (not an array).

Body

All response bodies must be a valid JSON Object ({}). This is for future compatibility; it will be easier to add fields in the future if JSON is expected rather than to support the cases when a JSON body may or may not be returned.

For error responses, the following fields are valid. Others will be ignored. If an empty JSON object is returned in the body {}, a generic message containing the HTTP response code returned by the broker will be displayed to the requestor.

Response Field	Type	Description
description	string	An error message explaining why the request failed. This message will be displayed to the user who initiated the request.

```
{
  "description": "Something went wrong. Please contact support at http://support.example.com."
}
```

Orphans

The Cloud Controller is the source of truth for service instances and bindings. Service brokers are expected to have successfully provisioned all the instances and bindings Cloud Controller knows about, and none that it doesn't.

Orphans can result if the broker does not return a response before a request from Cloud Controller times out (typically 60 seconds). For example, if a broker does not return a response to a provision request before Cloud Controller times out, the broker might eventually succeed in provisioning an instance after Cloud Controller considers the request a failure. This results in an orphan instance on the service side.

To mitigate orphan instances and bindings, Cloud Controller will attempt to delete resources it cannot be sure were successfully created, and will keep trying to delete them until the broker responds with a success.

More specifically, when a provision or bind request to the broker fails, Cloud Controller will immediately send a corresponding delete or unbind request. If the delete or unbind request fails, Cloud Controller will retry the delete or unbind request ten times with an exponential backoff schedule (over a period of 34 hours).

Status code	Result	Orphan mitigation
200	Success	
200 with malformed response	Failure	
201	Success	
201 with malformed response	Failure	Yes
All other 2xx	Failure	Yes
408	Failure due to timeout	Yes
All other 4xx	Broker rejects request	
5xx	Broker error	Yes
Timeout	Failure	Yes

If the Cloud Controller encounters an internal error provisioning an instance or binding (for example, saving to the database fails), then the Cloud Controller will send a single delete or unbind request to the broker but will not retry.

This orphan mitigation behavior was introduced in cf-release v196.

Managing Service Brokers

This page assumes you are using cf CLI v6.

In order to run any of the commands below, you must be authenticated with Cloud Foundry as an admin user.

Quick Start

Given a service broker that has implemented the [Service Broker API](#), two steps are required to make its services available to end users.

1. [Register a Broker](#)
2. [Make Plans Public](#)

Register a Broker

Registering a broker causes Cloud Controller to fetch and validate the catalog from your broker, and save the catalog to the Cloud Controller database. The basic auth username and password which are provided when adding a broker are encrypted in Cloud Controller database, and used by the Cloud Controller to authenticate with the broker when making all API calls. Your service broker should validate the username and password sent in every request; otherwise, anyone could curl your broker to delete service instances.

```
$ cf create-service-broker mybrokername someuser somethingsecure http://mybroker.example.com/
```

Make Plans Public

New service plans are private by default. To make plans available to end users, see [Make Plans Public](#). Instances of a private plan can not be provisioned until either the plan is made public or is made available to an organization.

Multiple Brokers, Services, Plans

Many service brokers may be added to a Cloud Foundry instance, each offering many services and plans. The following constraints should be kept in mind:

- It is not possible to have multiple brokers with the same name
- It is not possible to have multiple brokers with the same base URL
- The service ID and plan IDs of each service advertised by the broker must be unique across Cloud Foundry. GUIDs are recommended for these fields.

See [Possible Errors](#) below for error messages and what do to when you see them.

List Service Brokers

```
$ cf service-brokers
Getting service brokers as admin...Cloud Controller
OK

Name          URL
my-service-name http://mybroker.example.com
```

Update a Broker

Updating a broker is how to ingest changes a broker author has made into Cloud Foundry. Similar to adding a broker, update causes Cloud Controller to fetch the catalog from a broker, validate it, and update the Cloud Controller database with any changes found in the catalog.

Update also provides a means to change the basic auth credentials cloud controller uses to authenticate with a broker,

as well as the base URL of the broker's API endpoints.

```
$ cf update-service-broker mybrokername someuser somethingsecure http://mybroker.example.com/
```

Rename a Broker

A service broker can be renamed with the `rename-service-broker` command. This name is used only by the Cloud Foundry operator to identify brokers, and has no relation to configuration of the broker itself.

```
$ cf rename-service-broker mybrokername mynewbrokername
```

Remove a Broker

Removing a service broker will remove all services and plans in the broker's catalog from the Cloud Foundry Marketplace.

```
$ cf delete-service-broker mybrokername
```

Note: Attempting to remove a service broker will fail if there are service instances for any service plan in its catalog. When planning to shut down or delete a broker, make sure to remove all service instances first. Failure to do so will leave [orphaned service instances](#) in the Cloud Foundry database. If a service broker has been shut down without first deleting service instances, you can remove the instances with the CLI; see [Purge a Service](#).

Purge a Service

If a service broker has been shut down or removed without first deleting service instances from Cloud Foundry, you will be unable to remove the service broker or its services and plans from the Marketplace. In development environments, broker authors often destroy their broker deployments and need a way to clean up the Cloud Controller database.

The following command will delete a service offering, all of its plans, as well as all associated service instances and bindings from the Cloud Controller database, without making any API calls to a service broker. For services from v1 brokers, you must provide a provider with `-p PROVIDER`. Once all services for a broker have been purged, the broker can be removed normally.

```
$ cf purge-service-offering v1-test -p pivotal-software
Warning: This operation assumes that the service broker responsible for this
service offering is no longer available, and all service instances have been
deleted, leaving orphan records in Cloud Foundry's database. All knowledge of
the service will be removed from Cloud Foundry, including service instances and
service bindings. No attempt will be made to contact the service broker; running
this command without destroying the service broker will cause orphan service
instances. After running this command you may want to run either
delete-service-auth-token or delete-service-broker to complete the cleanup.

Really purge service offering v1-test from Cloud Foundry? y
OK
```

`purge-service-offering` requires cf-release v160 and edge of cf CLI 6.0.2.

Possible Errors

If incorrect basic auth credentials are provided:

```
Server error, status code: 500, error code: 10001, message: Authentication failed for the service broker API.
Double-check that the username and password are correct:
http://github-broker.a1-app.cf-app.com/v2/catalog
```

If you receive the following errors, check your broker logs. You may have an internal error.

```
Server error, status code: 500, error code: 10001, message:  
  The service broker response was not understood  
  
Server error, status code: 500, error code: 10001, message:  
  The service broker API returned an error from  
  http://github-broker.al-app.cf-app.com/v2/catalog: 404 Not Found  
  
Server error, status code: 500, error code: 10001, message:  
  The service broker API returned an error from  
  http://github-broker.primo.cf-app.com/v2/catalog: 500 Internal Server Error
```

If your broker's catalog of services and plans violates validation of presence, uniqueness, and type, you will receive meaningful errors.

```
Server error, status code: 502, error code: 270012, message: Service broker catalog is invalid:  
Service service-name-1  
  service id must be unique  
  service description is required  
  service "bindable" field must be a boolean, but has value "true"  
Plan plan-name-1  
  plan metadata must be a hash, but has value [{"bullets"=>["bullet1", "bullet2"]}]]
```

Access Control

By default, all new service plans are private. This means that when adding a new broker, or when adding a new plan to an existing broker's catalog, new service plans won't immediately be available to end users. This enables an admin to control which service plans are available to end users, and to manage limited availability.

Using the CLI

If your CLI and/or deployment of cf-release do not meet the following prerequisites, you can manage access control with [cf curl](#).

Prerequisites

- CLI v6.4.0
- Cloud Controller API v2.9.0 (cf-release v179)
- Admin user access; the following commands can be run only by an admin user

To determine your API version, curl `/v2/info` and look for `api_version`.

```
$ cf curl /v2/info
{
  "name": "vcap",
  "build": "2222",
  "support": "http://support.cloudfoundry.com",
  "version": 2,
  "description": "Cloud Foundry sponsored by Pivotal",
  "authorization_endpoint": "https://login.system-domain.com",
  "token_endpoint": "https://uaa.system-domain.com",
  "api_version": "2.13.0",
  "logging_endpoint": "wss://loggregator.system-domain.com:443"
}
```

Display Access to Service Plans

The `service-access` CLI command enables an admin to see the current access control setting for every service plan in the marketplace, across all service brokers.

```
$ cf service-access
getting service access as admin...
broker: p-riakcs
  service   plan      access    orgs
  p-riakcs  developer  limited

broker: p-mysql
  service   plan      access    orgs
  p-mysql   100mb-dev  all
```

The `access` column has values `all`, `limited`, or `none`. `all` means a service plan is available to all users of the Cloud Foundry instance; this is what we mean when we say the plan is “public”. `none` means the plan is not available to anyone; this is what we mean when we say the plan is “private”. `limited` means that the service plan is available to users of one or more select organizations. When a plan is `limited`, organizations that have been granted access are listed.

Flags provide filtering by broker, service, and organization.

```
$ cf help service-access
NAME:
  service-access - List service access settings

USAGE:
  cf service-access [-b BROKER] [-e SERVICE] [-o ORG]

OPTIONS:
  -b   access for plans of a particular broker
  -e   access for plans of a particular service offering
  -o   plans accessible by a particular organization
```

Enable Access to Service Plans

Service access is managed at the granularity of service plans, though CLI commands allow an admin to modify all plans of a service at once.

Enabling access to a service plan for organizations allows users of those organizations to see the plan listed in the marketplace (`cf marketplace`), and if users have the Space Developer role in a targeted space, to provision instances of the plan.

```
$ cf enable-service-access p-riakcs
Enabling access to all plans of service p-riakcs for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-riakcs
  service   plan       access   orgs
  p-riakcs developer   all
```

An admin can use `enable-service-access` to:

- Enable access to all plans of a service for users of all orgs (access: `all`)
- Enable access to one plan of a service for users of all orgs (access: `all`)
- Enable access to all plans of a service for users of a specified organization (access: `limited`)
- Enable access to one plan of a service for users of a specified organization (access: `limited`)

```
$ cf help enable-service-access
NAME:
  enable-service-access - Enable access to a service or service plan for one or all orgs

USAGE:
  cf enable-service-access SERVICE [-p PLAN] [-o ORG]

OPTIONS:
  -p   Enable access to a particular service plan
  -o   Enable access to a particular organization
```

Disable Access to Service Plans

```
$ cf disable-service-access p-riakcs
Disabling access to all plans of service p-riakcs for all orgs as admin...
OK

$ cf service-access
getting service access as admin...
broker: p-riakcs
  service   plan       access   orgs
  p-riakcs developer   none
```

An admin can use the `disable-service-access` command to:

- Disable access to all plans of a service for users of all orgs (access: `all`)
- Disable access to one plan of a service for users of all orgs (access: `all`)
- Disable access to all plans of a service for users of select orgs (access: `limited`)
- Disable access to one plan of a service for users of select orgs (access: `limited`)

```
$ cf help disable-service-access
NAME:
  disable-service-access - Disable access to a service or service plan for one or all orgs

USAGE:
  cf disable-service-access SERVICE [-p PLAN] [-o ORG]

OPTIONS:
  -p   Disable access to a particular service plan
  -o   Disable access to a particular organization
```

Limitations

- You cannot disable access to a service plan for an organization if the plan is currently available to all organizations.
You must first disable access for all organizations; then you can enable access for a particular organization.

Using cf curl

The following commands must be run as a system admin user.

Enable Access to Service Plans

Access can be enabled for users of all organizations, or for users of particular organizations. Service plans which are available to all users are said to be “public”. Plans that are available to no organizations, or to particular organizations, are said to be “private”.

Enable access to a plan for all organizations

Once made public, the service plan can be seen by all users in the list of available services. See [Managing Services](#) for more information.

To make a service plan public, you need the service plan GUID. To find the service plan GUID, run:

```
cf curl /v2/service_plans -X 'GET'
```

This command returns a filtered JSON response listing every service plan. Data about each plan shows in two sections: `metadata` and `entity`. The `metadata` section shows the service plan GUID, while the `entity` section lists the name of the plan. Note: Because `metadata` is listed before `entity` for each service plan, the GUID of a plan is shown six lines above the name.

Example:

```
$ cf curl /v2/service_plans
...
{
  "metadata": {
    "guid": "1afd5050-664e-4be2-9389-6bf0c967c0c6",
    "url": "/v2/service_plans/1afd5050-664e-4be2-9389-6bf0c967c0c6",
    "created_at": "2014-02-12T06:24:04+00:00",
    "updated_at": "2014-02-12T18:46:52+00:00"
  },
  "entity": {
    "name": "plan-name-1",
    "free": true,
    "description": "plan-desc-1",
    "service_guid": "d9011411-1463-477c-b223-82e04996b91f",
    "extra": "{\"bullets\":[\"bullet1\",\"bullet2\"]}",
    "unique_id": "plan-id-1",
    "public": false,
    "service_url": "/v2/services/d9011411-1463-477c-b223-82e04996b91f",
    "service_instances_url": "/v2/service_plans/1afd5050-664e-4be2-9389-6bf0c967c0c6/service_instances"
  }
}
```

In this example, the GUID of plan-name-1 is 1afd5050-664e-4be2-9389-6bf0c967c0c6.

To make a service plan public, run: `cf curl /v2/service_plans/SERVICE_PLAN_GUID -X 'PUT' -d '{"public":true}'`

As verification, the “entity” section of the JSON response shows the `"public":true` key-value pair.

```
$ cf curl /v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111 -X 'PUT' -d '{"public":true}'

{
  "metadata": {
    "guid": "1113aa0-124e-4af2-1526-6bfacf61b111",
    "url": "/v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111",
    "created_at": "2014-02-12T06:24:04+00:00",
    "updated_at": "2014-02-12T20:55:10+00:00"
  },
  "entity": {
    "name": "plan-name-1",
    "free": true,
    "description": "plan-desc-1",
    "service_guid": "d9011411-1463-477c-b223-82e04996b91f",
    "extra": "{\"bullets\": [\"bullet1\", \"bullet2\"]}",
    "unique_id": "plan-id-1",
    "public": true,
    "service_url": "/v2/services/d9011411-1463-477c-b223-82e04996b91f",
    "service_instances_url": "/v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111/service_instances"
  }
}
```

Enable access to a private plan for a particular organization

Users have access to private plans that have been enabled for an organization only when targeting a space of that organization. See [Managing Services](#) for more information.

To make a service plan available to users of a specific organization, you need the GUID of both the organization and the service plan. To get the GUID of the service plan, run the same command described above for [enabling access to a plan for all organizations](#):

```
cf curl -X 'GET' /v2/service_plans
```

To find the organization GUIDs, run:

```
cf curl /v2/organizations?q=name:YOUR-ORG-NAME
```

The `metadata` section shows the organization GUID, while the `entity` section lists the name of the organization. Note: Because `metadata` is listed before `entity` for each organization, the GUID of an organization is shown six lines above the name.

Example:

```
$ cf curl /v2/organizations?q=name:my-org

{
  "metadata": {
    "guid": "c54bf317-d791-4d12-89f0-b56d0936cfcdc",
    "url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc",
    "created_at": "2013-05-06T16:34:56+00:00",
    "updated_at": "2013-09-25T18:44:35+00:00"
  },
  "entity": {
    "name": "my-org",
    "billing_enabled": true,
    "quota_definition_guid": "52c5413c-869f-455a-8873-7972ecb85ca8",
    "status": "active",
    "quota_definition_url": "/v2/quota_definitions/52c5413c-869f-455a-8873-7972ecb85ca8",
    "spaces_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/spaces",
    "domains_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/domains",
    "private_domains_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/private_domains",
    "users_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/users",
    "managers_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/managers",
    "billing_managers_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/billing_managers",
    "auditors_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/auditors",
    "app_events_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfcdc/app_events"
  }
}
```

In this example, the GUID of my-org is c54bf317-d791-4d12-89f0-b56d0936cfcdc.

To make a private plan available to a specific organization, run:

```
cf curl /v2/service_plan_visibilities -X POST -d '{"service_plan_guid":"SERVICE_PLAN_GUID", "organization_guid":"ORG_GUID"}'
```

Example:

```
$ cf curl /v2/service_plan_visibilities -X 'POST' -d '{"service_plan_guid":"1113aa0-124e-4af2-1526-6bfacf61b111","organization_guid":"aaaaa1234-da91-4f12-8ffa-b51d0336aaaa"}'

{
  "metadata": {
    "guid": "99993789-a368-483e-ae7c-ebe79e199999",
    "url": "/v2/service_plan_visibilities/99993789-a368-483e-ae7c-ebe79e199999",
    "created_at": "2014-02-12T21:03:42+00:00",
    "updated_at": null
  },
  "entity": {
    "service_plan_guid": "1113aa0-124e-4af2-1526-6bfacf61b111",
    "organization_guid": "aaaaa1234-da91-4f12-8ffa-b51d0336aaaa",
    "service_plan_url": "/v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111",
    "organization_url": "/v2/organizations/c54bf317-d791-4d12-89f0-b56d0936cfdc"
  }
}
```

Members of my-org can now see the plan-name-1 service plan in the list of available services when a space of my-org is targeted.

Note: The `guid` field in the `metadata` section of this JSON response is the id of the “service plan visibility”, and can be used to revoke access to the plan for the organization as described below.

Disable Access to Service Plans

Disable access to a plan for all organizations

To make a service plan private, follow the instructions above for [Enable Access](#), but replace `"public":true` with `"public":false`.

Note: organizations that have explicitly been granted access will retain access once a plan is private. To be sure access is removed for all organizations, access must be explicitly revoked for organizations to which access has been explicitly granted. For details see below.

Example making plan-name-1 private:

```
$ cf curl /v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111 -X 'PUT' -d '{"public":false}'

{
  "metadata": {
    "guid": "1113aa0-124e-4af2-1526-6bfacf61b111",
    "url": "/v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111",
    "created_at": "2014-02-12T06:24:04+00:00",
    "updated_at": "2014-02-12T20:55:10+00:00"
  },
  "entity": {
    "name": "plan-name-1",
    "free": true,
    "description": "plan-desc-1",
    "service_guid": "d9011411-1463-477c-b223-82e04996b91f",
    "extra": "{\"bullets\":[\"bullet1\",\"bullet2\"]}",
    "unique_id": "plan-id-1",
    "public": false,
    "service_url": "/v2/services/d9011411-1463-477c-b223-82e04996b91f",
    "service_instances_url": "/v2/service_plans/1113aa0-124e-4af2-1526-6bfacf61b111/service_instances"
  }
}
```

Disable access to a private plan for a particular organization

To revoke access to a service plan for a particular organization, run:

```
cf curl /v2/service_plan_visibilities/SERVICE_PLAN_VISIBILITIES_GUID -X 'DELETE'
```

Example:

```
$ cf curl /v2/service_plan_visibilities/99993789-a368-483e-ae7c-ebe79e199999 -X DELETE
```

Catalog Metadata

The Services Marketplace is defined as the aggregate catalog of services and plans exposed to end users of a Cloud Foundry instance. Marketplace services may come from one or many service brokers. The Marketplace is exposed to end users by cloud controller clients (web, CLI, IDEs, etc), and the Cloud Foundry community is welcome to develop their own clients. All clients are not expected to have the same requirements for information to expose about services and plans. This document discusses user-facing metadata for services and plans, and how the broker API enables broker authors to provide metadata required by different cloud controller clients.

As described in the [Service Broker API](#), the only required user-facing fields are `label` and `description` for services, and `name` and `description` for service plans. Rather than attempt to anticipate all potential fields that clients will want, or add endless fields to the API spec over time, the broker API provides a mechanism for brokers to advertise any fields a client requires. This mechanism is the `metadata` field.

The contents of the `metadata` field are not validated by cloud controller but may be by cloud controller clients. Not all clients will make use of the value of `metadata`, and not all brokers have to provide it. If a broker does advertise the `metadata` field, client developers can choose to display some or all fields available.

 **Note:** In the [v1 broker API](#), the `metadata` field was called `extra`.

Community-Driven Standards

This page provides a place to publish the metadata fields required by popular cloud controller clients. Client authors can add their metadata requirements to this document, so that broker authors can see what metadata they should advertise in their catalogs.

Before adding new fields, consider whether an existing one will suffice.

 **Note:** “CLI strings” are all lowercase, no spaces. Keep it short; imagine someone having to type it as an argument for a longer CLI command.

Services Metadata Fields

Broker API Field	Type	Description	CC API Field	Pivotal CLI	Pivotal Apps Manager
name	CLI string	A short name for the service to be displayed in a catalog.	label	X	X
description	string	A short 1-line description for the service, usually a single sentence or phrase.	description	X	X
metadata.displayName	string	The name of the service to be displayed in graphical clients	extra.displayName		X
metadata.imageUrl	string	The URL to an image.	extra.imageUrl		X
metadata.longDescription	string	Long description	extra.longDescription		X
metadata.providerDisplayName	string	The name of the upstream entity providing the actual service	extra.providerDisplayName		X
metadata.documentationUrl	string	Link to documentation page for service	extra.documentationUrl		X
metadata.supportUrl	string	Link to support for the service	extra.supportUrl		X

Plan Metadata Fields

Broker API Field	Type	Description	CC API Field	Pivotal CLI	Pivotal Apps Manager
name	CLI string	A short name for the service plan to be displayed in a catalog.	name	X	
description	string	A description of the service plan to be displayed in a catalog.	description		
metadata.bullets	array-of-strings	Features of this plan, to be displayed in a bulleted-list	extra.bullets		X
metadata.costs	cost object	An array-of-objects that describes the costs of a service, in what currency, and the unit of measure. If there are multiple costs, all of them could be billed to the user (such as a monthly + usage costs at once). Each object must provide the following keys: amount: { usd: float }, unit: string This indicates the cost in USD of the service plan, and how frequently the cost is occurred, such as "MONTHLY" or "per 1000 messages".	extra.costs		X
metadata.displayName	string	Name of the plan to be display in graphical clients.	extra.displayName		X

Example Broker Response Body

The example below contains a catalog of one service, having one service plan. Of course, a broker can offering a catalog of many services, each having many plans.

```
{
  "services": [
    {
      "id": "766fa866-a950-4b12-adff-c11fa4cf8fdc",
      "name": "cloudamqp",
      "description": "Managed HA RabbitMQ servers in the cloud",
      "requires": [
        ],
      "tags": [
        "amqp",
        "rabbitmq",
        "messaging"
      ],
      "metadata": {
        "displayName": "CloudAMQP",
        "imageUrl": "https://d33na3ni6eqf5j.cloudfront.net/app_resources/18492/thumbs_112/img9069612145282015279.png",
        "longDescription": "Managed, highly available, RabbitMQ clusters in the cloud",
        "providerDisplayName": "84codes AB",
        "documentationUrl": "http://docs.cloudfoundry.com/docs/dotcom/marketplace/services/cloudamqp.html",
        "supportUrl": "http://www.cloudamqp.com/support.html"
      },
      "dashboard_client": {
        "id": "p-mysql-client",
        "secret": "p-mysql-secret",
        "redirect_uri": "http://p-mysql.example.com/auth/create"
      },
      "plans": [
        {
          "id": "024f3452-67f8-40bc-a724-a20c4ea24b1c",
          "name": "bunny",
          "description": "A mid-sized plan",
          "metadata": {
            "bullets": [
              "20 GB of messages",
              "20 connections"
            ],
            "costs": [
              {
                "amount": {
                  "usd": 99.0,
                  "eur": 49.0
                },
                "unit": "MONTHLY"
              },
              {
                "amount": {
                  "usd": 0.99,
                  "eur": 0.49
                },
                "unit": "1GB of messages over 20GB"
              }
            ],
            "displayName": "Big Bunny"
          }
        }
      ]
    }
  ]
}
```

Example Cloud Controller Response Body

```
{
  "metadata": {
    "guid": "bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
    "url": "/v2/services/bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
    "created_at": "2014-01-08T18:52:16+00:00",
    "updated_at": "2014-01-09T03:19:16+00:00"
  },
  "entity": {
    "label": "cloudamqp",
    "provider": "cloudamqp",
    "url": "http://adgw.a1.cf-app.com",
    "description": "Managed HA RabbitMQ servers in the cloud",
    "long_description": null,
    "version": "n/a",
    "info_url": null,
    "active": true,
    "bindable": true,
    "unique_id": "18723",
    "extra": {
      "displayName": "CloudAMQP",
      "imageUrl": "https://d33na3ni6eqf5j.cloudfront.net/app_resources/18723/thumbs_112/img9069612145282015279.png",
      "longDescription": "Managed, highly available, RabbitMQ clusters in the cloud",
      "providerDisplayName": "84codesAB",
      "documentationUrl": null,
      "supportUrl": null
    },
    "tags": [
      "amqp",
      "rabbitmq"
    ],
    "requires": [
    ],
    "documentation_url": null,
    "service_plans": [
      {
        "metadata": {
          "guid": "6c4903ab-14ce-41de-adb2-632cf06117a5",
          "url": "/v2/services/6c4903ab-14ce-41de-adb2-632cf06117a5",
          "created_at": "2013-11-01T00:21:25+00:00",
          "updated_at": "2014-01-09T03:19:16+00:00"
        },
        "entity": {
          "name": "bunny",
          "free": true,
          "description": "Big Bunny",
          "service_guid": "bc8748f1-fe05-444d-ab7e-9798e1f9aef6",
          "extra": {
            "bullets": [
              "20 GB of messages",
              "20 connections"
            ],
            "costs": [
              {
                "amount": {
                  "usd": 99.0,
                  "eur": 49.0
                },
                "unit": "MONTHLY"
              },
              {
                "amount": {
                  "usd": 0.99,
                  "eur": 0.49
                },
                "unit": "1GB of messages over 20GB"
              }
            ],
            "displayName": "Big Bunny"
          },
          "unique_id": "addonOffering_1889",
          "public": true
        }
      }
    ]
  }
}
```

Binding Credentials

A bindable service returns credentials that an application can consume in response to the `cf bind` API call. Cloud Foundry writes these credentials to the `VCAP_SERVICES` environment variable. In some cases, buildpacks write a subset of these credentials to other environment variables that frameworks might need.

Choose from the following list of credential fields if possible, though you can provide additional fields as needed. Refer to the [Using Bound Services](#) section of the *Managing Service Instances with the CLI* topic for information on how these credentials are consumed.

Note: If you provide a service that supports a connection string, provide the `uri` key for buildpacks and application libraries to use.

CREDENTIALS	DESCRIPTION
uri	Connection string of the form <code>DB-TYPE://USERNAME:PASSWORD@HOSTNAME:PORT/NAME</code> , where <code>DB-TYPE</code> is a type of database such as mysql, postgres, mongodb, or amqp.
hostname	FQDN of the server host
port	Port of the server host
name	Name of the service instance
vhost	Name of the messaging server virtual host - a replacement for a <code>name</code> specific to AMQP providers
username	Server user
password	Server password

The following is an example output of `ENV['VCAP_SERVICES']`.

Note: Depending on the types of databases you are using, each database might return different credentials.

```
VCAP_SERVICES=
{
  cleardb: [
    {
      name: "cleardb-1",
      label: "cleardb",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
      }
    }
  ],
  cloudamqp: [
    {
      name: "cloudamqp-6",
      label: "cloudamqp",
      plan: "lemur",
      credentials: {
        uri: "amqp://ksvyjmiv:IwN6dCdZmeQD4O0ZPKpu1Y0aLx1he8wo@lemur.cloudamqp.com/ksvyjmiv"
      }
    }
  ],
  {
    name: "cloudamqp-9dbc6",
    label: "cloudamqp",
    plan: "lemur",
    credentials: {
      uri: "amqp://vhuklnxa:9INFxpTujsAdTts98vQIdKHW3MojyMyV@lemur.cloudamqp.com/vhuklnxa"
    }
  }
],
  rediscloud: [
    {
      name: "rediscloud-1",
      label: "rediscloud",
      plan: "20mb",
      credentials: {

```

```
    port: "6379",
    host: "pub-redis-6379.us-east-1-2.3.ec2.redislabs.com",
    password: "1M5zd3QfWi9nUyya"
  },
],
}
```

Dashboard Single Sign-On

Introduction

Single sign-on (SSO) enables Cloud Foundry users to authenticate with third-party service dashboards using their Cloud Foundry credentials. Service dashboards are web interfaces which enable users to interact with some or all of the features the service offers. SSO provides a streamlined experience for users, limiting repeated logins and multiple accounts across their managed services. The user's credentials are never directly transmitted to the service since the OAuth2 protocol handles authentication.

Dashboard SSO was introduced in [cf-release v169](#) so this or a newer version is required to support the feature.

Enabling the feature in Cloud Foundry

To enable the SSO feature, the Cloud Controller requires a UAA client with sufficient permissions to create and delete clients for the service brokers that request them. This client can be configured by including the following snippet in the cf-release manifest:

```
properties:
  uaa:
    clients:
      cc-service-dashboards:
        secret: cc-broker-secret
        scope: openid,cloud_controller_service_permissions.read
        authorities: clients.read,clients.write,clients.admin
        authorized-grant-types: client_credentials
```

When this client is not present in the cf-release manifest, Cloud Controller cannot manage UAA clients and an operator will receive a warning when creating or updating service brokers that advertise the `dashboard_client` properties discussed below.

Service Broker Responsibilities

Registering the Dashboard Client

1. A service broker must include the `dashboard_client` field in the JSON response from its [catalog endpoint](#) for each service implementing this feature. A valid response would appear as follows:

```
{
  "services": [
    {
      "id": "44b26033-1f54-4087-b7bc-da9652c2a539",
      ...
      "dashboard_client": {
        "id": "p-mysql-client",
        "secret": "p-mysql-secret",
        "redirect_uri": "http://p-mysql.example.com"
      }
    }
  ]
}
```

The `dashboard_client` field is a hash containing three fields:

- `id` is the unique identifier for the OAuth2 client that will be created for your service dashboard on the token server (UAA), and will be used by your dashboard to authenticate with the token server (UAA).
- `secret` is the shared secret your dashboard will use to authenticate with the token server (UAA).
- `redirect_uri` is used by the token server as an additional security precaution. UAA will not provide a token if the callback URL declared by the service dashboard doesn't match the domain name in `redirect_uri`. The token server matches on the domain name, so any paths will also match; e.g. a service dashboard requesting a token and declaring a callback URL of `http://p-mysql.example.com/manage/auth` would be approved if `redirect_uri` for its client is `http://p-mysql.example.com/`.

2. When a service broker which advertises the `dashboard_client` property for any of its services is [added or updated](#),

Cloud Controller will create or update UAA clients as necessary. This client will be used by the service dashboard to authenticate users.

Dashboard URL

A service broker should return a URL for the `dashboard_url` field in response to a [provision request](#). Cloud Controller clients should expose this URL to users. `dashboard_url` can be found in the response from Cloud Controller to create a service instance, enumerate service instances, space summary, and other endpoints.

Users can then navigate to the service dashboard at the URL provided by `dashboard_url`, initiating the OAuth2 login flow.

Service Dashboard Responsibilities

OAuth2 Flow

When a user navigates to the URL from `dashboard_url`, the service dashboard should initiate the OAuth2 login flow. A summary of the flow can be found in [section 1.2 of the OAuth2 RFC](#). OAuth2 expects the presence of an [Authorization Endpoint](#) and a [Token Endpoint](#). In Cloud Foundry, these endpoints are provided by the UAA. Clients can discover the location of UAA from Cloud Controller's info endpoint; in the response the location can be found in the `token_endpoint` field.

```
$ curl api.example-cf.com/info
{"name":"vcap","build":"2222","support":"http://support.example-cf.com","version":2,"description":"Cloud Foundry sponsored by Pivotal","aut
```

 To enable service dashboards to support SSO for service instances created from different Cloud Foundry instances, the `/v2/info` url is sent to service brokers in the `X-Api-Info-Location` header of every API call. A service dashboard should be able to discover this URL from the broker, and enabling the dashboard to contact the appropriate UAA for a particular service instance.

A service dashboard should implement the OAuth2 Authorization Code Grant type ([UAA docs](#), [RFC docs](#)).

1. When a user visits the service dashboard at the value of `dashboard_url`, the dashboard should redirect the user's browser to the Authorization Endpoint and include its `client_id`, a `redirect_uri` (callback URL with domain matching the value of `dashboard_client.redirect_uri`), and list of requested scopes. Scopes are permissions included in the token a dashboard client will receive from UAA, and which Cloud Controller uses to enforce access. A client should request the minimum scopes it requires. The minimum scopes required for this workflow are `cloud_controller_service_permissions.read` and `openid`. For an explanation of the scopes available to dashboard clients, see [On Scopes](#).

2. UAA authenticates the user by redirecting the user to the Login Server, where the user then approves or denies the scopes requested by the service dashboard. The user is presented with human readable descriptions for permissions representing each scope. After authentication, the user's browser is redirected back to the Authorization endpoint on UAA with an authentication cookie for the UAA.
3. Assuming the user grants access, UAA redirects the user's browser back to the value of `redirect_uri` the dashboard provided in its request to the Authorization Endpoint. The `Location` header in the response includes an authorization code.

```
HTTP/1.1 302 Found
Location: https://p-mysql.example.com/manage/auth?code=F45jH
```

4. The dashboard UI should then request an access token from the Token Endpoint by including the authorization code received in the previous step. When making the request the dashboard must authenticate with UAA by passing the client `id` and `secret` in a basic auth header. UAA will verify that the client id matches the client it issued the code to. The dashboard should also include the `redirect_uri` used to obtain the authorization code for verification.
5. UAA authenticates the dashboard client, validates the authorization code, and ensures that the redirect URI received matches the URI used to redirect the client when the authorization code was issued. If valid, UAA responds back with an access token and a refresh token.

Checking User Permissions

UAA is responsible for authenticating a user and providing the service with an access token with the requested permissions. However, after the user has been logged in, it is the responsibility of the service dashboard to verify that the user making the request to manage an instance currently has access to that service instance.

The service can accomplish this with a GET to the `/v2/service_instances/:guid/permissions` endpoint on the Cloud Controller. The request must include a token for an authenticated user and the service instance guid. The token is the same one obtained from the UAA in response to a request to the Token Endpoint, described above. .

Example Request:

```
curl -H 'Content-Type: application/json' \
-H 'Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyX2lkIjoid' \
http://api.cloudfoundry.com/v2/service_instances/44b26033-1f54-4087-b7bc-da9652c2a539/permissions
```

Response:

```
{  
  "manage": true  
}
```

The response will indicate to the service whether this user is allowed to manage the given instance. A `true` value for the `manage` key indicates sufficient permissions; `false` would indicate insufficient permissions. Since administrators may change the permissions of users, the service should check this endpoint whenever a user uses the SSO flow to access the service's UI.

On Scopes

Scopes let you specify exactly what type of access you need. Scopes limit access for OAuth tokens. They do not grant any additional permission beyond that which the user already has.

Minimum Scopes

The following two scopes are necessary to implement the integration. Most dashboard shouldn't need more permissions than these scopes enabled.

Scope	Permissions
<code>openid</code>	Allows access to basic data about the user, such as email addresses
<code>cloud_controller_service_permissions.read</code>	Allows access to the CC endpoint that specifies whether the user can manage a given service instance

Additional Scopes

Dashboards with extended capabilities may need to request these additional scopes:

Scope	Permissions
<code>cloud_controller.read</code>	Allows read access to all resources the user is authorized to read
<code>cloud_controller.write</code>	Allows write access to all resources the user is authorized to update / create / delete

Reference Implementation

The [MySQL Service Broker](#) is an example of a broker that also implements a SSO dashboard. The login flow is implemented using the [OmniAuth library](#) and a custom [UAA OmniAuth Strategy](#). See this [OmniAuth wiki page](#) for instructions on how to create your own strategy.

The UAA OmniAuth strategy is used to first get an authorization code, as documented in [this section](#) of the UAA documentation. The user is redirected back to the service (as specified by the `callback_path` option or the default `auth/cloudfoundry/callback` path) with the authorization code. Before the application / action is dispatched, the OmniAuth strategy uses the authorization code to [get a token](#) and uses the token to request information from UAA to fill the `omniauth.auth` environment variable. When OmniAuth returns control to the application, the `omniauth.auth` environment variable hash will be filled with the token and user information obtained from UAA as seen in the [Auth Controller](#).

Restrictions

- UAA clients are scoped to services. There must be a `dashboard_client` entry for each service that uses SSO integration.
- Each `dashboard_client_id` must be unique across the CloudFoundry deployment.

Resources

- [OAuth2 ↗](#)
- [Example broker with SSO implementation ↗](#)
- [Cloud Controller API Docs ↗](#)
- [User Account and Authentication \(UAA\) Service APIs ↗](#)

Example Service Brokers

The following example service broker applications have been developed - these are a great starting point if you are developing your own service broker.

Ruby

- [GitHub repo service](#) - this is designed to be an easy-to-read example of a service broker, with complete documentation, and comes with a demo app that uses the service. The broker can be deployed as an application to any Cloud Foundry instance or hosted elsewhere. The service broker uses GitHub as the service back end.
- [MySQL database service](#) - this broker and its accompanying MySQL server are designed to be deployed together as a [BOSH](#) release. BOSH is used to deploy or upgrade the release, monitors the health of running components, and restarts or recreates unhealthy VMs. The broker code alone can be found [here](#).

Java

- [Spring Boot Service Broker](#) - This implements the rest contract for service brokers and the artifacts are published to the spring maven repo. This greatly simplifies development: include a single dependency in gradle, implement interfaces, and configure. A sample implementation has been provided for [MongoDB](#).
- [MySQL Java Broker](#) - a Java port of the Ruby-based [MySQL broker](#) above.

Go

- [Asynchronous Service Broker for AWS EC2](#) - This broker implements support for the experimental [Asynchronous Service Operations](#), and calls AWS APIs to provision EC2 VMs.

Application Log Streaming

By binding an application to an instance of an applicable service, Cloud Foundry will stream logs for the bound application to the service instance.

- Logs for all apps bound to a log-consuming service instance will be streamed to that instance
- Logs for an app bound to multiple log-consuming service instances will be streamed to all instances

To enable this functionality, a service broker must implement the following:

1. In the [catalog](#) endpoint, the broker must include `requires: syslog_drain`. This minor security measure validates that a service returning a `syslog_drain_url` in response to the [bind](#) operation has also declared that it expects log streaming. If the broker does not include `requires: syslog_drain`, and the bind request returns a value for `syslog_drain_url`, Cloud Foundry will return an error for the bind operation.
2. In response to a [bind](#) request, the broker should return a value for `syslog_drain_url`. The syslog URL has a scheme of syslog, syslog-tls, or https and can include a port number. For example:
`"syslog_drain_url": "syslog://logs.example.com:1234"`

How does it work?

1. Service broker returns a value for `syslog_drain_url` in response to bind
2. Loggregator periodically polls CC `/v2/syslog_drain_urls` for updates
3. Upon discovering a new `syslog_drain_url`, Loggregator identifies the associated app
4. Loggregator streams app logs for that app to the locations specified by the service instances' `syslog_drain_url`s

Users can manually configure app logs to be streamed to a location of their choice using User-provided Service Instances. For details, see [Using Third-Party Log Management Services](#).

Supporting Multiple Cloud Foundry Instances

It is possible to register a service broker with multiple Cloud Foundry instances. It may be necessary for the broker to know which Cloud Foundry instance is making a given request. For example, when using [Dashboard Single Sign-On](#), the broker is expected to interact with the authorization and token endpoints for a given Cloud Foundry instance.

There are two strategies that can be used to discover which Cloud Foundry instance is making a given request.

Routing & Authentication

The broker can use unique credentials and/or a unique url for each Cloud Foundry instance. When registering the broker, different Cloud Foundry instances can be configured to use different base urls that include a unique id. For example:

- On Cloud Foundry instance 1, the service broker is registered with the url `broker.example.com/123`
- On Cloud Foundry instance 2, the service broker is registered with the url `broker.example.com/456`

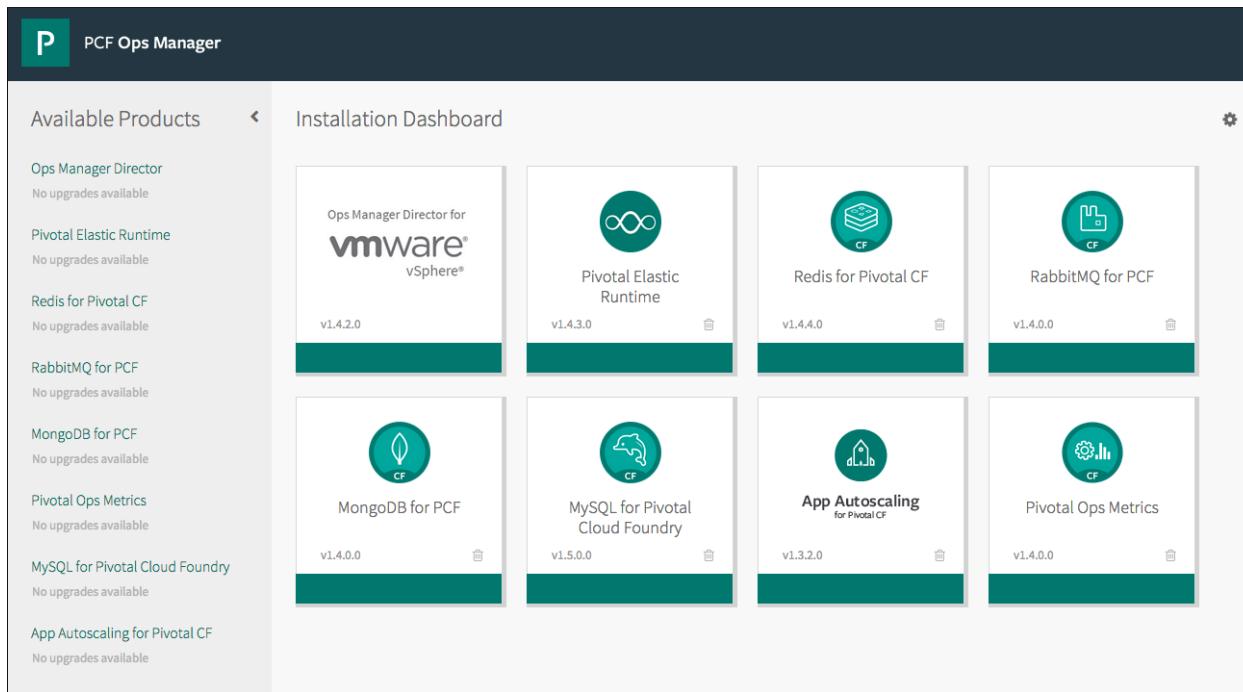
X-Api-Info-Location Header

All calls to the broker from Cloud Foundry include an `X-Api-Info-Location` header containing the `/v2/info` url for that instance. The `/v2/info` endpoint will return further information, including the location of that Cloud Foundry instance's UAA.

Support for this header was introduced in cf-release v212.

Creating a Pivotal Cloud Foundry Product Tile

Installed software products appear in the [Pivotal Cloud Foundry \(PCF\)](#)  Ops Manager Installation Dashboard as tiles. This tutorial describes the procedure for creating a product tile for use in Ops Manager.



The screenshot shows the 'Installation Dashboard' in PCF Ops Manager. On the left, a sidebar lists 'Available Products' with their versions and upgrade status:

- Ops Manager Director (v1.4.2.0, No upgrades available)
- Pivotal Elastic Runtime (v1.43.0, No upgrades available)
- Redis for Pivotal CF (v1.4.4.0, No upgrades available)
- RabbitMQ for PCF (v1.4.0.0, No upgrades available)
- MongoDB for PCF (v1.4.0.0, No upgrades available)
- Pivotal Ops Metrics (v1.4.0.0, No upgrades available)
- MySQL for Pivotal Cloud Foundry (v1.5.0.0, No upgrades available)
- App Autoscaling for Pivotal CF (v1.3.2.0, No upgrades available)

The main area displays eight product tiles in a grid:

Product	Version
Ops Manager Director for VMware vSphere	v1.4.2.0
Pivotal Elastic Runtime	v1.43.0
Redis for Pivotal CF	v1.4.4.0
RabbitMQ for PCF	v1.4.0.0
MongoDB for PCF	v1.4.0.0
MySQL for Pivotal Cloud Foundry	v1.5.0.0
App Autoscaling for Pivotal CF	v1.3.2.0
Pivotal Ops Metrics	v1.4.0.0

Prerequisites

To complete this tutorial, you will need:

- An Amazon Web Services (AWS) account with Virtual Private Cloud (VPC).
- The [AWS CLI](#)  installed on your machine.
- Some familiarity with [BOSH](#) .
- The [BOSH CLI](#)  installed on your machine.

Procedures

Ops Manager automates the BOSH deployment of software products, but introduces additional complexity as well. As a tile author, you should first confirm that your product successfully deploys with BOSH before attempting to deploy it with Ops Manager.

1. [Deploying A PCF Product Using BOSH](#) describes the procedure for deploying a release of your software product with BOSH. Confirm that this is successful before proceeding to deploy with Ops Manager.
2. [Deploying Your Product as a Tile in Ops Manager](#) describes the procedure for creating a tile for your product and deploying to Ops Manager.

Reference Documentation

[Product Template Reference](#) provides explanations for the BOSH manifest key-value pairs that are required for an Ops Manager product template.

Deploying a PCF Product Using BOSH

BOSH is Cloud Foundry's orchestration tool for installing and managing software and virtual machines. A product release should install successfully using BOSH before it can be crafted into an Ops Manager tile.

Refer to the [BOSH documentation](#) for more information about how to create a BOSH release of your own.

This topic leads you through the process of deploying an example BOSH release consisting of the [Nginx](#) webserver.

Step 1: Prepare an Environment

First, you must prepare an environment on your IaaS. For this tutorial, complete the following procedure to prepare an environment on Amazon Web Services (AWS). The steps differ slightly for other IaaSes, such as vSphere or OpenStack.

- Run the following command to clone the Example Tile Docs Resources git repository.

```
$ git clone git@github.com:pivotal-cf-experimental/example-tile-docs-resources.git
```

- cd into the cloned directory:

```
$ cd example-tile-docs-resources
```

- Create a key pair for your AWS Virtual Machines and pipe it to a local file named MyKeyPair.pem:

```
$ aws ec2 create-key-pair --key-name MyKeyPair \
| python -c 'import json,sys;obj=json.load(sys.stdin);print obj["KeyMaterial"]' \
> MyKeyPair.pem
```

- Provision your AWS account with the resources specified in the template:

```
$ aws cloudformation create-stack \
--stack-name examplestack \
--template-body file://cloudformation.json \
--parameters \
ParameterKey=KeyName,ParameterValue=MyKeyPair \
--capabilities CAPABILITY_IAM
```

- Allocate an Elastic IP for your BOSH instance by running the following command. The terminal displays the IP address for your BOSH instance. Record this IP address for use in a step below.

```
$ aws ec2 allocate-address --domain vpc
{
  "PublicIp": "52.8.233.151",
  "Domain": "vpc",
  "AllocationId": "eipalloc-13618976"
}
```

Step 2: Install a BOSH on Your IaaS

Complete the following procedure to install a BOSH on your IaaS. You use this BOSH to deploy the example release. See [Initializing an environment](#) to learn about deploying BOSH.

The Example Tile Docs repo you cloned in the previous step contains a BOSH manifest, `bosh.yml`, for the BOSH we will use to deploy the example release.

- cd into `example-tile-docs-resources`.
- Open `bosh.yml`. This is the BOSH manifest file for the BOSH you will use to deploy the example release.
- In `bosh.yml`, follow the instructions in comments to fill in the details specific to your own AWS account.
- Run the following command to deploy your BOSH instance:

```
$ bosh init deploy ./bosh.yml
Deployment manifest: '/Users/d/my-bosh/bosh.yml'
Deployment state: '/Users/d/my-bosh/bosh-state.json'
Finished deploying (00:16:35)
Stopping registry... Finished (00:00:00)
Cleaning up rendered CPI jobs... Finished (00:00:00)
```

Step 3: Prepare the BOSH Manifest for the Release

In this step, we prepare a BOSH manifest for the example product release. The BOSH instance deployed in the last step will use this manifest to deploy the example product. When [deploying a product tile with Ops Manager](#), Ops Manager generates a manifest like this using a template we provide. However, using a manifest we create manually to deploy without Ops Manager allows us to confirm that the software release is composed properly, as well as gives us an opportunity to understand what goes into a BOSH manifest.

1. Target your BOSH:

```
$ bosh target 52.8.50.119
Target set to `my-bosh'
Your username: admin
Enter password: admin
```

2. Upload your release by running:

```
$ bosh upload release example-release-7+dev.2.tgz
Acting as user 'admin' on 'my-bosh'
Verifying release...
Task 29 done
Started 2015-07-31 00:02:27 UTC
Finished 2015-07-31 00:02:27 UTC
Duration 00:00:00
Release uploaded
```

3. Upload the latest AWS Xen-HVM Light stemcell to your BOSH by running:

```
$ bosh upload stemcell https://bosh.io/d/stemcells/bosh-aws-xen-hvm-ubuntu-trusty-go_agent?v=3026
Acting as user 'admin' on 'my-bosh'
Using remote stemcell `https://bosh.io/d/stemcells/bosh-aws-xen-hvm-ubuntu-trusty-go_agent?v=3026'
Director task 15
Started update stemcell
Started update stemcell > Downloading remote stemcell. Done (00:00:02)
Started update stemcell > Extracting stemcell archive. Done (00:00:00)
Started update stemcell > Verifying stemcell manifest. Done (00:00:00)
Started update stemcell
Done update stemcell (00:00:11)
Task 15 done
Started 2015-07-30 21:23:09 UTC
Finished 2015-07-30 21:23:20 UTC
Duration 00:00:11
Stemcell uploaded and created.
```

4. Get your BOSH Director (BOSH) **UUID** by running the following command. You will need to enter this ID in the BOSH manifest in the next step.

```
$ bosh status
Config
  /Users/d/.bosh_config
Director
  Name      my-bosh
  URL       https://52.8.50.119:25555
  Version   1.3012.0 (00000000)
  User      admin
  UUID      729452b0-2c1f-4763-a6ec-cf43e7f0ba89
```

5. Open `example-release.yml`. This is the BOSH manifest file for the example release. The BOSH you deployed previously will use this manifest to deploy the example product.
6. In `example-release.yml`, follow the instructions in comments to fill in the details specific to your BOSH deployment.
7. Run the following command to set the deployment.

```
$ bosh deployment ./example-release.yml
```

8. Deploy the example release:

```
$ bosh deploy  
Acting as user 'admin'  
Are you sure you want to deploy? (type 'yes' to continue): yes  
Task 31 done  
Started 2015-07-31 00:04:19 UTC  
Finished 2015-07-31 00:12:52 UTC  
Duration 00:08:33  
  
Deployed example-product-906ec4aecbec172c9706' to my-bosh'
```

Step 4: Confirm Successful Deployment

1. Confirm that the webserver VM is running:

```
$ bosh vms  
Acting as user 'admin' on 'my-bosh'  
Deployment `example-product-906ec4aecbec172c9706'  
Director task 32  
Task 32 done  
+-----+-----+-----+  
| Job/index | State | Resource Pool | IPs |  
+-----+-----+-----+  
| web_server/0 | running | vms | 10.0.16.5 |  
+-----+-----+-----+
```

2. SSH into your VPC using the IP that you [allocated](#) and the key pair that you [created](#).

```
$ ssh-add MyKeyPair.pem  
Identity added: MyKeyPair.pem (MyKeyPair.pem)  
$ ssh ubuntu@BOSH-IP-ADDRESS  
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-59-generic x86_64)
```

3. To confirm successful deployment, view the 'hello world' page served by the webserver in the example product.

```
$ curl 10.0.16.5  
<html>  
 <body>  
   <h1>Hello World</h1>  
 </body>  
</html>
```

Step 5: Terminate VMs

Use the AWS EC2 Dashboard to terminate the instances in your VPC before proceeding to [Deploying Your Product as a Tile](#). See [the AWS documentation on terminating instances](#).

Deploying Your Product as a Tile in Ops Manager

This topic describes the procedure for deploying a Cloud Foundry software product as a tile in Ops Manager. Ops Manager offers a web GUI and eases the process of deploying software with BOSH by providing the following:

- Automated error checking
- Automated manifest generation
- Life cycle management for upgrading a product when a new release is published.

Note: Before beginning this procedure, you should follow the procedure described in [Deploying a PCF Product Using BOSH](#).

Step 1: Deploy Ops Manager

Ops Manager is available on [Pivotal Network](#). For the purpose of this step we will launch an Amazon AMI in the VPC we used in [Deploying a PCF Product Using BOSH](#).

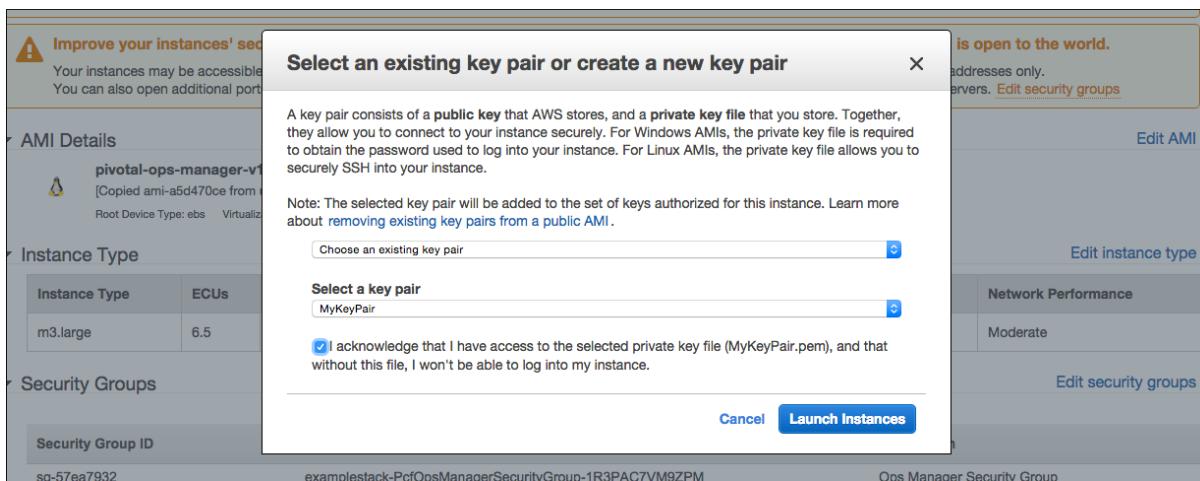
1. Login to [Pivotal Network](#) and download the latest Ops Manager for AWS. This is a PDF with the AMI ID.
2. Login to the **AWS Console**.
3. Navigate to the **EC2 Dashboard**.
4. Click **Launch Instance**.
5. Select **Community AMIs** and search for the Ops Manager AMI as printed in the PDF you downloaded from Pivotal Network.



6. Click the **Select** button next to the Ops Manager AMI.
7. Choose an m3.large instance and click Next: Configure Instance Details.
8. Click the dropdown for the **Network** key and choose the VPC you created in [Deploying Your Product with Bosh](#).
9. Click the dropdown for the **Subnet** key and choose the public subnet you created in [Deploying Your Product with Bosh](#).
10. Set **Auto-assign Public IP** to **Enable**.

Number of instances	<input type="text" value="1"/>
Purchasing option	<input type="checkbox"/> Request Spot Instances
Network	vpc-e0f25785 (10.0.0.0/16) pcf-vpc <input type="button" value="Create new VPC"/>
Subnet	subnet-3e96105b(10.0.0.0/24) pcf-public-subnet us-west-2 <input type="button" value="Create new subnet"/> 249 IP Addresses available
Auto-assign Public IP	Enable <input type="button"/>
IAM role	None <input type="button" value="Create new IAM role"/>
Shutdown behavior	Stop <input type="button"/>
Enable termination protection	<input type="checkbox"/> Protect against accidental termination
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring <small>Additional charges apply.</small>
Tenancy	Shared tenancy (multi-tenant hardware) <input type="button"/> <small>Additional charges will apply for dedicated tenancy.</small>

11. Click **Review and Launch**.
12. Edit the Security Group and set it to the OpsManagerSecurityGroup you used in [Deploying Your Product with Bosh](#).
13. Click **Launch**.
14. Choose the keypair you created in [Deploying Your Product with Bosh](#) and accept the acknowledgement checkbox.



15. Click **Launch Instance**.

Step 2: Configure and Install Ops Manager Director

1. Find the public IP address of the Ops Manager you deployed in the EC2 Dashboard.
2. Open a browser and navigate to the Ops Manager IP address.
3. Create a username and password, and agree to the terms and conditions.

PCF Ops Manager

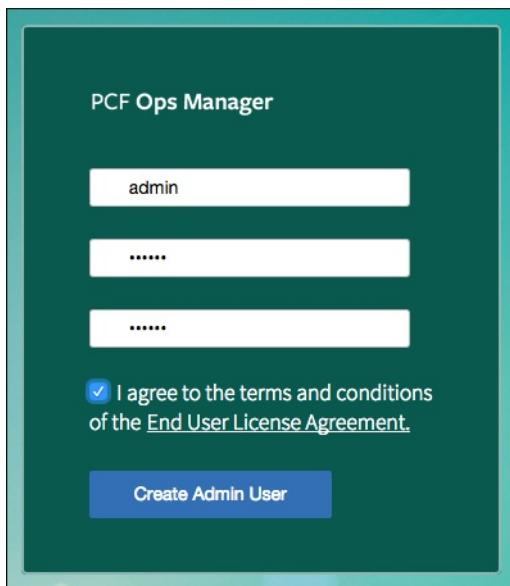
admin

.....

.....

I agree to the terms and conditions
of the [End User License Agreement](#).

Create Admin User



4. Click the **Ops Manager Director** Tile.



5. Enter the **Access Key**, **Secret**, **VPC ID**, **Security Group**, **Key Pair**, **Private Key** and **Region** with the values you used in [Deploying a PCF Product Using BOSH](#).

Settings Status Credentials

- AWS Config
- Director Config
- Create Availability Zones
- Assign Availability Zones
- Create Networks
- Assign Networks
- Security
- Resource Config

AWS Management Console Config

Access Key ID*
AKIAI6MOKKM7V7HSEGJA

AWS Secret Key*
.....

VPC ID*
vpc-e0f25785

Security Group Name*
examplestack-PcfVmsSecurityGroup-1JLA6l

Key Pair Name*
myKeyPair

SSH Private Key*
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAgBSvYpe7fyt3oJ1ICP
EZF8cKeQ99zft7qMhKKAzE7JuGDjoWV
92xdGyQsOq
cR540BvHOZ0zcGT0Ok5/UwolGv3Otbek
-----END RSA PRIVATE KEY-----

Region*
us-west-1

Save

6. Click **Save**.
7. Select **Director Config**.
8. Enter an Amazon NTP Server (0.amazon.pool.ntp.org) and click Save.
9. Select **Create Availability Zone**.
10. Enter the Availability Zone for the private subnet of your VPC (example: us-west-1a) and click **Save**.
11. Select **Assign Availability Zone**.
12. Choose the Availability Zone you created and click **Save**.
13. Select **Create Networks**.
14. Enter a **network name**, **Subnet ID**, **CIDR Range**, **Excluded IP range**, **DNS**, and **gateway** using the same values you used in [Deploying a PCF Product Using BOSH](#).

Note: you will get errors that Ops Manager cannot reach the gateway or DNS server.

Create Network

Name*
default

VPC Subnet ID*
subnet-3996105c

Subnet (CIDR Range)*
10.0.16.0/20

Excluded IP Ranges
10.0.16.0-10.0.16.20

DNS*
10.0.0.2

Gateway*
10.0.16.1

Save

1. Select **Assign Networks**.
2. Choose the network you created and click **Save**.
3. Click **Apply Changes** and wait until the Ops Manager Director is successfully installed.

Note: If you have existing VMs in the VPC you will receive errors that IP addresses are already in use.

Step 3: Download and Examine the Example Product Tile

We will learn how to build an product tile by examining its contents.

1. Download the latest release of the [Example Product Tile](#).
2. Change the extension from `.pivotl` to `.zip` and unzip the file on your local workstation.

Study the files contained in the following directories using these explanations as a guide. Once you have looked at the files, we will deploy the tile, SSH into the Ops Manager, and explore how the tile functions.

- Content_migrations (**optional**): Migrations for when an earlier version is upgraded.
- Metadata (**required**): This directory includes a product template, which is how a product is rendered in Ops Manager, and instructions for how its BOSH manifest is generated.
- Releases (**optional**): BOSH releases that contain the source code for a product. In this case, the release contains the NGINX webserver.

Step 4: Build your own Product Tile

Create your own product tile by following these high level instructions. Each instruction represents hours of work in which can refer to reference documentation for help.

1. Build and create a BOSH release. Information about how to build a release is found in the [BOSH documentation](#).

You can ask questions of the BOSH community using the [CF-BOSH mailing list](#) and [IRC Channel on Freenode](#).

2. Create a Product Template by using the Example Product Tile as a model. Refer to the [Product Template Reference](#) for clarification.
3. Consider adding Lifecycle Errands to both your BOSH release and your Product Template. Refer to the [BOSH documentation](#) and the [Lifecycle Errands](#) topic for more information. Consider what errands to include in your product tile.
4. Create folders according to the previous step, archive the folders into a `.zip` file and import the file into a deployed Ops Manager.
5. The fastest way to make changes to your product tile is to directly SSH into the Ops Manager VM and edit the product template at `/var/tempest/workspace/default/metadata`. After making changes, refresh your browser to see the change. You can click **Apply Changes** in the PCF Ops Manager Installation Dashboard and view the resulting BOSH manifest in `/var/tempest/workspace/default/deployments`.
6. You must create a migration file to migrate values in subsequent releases of your product. These files use a ruby gem known as Transmogrifier. Refer to the [Migrations](#) section more information about building migrations.

Product Template Reference

This document defines the separate pieces of a product template. For the purpose of explanation we will use the [1.6 example product tile](#), a functional tile provided by the Ops Manager engineering team that deploys the NGINX web server.

Top Level Properties

The following is an example of the properties that appear at the top of a product template. Following this example are definitions of each property.

```
---
name: example-product
product_version: "1.6.0.0-alpha.3.3062"
metadata_version: "1.5"
label: 'Ops Manager: Example Product'
description: An example product to demonstrate Ops Manager product-author features
rank: 1
stemcell_criteria:
  os: ubuntu-trusty
  requires_cpi: false
  version: "3062" # Bosh stemcell should match Ops Manager with same version
releases:
- name: example-release
  file: example-release-8.tgz
  version: "8"
post_deploy_errands:
- name: example-errand
pre_delete_errands:
- name: example-errand
icon_image: R0lGODhAgACIAAAAAAP//ywAAAAAAgACAAACoRRADs=
```

Note: The icon_image is a single-pixel gif to keep the example readable.

name

String. Required. The internal name of the product. You must keep the name of your product consistent for migrations to function properly. Changing the name indicates the installation of a completely different product.

product_version

String. Required. The version of the product. At present you can only import this version into Ops Manager once. If you intend to import the same product / version, you must delete the existing one from the /metadata folder and delete the installation files from Ops Manager's disk. The version number is important for [migrations](#).

metadata_version

String. Required. The versioned structure of the product template (the file you are editing). Changing the version number can unlock new properties, and also break properties that changed from previous versions. The metadata version does not always correlate to Ops Manager's version number and depends on what, or if, new metadata properties were introduced.

label

String. Optional. The label that appears in the product tile when it displays in the Ops Manager Dashboard.

description

String. Optional. A description of the product. This is not currently used but may be displayed in a future version of Ops Manager.

rank

Integer. Required. The order in which a product tile appears on the dashboard. The Ops Manager Director always

appears at rank 100. For your product to appear to the right of Ops Manager Director (preferable) you must set this value to an integer less than 100. It is recommended that you set it to 1. Ops Manager will sort tiles alphabetically if all tiles have the same rank. This is a known weak point and will be improved in a future version of Ops Manager.

stemcell_criteria

Hash. Required. Refer to [the BOSH hub](#) for a list of stemcells, including os and version. Notice that you do not specify which IaaS the Stemcell targets. This keeps your product template IaaS agnostic so that one product template can be deployed on any IaaS. At the time of this writing, none of the BOSH stemcells require a Cloud Provider Interface (CPI). This is expected to change in a future release of BOSH.

releases

Array of Hashes. Required. The list of releases contained in your product's releases directory. The version of the release must be exactly the same as the version contained in the release (BOSH releases are versioned and signed by BOSH).

post_deploy_errands

Array of Hashes. Optional. A list of errands that will run after a deploy succeeds. See [Lifecycle Errands](#)

pre_delete_errands

Array of Hashes. Optional. A list of errands that will run before a deployment is deleted. [Lifecycle Errands](#)

Form Properties

The following is an example of the properties that appear in the form_types section of a product template. These forms appear on the left hand side, as links, after a user clicks on the tile itself.

As shown in a later section, form properties reference property_blueprints for the definition of the type of data (url, IP address, list, etc.) being saved. Form properties are themselves referenced in the manifest section of the job_types section, which will also be explained later in this document. The eventual purpose of these properties is to hydrate a BOSH manifest, which Ops Manager generates on the user's behalf.

Following this example are definitions of each property.

```
form_types:
- name: example_form
  label: Configurable Properties
  description: All the properties that you can configure!
  property_inputs:
    - reference: .properties.example_string
      label: Example string
      description: 'Configure a property of type string'
    - reference: .properties.example_migrated_integer
      label: Example integer
      description: 'Configure a property of type integer'
- name: example_collections_form
  label: 'Collection example'
  description: 'A collection example form'
  property_inputs:
    - reference: .properties.example_collection
      label: 'Albums collection'
      description: 'The albums'
      property_inputs:
        - reference: album
          label: 'Name of the Album'
          description: 'ex Graceland'
        - reference: artist
          label: 'Name of the Artist'
          description: 'ex Paul Simon'
        - reference: explicit
          label: 'Explicit?'
          description: '$#!%&'
- name: example_selector_form
  label: 'Selector Example'
  description: 'A selector example form'
  property_inputs:
    - reference: .properties.example_selector
      label: 'Food Choices'
      selector_property_inputs:
        - reference: .properties.example_selector.pizza_option
          label: 'Pizza'
          property_inputs:
            - reference: .properties.example_selector.pizza_option.pepperoni
              label: 'Add Pepperoni'
            - reference: .properties.example_selector.pizza_option.pineapple
              label: 'Add Pineapple'
            - reference: .properties.example_selector.pizza_option.other_toppings
              label: 'Other toppings'
        - reference: .properties.example_selector.filet_mignon_option
          label: 'Filet Mignon'
          property_inputs:
            - reference: .properties.example_selector.filet_mignon_option.rarity_dropdown
              label: 'How rare?'
```

name

String. Required. The internal name of the form.

label

String. Required. The label of the form as it appears as a link on the left hand side of each form.

description

String. Optional. The description of the form. Appears at the top of the form as a header.

property_inputs

Array of Hashes. Required. References to properties defined in the property_blueprints section of the product template.

Simple vs. Complex Inputs (Selectors and Collections)

Most properties are simple values such as strings, integers, url addresses, or IP addresses. Others are complex, such as selectors or collections.

Selectors are a means of giving the user a choice of a set of inputs. Collections are a means of giving the user the ability

to enter an array of values to create a hash.

Selectors appear as follows:

A selector example form

Food Choices*

Pizza

Add Pepperoni

Add Pineapple

Other toppings

Filet Mignon

How rare?*

Rare

Save

Collections appear as follows:

A collection example form

Albums collection

The albums

Add

▶ Total Eclipse of the Heart

Remove

Remove

Name of the Album *

Name of the Artist *

Explicit?

Save

Property Blueprints

The following is an example of the property_blueprints that appear in a product template. These blueprints define anything that will eventually end up in the BOSH manifest generated by Ops Manager. References to these blueprints.

Note that one of these blueprints references a migrated value, which came from the Example Product v. 1.5 using [migrations](#).

No property will be viewable in a form if unless **configurable** is set to **true**. Some properties are not viewable. Rather

than giving the user the ability to enter a value, the value is generated by Ops Manager.

```

property_blueprints:
  - name: example_string
    type: string
    configurable: true
    default: 'Hello world'
  - name: example_migrated_integer
    type: integer
    configurable: true
    default: 1
  - name: example_collection
    type: collection
    configurable: true
    optional: true
  property_blueprints:
    - name: album
      type: string
    - name: artist
      type: string
    - name: explicit
      type: boolean
  - name: example_selector
    type: selector
    configurable: true
    default: Pizza
  option_templates:
    - name: pizza_option
      select_value: Pizza
  named_manifests:
    - name: my_snippet
      manifest: |
        pizza_toppings:
          pepperoni: (( .properties.example_selector.pizza_option.pepperoni.value ))
          pineapple: (( .properties.example_selector.pizza_option.pineapple.value ))
          other: (( .properties.example_selector.pizza_option.other_toppings.value ))
  property_blueprints:
    - name: pepperoni
      type: boolean
      configurable: true
    - name: pineapple
      type: boolean
      configurable: true
    - name: other_toppings
      type: string
      configurable: true
      optional: true
    - name: filet_mignon_option
      select_value: Filet Mignon
  named_manifests:
    - name: my_snippet
      manifest: |
        rarity: (( .properties.example_selector.filet_mignon_option.rarity_dropdown.value ))
  property_blueprints:
    - name: rarity_dropdown
      type: dropdown_select
      configurable: true
      default: rare
      options:
        - name: rare
          label: 'Rare'
        - name: medium
          label: 'Medium'
        - name: well-done
          label: 'Well done'

```

Configurable Properties

Many of these properties are strings, but can be used with validators in order to check that the user typed in the correct format for a url, IP, address, domain, etc.

string

A string.

integer

An integer.

boolean

A boolean. Viewed as a checkbox.

dropdown_select

A list of options. The user chooses one viewed as an HTML select box.

domain

A second, third, fourth, etc level domain.

wildcard_domain

A domain with a wildcard in front of it. Example: `*.domain.com`

text

A string. Appears as an HTML textarea.

ldap_url

A url prefaced by `ldap://`.

email

An email address.

ip_ranges

A range of IP addresses, with dashes and commas allowed. Example: `1.1.1.1-1.1.1.4,2.2.2.1-2.2.2.4`

port

An integer representing a network port.

network_address

A single IP address or domain. Example: `1.1.1.1`

network_address_list

A list of IP addresses or domains. Example: `1.1.1.1,example.com,2.2.2.2`

Generated Properties (can also be configurable)

The following properties are configurable, but can also be generated by Ops manager if configurable is false, or the configurable key is omitted. The exceptions are the uuid and salted credentials properties, which are never configurable.

rsa_cert_credentials

An RSA certificate.

rsa_pkey_credentials

An RSA private key.

salted_credentials

Username and password created using a non-reversible hash algorithm.

simple_credentials

Username and password.

secret

A random string or password.

uuid

A universal unique identifier.

Complex Properties (Selectors and Collections)

The selector and collections inputs are referenced by their selector and collection property blueprints. These are more complicated than simple properties in that they contain manifest snippets, which are further referenced in other manifest snippets. We will learn about manifest snippets in the next section.

Job Types

The following is an example of the job_types section that appear in a product template. This section defines the jobs that end up in a BOSH Manifest. Those jobs are defined in your BOSH Release. Jobs require many different settings in order to function properly, and that is the crux of what Ops Manager does for you: it asks a user for values to those settings, and generates a manifest based on what was entered.

At the time of this writing, every job requires a property_blueprint reference to vm_credentials as appears below. This will be removed as a requirement in a future version of Ops Manager.

```

job_types:
  - name: web_server
    resource_label: Web Server
    templates:
      - name: web_server
        release: example-release
      - name: time_logger
        release: example-release
    release: example-release
    static_ip: 1
    dynamic_ip: 0
    max_in_flight: 1
  resource_definitions:
    - name: ram
      type: integer
      configurable: true
      default: 1024
    - name: ephemeral_disk
      type: integer
      configurable: true
      default: 2048
    - name: persistent_disk
      type: integer
      configurable: true
      default: 1024
      constraints:
        min: 1024
    - name: cpu
      type: integer
      configurable: true
      default: 1
  instance_definitions:
    - name: instances
      type: integer
      configurable: true
      default: 1
      constraints:
        max: 1
  property_blueprints:
    - name: vm_credentials
      type: salted_credentials
      default:
        identity: vcap
  manifest: |
    configured:
      string: (( .properties.example_string.value ))
      integer: (( .properties.example_migrated_integer.value ))

      record_collection: (( .properties.example_collection.value || [] ))
      selector: (( .properties.example_selector.selected_option.parsed_manifest(my_snippet) ))
  ops_manager_provided_accessors:
    name: (( name ))
    ram: (( ram ))
    ephemeral_disk: (( ephemeral_disk ))
    persistent_disk: (( persistent_disk ))
    instances: (( instances ))
    availability_zone: (( availability_zone ))
    first_ip: (( first_ip ))
    ips: (( ips ))
    ips_by_availability_zone: (( ips_by_availability_zone ))
    $ops_manager.ca_certificate: (( $ops_manager.ca_certificate ))

```

name

String. Required. The name of the job as it will be created in the Ops Manager generated BOSH manifest.

resource_label

String. Required. The label of the job as it will appear in the resources page of the tile.

templates

Array of Hashes. Required. This is a BOSH feature (creating jobs from different releases). See the [BOSH documentation](#) for more information.

release

String. Required. The name of the BOSH release contained in your product archive (.pivotal file).

static_ip

Boolean. Required. Sets whether the BOSH job should have a static or dynamic IP. Static IPs are set by the user, and reserved, while Dynamic IPs are set by BOSH. Both are, in effect, static, in that they should not change between deployments.

dynamic_ip

Boolean. Required. Set the opposite of static_ip. This will eventually be eliminated as a property as it is obviously redundant and unnecessary.

max_in_flight

Integer. Required. A BOSH setting that controls the number of instances of this job that BOSH will deploy in parallel.

resource_definition

Array of Hashes. Required. A set of resource settings for the job along with max and min constraints, defaults, and whether or not the user can configure (change) the setting. The resources that can be set are:

- ram
- ephemeral_disk
- persistent_disk
- cpu

instance_definition

Hash. Required. The number of default instances for a job along with max, min, odd, and the ability to decrease sizing after deploy constraints.

manifest

Text Snippet, prefaced by pipe symbol: |. Optional. A BOSH manifest will be created by Ops Manager. The sections of that manifest will include each job, and the properties for that job. These properties come directly from the manifest snippet here, and use a syntax known as “double-parens” which consists of a variable name surrounded by two sets of parentheses.

The double-parens syntax will be evaluated when a user clicks the Apply Changes button in Ops Manager. A manifest is created and the double-parens variables are evaluated. The values, either entered by a user, or generated by Ops Manager, are dynamically inserted in place of the double-parens references.

Selector Manifest Snippets

Selector snippets are evaluated twice. As you saw in the property_blueprint, the selector has a manifest snippet for both sets of inputs that the user might choose. Only one of these sets is evaluated and inserted into the job’s manifest.

Ops Manager Provided Snippets

There are double-parens accessors for a job’s name, ram, ephemeral disk, persistent disk, instances, and availability zones in which jobs are placed. See the example of job types above for the example of these accessors.

In addition, Ops Manager supports accessors that are global to the entire installation rather than job specific. Currently, the only such accessor is `$ops_manager.ca_certificate`, which is the internal SSL CA certificate used to sign all SSL certificates generated by this Ops Manager instance, such as when the user clicks a **Generate Self-Signed RSA Certificate** link.

Understanding Lifecycle Errands

Lifecycle errands are BOSH Errands (scripts) that run at designated points in time during a product installation. Product teams create errands as part of a product package, and a product can only run errands it includes. Refer to [BOSH documentation](#) for more information about BOSH Errands

Products can have two kinds of lifecycle errands:

Post-Install: Post-install errands run after a product installs, before Ops Manager makes the product available for use.

Most post-install errands run by default. An operator can prevent a post-install errand from running by deselecting the checkbox for the errand on the **Settings** tab of the product tile in Ops Manager before installing the product.

The screenshot shows the 'Installation Dashboard' for 'Pivotal MySQL Dev'. The 'Settings' tab is active. In the 'Lifecycle Errands' section, there is a checked checkbox for 'Broker Registrar'. A tooltip for this checkbox provides the description: 'Registers broker with Cloud Controller and makes the 100mb plan public'.

Typical post-install errands include smoke or acceptance tests, databases initialization or database migration, and service broker registration.

Pre-Delete: Pre-delete errands run after an operator chooses to delete a product, before Ops Manager deletes the product. Ops Manager does not display pre-delete errands, and an operator cannot prevent a pre-delete errand from running.

Typical pre-delete errands include clean up of application artifacts and service broker de-registration.

Post-Install Errand Example

Pivotal MySQL has a **Broker Registrar** post-install errand. This errand registers the service broker with the Cloud Controller and makes service plans public. After successfully installing Elastic Runtime, but before making Elastic Runtime available for use, Ops Manager runs the **Broker Registrar** post-install errand.

If an operator deselects the checkbox for the **Broker Registrar** errand before installing Elastic Runtime, the service broker is not registered with the Cloud Controller, and the service plans are not made public.

Pre-Delete Errand Example

Pivotal MySQL has a **Broker Deregistrar** pre-delete errand. This errand:

- Purges the service offering
- Purges all service instances
- Purges all application bindings
- Deletes the service broker from the Cloud Controller

When an operator chooses to delete the Pivotal MySQL product, Ops Manager first runs the **Broker Deregistrar** pre-delete errand, then deletes the product.

Ops Manager does not display the **Broker Deregistrar** pre-delete errand, and an operator cannot prevent the errand from running when Pivotal MySQL is deleted.

Decrypting and Encrypting Installation Files

This topic is intended for product teams who package, distribute, and upgrade [Pivotal Cloud Foundry](#) (PCF) products across releases.

This topic describes the installation and product template YAML files in your Pivotal Cloud Foundry Operations Manager VM. It also provides instructions on viewing and modifying the installation YAML files so that product teams can migrate content.

Understanding Ops Manager Installation and Product Template Files

During the installation process, Ops Manager combines information from the installation and product template YAML files to generate the manifests that define your deployment.

Installation files: PCF stores user-entered data and automatically generated values for Ops Manager in installation files. PCF encrypts and stores these files in the following IaaS-specific directory:

- **AWS:** `/var/ubuntu/workspaces/default`
- **VMWare IaaS Product:** `/var/tempest/workspaces/default`

You must decrypt them to view the contents, edit the files as necessary for your content migration, then re-encrypt them.

Product templates: Ops Manager uses product templates to create forms and obtain user input. The `job_types` and `property_blueprint` key-value pairs in a product template determine how the `jobs` and `properties` sections display in the installation file. Ops Manager stores product templates in the following IaaS-specific directory:

- **AWS:** `/var/ubuntu/meta-data`
- **VMWare IaaS Product:** `/var/tempest/workspaces/default/metadata`

You can edit these files. User input does not alter these files.

Decrypting and Encrypting Installation Files

1. Clone the GitHub `pivotal-cf encrypt-decrypt scripts` repo:

```
git clone https://github.com/pivotal-cf/encrypt-decrypt-scripts
```

This repo contains the following files:

- `decrypt-ops.sh`
- `encrypt-ops.sh`
- `eos.rb`

2. Copy the files from your local machine to your Ops Manager VM.

Replace `LOCAL_USER@LOCAL_SERVER` with the username and the IP address of your machine.

Use the correct `REMOTE_USER` name for your IaaS, as follows:

- **AWS:** `ubuntu`
- **VMWare IaaS Product:** `tempest`

```
$ ssh LOCAL_USER@LOCAL_SERVER
$ scp ~/PATH_TO_FILES/*.sh REMOTE_USER@REMOTE_SERVER:/home/REMOTE_USER
$ scp ~/PATH_TO_FILES/eos.rb REMOTE_USER@REMOTE_SERVER:/home/REMOTE_USER
```

3. Log in to the Ops Manager VM using SSH and the correct username for your IaaS.

AWS: Replace `REMOTE_SERVER` with the IP address for your Ops Manager VM. When prompted, provide your `ops_manager` SSH key information that you created when configuring AWS components for PCF.

```
$ ssh ubuntu@REMOTE_SERVER
```

VMWare IaaS Product: Replace `REMOTE_SERVER` with the IP address for your Ops Manager VM. Also provide the

admin password that you set when you installed Ops Manager to your vSphere, vCloud Air, or vCloud IaaS. For more information, refer to the following:

- vCloud Air and vCloud: Refer to the step on defining password information for the Ops Manager vApp in the [Complete the vApp Wizard and Deploy Ops Manager](#) section of the *Deploying Operations Manager to vCloud Air and vCloud* topic.
- vSphere: Refer to the step on defining password information for the VM admin user in the [Deploying Operations Manager to vSphere](#) topic.

```
$ ssh tempest@REMOTE_SERVER  
tempest@REMOTE_SERVER's password:
```

4. Use `sudo cp` to copy the files to the protected `/var/REMOTE_USER/workspaces/default` directory.

```
$ sudo cp *.sh /var/REMOTE_USER/workspaces/default  
$ sudo cp eos.rb /var/REMOTE_USER/workspaces/default
```

5. In the `/var/REMOTE_USER/workspaces/default` directory, change the permissions of the `decrypt-ops.sh` and `encrypt-ops.sh` scripts.

```
$ sudo chmod 755 decrypt.sh  
$ sudo chmod 755 encrypt.sh
```

6. In the `/var/REMOTE_USER/workspaces/default` directory, run the `decrypt` command to create the `decrypted-installation.yml` and `decrypted-installed.yml` files.

Replace `PASSWORD` with your Ops Manager `REMOTE_USER` UserID password.

```
$ sudo ./decrypt.sh PASSWORD
```

This script does the following:

- Searches the `/var/REMOTE_USER/workspaces/default` for the installation YAML files and creates decrypted copies.
- Creates a directory `decrypted` and stores the decrypted files in this directory.

7. Review and edit the files as necessary for your content migration.

8. Run the `encrypt` command on the modified decrypted files.

Replace `PASSWORD` with your Ops Manager `REMOTE_USER` UserID password.

```
$ sudo ./encrypt.sh PASSWORD
```

This script encrypts the decrypted files and stores the resulting output in the `/var/REMOTE_USER/workspaces/default` directory, which overwrites the existing `installation.yml` file and adds a new file, `installed-installation.yml`. `encrypt-ops.sh` does not overwrite the `actual-installation.yml` file.

 **Note:** Ensure that you back up the edited and encrypted `installation.yml` and `installed-installation.yml` files for future reference.

Overview of the Loggregator System

Loggregator is the next generation system for aggregating and streaming logs and metrics from all of the user apps and system components in an Elastic Runtime deployment.

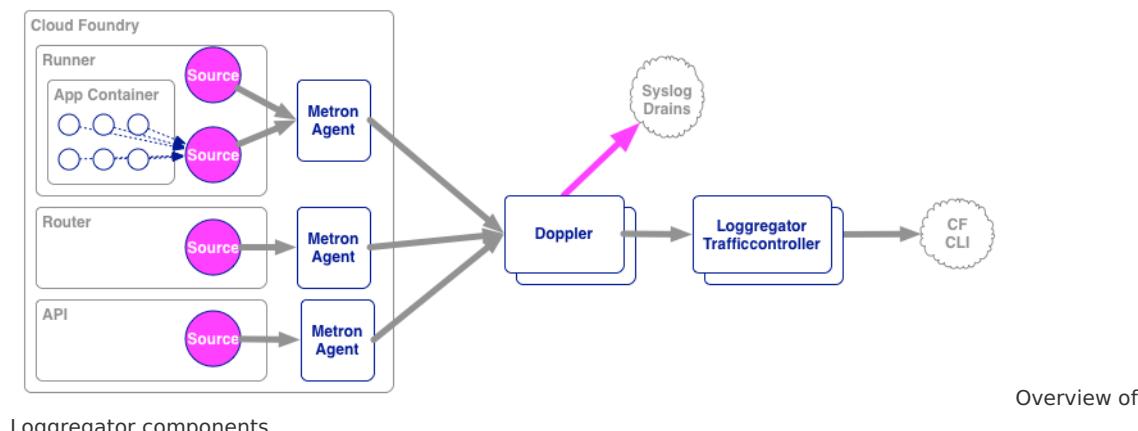
[Loggregator on GitHub](#)

Using Loggregator

The main use cases are as follows:

- App developers can tail their application logs or dump the recent logs from the CF CLI, or stream these to a third party log archive and analysis service.
- Operators and administrators can access the **Loggregator Firehose**, the combined stream of logs from all apps, plus metrics data from CF components.
- Operators can deploy ‘nozzles’ to the Firehose. A nozzle is a component that listens to the Firehose for specified events and metrics and streams this data to external services.

Loggregator Components



Source

Sources are logging agents that run on the Cloud Foundry components.

Metron

Metron agents are co-located with sources. They collect logs and forward them to the Doppler servers.

Doppler

Dopplers gather logs from the Metron agents, store them in temporary buffers, and forward them to the Traffic Controller or to third party syslog drains.

Traffic Controller

Handles client requests for logs. Gathers and collates messages from all Doppler servers, and provides external API and message translation (as needed for legacy APIs). Exposes the Firehose.

Firehose

The Firehose is a websocket endpoint which streams all the event data coming from an Elastic Runtime deployment, including logs, HTTP events and container metrics from all applications. It also includes metrics from all Elastic Runtime

system components and any errors in the system.

Because the data coming from the Firehose may contain sensitive information (for example, customer information in the application logs), the Firehose is only accessible by users who have the right permissions.

The Traffic Controller serves the Firehose over websocket at the `/firehose` endpoint. The events coming out of the Firehose are formatted as protobuf messages conforming to the [dropsonde protocol](#).

The address of the traffic controller can be discovered by hitting the info endpoint on the API and getting the value of the `doppler_logging_endpoint`.

Example output for a BOSH Lite CF environment:

```
$ cf curl /v2/info | jq .doppler_logging_endpoint
wss://doppler.10.244.0.34.xip.io:443
```

Nozzles

Nozzles are programs which consume data from the Loggregator Firehose. Nozzles can be configured to select, buffer, and transform data, and forward it to other applications and services. For example:

- The [Datadog nozzle](#) publishes metrics coming from the Firehose to Datadog.
- The [Syslog nozzle](#) filters out log messages coming from the Firehose and sends it to a syslog server.

See our [Nozzle Tutorial](#).

Loggregator Guide for Cloud Foundry Operators

This topic contains information about the [Loggregator System](#) that may be of use to operators of Cloud Foundry deployments.

Scaling Loggregator

[Dopplers](#) are components that take in log and metric data from Elastic Runtime system components and store this data in a buffer before periodically forwarding it to the [Traffic Controller](#), which serves up the aggregated data stream through the Firehose WebSocket endpoint. When input from [Metron](#) agents exceeds a doppler's buffer size for a given interval, log and metric data can be lost. There are several ways to minimize this loss.

Increase buffer size

1. In the [PCF Ops Manager Installation Dashboard](#), click the **Elastic Runtime** tile.
2. Select **System Logging**.
3. Increase the drain buffer size to prevent loss of log data.
4. Click **Save**.
5. Click **Apply Changes**.

The screenshot shows the PCF Ops Manager Installation Dashboard with the 'Elastic Runtime' tile selected. Under 'System Logging', the 'Logs' tab is active. On the left is a sidebar with checkboxes for various configuration sections: Assign Networks, Assign Availability Zones, System Database Config, File Storage Config, IPs and Ports, Security Config, MySQL Proxy Config, Cloud Controller, System Logging (which is highlighted), SSO Config, LDAP Config, and SMTP Config. The main panel displays configuration options for 'System Logging'. It includes fields for 'External Syslog Aggregator Hostname' (with a placeholder 'localhost'), 'External Syslog Aggregator Port' (with a placeholder '514'), 'External Syslog Network Protocol' (with a dropdown menu showing 'TCP' and 'UDP'), and 'Syslog Drain Buffer Size (# of messages)' (set to '100'). A 'Save' button is at the bottom of the form.

Add additional Doppler instances

1. In the [PCF Ops Manager Installation Dashboard](#), click the **Elastic Runtime** tile.
2. Select **Resource Config**.
3. Increase the number in the **Instances** column and the **Doppler Server** row.

<input checked="" type="checkbox"/> Resource Config	Doppler Server	1	2048
	Loggregator Trafficcontroller	1	2048

4. Click **Save**.
5. Click **Apply Changes**.

Add additional Traffic Controller instances

1. In the **PCF Ops Manager Installation Dashboard**, click the **Elastic Runtime** tile.
2. Select **Resource Config**.
3. Increase the number in the **Instances** column and the **Loggregator Trafficcontroller** row.

<input checked="" type="checkbox"/> Resource Config	Doppler Server	1	2048
	Loggregator Trafficcontroller	1	2048

4. Click **Save**.
5. Click **Apply Changes**.

Scaling Nozzles

Scale [nozzles](#) by using the subscription ID, which is specified when the nozzle connects to the Firehose. If you use the same subscription ID on each nozzle instance, the Firehose will evenly distribute events across all instances of the nozzle. For example, if you have two nozzles with the same subscription ID, then half the events will go to one nozzle and half to the other. Similarly, if there were three instances of the nozzle, then each instance would get one-third the traffic.

A stateless nozzle should handle scaling gracefully. If the nozzle buffers or caches the data, the nozzle author must test what happens when the nozzle is scaled up or scaled down.

Slow Nozzle Alerts

The [Traffic Controller](#) alerts nozzles if they are consuming events too slowly. If the nozzle falls behind, Loggregator will alert the nozzle in two ways:

- **TruncatingBuffer** alerts: If the nozzle is consuming messages more slowly than they are being produced, the loggregator system may drop messages. In this case, the loggregator system sends the log message `TB: Output channel too full. Dropped (n) messages`, where "n" is the number of dropped messages. It also emits a CounterEvent with the name `TruncatingBuffer.DroppedMessages`. The nozzle receives both messages from the Firehose, alerting the operator to the performance issue.
- **PolicyViolation** error: The Traffic Controller periodically sends `ping` control messages over the Firehose WebSocket connection. If a client does not respond to a `ping` message with a `pong` message within 30 seconds, the Traffic Controller closes the WebSocket connection with the WebSocket error code "ClosePolicyViolation" (1008). The nozzle should intercept this WebSocket close error, alerting the operator to the performance issue.

An operator can choose to scale her nozzles in response to these alerts, in order to minimize the loss of data.

Managing Syslog Forwarding from Elastic Runtime Components

Syslog data can be forwarded directly from Elastic Runtime components to an external aggregator.

Syslog forwarding for Pivotal Cloud Foundry is managed through the **PCF Ops Manager Installation Dashboard**. To enable syslog forwarding:

1. Click the **Elastic Runtime** tile.

2. Select **System Logging**.
3. Enter the Hostname, Port and Protocol for your third part log management service.
4. (Optional) Increase the drain buffer size to prevent loss of log data.
5. Click **Save**.
6. Click **Apply Changes**.

The screenshot shows the PCF Ops Manager interface for Pivotal Elastic Runtime. The top navigation bar includes a logo, the title "PCF Ops Manager", and a user identifier "pivotalcf". Below the header, a breadcrumb trail shows "Installation Dashboard < Pivotal Elastic Runtime". A navigation menu on the left lists several configuration sections: "Assign Networks", "Assign Availability Zones", "System Database Config", "File Storage Config", "IPs and Ports", "Security Config", "MySQL Proxy Config", "Cloud Controller", "System Logging" (which is highlighted in grey), "SSO Config", "LDAP Config", and "SMTP Config". The main content area is titled "Configure system logging. Leave the External Syslog fields blank unless you wish to use an external syslog server." It contains four input fields: "External Syslog Aggregator Hostname" (empty), "External Syslog Aggregator Port" (empty), "External Syslog Network Protocol" (dropdown menu open), and "Syslog Drain Buffer Size (# of messages)" with a value of "100" entered. A blue "Save" button is located at the bottom of the form.

Application Logging in Cloud Foundry

This page assumes you are using cf CLI v6.

Loggregator, the Cloud Foundry component responsible for logging, provides a stream of log output from your application and from Cloud Foundry system components that interact with your app during updates and execution.

By default, Loggregator streams logs to your terminal. If you want to persist more than the limited amount of logging information that Loggregator can buffer, you can drain logs to a third-party log management service. See [Third-Party Log Management Services](#).

Cloud Foundry gathers and stores logs in a best-effort manner. If a client is unable to consume log lines quickly enough, the Loggregator buffer may need to overwrite some lines before the client has consumed them. A syslog drain or a CLI tail can usually keep up with the flow of application logs.

Contents of a Log Line

Every log line contains four fields:

1. Timestamp
2. Log type (origin code)
3. Channel: either `STDOUT` or `STDERR`
4. Message

Loggregator assigns the timestamp when it receives log data. The log data is opaque to Loggregator, which simply puts it in the message field of the log line. Applications or system components sending log data to Loggregator may include their own timestamps, which then appear in the message field.

Origin codes distinguish the different log types. Origin codes from system components have three letters. The application origin code is `APP` followed by slash and a digit that indicates the application instance.

Many frameworks write to an application log that is separate from `STDOUT` and `STDERR`. This is not supported by Loggregator. Your application must write to `STDOUT` or `STDERR` for its logs to be included in the Loggregator stream. Check the buildpack your application uses to determine whether it automatically insures that your application correctly writes logs to `STDOUT` and `STDERR` only. Some buildpacks do this, and some do not.

Log Types and Their Messages

Different types of logs have different message formats, as shown in the examples below.

API

Users make API calls to request changes in application state. Cloud Controller, the Cloud Foundry component responsible for the API, logs the actions that Cloud Controller takes in response.

For example:

```
2014-02-13T11:44:52.11-0800 [API]      OUT Updated app with guid e1ca6390-cf78-4fc7-9d86-5b7ed01e9c28 ({"instances":>2})
```

STG

The Droplet Execution Agent emits STG logs when staging or restaging an app. These actions implement the desired state requested by the user. Once the droplet has been uploaded, STG messages end and DEA messages begin.

For example:

```
2014-02-07T10:54:36.80-0800 [STG]      OUT -----> Downloading and installing node
```

DEA

The Droplet Execution Agent emits DEA logs beginning when it starts or stops the app. These actions implement the desired state requested by the user. The DEA also emits messages when an app crashes.

For example:

```
2014-02-13T11:44:52.07-0800 [DEA]      OUT Starting app instance (index 1) with guid e1ca6390-cf78-4fc7-9d86-5b7ed01e9c28
```

RTR

The Router emits RTR logs when it routes HTTP requests to the application. Router messages include the application name followed by a Router timestamp and then selections from the HTTP request.

For example:

```
2014-02-13T11:42:31.96-0800 [RTR]      OUT nifty-gui.example.com - [13/02/2014:19:42:31 +0000]
"GET /favicon.ico HTTP/1.1" 404 23 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36" 10.10.2.142:6609 response_time:0.004092262
app_id:e1ca6390-cf78-4fc7-9d86-5b7ed01e9c28
```

LGR

Loggregator emits LGR to indicate problems with the logging process. Examples include “can’t reach syslog drain url” and “dropped log messages due to high rate.”

APP

Every application emits logs according to choices by the developer. The digit appended to the code indicates app instance: 0 is the first instance, 1 is the second, and so on.

For example:

```
2014-02-13T11:44:27.71-0800 [App/0]      OUT Express server started
```

Writing to the Log from your Application

Your application must write logs to `STDERR` or `STDOUT`. Both are typically buffered, and you should flush the buffer before delivering the message to Loggregator.

Alternatively, you can write log messages to `STDERR` or `STDOUT` synchronously. This approach is mainly used for debugging because it may affect application performance.

Viewing Logs in the Command Line Interface

You view logs in the CLI using the `cf logs` command. You can tail, dump, or filter log output.

Tailing Logs

`cf logs APP_NAME` streams Loggregator output to the terminal.

For example:

```
$ cf logs nifty-gui
Connected, tailing logs for app nifty-gui in org janclouduser / space jancloudspace as admin...
2014-02-06T12:00:19.44-0800 [API]
    OUT Updated app with guid c8612fc2-85b1-464c-92f5-d4a1156eacbf
    {"route":>"2ef5796b-475a-4615-9c71-75bbe277022e"}
2014-02-06T12:00:27.54-0800 [DEA]
    OUT Got staging request for app with id
    c8612fc2-85b1-464c-92f5-d4a1156eacbf
2014-02-06T12:00:28.51-0800 [API]
    OUT Updated app with guid c8612fc2-85b1-464c-92f5-d4a1156eacbf
    {"state":>"STARTED"}
...
...
```

Use **Ctrl-C** (^C) to exit the real-time stream.

Dumping Logs

`cf logs APP_NAME --recent` displays all the lines in the Loggregator buffer.

Filtering Logs

To view some subset of log output, use `cf logs` in conjunction with filtering commands of your choice. In the example below, `grep -v` excludes all Router logs:

```
$ cf logs nifty-gui --recent | grep -v RTR
Connected, dumping recent logs for app nifty-gui in org jancloudspace-org / space development
as jancloudspace@example.com...
2014-02-07T10:54:40.41-0800 [STG]
    OUT -----> Uploading droplet (5.6M)
2014-02-07T10:54:44.44-0800 [DEA]
    OUT Starting app instance (index 0) with guid
    4d397313-20e0-478a-9a74-307446eb7640
2014-02-07T10:54:46.31-0800 [App/0]
    OUT Express server started
2014-02-07T10:57:53.60-0800 [API]
    OUT Updated app with guid 4d397313-20e0-478a-9a74-307446eb7640
    {"instances":>2}
2014-02-07T10:57:53.64-0800 [DEA]
    OUT Starting app instance (index 1) with guid
    4d397313-20e0-478a-9a74-307446eb7640
2014-02-07T10:57:55.88-0800 [App/1]
    OUT Express server started
...
...
```

Cloud Foundry System Metrics

Metrics by Component

Cloud Controller

Metric Name	Description
vcap_sinatra.requests.outstanding	Number of requests currently being processed
vcap_sinatra.requests.completed	Number of requests that have been completed
vcap_sinatra.http_status.[status_code]	Metrics on number of http response codes for all requests
vcap_sinatra.recent_errors	50 most recent errors
log_counts.[severity]	Metrics for number of log messages of various severity
cc_user_count	"Total number of users ever created (includes inactive users). Note: this field is updated at least every 10 minutes."
cc_job_queue_length.[queue_type]	"Number of delayed_jobs that have yet to run for the first time. Note: this field is updated at least once every 30 seconds"
cc_job_queue_length.total	Total number of delayed_jobs that have yet to run for the first time
thread_info.[rest below here]	Note: these fields are updated at least once every 30 seconds
thread_info.thread_count	Total number of threads that are either runnable or stopped
thread_info.event_machine.connection_count	Number of open connections to event machine
thread_info.event_machine.threadqueue.size	Number of unscheduled tasks in the threadqueue
thread_info.event_machine.threadqueue.num_waiting	Number of scheduled tasks in the threadqueue
thread_info.event_machine.resultqueue.size	Number of unscheduled tasks in the result
thread_info.event_machine.resultqueue.num_waiting	Number of scheduled tasks in the result
cc_failed_job_count.[queue_type]	"Number of failed jobs sorted by queue type. Updated once every 30 seconds. Note: this is unrelated to the worker"
cc_failed_job_count.total	Total number of failed jobs
uptime_in_seconds	"The uptime of the process
uptime	"The uptime of the process
mem_bytes	RSS bytes calculated by the OS
mem_used_bytes	total memory used according to the OS
mem_free_bytes	total memory available according to the OS
cpu_load_avg	system cpu load average over a minute according to the OS
cpu	the percent cpu usage by the process
num_cores	number of CPUs of the host machine

[Top](#)

Dea Logging Agent

Metric Name	Description
dea_logging_agent.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
dea_logging_agent.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
dea_logging_agent.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
dea_logging_agent.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator

Metric Name	Description
dea_logging_agent.memoryStats.numFrees	Lifetime number of memory deallocations
dea_logging_agent.memoryStats.numMallows	Lifetime number of memory allocations
dea_logging_agent.numCPUS	Number of CPUs on the machine
dea_logging_agent.numGoRoutines	Instantaneous number of active Goroutines in the Doppler process
dea_logging_agent.totalApps	The number of applications which the DEA logging agent is hooked onto

[Top](#)

Diego

Metric Name	Description
auctioneer.AuctioneerFetchStatesDuration	
auctioneer.AuctioneerLRPAuctionsFailed	
auctioneer.AuctioneerLRPAuctionsStarted	
auctioneer.AuctioneerTaskAuctionsFailed	
auctioneer.AuctioneerTaskAuctionsStarted	
auctioneer.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
auctioneer.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
auctioneer.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
auctioneer.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
auctioneer.memoryStats.numFrees	Lifetime number of memory deallocations
auctioneer.memoryStats.numMallows	Lifetime number of memory allocations
auctioneer.numCPUS	Number of CPUs on the machine
auctioneer.numGoRoutines	Instantaneous number of active Goroutines in the process
bbs.ConvergenceLRPRuns	
bbs.ConvergenceLRPDuration	
bbs.ConvergenceLRPPreProcessingDesiredLRPsDeleted	
bbs.ConvergenceLRPPreProcessingActualLRPsDeleted	
bbs.ConvergenceLRPsDeleted	
bbs.LRPsDesired	
bbs.LRPsStarting	
bbs.LRPsRunning	
bbs.CrashedActualLRPs	
bbs.CrashingDesiredLRPs	
bbs.ETCDLeader	
bbs.ETCDFollowers	
bbs.ETCDReceivedBandwidthRate	
bbs.ETCDSentBandwidthRate	
bbs.ETCDReceivedRequestRate	
bbs.ETCDSentRequestRate	
bbs.ETCDRaftTerm	
bbs.ETCDWatchers	
bbs.ConvergenceTaskRuns	

Metric Name	Description
bbs.ConvergenceTaskDuration	
bbs.ConvergenceTasksKicked	
bbs.ConvergenceTasksPruned	
bbs.TasksPending	
bbs.TasksRunning	
bbs.TasksCompleted	
bbs.TasksResolving	
bbs.RequestLatency	
bbs.RequestCount	
bbs.MetricsReportingDuration	
bbs.BBSMasterElected	
bbs.MigrationDuration	
bbs.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
bbs.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
bbs.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
bbs.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
bbs.memoryStats.numFrees	Lifetime number of memory deallocations
bbs.memoryStats.numMallocs	Lifetime number of memory allocations
bbs.numCPUS	Number of CPUs on the machine
bbs.numGoRoutines	Instantaneous number of active Goroutines in the process
cc_uploader.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
cc_uploader.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
cc_uploader.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
cc_uploader.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
cc_uploader.memoryStats.numFrees	Lifetime number of memory deallocations
cc_uploader.memoryStats.numMallocs	Lifetime number of memory allocations
cc_uploader.numCPUS	Number of CPUs on the machine
cc_uploader.numGoRoutines	Instantaneous number of active Goroutines in the process
converger.ConvergenceLRPDuration	
converger.ConvergenceLRPPreProcessingActualLRPsDeleted	
converger.ConvergenceLRPPreProcessingDesiredLRPsDeleted	
converger.ConvergenceLPRRuns	
converger.ConvergenceTaskDuration	
converger.ConvergenceTaskRuns	
converger.ConvergenceTasksKicked	
converger.ConvergenceTasksPruned	
converger.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
converger.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
converger.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use

Metric Name	Description
converger.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
converger.memoryStats.numFrees	Lifetime number of memory deallocations
converger.memoryStats.numMallocs	Lifetime number of memory allocations
converger.numCPUS	Number of CPUs on the machine
converger.numGoRoutines	Instantaneous number of active Goroutines in the process
file_server.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
file_server.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
file_server.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
file_server.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
file_server.memoryStats.numFrees	Lifetime number of memory deallocations
file_server.memoryStats.numMallocs	Lifetime number of memory allocations
file_server.numCPUS	Number of CPUs on the machine
file_server.numGoRoutines	Instantaneous number of active Goroutines in the process
garden_linux.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
garden_linux.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
garden_linux.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
garden_linux.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
garden_linux.memoryStats.numFrees	Lifetime number of memory deallocations
garden_linux.memoryStats.numMallocs	Lifetime number of memory allocations
garden_linux.numCPUS	Number of CPUs on the machine
garden_linux.numGoRoutines	Instantaneous number of active Goroutines in the process
nsync_bulker.DesiredLRPSyncDuration	
nsync_bulker.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
nsync_bulker.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
nsync_bulker.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
nsync_bulker.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
nsync_bulker.memoryStats.numFrees	Lifetime number of memory deallocations
nsync_bulker.memoryStats.numMallocs	Lifetime number of memory allocations
nsync_bulker.numCPUS	Number of CPUs on the machine
nsync_bulker.numGoRoutines	Instantaneous number of active Goroutines in the process
nsync_listener.LRPsDesired	
nsync_listener.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
nsync_listener.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
nsync_listener.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use

Metric Name	Description
nsync_listener.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
nsync_listener.memoryStats.numFrees	Lifetime number of memory deallocations
nsync_listener.memoryStats.numMallocs	Lifetime number of memory allocations
nsync_listener.numCPUS	Number of CPUs on the machine
nsync_listener.numGoRoutines	Instantaneous number of active Goroutines in the process
receptor.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
receptor.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
receptor.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
receptor.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
receptor.memoryStats.numFrees	Lifetime number of memory deallocations
receptor.memoryStats.numMallocs	Lifetime number of memory allocations
receptor.numCPUS	Number of CPUs on the machine
receptor.numGoRoutines	Instantaneous number of active Goroutines in the process
rep.CapacityRemainingContainers	
rep.CapacityRemainingDisk	
rep.CapacityRemainingMemory	
rep.CapacityTotalContainers	
rep.CapacityTotalDisk	
rep.CapacityTotalMemory	
rep.ContainerCount	
rep.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
rep.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
rep.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
rep.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
rep.memoryStats.numFrees	Lifetime number of memory deallocations
rep.memoryStats.numMallocs	Lifetime number of memory allocations
rep.numCPUS	Number of CPUs on the machine
rep.numGoRoutines	Instantaneous number of active Goroutines in the process
rep.RepBulkSyncDuration	
route_emitter.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
route_emitter.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
route_emitter.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
route_emitter.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
route_emitter.memoryStats.numFrees	Lifetime number of memory deallocations
route_emitter.memoryStats.numMallocs	Lifetime number of memory allocations
route_emitter.numCPUS	Number of CPUs on the machine

Metric Name	Description
route_emitter.numGoRoutines	Instantaneous number of active Goroutines in the process
route_emitter.RouteEmitterSyncDuration	
route_emitter.RoutesRegistered	
route_emitter.RoutesSynced	
route_emitter.RoutesTotal	
route_emitter.RoutesUnregistered	
ssh_proxy.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
ssh_proxy.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
ssh_proxy.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
ssh_proxy.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
ssh_proxy.memoryStats.numFrees	Lifetime number of memory deallocations
ssh_proxy.memoryStats.numMallocs	Lifetime number of memory allocations
ssh_proxy.numCPUS	Number of CPUs on the machine
ssh_proxy.numGoRoutines	Instantaneous number of active Goroutines in the process
stager.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
stager.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
stager.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
stager.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
stager.memoryStats.numFrees	Lifetime number of memory deallocations
stager.memoryStats.numMallocs	Lifetime number of memory allocations
stager.numCPUS	Number of CPUs on the machine
stager.numGoRoutines	Instantaneous number of active Goroutines in the process
stager.StagingRequestFailedDuration	
stager.StagingRequestsFailed	
stager.StagingRequestsSucceeded	
stager.StagingRequestSucceededDuration	
stager.StagingStartRequestsReceived	
tps_listener.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
tps_listener.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
tps_listener.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
tps_listener.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
tps_listener.memoryStats.numFrees	Lifetime number of memory deallocations
tps_listener.memoryStats.numMallocs	Lifetime number of memory allocations
tps_listener.numCPUS	Number of CPUs on the machine
tps_listener.numGoRoutines	Instantaneous number of active Goroutines in the process
tps_watcher.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds

Metric Name	Description
tps_watcher.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
tps_watcher.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
tps_watcher.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
tps_watcher.memoryStats.numFrees	Lifetime number of memory deallocations
tps_watcher.memoryStats.numMallocs	Lifetime number of memory allocations
tps_watcher.numCPUS	Number of CPUs on the machine
tps_watcher.numGoRoutines	Instantaneous number of active Goroutines in the process

[Top](#)

Doppler

Metric Name	Description
DopplerServer.dropsondeListener.currentBufferCount	Instantaneous number of messages read by UDP socket but not yet unmarshalled
DopplerServer.dropsondeListener.receivedByteCount	Lifetime number of bytes read by UDP socket
DopplerServer.dropsondeListener.receivedMessageCount	Lifetime number of messages read by UDP socket
DopplerServer.dropsondeUnmarshaller.containerMetricReceived	Lifetime number of ContainerMetric messages unmarshalled
DopplerServer.dropsondeUnmarshaller.counterEventReceived	Lifetime number of CounterEvent messages unmarshalled
DopplerServer.dropsondeUnmarshaller.errorReceived	Lifetime number of Error messages unmarshalled
DopplerServer.dropsondeUnmarshaller.heartbeatReceived	Lifetime number of Heartbeat messages unmarshalled
DopplerServer.dropsondeUnmarshaller.httpStartReceived	Lifetime number of HttpStart messages unmarshalled
DopplerServer.dropsondeUnmarshaller.httpStartStopReceived	Lifetime number of HttpStartStop messages unmarshalled
DopplerServer.dropsondeUnmarshaller.httpStopReceived	Lifetime number of HttpStop messages unmarshalled
DopplerServer.dropsondeUnmarshaller.logMessageTotal	Lifetime number of LogMessage messages unmarshalled
DopplerServer.dropsondeUnmarshaller.unmarshalErrors	Lifetime number of errors when unmarshalling messages
DopplerServer.dropsondeUnmarshaller.valueMetricReceived	Lifetime number of ValueMetric messages unmarshalled
DopplerServer.httpServer.receivedMessages	Number of messages received by Doppler's internal MessageRouter
DopplerServer.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
DopplerServer.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
DopplerServer.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
DopplerServer.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
DopplerServer.memoryStats.numFrees	Lifetime number of memory deallocations
DopplerServer.memoryStats.numMallocs	Lifetime number of memory allocations
DopplerServer.messageRouter.numberOfDumpSinks	Instantaneous number of dump sinks known to the SinkManager

Metric Name	Description
DopplerServer.messageRouter.numberOfFirehoseSinks	Instantaneous number of firehose sinks known to the SinkManager
DopplerServer.messageRouter.numberOfSyslogSinks	Instantaneous number of syslog sinks known to the SinkManager
DopplerServer.messageRouter.numberOfWebSocketSinks	Instantaneous number of websocket sinks known to the SinkManager
DopplerServer.messageRouter.totalDroppedMessages	Lifetime number of messages dropped inside Doppler for various reasons (downstream consumer can't keep up internal object wasn't ready for message, etc.)
DopplerServer.numCPUs	Number of CPUs on the machine
DopplerServer.numGoRoutines	Instantaneous number of active Goroutines in the Doppler process
DopplerServer.signatureVerifier.invalidSignatureErrors	Lifetime number of messages received with an invalid signature
DopplerServer.signatureVerifier.missingSignatureErrors	Lifetime number of messages received that are too small to contain a signature
DopplerServer.signatureVerifier.validSignatures	Lifetime number of messages received with valid signatures

[Top](#)

Etcd

Visit [etcd stats API](#)

Metric Name	Description
etcd.leader.Followers	number of etcd followers
etcd.leader.Latency	current latency to a specific follower
etcd.server.IsLeader	1 if the current server is the leader, 0 if it is a follower
etcd.server.ReceivedAppendRequests	number of append requests this node has processed
etcd.server.ReceivingBandwidthRate	number of bytes per second this node is receiving (follower only)
etcd.server.ReceivingRequestRate	number of requests per second this node is receiving (follower only)
etcd.server.SendingBandwidthRate	number of bytes per second this node is sending (leader only). This value is undefined on single member clusters.
etcd.server.SendingRequestRate	number of requests per second this node is sending (leader only). This value is undefined on single member clusters.
etcd.server.SentAppendRequests	number of requests that this node has sent
etcd.store.RaftIndex	X-Raft-Index value from the /stats/store endpoint
etcd.store.RaftTerm	X-Raft-Term value from the /stats/store endpoint
etcd.store.EtcdIndex	X-Etcd-Index value from the /stats/store endpoint
etcd.store.CompareAndDeleteFail	CompareAndDeleteFail operation count
etcd.store.CompareAndDeleteSuccess	CompareAndDeleteSuccess operation count
etcd.store.CompareAndSwapFail	CompareAndSwapFail operation count
etcd.store.CompareAndSwapSuccess	CompareAndSwapSuccess operation count
etcd.store.CreateFail	CreateFail operation count
etcd.store.CreateSuccess	CreateSuccess operation count
etcd.store.DeleteFail	DeleteFail operation count
etcd.store.DeleteSuccess	DeleteSuccess operation count

Metric Name	Description
etcd.store.ExpireCount	ExpireCount operation count
etcd.store.GetsFail	GetsFail operation count
etcd.store.GetsSuccess	GetsSuccess operation count
etcd.store.SetsFail	SetsFail operation count
etcd.store.SetsSuccess	SetsSuccess operation count
etcd.store.UpdateFail	UpdateFail operation count
etcd.store.UpdateSuccess	UpdateSuccess operation count
etcd.store.Watchers	Watchers operation count

[Top](#)

Hm9000

Metric Name	Description
HM9000.numCpus	Number of CPUs for the HM9000 box with the lock
HM9000.numGoRoutines	Number of go routines running on the metrics server on the HM9000 with the lock
HM9000.HM9000.StartCrashed	Increments each time HM9000 tries to start a crashed instance
HM9000.HM9000.StartEvacuating	Increments each time HM9000 is evacuating an instance and sends an immediate start and sometimes stop
HM9000.HM9000.StartMissing	Increments each time HM9000 starts an instance that is missing
HM9000.HM9000.StopDuplicate	Increments each time a stop is scheduled for a running instance at a duplicated index
HM9000.HM9000.StopEvacuationComplete	Increments each time HM9000 is done evacuating an instance and stops the instance

[Top](#)

Metron Agent

Metric Name	Description
MessageAggregator.counterEventReceived	Lifetime number of CounterEvents aggregated in Metron
MessageAggregator.httpStartReceived	Lifetime number of HTTPStart aggregated in Metron
MessageAggregator.httpStartStopEmitted	Lifetime number of HTTPStartStop events emitted by Metron (created by combining HTTPStart and HTTPStop events)
MessageAggregator.httpStopReceived	Lifetime number of HTTPStop aggregated in Metron
MessageAggregator.httpUnmatchedStartReceived	Lifetime number of HTTPStart events for which no HTTPStop was received
MessageAggregator.httpUnmatchedStopReceived	Lifetime number of HTTPStop events for which no HTTPStart was received
MessageAggregator.uncategorizedEvents	Lifetime number of non-(CounterEvent HTTPStart HTTPStop) events processed by aggregator
dropsondeAgentListener.currentBufferCount	Instantaneous number of Dropsonde messages read by UDP socket but not yet unmarshalled
dropsondeAgentListener.receivedByteCount	Lifetime number of bytes of Dropsonde messages read by UDP socket
dropsondeAgentListener.receivedMessageCount	Lifetime number of Dropsonde messages read by UDP socket
dropsondeMarshaller.containerMetricMarshalled	Lifetime number of ContainerMetric messages marshalled
dropsondeMarshaller.counterEventMarshalled	Lifetime number of CounterEvent messages marshalled
dropsondeMarshaller.errorMarshalled	Lifetime number of Error messages marshalled
dropsondeMarshaller.heartbeatMarshalled	Lifetime number of Heartbeat messages marshalled

Metric Name	Description
dropsondeMarshaller.httpStartMarshalled	Lifetime number of HttpStart messages marshalled
dropsondeMarshaller.httpStartStopMarshalled	Lifetime number of HttpStartStop messages marshalled
dropsondeMarshaller.httpStopMarshalled	Lifetime number of HttpStop messages marshalled
dropsondeMarshaller.logMessageMarshalled	Lifetime number of LogMessage messages marshalled
dropsondeMarshaller.marshalErrors	Lifetime number of errors when marshalling messages
dropsondeMarshaller.valueMetricMarshalled	Lifetime number of ValueMetric messages marshalled
dropsondeUnmarshaller.containerMetricReceived	Lifetime number of ContainerMetric messages unmarshalled
dropsondeUnmarshaller.counterEventReceived	Lifetime number of CounterEvent messages unmarshalled
dropsondeUnmarshaller.errorReceived	Lifetime number of Error messages unmarshalled
dropsondeUnmarshaller.heartbeatReceived	Lifetime number of Heartbeat messages unmarshalled
dropsondeUnmarshaller.httpStartReceived	Lifetime number of HttpStart messages unmarshalled
dropsondeUnmarshaller.httpStartStopReceived	Lifetime number of HttpStartStop messages unmarshalled
dropsondeUnmarshaller.httpStopReceived	Lifetime number of HttpStop messages unmarshalled
dropsondeUnmarshaller.logMessageTotal	Lifetime number of LogMessage messages unmarshalled
dropsondeUnmarshaller.unmarshalErrors	Lifetime number of errors when unmarshalling messages
dropsondeUnmarshaller.valueMetricReceived	Lifetime number of ValueMetric messages unmarshalled
legacyAgentListener.currentBufferCount	Instantaneous number of Legacy messages read by UDP socket but not yet unmarshalled
legacyAgentListener.receivedByteCount	Lifetime number of bytes of Legacy messages read by UDP socket
legacyAgentListener.receivedMessageCount	Lifetime number of Legacy messages read by UDP socket
memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
memoryStats.numFrees	Lifetime number of memory deallocations
memoryStats.numMallocs	Lifetime number of memory allocations
numCpus	Number of CPUs on the machine
numGoRoutines	Instantaneous number of active Goroutines in the Doppler process

[Top](#)

Syslog Drain Binder

Metric Name	Description
syslog_drain_binder.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
syslog_drain_binder.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
syslog_drain_binder.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
syslog_drain_binder.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
syslog_drain_binder.memoryStats.numFrees	Lifetime number of memory deallocations
syslog_drain_binder.memoryStats.numMallocs	Lifetime number of memory allocations
syslog_drain_binder.numCPUS	Number of CPUs on the machine
syslog_drain_binder.numGoRoutines	Instantaneous number of active Goroutines in the Doppler process
syslog_drain_binder.pollCount	Number of times the syslog drain binder has polled the cloud controller for syslog drain bindings
syslog_drain_binder.totalDrains	Number of syslog drains returned by cloud controller

[Top](#)

Traffic Controller

Metric Name	Description
LoggregatorTrafficController.memoryStats.lastGCPauseTimeNS	Duration of the last Garbage Collector pause in nanoseconds
LoggregatorTrafficController.memoryStats.numBytesAllocated	Instantaneous count of bytes allocated and still in use
LoggregatorTrafficController.memoryStats.numBytesAllocatedHeap	Instantaneous count of bytes allocated on the main heap and still in use
LoggregatorTrafficController.memoryStats.numBytesAllocatedStack	Instantaneous count of bytes used by the stack allocator
LoggregatorTrafficController.memoryStats.numFrees	Lifetime number of memory deallocations
LoggregatorTrafficController.memoryStats.numMallocs	Lifetime number of memory allocations
LoggregatorTrafficController.numCPUS	Number of CPUs on the machine
LoggregatorTrafficController.numGoRoutines	Instantaneous number of active Goroutines in the Doppler process

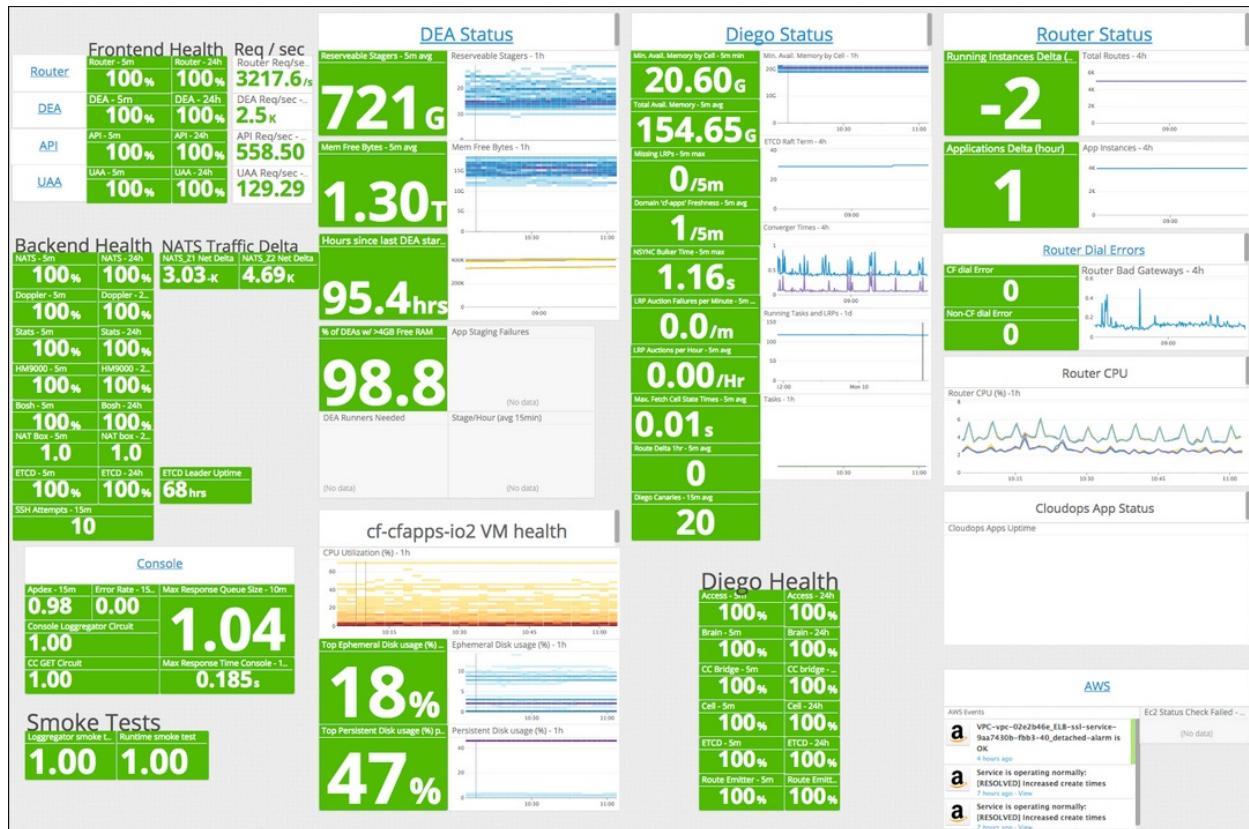
[Top](#)

Key Metrics for Monitoring a Pivotal Cloud Foundry Deployment

The Pivotal Cloud Ops team monitors the health of its Cloud Foundry deployments using a customized Datadog dashboard. This topic describes each of the key metrics as they are rendered in the custom dashboard, and why they are useful for monitoring the health of a Cloud Foundry deployment.

Cloud Ops' practices are tailored to the specific details of the Cloud Foundry deployments they operate. Therefore, the descriptions here are meant to be informative examples rather than general prescriptions. Pivotal recommends that operators experiment with different combinations of metrics and alerts appropriate to their specific requirements.

The Cloud Ops team's custom configuration of Datadog's dashboards, alerts and screenboards can be found in the [Datadog Config repository](#).



BOSH Health Monitor

Backend Health	
NATS - 5m	NATS - 24h
100%	100%
Doppler - 5m	Doppler - 2...
100%	100%
Stats - 5m	Stats - 24h
100%	100%
HM9000 - 5m	HM9000 - 2...
100%	100%
Bosh - 5m	Bosh - 24h
100%	100%
NAT Box - 5m	NAT box - 2...
1.0	1.0
ETCD - 5m	ETCD - 24h
100%	100%

Frontend Health	
Router - 5m	Router - 24h
100%	100%
DEA - 5m	DEA - 24h
100%	100%
API - 5m	API - 24h
100%	100%
UAA - 5m	UAA - 24h
100%	100%

What we monitor	Health, broken down by component. Each row displays the average percentage of healthy instances for the relevant component over the last 5 minutes and 24 hours. For example, suppose your Router has ten instances. If one instance is not healthy, the stoplight turns red and shows 90%.
	We monitor health for the following components:
	<ul style="list-style-type: none"> • Backend: <ul style="list-style-type: none"> ◦ NATS ◦ Doppler ◦ Stats ◦ HM9000 ◦ BOSH ◦ NAT Box ◦ ETCD • Frontend: <ul style="list-style-type: none"> ◦ Router ◦ DEA ◦ API ◦ UAA
Why we monitor it	To ensure all VMs are functioning properly.
System metric	bosh.healthmonitor.system.healthy
Alerts triggered	None
Notes	Alerts generated from this metric go to a buffer queue in our alerting system, Pagerduty. Because BOSH restores systems quickly when they go down, we wait two minutes before forwarding unresolved alerts to our operators.

Requests per Second

What we monitor	Requests per second for each of the following components: <ul style="list-style-type: none">• Router• DEA• API• UAA
Why we monitor it	To track the flow of traffic through the components in the system.
System metric	<code>cf.collector.router.requests(component: app/cloudcontroller/uaa)</code>
Alerts triggered	None
Notes	None

NATS Traffic Delta

What we monitor	Delta of average NATS traffic over the last hour. The displayed metric is the difference between the average NATS traffic over the last 30 minutes and the average NATS traffic over the interval from 90 to 60 minutes prior.
Why we monitor it	To detect significant drops in NATS traffic. A sudden drop might indicate a problem with the health of the NATS VMs.
System metric	<code>aws.ec2.network_in</code>
Alerts triggered	None
Notes	None

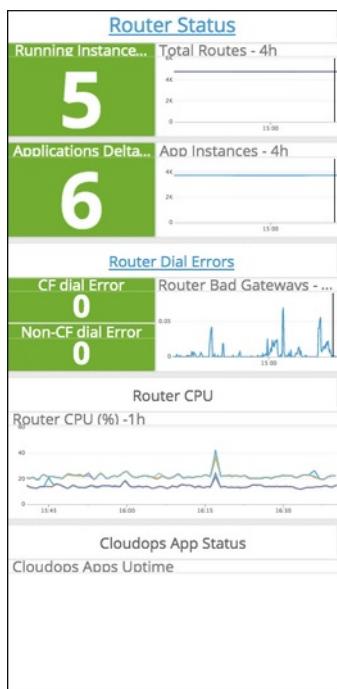
ETCD Leader Uptime

What we monitor	Time since the ETCD leader last was down.
Why we monitor it	When the ETCD leader goes down, it usually indicates a push failure.
System metric	<code>cloudops_tools.etcd_leader_health</code>
Alerts triggered	None
Notes	The <code>cloudops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

SSH Attempts

What we monitor	Total SSH attempts. We log the count of connection attempts to our systems on the SSH port (port 22).
Why we monitor it	A spike in SSH attempts is a good indicator of SSH-cracker attacks.
System metric	<code>cloudops_tools.ssh-abuse-monitor</code>
Alerts triggered	None
Notes	<ul style="list-style-type: none"> • DEAs send their iptables logs to Logsearch. A Cloud Ops internal app polls Logsearch for first packets and pushes the count to Datadog. • The <code>cloudops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

The Router Status Column



App Instance Count

What we monitor	Count of running app instances.
Why we monitor it	Unexpected large fluctuations in app count can indicate malicious user behavior or CF component issues.
System metric	<code>avg:cf.collector.HM9000.HM9000.NumberOfAppsWithAllInstancesReporting</code>
Alerts triggered	running app number change rate
Notes	Spikes in this metric might indicate the need to add more resources. Pay attention to Reservable Stagers and DEA Runners Needed .

Total Routes

What we monitor	Count of routes held by the GoRouter.
Why we monitor it	The count on all routers should be the same. If this differs between routers, it usually indicates a NATS problem.
System metric	<code>cf.collector.router.total_routes</code>
Alerts triggered	prod CF: Number of routes in the router's routing table is too low
Notes	<ul style="list-style-type: none"> We show the delta over the last N min so that an operator can see an absolute change rather than counting on the shape of the graph, which can be misleading. The router are “the” way into all CF systems (both CF components and customer apps). If they go down (or don’t have routes) the system is down. It’s a keystone component and understanding what its metrics tell us is important. This is a metric with a “cliff”: nominal changes are < 20, if we are doing a marketing event we might see ~200. Those are okay. Outside of that we really only see full failure (minus 4000+ routes). There is nothing in between.

Router Dial Errors

What we monitor	5xx from the routers to backend CF components and Apps. There are separate measures for CF Components and User pushed apps.
Why we monitor it	This tells us if we have failures connecting to components.

System metric	<code>avg:cloudops_tools.app_instance_monitor.router.dial.errors{domain:run.pivotal.io} / avg:cloudops_tools.app_instance_monitor.router.dial.errors{cf_component:false}</code>
Alerts triggered	<ul style="list-style-type: none"> No data for router dial errors Router dial errors for <code>console.run.pivotal.io</code> Too many router dial errors for cf components
Notes	<ul style="list-style-type: none"> Any dial errors to CC, UAA, Dopplers and any other BOSH-deployed CF component must be investigated. Same for admin domain apps (<code>*run.pivotal.io</code> in our case). We expect dial errors from our large population of customer app (4000+). People push bad apps, or are running dev iterations, etc and will have 502s. We don't alert on them, we observe that 5xx in the 500/10 min range are normal. If we saw this number to jump to 1000+/10 min, we'd investigate. The <code>cloudops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

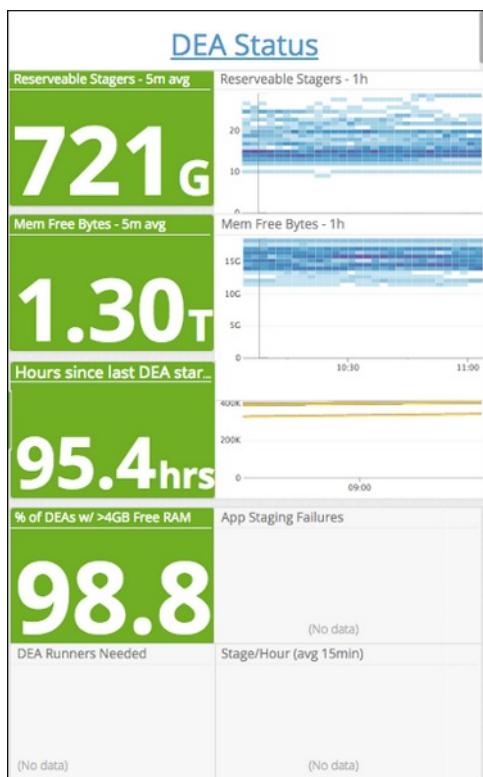
Router CPU

What we monitor	OS-level CPU usage.
Why we monitor it	Routers are multi-threaded and like a lot of CPU, if they are using too much CPU we will scale them via BOSH.
System metric	<code>bosh.healthmonitor.system.cpu.user{deployment:cf-cfapps-io2,job:}</code>
Alerts triggered	None
Notes	Our rule of thumb is: Scaling is easy, if in doubt, add some routers.

AWS Events

What we monitor	Feed from 'aws ec2 events'
Why we monitor it	Tells us about VM, RDS and other things critical to know from our IaaS.
System metric	N/A
Alerts triggered	None
Notes	Only applies to CloudFoundry deployments on AWS.

The DEA Status Column



Reservable Stagers

What we monitor	How many 1GB apps can be staged.
Why we monitor it	Better capacity planning. If the number is too low users will not be able to stage more apps or even scale their apps.
System metric	<code>sum:cf.collector.reserveable_stagers</code>
Alerts triggered	Alerts when there aren't enough reservable staggers.
Notes	Used to calculate DEA runners needed.

Mem Free Bytes

What we monitor	Total free memory on the DEAs
Why we monitor it	Better capacity planning. If the number is too low, users will not be able to stage more apps or scale staged apps.
System metric	<code>cf.collector.mem_free_bytes{job:dea}</code>
Alerts triggered	None
Notes	Used to calculate DEA runners needed.

Hours Since Last DEA Start

What we monitor	Number of hours since the last DEA was started.
Why we monitor it	If DEAs are restarting frequently this can indicate a bug in the DEA. This can directly impact customer-facing performance as it can cause apps to constantly be restaged.
System metric	<code>(min:cf.collector.uptime_in_seconds{job:dea}) / 3600</code>
Alerts triggered	None
Notes	None

Percent of DEAs with At Least 4 GB Free RAM

What we monitor	Percentage of existing DEAs available to stage 4 GB apps.
Why we monitor it	To ensure sure a user can push a 4 GB app.
System metric	<code>avg:clouddops_tools.dea_monitor.reservable_stagers_aggregate * 100</code>
Alerts triggered	None
Notes	The <code>clouddops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

Application Staging Failures

What we monitor	How many app staging requests failed on the runner.
Why we monitor it	To ensure that users are not facing issues staging apps.
System metric	<code>avg:clouddops_tools.failed_stages.counts_per_5min</code>
Alerts triggered	None
Notes	The <code>clouddops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

DEA runners needed

What we monitor	If there are any DEA runners needed.
Why we monitor it	To tell us how many more runners we might need to add.
System metric	<code>avg:clouddops_tools.dea_monitor.additional_runners_needed</code>
Alerts triggered	None
Notes	The <code>clouddops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.

Stages per hour

What we monitor	How many apps are being staged every hour.
Why we monitor it	This metric is useful for statistical purposes and debugging issues with the system. If there are no apps being staged for 15 minutes, we might want to look at the system and see if anything is wrong.
System metric	<code>avg:clouddops_tools.stages.counts_per_5min.rollup(sum,3600)</code>
Alerts triggered	The <code>clouddops_tools</code> metrics are generated by an internal app that the Pivotal Cloud Ops team developed. These metrics are not available on other Cloud Foundry deployments.
Notes	None

Deploying a Nozzle to the Loggregator Firehose

This topic will walk through the procedure for deploying a ‘nozzle’ application to the Cloud Foundry [Loggregator Firehose](#). The Loggregator team has prepared an example nozzle application for use with this tutorial.

The procedure described here will deploy this example nozzle to the Firehose of a Cloud Foundry deployed locally, with BOSH Lite. This is generally the easiest point of entry into Cloud Foundry operations.

Prerequisites:

- [BOSH CLI](#) installed locally.
- Spiff installed locally and added to your shell’s load path. [Spiff on GitHub](#).
- A BOSH Lite deployed locally using VirtualBox. See [BOSH lite on GitHub](#).
- A working [Cloud Foundry](#), including Loggregator, deployed with your local BOSH Lite. This will serve as our source of data. See [Deploying Cloud Foundry using BOSH Lite](#), or use the `provision_cf` script included in the [BOSH Lite release](#).

Note: Even using the automated `provision_cf` script, Deploying Cloud Foundry can take up to several hours, depending on your internet bandwidth.

Step 1: Download Your Cloud Foundry BOSH Manifest

1. Run `bosh deployments` to identify the name of the current BOSH deployment:

```
$ bosh deployments
+-----+-----+
| Name      | Release(s)          | Stemcell(s)           |
+-----+-----+
| cf-example | cf-mysql/10        | bosh-vsphere-esxi-ubuntu-trusty-go_agent/2690.3 |
|           | cf/183.2             |                      |
+-----+-----+
```

2. Run `bosh download manifest DEPLOYMENT-NAME LOCAL-SAVE-NAME` to download and save the current BOSH deployment manifest. You need this manifest to locate information about your databases. Replace `DEPLOYMENT-NAME` with the name of the current BOSH deployment. For this procedure, use `cf.yml` as the `LOCAL-SAVE-NAME`.

```
$ bosh download manifest cf-example cf.yml
Deployment manifest saved to `cf.yml'
```

Step 2: Add UAA client

You must authorize the example nozzle as a UAA client for your CF deployment. To do this, add an entry for the example nozzle as `client` for `uaa` under `properties` key in your CF deployment manifest. You must enter the example nozzle object in the right location in the manifest, and with the correct indentation, as described below.

Note: Deployment manifests have the `.yml` extension, and are therefore YAML files. Visit [YAML](#) on the web to learn about YAML syntax.

1. Find the `properties` key that is at 0 indentation (left aligned).
2. Under that `properties` key, find `uaa` at the next level of indentation.
3. Under that `uaa` key, find the `clients` key at the next level of indentation.
4. Enter properties for the `example-nozzle` at the next level of indentation, exactly as below. The `...` in the figure below indicate that other properties may populate the manifest at each level in the hierarchy.

```
properties:  
...  
uaa:  
...  
clients:  
...  
example-nozzle:  
  access-token-validity: 1209600  
  authorized-grant-types: authorization_code,client_credentials,refresh_token  
  override: true  
  secret: example-nozzle  
  scope: openid,oauth.approvals,doppler.firehose  
  authorities: oauth.login,doppler.firehose
```

5. Save the manifest file `cf.yml`.

Step 3: Redeploy Cloud Foundry

1. Select `cf.yml` as the manifest for your deployment.

```
$ bosh deployment cf.yml  
Deployment set to `/Users/example_user/workspace/bosh-lite/cf.yml'
```

2. Deploy your Cloud Foundry with BOSH.

```
$ bosh deploy
Acting as user 'admin' on deployment 'cf-warden' on 'Bosh Lite Director'
Getting deployment properties from director...

Detecting deployment changes
-----
Releases
No changes

Compilation
No changes

Update
No changes

Resource pools
No changes

Disk pools
No changes

Networks
No changes

Jobs
No changes

Properties
uaa
clients
example-nozzle
+ access-token-validity: 1209600
+ authorized-grant-types: authorizationcode,clientcredentials,refresh_token
+ override: true
+ secret: example-nozzle
+ scope: openid,oauth.approvals,doppler.firehose
+ authorities: oauth.login,doppler.firehose

Meta
No changes

Please review all changes carefully

Deploying
-----
Are you sure you want to deploy? (type 'yes' to continue):
```

Step 4: Clone example release

The Loggregator team has prepared an example nozzle application for use with this tutorial.

1. Run `git clone` to clone the main release repository from [GitHub](#).

```
$ git clone git@github.com:cloudfoundry-incubator/example-nozzle-release.git
Cloning into 'example-nozzle-release'...
```

2. Run `git submodule update --init --recursive` to update all the included submodules.

```
$ git submodule update --init --recursive
Submodule 'src/github.com/cloudfoundry-incubator/example-nozzle' (git@github.com:cloudfoundry-incubator/example-nozzle)
Submodule 'src/github.com/cloudfoundry-incubator/uaago' (git@github.com:cloudfoundry-incubator/uaago.git) registered for tracking
...
Cloning into 'src/github.com/cloudfoundry-incubator/example-nozzle'...
...
```

3. `cd` into the `example-release` directory.

```
$ cd example-nozzle-release
```

Step 5: Prepare the Nozzle manifest

There are two options for preparing the nozzle deployment manifest:

- (Quicker option) Our engineers have included a script to prepare the manifest in a single step. To use this option simply run the following in the terminal:

```
$ scripts/make_manifest_spiff_bosh_lite
```

- (Recommended, in order to fully understand the deployment procedure) Complete the following steps:

1. Inside `example-nozzle-release`, change into the `templates` directory.

```
$ cd templates
```

There are two yaml files inside:

- `template.yml` is template for the manifest we will use to deploy the nozzle.
- `bosh-lite-stub.yml` contains the values that will populate the missing information in the manifest template to complete our deployment manifest.

2. Make a `tmp` directory to house our compiled manifest.
3. Use spiff to compile a deployment manifest from the template and stub, and save this manifest.

```
$ spiff merge templates/template.yml templates/bosh-lite-stub.yml > tmp/manifest_bosh_lite.yml
```

4. Get your BOSH director UUID by running:

```
$ bosh status --uuid
```

5. In the compiled nozzle deployment manifest, find the `director_uuid` property and replace its value of `PLACEHOLDER-DIRECTOR-UUID` with your BOSH director UUID.

```
compilation:
  cloud_properties:
    name: default
  network: example-nozzle-net
  reuse_compilation_vms: true
  workers: 1
  director_uuid: PLACEHOLDER-DIRECTOR-UUID
```

Step 6: Select the new manifest as the deployment manifest for the nozzle

```
$ bosh deployment tmp/manifest_bosh_lite.yml
Deployment set to `/Users/example_user/workspace/example-nozzle-release/templates/tmp/manifest_bosh_lite.yml'
```

Step 7: Create a BOSH release for the nozzle

Run the following terminal command, using the name `example-nozzle`, which matches the [UAA client created earlier](#) in the CF deployment manifest:

```
$ bosh create release --name example-nozzle
Syncing blobs...
...
```

Step 8: Upload the BOSH release

Run the following terminal command:

```
$ bosh upload release
Acting as user 'admin' on 'Bosh Lite Director'

Copying packages
-----
example-nozzle
golang1.4

Copying jobs
-----
example-nozzle

Generated /var/folders/4n/qs1rjbmd1c5gfb78m3_06j6r0000gn/T/d20151009-71219-17a5m49/d20151009-71219-rts928/release.tgz
Release size: 59.2M

Verifying release...
...
Release info
-----
Name: nozzle-test
Version: 0+dev.2

Packages
-
- example-nozzle (b0944f95eb5a332e9be2adfb4db1bc88f9755894)
- golang1.4 (b68dc9557ef296cb21e577c31ba97e2584a5154b)

Jobs
-
- example-nozzle (112e01c6ee91e8b268a42239e58e8e18e0360f58)

License
-
- none

Uploading release
```

Step 9: Deploy the nozzle

Run the following terminal command:

```
$ bosh deploy
Acting as user 'admin' on deployment 'example-nozzle-lite' on 'Bosh Lite Director'
Getting deployment properties from director...
Unable to get properties list from director, trying without it...
Cannot get current deployment information from director, possibly a new deployment
Please review all changes carefully

Deploying
-----
Are you sure you want to deploy? (type 'yes' to continue):
```

Step 10: View the nozzle output

Congratulations! You have successfully deployed a Loggregator firehose nozzle.

The example nozzle simply dumps all of the data coming from the firehose to its log files. To view this data, SSH into the example-nozzle VM and examine the logs.

1. `ssh` into the nozzle vm at the IP configured in the nozzle's manifest template stub `./templates/bosh-lite-stub.yml`.

```
bosh ssh example-nozzle --stricthostkeychecking no

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.19.0-25-generic x86_64)

  • Documentation: https://help.ubuntu.com/
Last login: Wed Sep 23 21:29:50 2015 from 192.168.50.1
```

2. `cat` out the `stdout` log file.

```
$ cat /var/vcap/sys/log/example-nozzle/example-nozzle.stdout.log

===== Streaming Firehose (will only succeed if you have admin credentials)
origin:"DopplerServer" eventType:ValueMetric timestamp:1443046217700750747 deployment:"cf-warden" job:"dopplerz1" index:"0"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910193187 deployment:"cf-warden" job:"loggregatortraffic" index:"0"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910360012 deployment:"cf-warden" job:"loggregatortraffic" index:"1"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910252169 deployment:"cf-warden" job:"loggregatortraffic" index:"2"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910294255 deployment:"cf-warden" job:"loggregatortraffic" index:"3"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910318582 deployment:"cf-warden" job:"loggregatortraffic" index:"4"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910339088 deployment:"cf-warden" job:"loggregatortraffic" index:"5"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910379199 deployment:"cf-warden" job:"loggregatortraffic" index:"6"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046218910394886 deployment:"cf-warden" job:"loggregatortraffic" index:"7"
origin:"router0" eventType:HttpStartStop timestamp:1443046219105062148 deployment:"cf-warden" job:"routerz1" index:"0"
origin:"apiZ0" eventType:HttpStartStop timestamp:1443046219109842455 deployment:"cf-warden" job:"api_z1" index:"0" ip:"192.168.50.1"
origin:"router0" eventType:HttpStartStop timestamp:1443046219110064368 deployment:"cf-warden" job:"routerz1" index:"0" ip:"192.168.50.1"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177165446 deployment:"cf-warden" job:"dopplerz1" index:"0"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177288325 deployment:"cf-warden" job:"dopplerz1" index:"1"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177346726 deployment:"cf-warden" job:"dopplerz1" index:"2"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177274975 deployment:"cf-warden" job:"dopplerz1" index:"3"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177310389 deployment:"cf-warden" job:"dopplerz1" index:"4"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177330214 deployment:"cf-warden" job:"dopplerz1" index:"5"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177353454 deployment:"cf-warden" job:"dopplerz1" index:"6"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177360052 deployment:"cf-warden" job:"dopplerz1" index:"7"
origin:"syslogdrainbinder" eventType:ValueMetric timestamp:1443046219177481456 deployment:"cf-warden" job:"dopplerz1" index:"8"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880585603 deployment:"cf-warden" job:"dopplerz1" index:"9"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880895040 deployment:"cf-warden" job:"dopplerz1" index:"10"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881017888 deployment:"cf-warden" job:"dopplerz1" index:"11"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881011670 deployment:"cf-warden" job:"dopplerz1" index:"12"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880929574 deployment:"cf-warden" job:"dopplerz1" index:"13"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219881004417 deployment:"cf-warden" job:"dopplerz1" index:"14"
origin:"DopplerServer" eventType:CounterEvent timestamp:1443046219880929568 deployment:"cf-warden" job:"dopplerz1" index:"15"
origin:"MetronAgent" eventType:CounterEvent timestamp:1443046220058280679 deployment:"cf-warden" job:"api_z1" index:"0"
```

Cloud Foundry Data Sources

Currently, Cloud Foundry logs and metrics come from several sources:

- Loggregator is the next generation logging and metrics system for Cloud Foundry. It aggregates metrics from applications and CF system components and streams these out to the CF cli or to third party log management services.
- The [Collector](#) is Cloud Foundry's original metric aggregation system. It gathers metrics from all Cloud Foundry system components by querying their `/healthz` and `/varz` endpoints, and then publishes this data to external systems such as Datadog, AWS CloudWatch and OpenTSDB.

 **Note:** The Collector will eventually be deprecated in favor of the Loggregator system.

- The BOSH Health Monitor continually listens for one 'heartbeat' per minute from each deployed VM. These heartbeats contain status updates and lifecycle events. Health Monitor can be extended by plugins to forward heartbeat data to other CF components or third party services.
- Logs from CF components can also be forwarded directly to your own server, bypassing loggregator. See [Loggregator for Operators](#) for more information.

Currently, Cloud Foundry supports all of these metrics pipelines. Data from each of these sources can be streamed to a variety of services including the following:

- Ops Metrics
- Datadog
- AWS CloudWatch

See [Using Log Management Services](#) for more information about draining logs from Elastic Runtime.

Installing the Loggregator Firehose Plugin for cf CLI

The Loggregator Firehose plugin for the Cloud Foundry Command Line Interface (cf CLI) allows Cloud Foundry (CF) administrators access to the output of the [Loggregator Firehose](#), which includes logs and metrics from all CF components.

See [Using cf CLI Plugins](#) for more information about using plugins with the cf CLI.

Prerequisites

- Administrator access to the Cloud Foundry deployment that you wish to monitor
- Cloud Foundry Command Line Interface (cf CLI) 6.12.2 or later

Refer to the [Installing the cf Command Line Interface](#) topic for information about downloading, installing, and uninstalling the cf CLI.

Install the Plugin

1. Run `cf add-plugin-repo REPO_NAME URL` to add the Cloud Foundry Community plugin repository to your cf CLI plugins.

```
$ cf add-plugin-repo CF-Community http://plugins.cloudfoundry.org
```

2. Run `cf install-plugin PLUGIN-NAME -r PLUGIN-REPO` to install the Firehose plugin from the CF Community plugin repository.

```
$ cf install-plugin "Firehose Plugin" -r CF-Community
```

View the Firehose

Run `cf nozzle --debug` to view the streaming output of the Firehose, which includes logging events and metrics from CF system components. For more information about logging and metrics in CF, see [Overview of the Loggregator System](#).

```
$ cf nozzle --debug
```

 **Note:** You must be logged in as a Cloud Foundry administrator to access the Firehose.

Uninstall the Plugin

Run `cf plugins` to see a list of installed plugins.

```
$ cf plugins
Listing Installed Plugins...
OK
Plugin Name      Version    Command Name   Command Help
FirehosePlugin   0.6.0     nozzle        Command to print out messages from the firehose
```

Run `cf uninstall-plugin PLUGIN-NAME` to uninstall the plugin.

```
$ cf uninstall-plugin FirehosePlugin
```

Using SSL with a Self-Signed Certificate in Pivotal Ops Metrics

Secure Socket Layer (SSL) is a standard protocol for establishing an encrypted link between a server and a client. To communicate over SSL, a client needs to trust the SSL certificate of the server.

There are two kinds of SSL certificates: signed and self-signed.

- **Signed:** A Certificate Authority (CA) signs the certificate. A CA is a trusted third party that verifies your identity and certificate request, then sends you a digitally signed certificate for your secure server. Client computers automatically trust signed certificates.
- **Self-signed:** Your own server generates and signs the certificate. Clients do not automatically trust self-signed certificates. To communicate over SSL with a server providing a self-signed certificate, a client must be explicitly configured to trust the certificate.

Note: Certificates generated in Elastic Runtime are signed by the Operations Manager Certificate Authority. They are not technically self-signed, but they are referred to as 'Self-Signed Certificates' in the Ops Manager GUI and throughout this documentation.

Clients keep all trusted certificates in a kind of keystore called a truststore. To configure a client to trust a self-signed certificate, import the self-signed certificate to a truststore on the client.

Refer to the following procedures to use SSL with a self-signed certificate in Pivotal Ops Metrics.

Step 1: Enable SSL

To enable SSL, follow the step below to configure a PKCS#1 certificate in your release manifest.

1. Use `openssl genrsa` with the `-out` flag to generate a new private key. For example, run the following command to generate a 2048-bit private key named `privkey.pem`:

```
$ openssl genrsa -out privkey.pem 2048
```

2. Use `openssl req` to generate a new certificate signing request based on your private key.

```
$ openssl req -new -key privkey.pem -out server.csr
```

3. Use the command below to create a self-signed certificate, `server.crt`, from the existing private key, `privkey.pem` and the existing certificate signing request, `server.csr`:

```
$ openssl x509 -req -days 365 -in server.csr -signkey privkey.pem -out server.crt
```

4. Use `keytool -import` to create a client keystore. Note that the local JMX client uses this keystore, not the server.

```
$ keytool -import -alias ops-metrics-ssl -file ops-metrics.cer -keystore localhost.truststore
```

5. Enter a password for the keystore when prompted. Record this password.

6. Enter `yes` when prompted to trust the certificate.

7. Use `bosh edit deployment` to open your release manifest for editing.

```
$ bosh edit deployment
```

In your release manifest, copy your private key into the `ssl_private_key` field and your public certificate into the `ssl_cert` field.

```

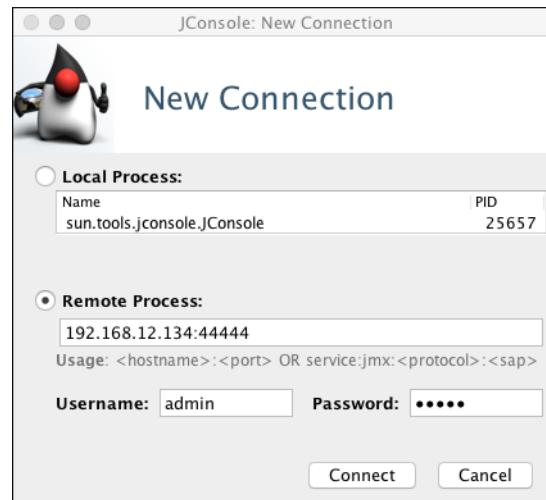
properties:
maximus:
  username: USERNAME
  password: PASSWORD
  public_hostname: HOSTNAME
  use_ssl: true # This defaults to true
  ssl_cert: ! '-----BEGIN CERTIFICATE-----
  ...
  -----END CERTIFICATE-----'
  ...
  ssl_private_key: ! '-----BEGIN PRIVATE KEY-----
  ...
  -----END PRIVATE KEY----- '

```

8. Verify your settings using the local JMX client of your choice. The example below uses `jconsole`. Replace `KEYSTORE_PASSWORD` with the keystore password that you recorded in a previous step:

```
$ jconsole -J-Djavax.net.ssl.trustStore=/lib/home/jcert/localhost.truststore -J-Djavax.net.ssl.trustStorePassword=KEYSTORE_PASSWORD
```

- In the **Remote Process** field, enter the IP address of the Maximus server, port number `44444`.
- To complete the **Username** and **Password** fields, refer to the **Credentials** tab of the Metrics tile in Pivotal Ops Manager.



By default, these credentials are `admin` and `admin`.

Step 2: Generate a Self-Signed Certificate

Follow the steps below to generate a self-signed certificate on your server:

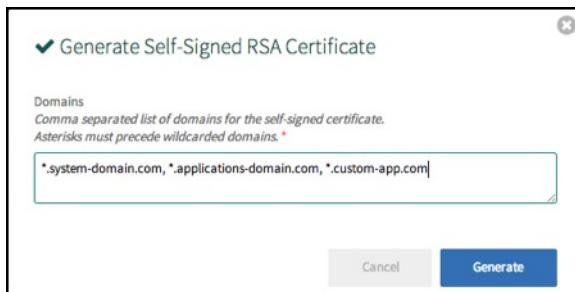
- In Pivotal Ops Manager, click the **Pivotal Ops Metrics** tile.
- Check **Enable SSL**.
- Click **Generate Self-Signed RSA Certificate** and check the **Trust Self-Signed Certificates** box.

[Generate Self-Signed RSA Certificate](#)

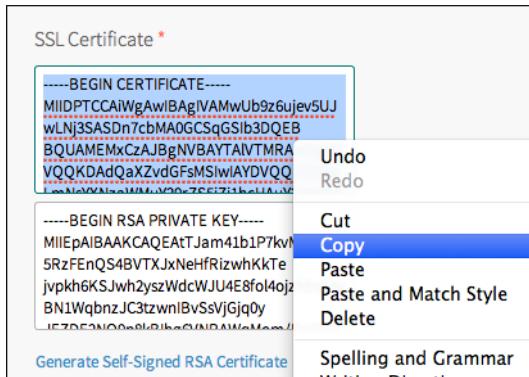
Trust Self-Signed Certificates

Save

4. Enter your system and application domains in wildcard format. Optionally, also add any custom domains in wildcard format. Click **Generate**.



5. Select and copy the certificate.



6. Paste the certificate into a text file and save as `ops-metrics.cer`.

Step 3: Import the Self-signed Certificate to a Truststore

Follow the steps below to import the self-signed certificate to your client:

1. Copy `ops-metrics.cer` from your server to your client.
2. Navigate to the client directory where you copied the saved certificate.
3. Use `keytool -import` to import the certificate with an alias of `ops-metrics-ssl` to the truststore `localhost.truststore`:

```
$ keytool -import -alias ops-metrics-ssl -file ops-metrics.cer -keystore localhost.truststore
```

- If `localhost.truststore` already exists, a password prompt appears. Enter the keystore password that you recorded in a previous step.
- If `localhost.truststore` does not exist, you must create a password.

4. Verify the details of the imported certificate.

Step 4: Start a Monitoring Tool with the Truststore

Once you import the self-signed certificate to the `localhost.truststore` truststore on the client, you must instruct your monitoring tool to use the truststore.

On a command line, start your monitoring tool with the location and password of the truststore as follows:

1. Pass the location of `localhost.truststore` to the monitoring tool using the `javax.net.ssl.trustStore` property.
 2. Pass the truststore password using the `javax.net.ssl.trustStorePassword` property.
- Example starting jConsole:

```
$ jconsole -J-Djavax.net.ssl.trustStore=/lib/home/jcert/localhost.truststore -J-Djavax.net.ssl.trustStorePassword=KEYST
```

3. Enter a fully-qualified hostname. This hostname must exist in a subdomain covered by your certificate.

 **Note:** You must enter a fully-qualified hostname, not an IP address.

Your monitoring tool should now communicate with your server through the SSL connection.

Deploying Pivotal Ops Metrics

The Pivotal Ops Metrics tool is a JMX extension for Elastic Runtime. Pivotal Ops Metrics collects and exposes system data from Cloud Foundry components via a JMX endpoint. Use this system data to monitor your installation and assist in troubleshooting.

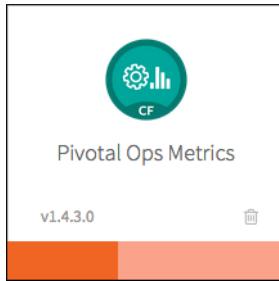
The Pivotal Ops Metrics tool is composed of the following three virtual machines:

- The JMX Provider
- A VM that governs compilation
- A nozzle for the [Loggregator Firehose](#)

Follow the instructions below to deploy Pivotal Ops Metrics using the [Pivotal Cloud Foundry](#) Operations Manager.

Step 1: Install the Ops Metrics Tile

1. [Download Ops Metrics](#).
2. Import Ops Metrics into Ops Manager by following the instructions for [Adding and Importing Products](#).
3. On the Installation Dashboard, click the **Pivotal Ops Metrics** tile.



The orange bar on the **Pivotal Ops Metrics** tile indicates that the product requires configuration.

Step 2: Configure Networks

1. Select **Assign Networks**.
2. Use the drop-down menu to select a Network.

Note: Ops Metrics uses the default Assigned Network if you do not select a different network.

Step 3: Configure Availability Zones

1. (vSphere and Amazon Web Services Only) Select **Assign Availability Zones**. These are the Availability Zones

that you [create](#) when configuring Ops Manager Director.

- Select an Availability Zone under **Place singleton jobs**. Ops Manager runs Metrics jobs with a single instance in this Availability Zone.
- Select one or more Availability Zones under **Balance other jobs**. Ops Manager balances instances of Metrics jobs with more than one instance across the Availability Zones that you specify.

The screenshot shows the 'Pivotal Ops Metrics' settings interface. On the left, there's a sidebar with tabs: 'Settings' (selected), 'Status', 'Credentials', and 'Logs'. Below the tabs is a list of configuration items with checkboxes:

- Assign Networks (checked)
- Assign Availability Zones (unchecked)
- JMX Provider (unchecked)
- OpenTSDB Firehose Nozzle (unchecked)
- Resource Config (checked)
- Stemcell (checked)

To the right, under 'Availability Zone Assignments', there are two sections:

- 'Place singleton jobs in': A radio button group where 'MY-ZONE' is selected (radio button is filled).
- 'Balance other jobs in': A checkbox group where 'MY-ZONE' is checked (checkbox has a checkmark).

A blue 'Save' button is located at the bottom right of the main content area.

Step 4: Configure JMX Provider

1. Select **JMX Provider**.
2. Enter a new username and password into the **JMX Provider credentials** username and password fields.
3. Record these credentials. You will use these to connect JMX clients to the JMX Provider.

Step 5: Configure SSL

1. **(Optional)** Select the **Enable SSL** checkbox. Enabling SSL requires JMX clients to use SSL to connect to the JMX Provider. If SSL is not enabled, JMX clients can connect to the JMX Provider without SSL credentials.

Assign Availability Zones

JMX Provider

Resource Config

JMX Provider credentials *

Enable SSL

SSL Certificate

Certificate PEM

Private Key PEM

[Generate Self-Signed RSA Certificate](#)

Save

If you select the **Enable SSL** checkbox, you must also provide an SSL certificate and private key. There are two ways to provide an SSL certificate and private key:

- If you are using a signed certificate, paste an X.509 certificate in the **Certificate PEM** field and a PKCS#1 private key in the **Private Key** field.
- If you want to use SSL but do not want to use a signed certificate, you must perform the following actions:
 1. Generate a self-signed certificate on the server.
 2. Import the self-signed certificate to a trust store on the client.
 3. Start jConsole, or another monitoring tool, with the trust store.

For more information, see [Using SSL with a Self-Signed Certificate](#).

Once you have provided an SSL certificate and private key, click **Save**.

Assign Networks

Assign Availability Zones

JMX Provider

Resource Config

Credentials to connect to JMX Provider

JMX Provider credentials *

Enable SSL

SSL Certificate

```
-----BEGIN CERTIFICATE-----
MIIDUTCCAg2gAwIBAgIVAP2UDTsMs56HI
USTM4RoypPMruhuMA0GCSqGSIb3DQE
B
BQUAMDsxCzAJBgNVBAYTAINTMRawDgY
-----END CERTIFICATE-----
```

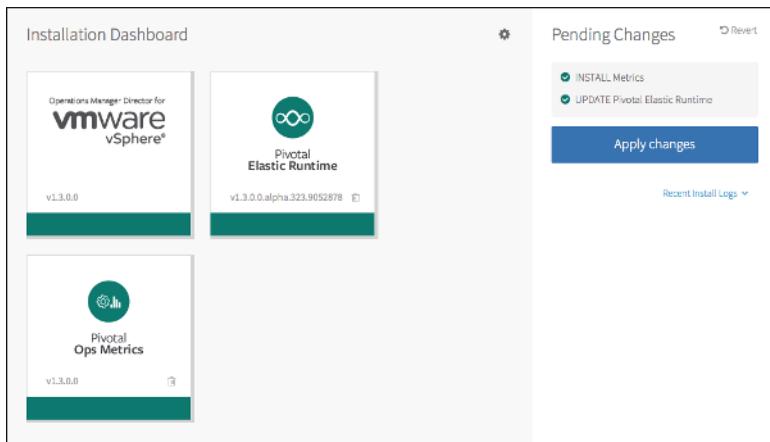
```
-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEazlnfMTS/rpn1rLyIKM
aK11y7FyGi/JN9Z6E51SwxKfCapgS
HSof6zER1Sijwtg27nHobvjmp3UnWV5q
IxSseKpgsJaQKM+u5/zbwZj4gA+FSO
-----END RSA PRIVATE KEY-----
```

[Generate Self-Signed RSA Certificate](#)

Save

Step 6: Apply Changes

1. Navigate to the PCF Ops Manager Installation Dashboard.
2. In the Pending Changes view, click **Apply Changes** to install Pivotal Ops Metrics.



- When complete, a “Changes Applied” message appears.

Step 7: Find the IP of the JMX Provider

- Click **Return to Product Dashboard**.
- Click the **Pivotal Ops Metrics** tile and select the **Status** tab.
- Record the IP address of the JMX Provider.

Note: After installation, your JMX client connects to this IP address at port 44444 using the credentials that you supplied.

The screenshot shows the 'Metrics' status page with tabs for Settings, Status (selected), Credentials, and Logs. It displays a table for 'Jobs on Availability Zone "default"' with one row for 'JMX Provider'.

JOB	INDEX	IPS	CID
JMX Provider	0	10.85.52.127	vm-26195407-c448-4a89

Step 8: Configure the Metrics IP Address

- Return to the **Installation Dashboard**. Click the **Ops Manager Director** tile and select **Director Config**.
- In the **Metrics IP Address** field, enter the IP address of the JMX Provider. Click **Save**.

[Installation Dashboard](#)

Ops Manager Director for VMware vSphere

Settings Status Credentials Logs

vCenter Config Director Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

NTP Servers (comma delimited)*
time.apple.com

Metrics IP Address
10.85.52.127

Enable VM Resurrector Plugin

Save

Step 9: Allocate a VM for the Collector

1. Return to the **Installation Dashboard**. Click the **Pivotal Elastic Runtime** tile and select **Resource Config**.
2. Change the number of **Instances** for the **Collector** job from **0** to **1**. Click **Save**.

[Installation Dashboard](#)

Pivotal Elastic Runtime

Settings Status Credentials Logs

Assign Networks Resource Config

Assign Availability Zones

HAProxy

Router IPs

Cloud Controller

External Endpoints

SSO Config

LDAP Config

SMTP Config

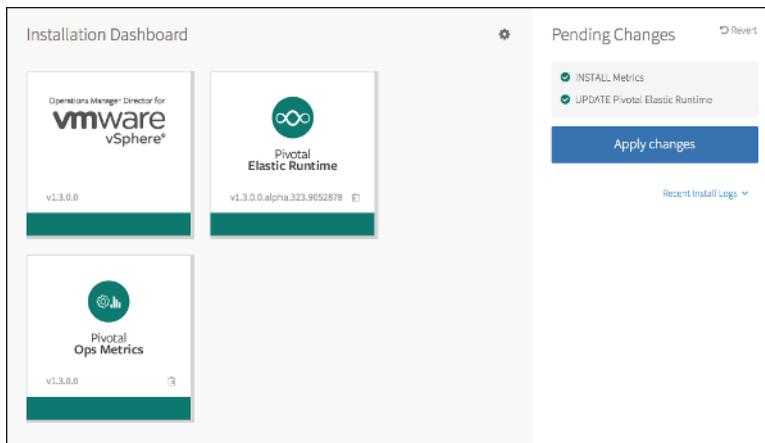
Lifecycle Errands

Resource Config

JOB	INSTANCES
HAProxy	1
NATS	1
etcd	1
Health Manager	1
NFS Server	1
Cloud Controller Database	1
Cloud Controller	1
Clock Global	1
Cloud Controller Worker	1
Router	1
Collector	1
UAA Database	1
UAA	1
Login	1
Console Database	1
MySQL Server	1
DEA	1

Step 10: Complete Installation

1. In the Pending Changes view, click **Apply Changes**.



- When complete, a “Changes Applied” message appears. Click **Return to Product Dashboard**. Pivotal Ops Metrics is now installed and configured.

Once installed and configured, metrics for Cloud Foundry components automatically report to the JMX endpoint. Your JMX client uses the credentials supplied to connect to the IP address of the Pivotal Ops Metrics JMX Provider at port 44444.

Using Pivotal Ops Metrics

Pivotal Ops Metrics is a Java Management Extensions (JMX) tool for Elastic Runtime. To help you monitor your installation and assist in troubleshooting, Pivotal Ops Metrics collects and exposes system data from Cloud Foundry components via a JMX endpoint.

Cloud Controller Metrics

Pivotal Ops Metrics reports the number of Cloud Controller API requests completed and the requests sent but not completed.

The number of requests sent but not completed represents the pending activity in your system, and can be higher under load. This number will vary over time, and the range it can vary over depends on specifics of your environment such as hardware, OS, processor speeds, load, etc. In any given environment, though, you can establish a typical range of values and maximum for this number.

Use the Cloud Controller metrics to ensure that the Cloud Controller is processing API requests in a timely manner. If the pending activity in your system increases significantly past the typical maximum and stays at an elevated level, Cloud Controller requests may be failing and additional troubleshooting may be necessary.

The following table shows the name of the Cloud Controller metric, what the metric represents, and the metric type (data type).

METRIC NAME	DEFINITION	METRIC TYPE (DATA TYPE)
cc.requests.completed	Number of Cloud Controller API requests completed since this instance of Cloud Controller started	Counter (Integer)
cc.requests.outstanding	Number of Cloud Controller API requests made but not completed since this instance of Cloud Controller started	Counter (Integer)

See the [Cloud Controller](#) topic for more information about the Cloud Controller.

Router Metrics

Pivotal Ops Metrics reports the number of sent requests and the number of completed requests for each Cloud Foundry component.

The difference between these two metrics is the number of requests made to a component but not completed, and represents the pending activity for that component. The number for each component can vary over time, and is typically higher under load. In any given environment, though, you can establish a typical range of values and maximum for this number for each component.

Use these metrics to ensure that the Router is passing requests to other components in a timely manner. If the pending activity for a particular component increase significantly past the typical maximum and stays at an elevated level, additional troubleshooting of that component may be necessary. If the pending activity for most or all components increases significantly and stays at elevated values, troubleshooting of the router may be necessary.

The following table shows the name of the Router metric, what the metric represents, and the metric type (data type).

METRIC NAME	DEFINITION	METRIC TYPE (DATA TYPE)
router.requests [component=c]	Number of requests the router has received for component c since this instance of the router has started c can be Cloud Controller or DEA	Counter (Integer)
router.responses [status=s,component=c]	Number of requests completed by component c since this instance of the router has started c can be Cloud Controller or DEA s is http status family: 2xx, 3xx, 4xx, 5xx, and other	Counter (Integer)

See the [Router](#) topic for more information about the Router.

Droplet Execution Agent Metrics

Pivotal Ops Metrics reports four data values for each Droplet Execution Agent (DEA) instance. If you have multiple DEA instances, the metrics reported are per DEA, and not the total across all of your DEAs.

Use these metrics to ensure that your system has enough disk space and memory to deploy and run your applications. For example, if `available_disk_ratio` or `available_memory_ratio` drop too low, the Cloud Controller will be unable to make reservations for a new application, and attempts to deploy that application will fail.

The following table shows the name of the DEA metric, what the metric represents, and the metric type (data type).

METRIC NAME	DEFINITION	METRIC TYPE (DATA TYPE)
available_disk_ratio	Percentage of disk available for allocation by future applications/staging requests	Gauge (Float, 0-1)
available_memory_ratio	Percentage of memory available for allocation by future applications/staging requests	Gauge (Float, 0-1)
mem_free_bytes	Current amount of memory free on the DEA	Gauge (Integer)
mem_used_bytes	Current amount of memory actually used, not just allocated, on the DEA	Gauge (Integer)

See the [Droplet Execution Agent](#) topic for more information about Droplet Execution Agents.

Virtual Machine Metrics

Pivotal Ops Metrics reports data for each virtual machine (VM) in a deployment. Use these metrics to monitor the health of your Virtual Machines.

The following table shows the name of the Virtual Machine metric, what the metric represents, and the metric type (data type).

METRIC NAME	DEFINITION	METRIC TYPE (DATA TYPE)
system.mem.percent	Percentage of memory used on the VM	Gauge (Float, 0-100)
system.swap.percent	Percentage of swap used on the VM	Gauge (Float, 0-100)
system.disk.ephemeral.percent	Percentage of ephemeral disk used on the VM	Gauge (Float, 0-100)
system.disk.system.percent	Percentage of system disk used on the VM	Gauge (Float, 0-100)
system.cpu.sys	Amount of CPU spent in system processes	Gauge (Float)
system.cpu.user	Amount of CPU spent in user processes	Gauge (Float)
system.cpu.wait	Amount of CPU spent in waiting processes	Gauge (Float)

Pivotal Cloud Foundry Release Notes and Known Issues

Release Notes

- [Pivotal Elastic Runtime](#)
- [Pivotal Operations Manager](#)

Known Issues

- [Pivotal Elastic Runtime](#)
- [Pivotal Operations Manager](#)

Pivotal Elastic Runtime v1.6.0.0 Release Notes

Changes since v1.5.0.0:

Elastic Runtime

- CF Release version: 222
- Diego version: 0.1437.0
- Garden version: 0.308.0
- CF MySQL version: 23

New Features

Default Stack: cflinuxfs2

This release completely removes the lucid64 stack from Pivotal Cloud Foundry. After upgrading to this release, apps that have not been migrated to cflinuxfs2 will fail to start.

Operators are encouraged to migrate all apps to cflinuxfs2, as mentioned in the v1.4 release notes, before upgrading to v1.6.

Further details can be found [here](#).

Cloud Foundry Runtime - Diego

Diego is the new application runtime that Pivotal Cloud Foundry will use by default to run your apps.

Diego enables many new features and enhancements, such as allowing the deployment of thousands of applications and resurrecting crashed applications within seconds. It also supports new workloads, such as .NET on Windows and Docker on Linux, and enables SSH access to containers.

Details about how Diego works can be found [here](#). Details about its architecture can be found [here](#).

Details about SSH access to application containers on Diego can be found [here](#).

There is a new field, “Application Containers Subnet Pool”, which allows you to configure a specific subnet pool for your applications running on Diego. You can leave this field as its default value unless you specifically want to use a different subnet, say if you have a third-party service or other VMs running with the same IPs in your network. This feature is only usable on Diego, not on DEAs. You can only specify a single CIDR subnet, no “excluded IPs” can be specified.

Windows 2012 stack

You can use the Windows .NET MSI to run your applications on Windows containers. Further details on enabling this feature can be found [here](#).

Docker

Docker support for Pivotal Cloud Foundry is in beta currently. To enable Docker support in your deployment, you can use a CLI command as a user with the admin role: `cf enable-feature-flag diego_docker`. Further details about Docker on Diego can be found [here](#).

Security Configuration of Runtime

This section allows you to configure security settings for your Pivotal Cloud Foundry Elastic Runtime components, such as HAProxy, Router, and the DEA/Diego Cells.

The SSL Termination Certificate input now applies to both HAProxy (if you use this component) and the Cloud Foundry Router. This field is not optional, even if you use an external load-balancer instead of HAProxy, since it will be applied to the Router. It does not have to be the same certificate as the one used by your external load-balancer, as this certificate is only used for SSL traffic between the load-balancer and Router. You can enable SSL termination on the Router with a

checkbox in this same section, “Enable TLS on the Router”. More details [here](#) about securing connections to the Router.

If you have multiple domains to map to the Router, such as separate system and apps domains, you can only use one SSL certificate. The Router does not yet support multiple certificates. However, one SSL certificate with multiple domains attributed to it is acceptable.

The “Enable cross-container traffic” checkbox now controls the restriction of cross-container traffic for both DEAs and Diego Cells, depending on which runtime backend platform is chosen. Pivotal does not recommend that you select this checkbox within multi-tenant environments. The feature enables using application containers to connect to other application containers directly, without going through the router. This setup may be desired for microservices, which may be optionally used with Pivotal Spring Cloud Services.

The Security Configuration section also includes fields that allow specifying the HAProxy and Router ciphers, both of which are optional fields.

Other Runtime Features

S3-Compatible Filestore Configuration

It is now possible to configure four different S3 buckets to comprise the Cloud Controller’s filesystem, instead of just one.

You can choose to use four different buckets if you are installing Pivotal Cloud Foundry for the first time, but not for an upgrade (i.e., from Pivotal Cloud Foundry v1.5 to v1.6). This is because the data will not be automatically migrated from the one bucket configuration in 1.5 to separate buckets in 1.6. It is recommended to use separate buckets for new deployments. Existing deployments are supported with one bucket.

Experimental Features

There are no experimental features being introduced in PCF Elastic Runtime for v1.6.

Improved MySQL Service

MySQL, used by some of the Pivotal Cloud Foundry system applications in the past, is now available as a data store for every Pivotal Cloud Foundry system component and application, including Cloud Controller and UAA. This is an improvement over the non-highly-available Postgresql databases that these components and applications relied upon in previous releases.

You can choose to use this MySQL service if you are installing Pivotal Cloud Foundry for the first time, but not for an upgrade (e.g.: from Pivotal Cloud Foundry v1.5 to v1.6). This is because the data will not be automatically migrated from an existing database to a new one. A future release is planned that will address data migration for installations that started with a postgres database.

If you do install Pivotal Cloud Foundry on the MySQL databases only, then you can scale down the Apps Manager Database (Postgres), Cloud Controller Database (Postgres), and UAA Database (Postgres) to zero instances. Those components will instead use the MySQL cluster.

API/cf CLI

Note: These features are available via API only; cf CLI support coming soon. - Org Managers and Space Managers can now manage roles without requiring admin privileges. - Max number of private domains can be specified in the Organization Quota. - Max number of app instances can be specified in the Organization Quota and Space Quota. - Admins can purge a single service instance and its bindings. This is a more targeted purge to resolve a single service instance than the purge service offering.

Apps Manager

This release adds support for the Diego Runtime, and features numerous fixes and enhancements for it.

Two new environment variables are introduced and one is removed in v1.6:

- The new `ENABLE_INTERNAL_USER_STORE` environment variable controls org and space user invitations, new user registrations, and password updates to the PCF internal user store. The default is set to `false`, which means that none of the features are exposed in Apps Manager. Set to `true` to expose these features.
- The new `ENABLE_NON_ADMIN_ROLE_MANAGEMENT` environment variable allows org managers and spaces managers to manage user roles regardless of where these users are created. The default is set to `false`, which means that none of

the features are exposed in Apps Manager. Set to `true` to expose these features.

- The previous `ENABLE_NON_ADMIN_USER_MANAGEMENT` environment variable that controlled many of these same features is removed in v1.6.

DEAs

Multiple stability and performance improvements were made to DEAs. They can now handle many more simultaneous connections, for instance.

Identity (aka UAA Server)

- UAA now enforces HTTPS only communication. This was an experimental feature in PCF 1.5.x
- Added support for UAA Log Rotation via BOSH
- SAML SSO Integration Updates
 - Allow Identity Provider Metadata with missing XML Declaration tag
 - Strict checking for duplicate SAML Entity IDs
- Token Format Updates
 - UAA tokens now contain an origin field which signifies the Identity Provider used for authentication. If SAML IDP is used for authentication, the Identity Provider alias is set in as the origin value.
 - Support for nonce parameter in OAuth requests to prevent against CSRF attacks

Logging and Metrics

- There is a new field, “Syslog Drain Buffer Size”, which allows you to configure the number of messages that your Doppler server will keep before starting to drop messages. Note that a larger buffer may overload your system.

Buildpacks

Smaller, More Secure Buildpacks

Buildpacks in PCF 1.6 are smaller than in PCF 1.5, since older, unsupported versions of third-party dependencies are no longer being packaged.

An example of this is Ruby 1.9.x, which reached [End Of Life on February 23, 2015](#) and was removed in [ruby-buildpack v1.4.0](#).

PCF 1.6 also is only providing the **last two** patch releases on any `major.minor` branch, with the goal of allowing an upgrade path, while also encouraging developers to upgrade to the latest security releases whenever possible.

The smaller buildpack sizes also means a faster PCF installation, as well as improved staging performance.

For any customers who want to customize the binaries that are shipped with a buildpack, the tooling for generating custom buildpacks has been open-sourced:

- [buildpack-packager](#)
- [binary-builder](#)

Binary Buildpack

PCF 1.6 also provides a “binary buildpack”, for customers to run precompiled binaries, for those occasions when developers want a bit more control over the executables that are running in the container.

The [binary-builder](#) tooling can also be used to help build executables for a specific rootfs in a maintainable manner.

Security - CVE fixes have been implemented since v1.5.0.0 and released via security patches.

- Canonical Ubuntu USN-2765-1
- Canonical Ubuntu USN-2756-1
- Canonical Ubuntu USN-2751-1
- Canonical Ubuntu USN-2740-1
- Canonical Ubuntu USN-2739-1
- Canonical Ubuntu USN-2738-1
- Canonical Ubuntu USN-2726-1
- Canonical Ubuntu USN-2722-2
- Canonical Ubuntu USN-2718-1
- Canonical Ubuntu USN-2710-1
- Canonical Ubuntu USN-2710-2
- Canonical Ubuntu USN-2698-1
- Canonical Ubuntu USN-2696-1
- Canonical Ubuntu USN-2694-1
- Canonical Ubuntu USN-2690-1
- Canonical Ubuntu USN-2684-1
- Canonical Ubuntu USN-2670-1
- Canonical Ubuntu USN-2639-1
- CVE 2015-3281
- CVE 2015-1420
- CVE 2015-1330

Other Bug Fixes

We fixed an issue with the v1.5.6 patch for Openstack deployments. The BOSH stemcell v3094, which this version of Elastic Runtime references, has a limitation affecting Openstack users only:

- Elastic Runtime 1.5.6 on Openstack does not work with S3/Swift blobstores.
- Elastic Runtime 1.5.6 on Openstack users must configure their object storage to use the internal blobstore option.
- vSphere, AWS and vCloud users are not affected.
- This is fixed in v1.6.0

Pivotal Cloud Foundry Ops Manager v1.6.0 Release Notes

Version 1.6.0 (changes since v1.5.5)

Major Features

- EBS encryption is now available for AWS deployments. This enables AWS customers to have encrypted data at rest on persistent disks.

API Changes

- Previously, Ops Manager provided an API endpoint called `/api/users` that allowed for the creation of multiple user accounts, though only the first one was functional. To correct this, `/api/users` has been replaced by `/api/setup` that provides the same functionality for creating the first user, but does not attempt to create secondary, tertiary, etc., users, since these user accounts would not be functional. Please see API documentation at <http://your-ops-man-ip/docs> for more information. **Please note that the API is not officially supported and may change without warning in the future.**

Security Features

- Multiple CVE fixes per pivotal.io/security.
- VM passwords are now scrubbed from the output logs and replaced with asterisks.

Minor Features

- On vSphere, the operator is now able to split availability zones by resource pool in the same cluster.

Bug Fixes

- Credentials no longer shows incorrect credentials when “Use default BOSH password” is selected.
- Upgrades no longer fail if product version contains a dash.
- A migration that exists but is null no longer results in a 500 error.

Product Author Features

- New product author documentation has been made available here:
<http://docs.pivotal.io/pivotalcf/packaging/index.html>

Notes

Embedded stemcell in 1.6.0 is 3100

Pivotal Elastic Runtime v1.5 Known Issues

New Issues

- The PCF Elastic Runtime tile v.1.6.0 will show “1.6.0-build.315” in the tile version. This is fine.
- Diego Cells may sometimes experience slow performance for various actions like staging new applications. If you experience slow performance, you can SSH into the Ops Manager VM and `bosh-recreate` the affected Diego Cell VM.
- The CLI command for viewing application files, `cf files`, does not work with applications on Diego.
- The Diego BBS may output a TLS handshake error every three seconds in the sys logs. This error is fine to ignore, and will be fixed in a future release of Diego.
- To SSH into an application container on Diego, your CF user must have the Space Developer role attached to it for the application space.
- The Smoke Tests may occasionally fail at the logging test suite. The Smoke Tests errand is fine to re-run in case of any failures.
- The version of Postgres has been upgraded to 9.4.2. If you have Postgresql databases in your deployment, Cloud Controller and UAA will be unavailable during the upgrade. In case that upgrade fails, there are mitigations [documented here ↗](#).
- After you upgrade to Pivotal Cloud Foundry 1.6, you may want to reclaim some blobstore space due to some orphaned blobs. To reclaim space, call [this end point ↗](#) to delete all buildpack caches from the blobstore after the v1.6 upgrade is complete.
- This upgrade includes a migration that modifies the events table on the Cloud Controller database. This table may be very large, and the migration may cause the upgrade to fail at the Cloud Controller job if it takes too long. If this happens, do not restart the deploy until the migration has finished running. The deploy can be restarted once the `space_id` foreign key constraint has been removed from the events table. To avoid the possibility of the migration causing a failure, truncate the events table before the deployment starts. The data in the events table are audit and log data, and PCF can function without it.
- .NET support on Windows cells does not support the same level of security and isolation as seen on Linux cells. At this time, it is only recommended for running “trusted” apps.
- Apps Manager now has a zero-downtime blue green deploy. As of v1.6, any `NON_ADMIN` environment variable changed from the default setting will not be applied to the second app version. You will need to set `NON-ADMIN` environment variables on both versions in order to keep the configuration consistent.

Existing Issues

- On the Security Config page of Elastic Runtime, after you supply your own cipher sets and click Save, do not erase the HAProxy and Router Cipher fields. The ciphers will not return to their default values when deleted, and will instead be interpreted as an empty cipher set.
- When selecting between internal and external System Database and/or File Storage config, if saved values for external systems fail verification (e.g. a host is not reachable from Ops Manager), the values will persist if you then select ‘Internal Databases’ or ‘Internal File Store’. To resolve this issue, return to your Ops Manager Installation Dashboard and click **Revert**, located in the upper right corner of the page.
- Recent versions of the cf CLI, including the version shipped with v1.6, will report a version mismatch with Elastic Runtime. You can ignore this message, and there is no need to upgrade the CLI. This error will be fixed in a future version of the cf CLI.

Pivotal Cloud Foundry Ops Manager v1.6.0 Known Issues

New Issues

Some of these issues may be fixed in subsequent patch releases to 1.6. Consult the Ops Manager 1.6 [Release Notes](#) for more information.

- If EBS encryption is initially disabled, and then later enabled, the change will not apply to the Director's persistent disk.
- The new "Trusted Certificates" feature does not insert the certificate (or certificate chain) that you provide into your application containers.
- The new "Trusted Certificates" feature does not insert the certificate (or certificate chain) that you provide into UAA's JVM store (which would be necessary to enable UAA -> SASL trust).
- Once the above issues with "Trusted Certificates" are resolved, "Trusted Certificates" will be moved out of the Experimental Features section.
- The "Generate Self-Signed Certificate" link no longer generates a self-signed certificate. (The certificate it generates is signed by Ops Manager's internal CA, which is generated once at installation time for each Ops Manager and persists across exports/imports.) The link will be renamed "Generate Certificate" in future versions of Ops Manager.

Existing Issues

- Ops Manager 1.5 does not support AWS S3 buckets in eu-central-1 (Frankfurt) due to an issue with Amazon's authentication APIs in this region. Therefore, Pivotal does not recommend customers use the eu-central-1 region for deployments that require high performance.
- On AWS environments, it is only possible to use "light-bosh" stemcells.
- If you are using OpenStack Juno, make sure you have applied this patch:
<https://bugs.launchpad.net/horizon/+bug/1394051>
- DNS and Gateway ICMP errors will occur on AWS even if ICMP is allowed.
- Resource Pools cannot be deleted from Availability Zones. If a resource pool is inadvertently deleted, run BOSH recreate.
- Compiled Packages are missing from installation exports, which can cause a failure if the product contained and referenced them. A fix is to SSH into the Ops Manager VM and remove references to compiled_packages in every file located in the `/var/tempest/workspace/metadata` directory.
- Unnecessary job instances must be reduced manually (MySQL, HA Proxy if RDS & ELBs are used).