Deploy Cloud Foundry

## 

[**Configure your AWS environment for Cloud Foundry**](#_siroehq7mcqu) **1**

[Allow Loggregator traffic](#_aingejohu0tw) 1

[Create a new Security Group](#_kuh2x2u2y4ki) 2

[Create an Elastic IP](#_s2vdrefwm1ir) 3

[**Prepare deployment**](#_nsy2dwkmiatr) **3**

[Clone the single-vm-cf repository](#_p78d3ewkvew8) 3

[Get the BOSH UUID](#_nv6frmej086v) 3

[Copy the manifest file](#_azrp4old0b1) 3

[Merge templates and create deployment manifest.](#_m0evicl205i0) 4

[**Deploy**](#_a2hxtfy2bxmd) **4**

[Upload a stemcell](#_x7jwerzhlv69) 4

[Upload the Cloud Foundry release](#_hzur38bzn8mg) 5

[Set the deployment manifest](#_spm6geg6kfa) 5

[Deploy!](#_1ugas0xnx5c4) 5

[**Verify the deployment**](#_r8gqsgt9n88v) **5**

[Run smoke tests](#_h47c8i37l8ny) 5

[Login to Cloud Foundry](#_vgtvmrw0x8aj) 5

## Configure your AWS environment for Cloud Foundry

Before deploying Cloud Foundry you need to configure your infrastructure. The following steps will guide you through necessary configurations for your AWS account, so that you can prepare it for the Cloud Foundry deployment.

#### Allow Loggregator traffic

Loggregator provides a highly-available (HA) and secure stream of logs and metrics for all applications and components on Cloud Foundry.

Add the following ingress rule to the BOSH security group in the bosh.tf file.

|  |
| --- |
| /\*\* Allow loggregator traffic \*\*/  ingress {  from\_port = 4443  to\_port = 4443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  } |

#### 

#### Create a new Security Group

Cloud Foundry uses its own security group to control what goes in into the Cloud Foundry deployment.

Open a new file named ~/deployment/cloudfoundry.tf and create a new security group:

|  |
| --- |
| resource "aws\_security\_group" "cf" {  name = "cf\_training\_sg"  description = "Security group for CF deployment"  vpc\_id = "${aws\_vpc.bosh.id}"  /\*\* Allow HTTP traffic \*\*/  ingress {  from\_port = 80  to\_port = 80  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  /\*\* Allow HTTPS traffic \*\*/  ingress {  from\_port = 443  to\_port = 443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  /\*\* Allow loggregator connections \*\*/  ingress {  from\_port = 4443  to\_port = 4443  protocol = "tcp"  cidr\_blocks = ["0.0.0.0/0"]  }  egress {  from\_port = "0"  to\_port = "0"  protocol = "-1"  cidr\_blocks = ["0.0.0.0/0"]  }  tags {  Name = "cf\_training\_sg"  }  } |

#### 

#### Create an Elastic IP

Cloud Foundry will need another Elastic IP to be able to serve external requests.

Create the new EIP and its output.

|  |
| --- |
| resource "aws\_eip" "cf" {  }  output "cf\_eip" {  value = "${aws\_eip.cf.public\_ip}"  } |

## Prepare deployment

Default Cloud Foundry installation is pretty large. Even if you reduce the number of instances to their minimum values, the resulting installation will require 18 instances. If you only plan to use your installation for educational or testing purposes, that is definitely an overhead. That's why we are going to use [single-vm-cf](https://github.com/Altoros/single-vm-cf) repository that contains special stubs that allow you to install all CF components to a single vm.

#### Clone the single-vm-cf repository

|  |
| --- |
| $ git clone https://github.com/Altoros/single-vm-cf ~/deployment/single-vm-cf $ cd ~/deployment/single-vm-cf $ git submodule init $ git submodule update $ cd ~/deployment/single-vm-cf/cf-release |

This repository includes standard cf-release repository as a submodule.

#### Get the BOSH UUID

Run the following command and take note of the output

|  |
| --- |
| $ bosh status --uuid |

#### Copy the manifest file

Save the following file as ~/deployment/single-vm-cf/cf-release/stub.yml

|  |
| --- |
| --- meta:  eip: YOUR\_CF\_ELASTIC\_IP  system\_domain: YOUR\_CF\_ELASTIC\_IP.nip.io  private\_ip: 10.0.0.10  aws:  availability\_zone: YOUR\_AVAILABILITY\_ZONE  subnet\_id: YOUR\_SUBNET\_ID  director\_uuid: YOUR\_BOSH\_DIRECTOR\_UUID networks: - name: default   subnets:  - range: 10.0.0.0/24  reserved:  - 10.0.0.2 - 10.0.0.9  static:  - 10.0.0.10 - 10.0.0.15  gateway: 10.0.0.1  dns:  - 10.0.0.2  cloud\_properties:  security\_groups:  - cf\_training\_sg  - bosh  subnet: (( meta.aws.subnet\_id )) - name: vip  type: vip |

#### 

#### Merge templates and create deployment manifest.

|  |
| --- |
| $ root\_dir=~/deployment/single-vm-cf/cf-release $ single\_vm\_templates="${root\_dir}/templates" $ cf\_templates="${root\_dir}/cf-release/templates" $ spiff merge \  "${cf\_templates}/generic-manifest-mask.yml" \  "${single\_vm\_templates}/jobs-single-vm-aws.yml" \  "${cf\_templates}/cf.yml" \  "${single\_vm\_templates}/infrastructure-single-vm-aws.yml" \  "${root\_dir}/stub.yml" > ~/deployment/cf-manifest.yml |

## 

## Deploy

Deploying Cloud Foundry with BOSH requires 3 easy steps.

#### Upload a stemcell

It is a versioned image of a bare-minimum OS skeleton, wrapped with IaaS-specific packaging. Deploying Cloud Foundry starts with specifying a stemcell, which BOSH installs on each component VM. To do that, run:

|  |
| --- |
| $ bosh upload stemcell https://bosh.io/d/stemcells/bosh-aws-xen-hvm-ubuntu-trusty-go\_agent?v=3312.24 |

#### Upload the Cloud Foundry release

|  |
| --- |
| $ bosh upload release https://bosh.io/d/github.com/cloudfoundry/cf-release?v=252 |

#### Set the deployment manifest

|  |
| --- |
| $ bosh deployment ~/deployment/cf-manifest.yml |

#### Deploy!

|  |
| --- |
| $ bosh deploy |

#### 

## Verify the deployment

#### Run smoke tests

|  |
| --- |
| $ bosh run errand smoke\_tests |

This test will fail, since we have to change our infrastructure to allow BOSH managed VMs to have access to external networks. This will be done in the “Avdanced” course, when we go thru how to do secure deployments.

#### Login to Cloud Foundry

There is a very convenient, manual way to test if you CF instance is ready:

|  |
| --- |
| $ cf login --skip-ssl-validation -a api.YOUR\_CF\_ELASTIC\_IP.nip.io -u admin -p admin $ cd ~/deployment $ git clone <https://github.com/Altoros/cf-example-sinatra.git> $ cd cf-example-sinatra $ cf create-space my-space -o cf-training $ cf target -o cf-training -s my-space $ cf push my-app |

Wait until the deployment process finishes and try access:

http://my-app.YOUR\_CF\_IP.nip.io