To implement the instructions in this post, you will need the following:

* AWS account
* Docker Hub account
* GitHub account

**Create a Jenkins CI server using Terraform**

Provisioning a Jenkins CI server manually can be error-prone and time-consuming, so I shall be configuring the Jenkins Continuous Server (CI) using Infrastructure as Code (IaC). For this post, I have decided to use Terraform. Log in to the AWS Management Console and [create an EC2 key pair](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair) .Using your GitHub account, fork the code sample repository at  <https://github.com/balasg70/myblock.git>.

From the AWS Cloud9 IDE, open a shell terminal and do the following (replace aws-samples with your GitHub account):

git clone <https://github.com/balasg70/myblock.git>

cd myblock/Eks-jenkins-terraform/terraform/

# terraform init

# terraform plan

# terraform apply -auto-approve

Terraform apply will also output the IP address of the Jenkins CI server as shown above.

Terraform will provision an AWS EC2 instance and install git, Apache Maven, Docker, Java 8, and Jenkins as shown in the install\_jenkins.sh file:

#!/bin/bash

sudo yum -y update

echo "Install Java JDK 8"

sudo yum remove -y java

sudo yum install -y java-1.8.0-openjdk

echo "Install Maven"

sudo yum install -y maven

echo "Install git"

sudo yum install -y git

echo "Install Docker engine"

sudo yum update -y

sudo yum install docker -y

sudo sudo chkconfig docker on

echo "Install Jenkins"

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo

sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

sudo yum install -y jenkins

sudo usermod -a -G docker jenkins

sudo chkconfig jenkins on

echo "Start Docker & Jenkins services"

sudo service docker start

sudo service jenkins start

Bash

Using a browser, open the page at [http://jenkins\_ip\_address:8080](http://ip-addess:8080/); the Jenkins admin page will be displayed:

Using the AWS EC2 instance shell terminal, log in to the Jenkins CI server, find the Administrator password by running the following command:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Enter this Administrator password on the Jenkins Console by pasting it into the input box, and click **Next**. Click Install suggested plugin.

**Install Halyard**

curl -O https://raw.githubusercontent.com/spinnaker/halyard/master/install/debian/InstallHalyard.sh

sudo bash InstallHalyard.sh

sudo update-halyard

hal -v

**Install Terraform**

wget https://releases.hashicorp.com/terraform/0.12.4/terraform\_0.12.4\_linux\_amd64.zip

unzip terraform\_0.12.4\_linux\_amd64.zip

sudo mv terraform /usr/local/bin/

export PATH=$PATH:/usr/local/bin/terraform

**Check the EKS on AWS Cloud**

**EKS cluster with AWS CLI** by using the command “aws eks update-kubeconfig --name eks\_spinnaker”. This command constructs a configuration with prepopulated server and certificate authority data values the cluster you specified.

aws eks update-kubeconfig --name eks-spinnaker --region us-east-2 \

--alias eks-spinnaker

aws eks update-kubeconfig --name eks-uat --region us-east-1 \

--alias eks-uat

aws eks update-kubeconfig --name eks-prod --region us-east-1 \

--alias eks-prod

kubectl apply --context $CONTEXT \

-f https://spinnaker.io/downloads/kubernetes/service-account.yml

**Configure Jenkins**

1. Plugins:

Log in to the Jenkins console, click Manage Jenkins → Manage Plugins → Available. Choose and install Docker plugin and GitHub Integration Plugin, then restart Jenkins by clicking the Restart Jenkins

2. Credentials:

Docker Hub: Click Credentials → global → Add Credentials, choose Username with password as Kind, enter the Docker Hub username and password and use dockerHubCredentials for ID.

GitHub: Click Credentials → Global → Add Credentials , choose Username with password as Kind, enter the GitHub username and password and use gitHubCredentials for ID.

**Configure the Jenkins job and pipeline**

From the Jenkins console, click New item. Choose Multibranch Pipeline, name it petclinic and click OK.

Choose GitHub and from the drop-down select the GitHub credentials. Enter the GitHub URL as shown below and click Save to save the Jenkins job.

The Jenkins build executor will check out and scan the GitHub repository and execute the stages in the pipeline as laid out in the Jenkins file shown below. Make sure that you replace the registry with your Docker registry URL inside the build stage.

Below is a screenshot of the final run; if all goes well, you will see a new Docker image pushed to your Docker registry.

**Create and configure Spinnaker pipelines**

A pipeline is a sequence of stages provided by Spinnaker, ranging from functions that manipulate infrastructure (deploy, resize, disable) to utility scaffolding functions (manual judgment, wait, run Jenkins job) that together precisely define your runbook for managing your deployments. Pipelines help you manage deployments consistently, repeatably, and safely.

1. Log in to the AWS EC2 instance environment and open a new terminal. Run the following command:

kubectl get svc -n spinnaker

2.  Using a browser, log in to the Spinnaker UI using the spin-deck-public services endpoint.

Select the Applications tab, then Actions → Create Application. Enter petclinic as Name and enter a valid email address, leave the rest of the fields blank.

3.  On the Pipelines tab, click Configure a new pipeline , enter DeployToUAT as the Pipeline Name and click Create.

4.  Click Add Artifact and choose GitHub → Kind , File path → kubernetes/petclinic.yaml, Display name → Petclinic-Manifest, Content URL → <https://github.com/balasg70/myblock/blob/master/Eks-jenkins-terraform/kubernetes/petclinic.yaml>

5.  Click Add Trigger and choose Type → Docker Registry, Registry Name → your Docker registry as configured in Spinnaker, Organization → your Docker registry name, Image → Docker image as created by Jenkins.

6.  Click Add Stage, choose Stage Type → Deploy (Manifest) , Account → eks-uat, Application → petclinic, Manifest Source → Artifact, Manifest Artifact → Petclinic-Manifest, Artifact Account → spinnaker-github.

7.  Click Save to save the changes to the DeployToUAT pipeline.

8.  Under the PIPELINES tab, click Create , enter ManualApproval as the Pipeline Name and click Create. Click Add Trigger and Choose Type → Pipeline, Application → petclinic, Pipeline → DeployToUAT.

9.  Click Add Stage, choose Stage Name → Manual Judgement, under Judgement Inputs add two options Approve and Reject as shown below:

10.  Click Save to save the changes to the ManualApproval pipeline.

11.  Under Pipelines tab, click Create , enter DeployToProd as the Pipeline Name and click **Create**. Click **Add Trigger** and Choose Type → Pipeline, Application → petclinic, Pipeline → DeployToProd.

12.  Click Add Artifact and choose GitHub → Kind , File path → kubernetes/petclinic.yaml, Display name → Petclinic-Manifest, Content URL → <https://github.com/balasg70/myblock/blob/master/Eks-jenkins-terraform/kubernetes/petclinic.yaml>

13.  Click Add Trigger and choose Type → Docker Registry, Registry Name → your Docker registry as configured in Spinnaker, Organization → your Docker registry name, Image → Docker image created by Jenkins.

14.  Click Add Stage, choose Stage Type → Deploy (Manifest) , Account → eks-prod, Application → petclinic, Manifest Source → Artifact, Manifest Artifact → Petclinic-Manifest, Artifact Account → spinnaker-github.

15.  Click Save to save the changes of the DeployToProd pipeline.

**Run Spinnaker pipelines manually**

Now run the three pipelines manually. Click Start Manual Execution, choose Pipeline → DeployToUAT, Type → Tag, Tag → enter a valid tag number. Click Run and watch the pipeline execution.