CLOUD FOUNDRY

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud Foundry

Version 1.11.a

Pivotal

Copyright Notice

Pivotal

Table of Contents

Requirements

1. Personal Experience

- Background in Software Development
- At ease with the command-line or "shell"
- Java/Spring experience is optional

2. Pivotal Cloud Foundry Environment

You will need a Pivotal Cloud Foundry environment to execute the labs. You may use a Pivotal Cloud Foundry instance or Pivotal Web Services.

If taking the course via e-learning, Pivotal Web Services is recommended, unless you have access to another Pivotal Cloud Foundry instance (typically already provisioned by your company). It is not recommended that you provision a personal Pivotal Cloud Foundry environment to complete the labs. The setup and installation of Pivotal Cloud Foundry is beyond the scope of this course.

If taking the course via instructor-led training, your instructor will provide guidance on the environment to use.

Note

If taking this course via e-learning, use a Pivotal Cloud Foundry account that your admin has setup for you (choose the `org` and `space` that works for you) or use Pivotal Web Services.

Throughout the labs you will see specific directions for Pivotal Cloud Foundry or Pivotal Web Services. *Make sure to follow the directions for the right environment.*

2.1. Option A: Pivotal Cloud Foundry

Pivotal Cloud Foundry may be used to complete these labs. The following dependencies are required:

- PCF Elastic Runtime version 1.7 or higher, with the following "tiles" added:

- MySQL for PCF
- PCF Metrics
- Redis for PCF

Tip

If using self-signed certificates, under the `Elastic Runtime Tile` -> `Settings` -> `Networking` page make sure the following options have been checked: `Disable SSL certificate verification for this environment` and `Ignore SSL certificate verification on route services`.

2.2. Option B: Pivotal Web Services

Pivotal Web Services may be used to complete these labs. Simply sign up for an account.

2.3. A Note About the Service Broker Lab

The Service Broker lab requires access to a MongoDB instance. Please review that lab for alternatives on how to meet that requirement. Options include using the AWS MongoDB AMI.

3. Local Machine

Minimum specification

- 4GB Memory
- 1G free disk space - if running MS Windows, 1G on your C: drive is preferred.
- Network access to the Pivotal Cloud Foundry environment and internet

You will need the ability to install software on your local machine. The following should be installed *prior* to the course:

- Java JDK is required for a couple of the labs
- Apache JMeter is provided for you and is required by the Auto Scaler Lab. (or download the most recent

version go here)

- A programmer-friendly text editor. Atom or Notepad++ (for Windows only) are popular options

3.1. Optional

- JSON Viewer (browser plugin)
  - JSON Formatter for Chrome
  - JSON Formatter for Firefox - you can download it directly from Mozilla
- Command Line HTTP client
  - Mac and Linux users already have `curl`. Windows Users can find an installer in the lab materials (or download the latest version here).
  - Alternatively, HTTPie is a nicer client if you prefer.
- Windows users may desire a better terminal experience than the *cmd.exe*.
  - Consider using Powershell (open a CMD windows and type "powershell" at the prompt).
  - Or use ConEmu. It provides multiple tabs and search which is helpful during these labs.
- A Java IDE such as the Spring Tool Suite can be useful for reading the implementation of the service broker and route service that we work with in those respective labs
- The Continuous Delivery Lab provides the option to use a pre-provisioned Jenkins instance or to install everything locally. If a you choose the latter option, then they’ll also need git and Maven

4. Windows VM

An alternative to installing and configuring the software on your local machine is to use our Windows VM.

If using the Windows VM see these requirements.

Windows VM

A Windows VM is available to complete the labs. The VM is hosted on Amazon Web Services. You will access it with remote desktop client.

1. Who Should Use?

Please consider using this VM if you have any of the following limitations:

- Your system does not meet the technical requirements
- You can't install software on your system
- You have limited network access

2. Requirements

- Remote Desktop Client on your system (most operating systems ship with one)
- Network connectivity and access to reach the Windows VM (RDP uses port 3389)

3. Provisioning

1. User must have access to an AWS account (can run from any account)
2. User must have an existing key pair in the account or create one
3. The AMI can be found be searching for `WinServer08STS` in the EC2 Dashboard. Must be in the N. Virginia region.

   Launch the AMI named `WinServer08STS`. In AWS, in the EC2 dashboard, using the US East (N. Virginia) region, select (or search for the public image) Images > AMIs > WinServer08STS. Click Launch.

   Note

   pick the AMI with the greatest number appended to the AMI Name. In this case, `WinServer08STS7` image::ami.png[search for ami]

4. When launching, accept defaults except:
   a. Choose an Instance Type > `t2.medium` (4 GB memory)
   b. Configure Instance Details > Auto-assign Public IP > `Enable`
   c. Tag Instance > Key = `Name`; Value = `<your name here-windows>`
5. Launch the instance. Once the instance is running, select it in list of EC2 instances. Obtain the public IP.
6. Using your Remote Desktop connect to the Windows VM using the the public IP.

   Username: `Administrator`

   Password: `KeepItSimple123`

4. Software Installed

The software listed below has been installed and configured on the VM:

- JDK 1.8 - License
- Maven - License
- git - License
- curl - License
- cf - License
- Spring Tool Suite - License
- Cygwin - License
- Chrome - License
- Json Formatter for Chrome

5. Best Practices

5.1. Work Directory

As part of the VM there is a `repos` directory. It is located at `C:\Users\Administrator\repos`. This is where you will clone repos from GitHub. This will be referred to as `$REPOS_HOME` throughout the labs.

5.2. ConEmu

The Windows console is very limited in the sense that it lacks tab support, search and other features one might like in a console/terminal experience.

Therefore, also installed is an alternative to the command prompt known as ConEmu.

We recommend using ConEmu to execute the labs. You can launch ConEmu from the desktop. It will open to your `$REPOS_HOME` directory.

5.2.1. Use Labeled Tabs

As part of the labs you will be starting many processes. Organizing the processes with tabs makes a ton of sense, otherwise you will have windows everywhere making it difficult to manage.

To label the tab execute the following:

```
title - <tab label>
```

For example, creating a tab for the `config-server`. This is where work done with the `config-server` would take place.

5.3. Odd Keyboard Behavior?

Sometimes when using Remote Desktop keys (`control`, `command`, `shift`, etc.) may become depressed by accident leading to what seems like very odd behavior on the remote machine. The problem can typically be resolved by toggling the depressed key(s). If needed the **On-Screen Keyboard** can also be brought up.

Open the **On-Screen Keyboard** by clicking the `Start` button, clicking `All Programs`, clicking `Accessories`, clicking `Ease of Access`, and then clicking `On-Screen Keyboard`.

Setting Up Mongo DB VM on AWS

1. Context

If you are attending an Instructor-led or Live-Online course, your instructor will do this for you.

If you are taking this course as E-Learning, you will need to do this yourself unless you already have access to a running MongoDB system.

2. Setup MongoDB VM

We will be using Terraform (from Hashicorp) to create our VM.

You should have a file called `mongo.tf` as part of your lab files in `pivotal-cloud-foundry-developer-workshop/terraform`.

Go to https://www.terraform.io, download and install the software. Ensure that `terraform` is on your `PATH`.

Login to your AWS account and navigate to the EC2 dashboard. *Make sure you have US East (N. Virginia) set as your active region.*

2.1. Keys and Configuration

You need a security key-pair. If you don’t have one already:

- In the left-hand panel, click `Key Pairs`.
- Then click `Create Key Pair` and specify your key-pair name
- Create the key-pair and download the PEM file when prompted.
- Copy the PEM file you have downloaded to the directory where you have placed the `mongo.tf` file
- If running on Unix (MacOS/Linux) restrict access using `chmod 400 <your-key>.pem`

- DO NOT lose this file, there is no way to recover it

The VM configuration is specified using the `mongo.tf` file that you should have downloaded previously. It takes a MongoDB AMI (VM template) and wraps security around it.

The `mongo.tf` file expects certain variables to make it work. Create a file called `variable.tf` and copy this text into it:

```
variable "access_key" {
    default = "AWS-ACCESS-KEY"
}

variable "secret_key" {
    default = "AWS-SECRET-KEY"
}

variable "key_name" {
    default = "YOUR-KEY-PAIR-NAME"
}

variable "key_path" {
    default = "/absolute/path/to/my-key-pair.pem"
}

variable "region" {
    default = "us-east-1"
}

variable "username" {
    default = "YOUR_AMS_LOGIN_NAME"
}
```

2.2. Creating the VM

For full details of using Terraform, go here: https://www.terraform.io/intro/getting-started/install.html.

Replace each of the defaults with their correct values.

Try and run your plan:

```
terraform mongo.tf plan
```

If it works, run

```
terraform mongo.tf apply
```

You may be prompted to go to AWS and purchase the VM instance you are about to create. Once AWS is happy, come back and run `terraform mongo.tf apply` again.

Once the script has run, it finishes by telling you the static IP address of your new VM. This is the `MONGODB_HOST` they need for the Service Btoker lab.

Return to EC2 dashboard and view your instance. Wait for it to finish starting up.

Select your VM, and in the Actions dropdown -> Instance Settings -> Get System Log. Scan the log for "Setting Bitnami application password to". See here for more details: https://docs.bitnami.com/aws/faq/#how-to-find-application-credentials. This is the `MONGODB_PASSWORD` needed for the lab.

2.3. Using the VM

This is all you need to do to finish the Service Broker Lab.

Once you have finished the lab, you can get rid of the VM by going to your AWS EC2 dashboard, selecting the MongoDB instance, and clicking `Actions` -> `Instance State` -> `Terminate`. The VM should stop, change to "Terminated" status and will eventually be removed from your list of VMs (The VM will shutdown and eventually be removed from your list of VMs (removal may not happen immediately)emoval may not happen immediately)

3. Accessing Your Mongo VM Remotely

If you wish you can access MongoDB and see what you have setup in the lab.

You can SSH into the VM using your private key, the `bitnami` username and the public IP address or hostname of your VM.

For example:

```
ssh -i my-key-pair.pem bitnami@53.91.185.0
```

For more information refer to the Amazon Reference Documentation

3.1. Managing MongoDB

Start MongoDB

```
sudo /opt/bitnami/ctlscript.sh start
```

Stop MongoDB

```
sudo /opt/bitnami/ctlscript.sh stop
```

Restart MongoDB

```
sudo /opt/bitnami/ctlscript.sh restart
```

See here for more information.

3.2. The Mongo Shell

The `mongo` shell is also loaded on the VM. It can be accessed in the terminal by doing the following:

```
mongo
```

Once in the shell, you'll have to authenticate. The username is `root`, and the password is the one you previously obtained from the VM system log:

```
use admin
db.auth('root', '<password>')
```

For more information refer to the Mongo Shell Reference

Amazon Web Services Jenkins AMI

A Jenkins VM (Ubuntu 14.04) is available to complete the continuous delivery lab. The VM is hosted on Amazon Web Services.

1. Requirements

Network connectivity and access for Jenkins to reach your Pivotal Cloud Foundry environment (push on your behalf).

2. Installed Software

The software listed below has been installed and configured on the VM:

- Jenkins - License
- JDK 1.8 - License
- Git - License
- GitHub Plugin - License
- Cloud Foundry Plugin - License

3. Provisioning

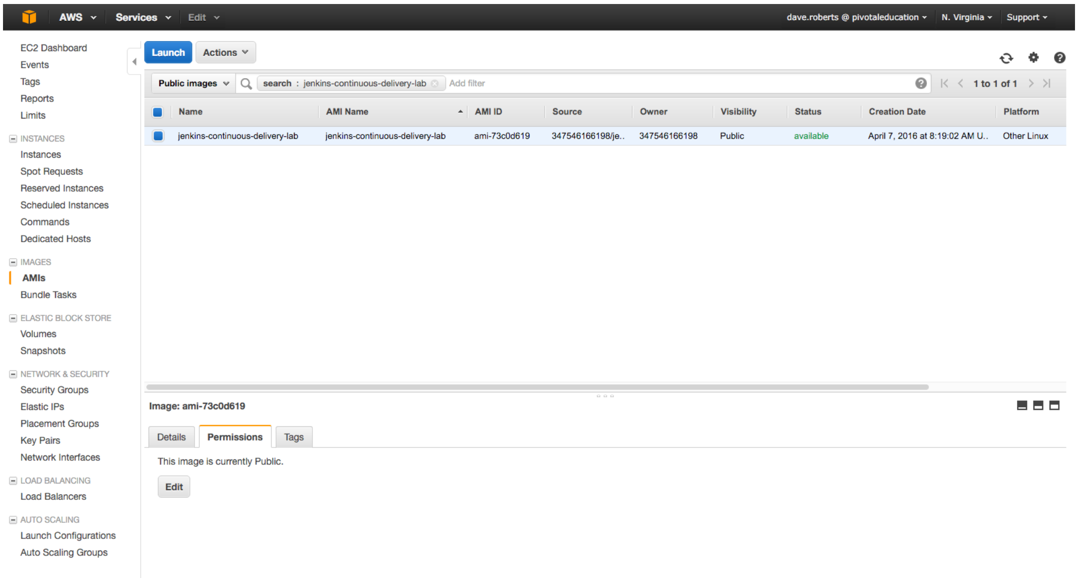
You can use the following `jenkins.tf` terraform script in your lab-files, in `pivotal-cloud-foundry-developer-workshop/terraform`.

Alternatively, you can provision the VM manually by following these directions:

1. User must have access to an AWS account (can run from any account)
2. User must have an existing key pair in the account or create one

3. The AMI can be found be searching for `jenkins-continuous-delivery-lab` in the EC2 Dashboard. Must be in the N. Virginia region.

In the left navbar of the EC2 dashboard, under `Images`, select `AMIs`. Then, to the left of the search bar, change `Owned by Me` to `Public images`. Search for `jenkins-continuous-delivery-lab`. Click Launch.



Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date	Platform
jenkins-continuous-delivery-lab	jenkins-continuous-delivery-lab	ami-73c0d619	347546166198/je..	347546166198	Public	available	April 7, 2016 at 8:19:02 AM U..	Other Linux

Image: ami-73c0d619

This image is currently Public.

3.1. Provisioning Wizard

When launching, accept defaults except where specified.

3.1.1. Choose an Instance Type

- Use `t2.medium` (4 GB memory)

E-learning Students: Please note that, though this instance type is not free-tier eligible, it is necessary for completing the continuous delivery lab and the PCF Developer course. Costs associated with its use are negligible. Charges cease as soon as the instance is deprovisioned (the steps for deprovisioning are at the end of the lab).

3.1.2. Configure Instance Details

Auto-assign Public IP = `Enable`

3.1.3. Add Storage

Accept defaults

3.1.4. Tag Instance

Create the following: Key = `Name`; Value = `{{{yourname}}}-jenkins`

3.1.5. Create a Security Group

Create a security group with the following rules:

Rule 1.

```
Type = `ssh`
Protocol = `TCP`
Port Range = `22`
Source = `anywhere` (0.0.0.0/0)
```

Rule 2.

```
Type = `HTTPS`
Protocol = `TCP`
Port Range = `443`
Source = `anywhere` (0.0.0.0/0)
```

3.1.6. Review Instance Launch

Select your key-pair an launch the instance.

Once the instance is running, select it in list of EC2 instances.

Obtain the public IP for use in the lab .

4. Accessing Your Jenkins VM Remotely

4.1. HTTPS

A self signed certificate is used, so browser warnings are expected.

Access Jenkins via the `https://{{{jenkins_ip_address}}}/` .

4.1.1. Credentials:

Username: `pivotal`

Password: `keepitsimple`

4.2. SSH

SSH using your private key, the `ubuntu` user and the public IP address of your VM.

For example:

```
ssh -i my-key-pair.pem ubuntu@{{jenkins_ip_address}}
```

[Amazon Reference Documentation]

5. Managing Jenkins

5.1. Start Jenkins

```
sudo service jenkins start
```

5.2. Stop Jenkins

```
sudo service jenkins stop
```

5.3. Restart Jenkins

```
sudo service jenkins restart
```