

## Assisted Practice: 5.3 Install Kubernetes

This section will guide you:

- To install Kubernetes on a standalone server in order to deploy a custom Docker container

This lab has mainly three subsections, namely:

5.1.1 Installing prerequisites packages

5.1.2 Configuring Kubernetes

5.1.3 Pushing the code to GitHub repositories

### **Step 5.1.1:** Installing prerequisites packages

- Kubernetes is already installed in your practice lab. (Refer FSD: Lab Guide - Phase 5)
- Run the **docker version** command to validate Docker.

#### **docker version**

```
root@ip-172-31-86-69:~# docker version
Client:
Version:           18.09.7
API version:       1.39
Go version:        go1.10.1
Git commit:        2d0083d
Built:            Wed Jul  3 12:13:59 2019
OS/Arch:          linux/amd64
Experimental:      false

Server:
Engine:
Version:          18.09.7
API version:      1.39 (minimum version 1.12)
Go version:       go1.10.1
Git commit:       2d0083d
Built:           Mon Jul  1 19:31:12 2019
OS/Arch:         linux/amd64
Experimental:     false
root@ip-172-31-86-69:~#
```

## Step 5.1.2: Configuring Kubernetes

- Configure Kubernetes using the procedure below.

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -  
echo "deb http://apt.kubernetes.io/ kubernetes-xenial main"  
>/etc/apt/sources.list.d/kubernetes.list  
apt-get update  
apt-get install -y kubelet kubeadm kubectl
```

```
root@ip-172-31-86-69:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -  
OK  
root@ip-172-31-86-69:~# echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >/etc/apt/sources.list.d/kubernetes.list  
root@ip-172-31-86-69:~# apt-get update  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease  
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]  
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [27.5 kB]  
Fetched 36.5 kB in 1s (64.3 kB/s)  
Reading package lists... Done  
root@ip-172-31-86-69:~# apt-get install -y kubelet kubeadm kubectl  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  conntrack cri-tools kubernetes-cni socat  
The following NEW packages will be installed:  
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni socat  
0 upgraded, 7 newly installed, 0 to remove and 2 not upgraded.  
Need to get 52.9 MB of archives.  
After this operation, 280 MB of additional disk space will be used.  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 conntrack amd64 1:1.4.4+snapshot20161117-6ubuntu2 [30.6 kB]
```

- Initialize Kubernetes to deploy containers using Kubernetes CLI.

### kubeadm init

```
root@ip-172-31-86-69:~# kubeadm init  
[init] Using Kubernetes version: v1.15.0  
[preflight] Running pre-flight checks  
[WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'  
[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver  
tes.io/docs/setup/cri/  
[preflight] Pulling images required for setting up a Kubernetes cluster  
[preflight] This might take a minute or two, depending on the speed of your internet connection  
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'  
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"  
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
```

- Once Kubernetes is initialized, configure Kubernetes to start using the Kubernetes cluster.

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.86.69:6443 --token 7jp400.ldgg81o8qzqwdriwa \
--discovery-token-ca-cert-hash sha256:50515e1fd7c9454ab794ba72f8d4f5ad30433b3be83126e868817e0114198e9d
root@ip-172-31-86-69:~#
```

- After the cluster gets started, deploy a weave network to the cluster.

```
export kubever=$(kubectl version | base64 | tr -d '\n')
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
```

```
root@ip-172-31-86-69:~# export kubever=$(kubectl version | base64 | tr -d '\n')
root@ip-172-31-86-69:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.extensions/weave-net created
root@ip-172-31-86-69:~# kubectl get node
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-86-69     NotReady  master   12m   v1.15.0
root@ip-172-31-86-69:~# kubectl get node
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-86-69     Ready     master   12m   v1.15.0
root@ip-172-31-86-69:~#
```

- With weave network deployment, validate that the node is up and running. That will help to deploy a Docker container to the Kubernetes cluster.

```
kubectl get node
```

```
kubectl get pods --all-namespaces
```

```

root@ip-172-31-86-69:~# kubectl get node
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-86-69    Ready    master   15m   v1.15.0
root@ip-172-31-86-69:~# kubectl get pods --all-namespaces
NAMESPACE   NAME                                                    READY   STATUS    RESTARTS   AGE
kube-system  coredns-5c98db65d4-6x7g2                              1/1     Running   0           15m
kube-system  coredns-5c98db65d4-zz14t                              1/1     Running   0           15m
kube-system  etcd-ip-172-31-86-69                                    1/1     Running   0           13m
kube-system  kube-apiserver-ip-172-31-86-69                        1/1     Running   0           14m
kube-system  kube-controller-manager-ip-172-31-86-69               1/1     Running   0           14m
kube-system  kube-proxy-4n9br                                       1/1     Running   0           15m
kube-system  kube-scheduler-ip-172-31-86-69                        1/1     Running   0           14m
kube-system  weave-net-ht9nf                                         2/2     Running   0           3m2s
root@ip-172-31-86-69:~#

```

### Step 5.1.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

- Initialize your repository using the following command:

**git init**

- Add all the files to your git repository using the following command:

**git add .**

- Commit the changes using the following command:

**git commit . -m "Changes have been committed."**

- Push the files to the folder you initially created using the following command:

**git push -u origin master**

# Assisted Practice: 5.4 Install Kubernetes on Cloud

This section will guide you to:

- Install Kubernetes on AWS cloud
- Understand EKS service to create an EKS cluster

This lab has mainly three subsections, namely:

5.2.1 Creating an AWS EKS cluster

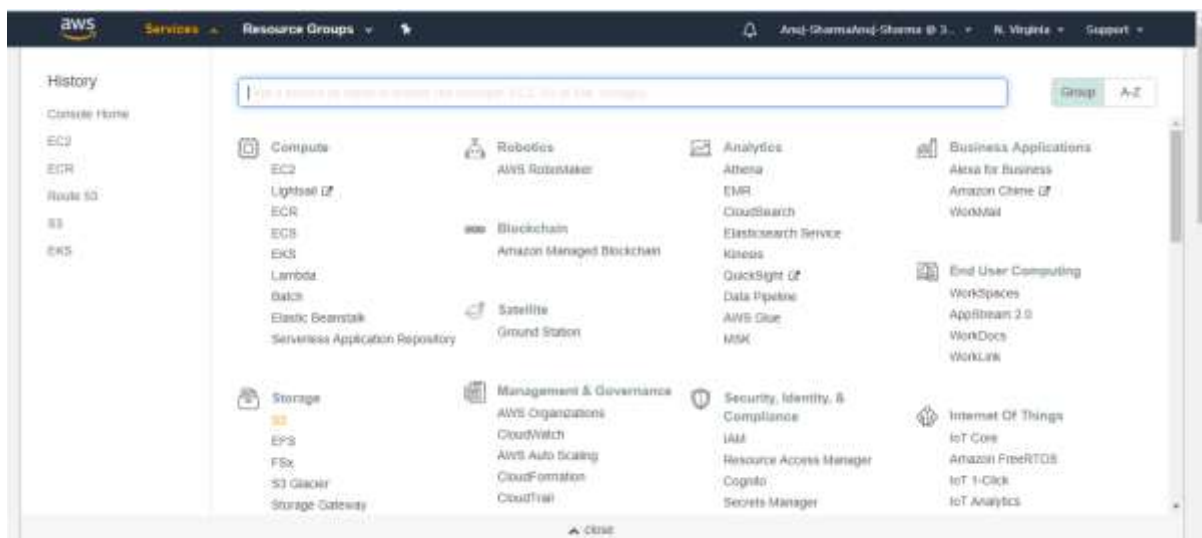
5.2.2 Setting up kubectl command line with EKS cluster

5.2.3 Pushing the code to GitHub repositories

- Kubernetes is already installed in your practice lab. (Refer FSD: Lab Guide - Phase 5)

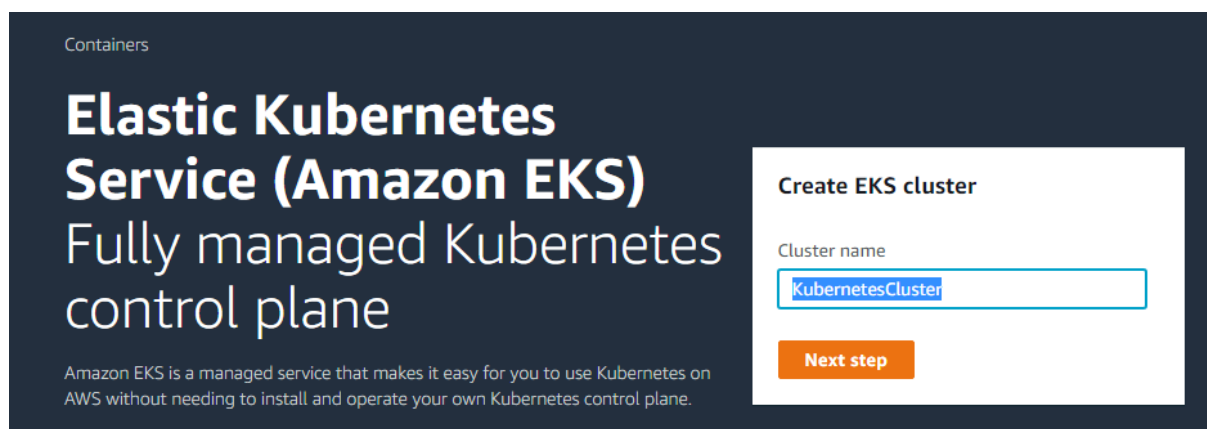
## **Step 5.2.1:** Creating an AWS EKS cluster

- Connect to AWS console and navigate to EKS service to create an EKS cluster.





- Provide a cluster name and click on **Next Step**.



- Configure **Role name** used by EKS rest.

#### [Role name](#)

Select the IAM Role to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.

- Click on **Create Role** and provide the policy details. Select **EKS** from the service list.

# Create role

- 1
- 2
- 3
- 4

## Select type of trusted entity

**AWS service**  
EC2, Lambda and others

**Another AWS account**  
Belonging to you or 3rd party

**Web identity**  
Cognito or any OpenID provider

**SAML 2.0 federation**  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

## Choose the service that will use this role

**EC2**  
Allows EC2 instances to call AWS services on your behalf.

**Lambda**  
Allows Lambda functions to call AWS services on your behalf.

API Gateway	Comprehend	ElastiCache	Lex	SMS
AWS Backup	Config	Elastic Beanstalk	License Manager	SNS
AWS Support	Connect	Elastic Container Service	Machine Learning	SWF
Amplify	DMS	Elastic Transcoder	Macie	SageMaker

# Create role

- 1
- 2
- 3
- 4

## Attached permissions policies

The type of role that you selected requires the following policy.

Filter policies <input type="text" value="Search"/>			Showing 2 results
Policy name	Used as	Description	
 AmazonEKSClusterPolicy	None	This policy provides Kubernetes the permissio...	
 AmazonEKSServicePolicy	None	This policy allows Amazon Elastic Container S...	

## Create role

1 2 3 4

### Review

Provide the required information below and review this role before you create it.

Role name\*

EKSRoleName

Use alphanumeric and "+=, @ \_ -" characters. Maximum 64 characters.

Role description

Allows EKS to manage clusters on your behalf.

Maximum 1000 characters. Use alphanumeric and "+=, @ \_ -" characters.

Trusted entities

AWS service: eks.amazonaws.com

Policies

 AmazonEKSClusterPolicy [↗](#)

 AmazonEKSServicePolicy [↗](#)

\* Required

Cancel

Previous

Create role

- Select the newly created role name from the list while creating the EKS cluster.

### General configuration

Cluster name

Enter a unique name for your Amazon EKS cluster.

Kubernetes Version

Select the Kubernetes version to install.

1.13 ▼

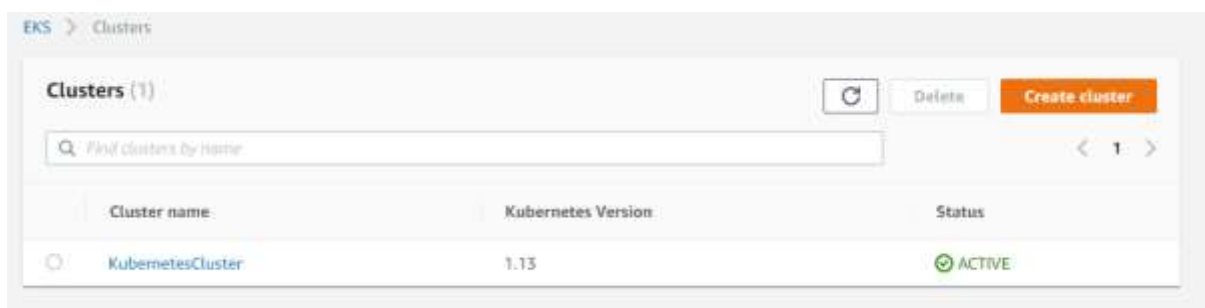
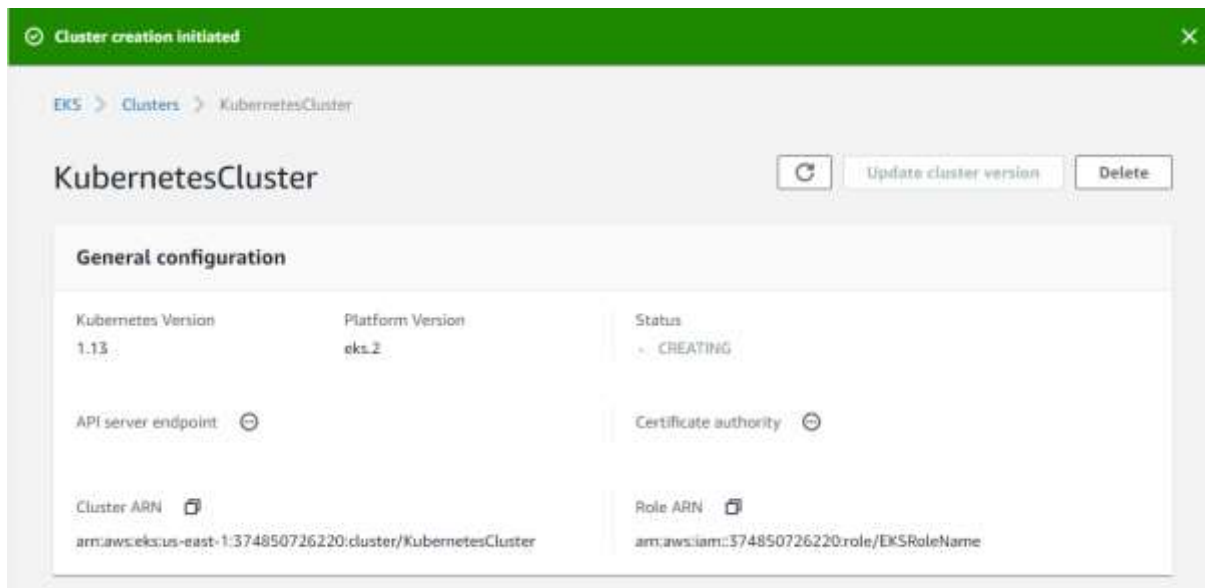
Role name [↗](#)

Select the IAM Role to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.

EKSRoleName ▼

**Please Note:** Once configurations are saved and the EKS cluster is created, it may take some time to bring the cluster online. Configure **kubectrl** command line only when EKS will be completely online.





### Step 5.2.2: Setting up kubectl command line with EKS cluster

**Please Note:** Amazon EKS clusters require kubectl and kubelet binaries and the AWS IAM Authenticator for Kubernetes to allow IAM authentication for our Kubernetes cluster.

- Download the Amazon EKS-vended kubectl binary from Amazon S3:

Linux: <https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl>

- Follow steps as shown in the screenshot below.

```
wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
```

```
chmod +x kubectl
```

## ./kubectl

```
root@ip-172-31-17-73:~# wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
--2018-07-28 02:03:07-- https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
Resolving amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)... 52.218.253.65
Connecting to amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)|52.218.253.65|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54146332 (52M) [binary/octet-stream]
Saving to: 'kubectl'

kubectl                                100%[=====] 51.64M  7.63MB/s
2018-07-28 02:03:14 (7.41 MB/s) - 'kubectl' saved [54146332/54146332]

root@ip-172-31-17-73:~# ./kubectl
-ma: ./kubectl: Permission denied
root@ip-172-31-17-73:~# chmod +x kubectl
root@ip-172-31-17-73:~# ./kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/
```

- Configure **kubectl** in **PATH** variable to call **kubectl** command globally. Follow the set of commands mentioned below to configure **PATH** variable.

## mkdir bin

**cp ./kubectl \$HOME/bin/kubectl && export PATH=\$HOME/bin:\$PATH**

## kubectl version

**kubectl version --short --client**

```
root@ip-172-31-17-73:~# mkdir bin
root@ip-172-31-17-73:~# cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
root@ip-172-31-17-73:~# kubectl version
Client Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.3", GitCommit:"2bba0
-26T20:40:11Z", GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}
```

- Configure AWS CLI and **aws-iam-authenticator**. Download the Amazon EKS-vended aws-iam-authenticator binary from Amazon S3:

Linux: <https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator>

**wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator**

**chmod +x ./aws-iam-authenticator**

**cp ./aws-iam-authenticator \$HOME/bin/aws-iam-authenticator && export PATH=\$HOME/bin:\$PATH**

**aws-iam-authenticator help**

```

root@ip-172-31-17-73:~# wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2019-07-26/bin/linux/amd64/aws-iam-authenticator
--2019-07-28 02:11:02-- https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2019-07-26/bin/linux/amd64/aws-iam-authenticator
Resolving amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)... 52.218.193.133
Connecting to amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)|52.218.193.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26349462 (25M) [binary/octet-stream]
Saving to: 'aws-iam-authenticator'

aws-iam-authenticator      100%[=====]
2019-07-28 02:11:05 (9.03 MB/s) - 'aws-iam-authenticator' saved [26349462/26349462]

root@ip-172-31-17-73:~# chmod +x ./aws-iam-authenticator
root@ip-172-31-17-73:~# cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH
root@ip-172-31-17-73:~# aws-iam-authenticator help
A tool to authenticate to Kubernetes using AWS IAM credentials

```

- Install AWS CLI using the sequence of commands mentioned below.

**apt install python-pip**

**pip install awscli**

**aws --version**

- Create **Access keys** in AWS IAM Console.

#### Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. [Learn more](#)

Create access key

Access key ID	Created	Last used	Status	
AKIAVORWYFFGC3WVPNWC	2019-07-24 08:28 UTC+0530	2019-07-26 13:51 UTC+0530 with sts in us-east-1	Active	Make inactive ✕

**Please Note:** Create Access keys and keep them saved in a document.

Create access key

Access key ID	Created	Last used	Status	
AKIAVORWYFFGC3WVPNWC	2019-07-24 08:28 UTC+0530	2019-07-26 13:51 UTC+0530 with sts in us-east-1	Active	Make inactive ✕
AKIAVORWYFFGE3YTFZFZ	2019-07-28 07:49 UTC+0530	N/A	Active	Make inactive ✕

- Configure AWS CLI and provide **Access Keys** and **Secret Access Keys** while configuring it.

```
root@ip-172-31-17-73:~# aws configure
AWS Access Key ID [None]: AKIAVORWYFFGE3YTF2FZ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
root@ip-172-31-17-73:~#
```

- Update the local **kubeconfig** to configure locally installed **kubectl** with EKS configurations.

```
aws eks --region us-east-1 update-kubeconfig --name KubernetesCluster
```

```
kubectl get svc
```

```
root@ip-172-31-17-73:~# aws eks --region us-east-1 update-kubeconfig --name KubernetesCluster
Updated context arn:aws:eks:us-east-1:374850726220:cluster/KubernetesCluster in /root/.kube/config
root@ip-172-31-17-73:~# kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.100.0.1    <none>         443/TCP    32m
root@ip-172-31-17-73:~#
```

### Step 5.2.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

```
cd <folder path>
```

- Initialize your repository using the following command:

```
git init
```

- Add all the files to your git repository using the following command:

```
git add .
```

- Commit the changes using the following command:

```
git commit . -m "Changes have been committed."
```

- Push the files to the folder you initially created using the following command:

**git push -u origin master**

## Assisted Practice: 5.5 Web Hosting

This section will guide you to:

- Build a custom Docker image with Spring Boot application and deploy it to AWS EKS

This lab has mainly three subsections, namely:

5.3.1 Creating a custom Docker image

5.3.2 Deploying a Spring Boot application to AWS EKS

5.3.3 Pushin the code to GitHub repositories

### **Step 5.3.1:** Creating a custom Docker image

- Follow the set of commands shown below to build a custom Docker image:

```
git clone https://github.com/Anuj1990/SpringBootDocker.git
```

```
ls -lart
```

```
root@ip-172-31-86-69:~# git clone https://github.com/Anuj1990/SpringBootDocker.git
Cloning into 'SpringBootDocker'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 52 (delta 4), reused 52 (delta 4), pack-reused 0
Unpacking objects: 100% (52/52), done.
root@ip-172-31-86-69:~# cd SpringBootDocker/
root@ip-172-31-86-69:~/SpringBootDocker# ls -lart
total 24
drwx----- 7 root root 4096 Jul 25 02:33 ..
drwxr-xr-x 4 root root 4096 Jul 25 02:33 src
-rw-r--r-- 1 root root 2601 Jul 25 02:33 pom.xml
-rw-r--r-- 1 root root 207 Jul 25 02:33 Dockerfile
drwxr-xr-x 8 root root 4096 Jul 25 02:33 .git
drwxr-xr-x 4 root root 4096 Jul 25 02:33 .
root@ip-172-31-86-69:~/SpringBootDocker#
```

- Build source code to generate artifacts which can be deployed on Docker host.

### mvn clean install

```
root@ip-172-31-86-69:~/SpringBootDocker# mvn clean install
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar)
Class(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:demo-docker >-----
[INFO] Building demo-docker 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.0.0:clean (default-clean) @ demo-docker ---
[INFO] Deleting /root/SpringBootDocker/target
[INFO]
```

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ demo-docker ---
[INFO] Building jar: /root/SpringBootDocker/target/demo-docker-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.0.5.RELEASE:repackage (default) @ demo-docker ---
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ demo-docker ---
[INFO] Installing /root/SpringBootDocker/target/demo-docker-0.0.1-SNAPSHOT.jar to /root/.m2/repository/com/example/demo-docker/0.0.1-SNAPSHOT/demo-docker-0.0.1-SNAPSHOT.jar
[INFO] Installing /root/SpringBootDocker/pom.xml to /root/.m2/repository/com/example/demo-docker/0.0.1-SNAPSHOT/pom.xml
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 8.341 s
[INFO] Finished at: 2019-07-25T02:35:10Z
[INFO] -----
root@ip-172-31-86-69:~/SpringBootDocker#
```

- Deploy this artifact inside the custom Docker image using **docker build** command line. Follow the steps shown below to create the custom Docker image:

**docker build -t springbootapp .**

```

root@ip-172-31-86-69:~/SpringBootDocker# docker build -t springbootapp .
Sending build context to Docker daemon 30.99MB
Step 1/5 : FROM java:8-jdk-alpine
--> 3fd9dd82815c
Step 2/5 : COPY ./target/demo-docker-0.0.1-SNAPSHOT.jar /usr/app/
--> 03af141fea64
Step 3/5 : WORKDIR /usr/app
--> Running in c5873bb5c094
Removing intermediate container c5873bb5c094
--> c7628e48b550
Step 4/5 : RUN sh -c 'touch demo-docker-0.0.1-SNAPSHOT.jar'
--> Running in 090cab39b1ed
Removing intermediate container 090cab39b1ed
--> 80f5bfb8c92e
Step 5/5 : ENTRYPOINT ["java","-jar","demo-docker-0.0.1-SNAPSHOT.jar"]
--> Running in e3d6aaa482cc
Removing intermediate container e3d6aaa482cc
--> 5a26279c1de0
Successfully built 5a26279c1de0
Successfully tagged springbootapp:latest
root@ip-172-31-86-69:~/SpringBootDocker# docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
springbootapp	latest	5a26279c1de0	4 seconds ago	177MB
java	8-jdk-alpine	3fd9dd82815c	2 years ago	145MB

```

root@ip-172-31-86-69:~/SpringBootDocker#

```

- Push this image to Docker Hub. Follow the command below to do so.

**docker images**

**docker tag springbootapp anujsharma1990/springboot**

**docker push anujsharma1990/springboot**

```

root@ip-172-31-86-69:~# docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
springbootapp	latest	5a26279c1de0	6 days ago	177MB
java	8-jdk-alpine	3fd9dd82815c	2 years ago	145MB

```

root@ip-172-31-86-69:~# docker tag springbootapp anujsharma1990/springboot
root@ip-172-31-86-69:~# docker push anujsharma1990/springboot
The push refers to repository [docker.io/anujsharma1990/springboot]
3b9dfb836448: Pushed
e817cce62ea5: Pushed
a1e7033f082e: Mounted from library/java
78075328e0da: Mounted from library/java
9f8566ee5135: Mounted from library/java
latest: digest: sha256:6705b88d681e987bb8ef39339b75421fec6a65675b128b90a36a3d8dfe51a93c8 size: 1371
root@ip-172-31-86-69:~#

```

### Step 5.3.2: Deploying a Spring Boot application to AWS EKS

- Configure **kubectl command line** and deploy containers to AWS EKS.

**export PATH=\$HOME/bin:\$PATH**



## kubectl get node

```
root@ip-172-31-86-69:~# export PATH=$HOME/bin:$PATH
root@ip-172-31-86-69:~# kubectl get node
NAME                                                    STATUS    ROLES    AGE    VERSION
ip-192-168-23-105.us-west-2.compute.internal          Ready     <none>   10m    v1.13.7-eks-c57ff8
ip-192-168-72-78.us-west-2.compute.internal           Ready     <none>   10m    v1.13.7-eks-c57ff8
root@ip-172-31-86-69:~#
```

- Create Kubernetes deployment and service using the set of commands given below:

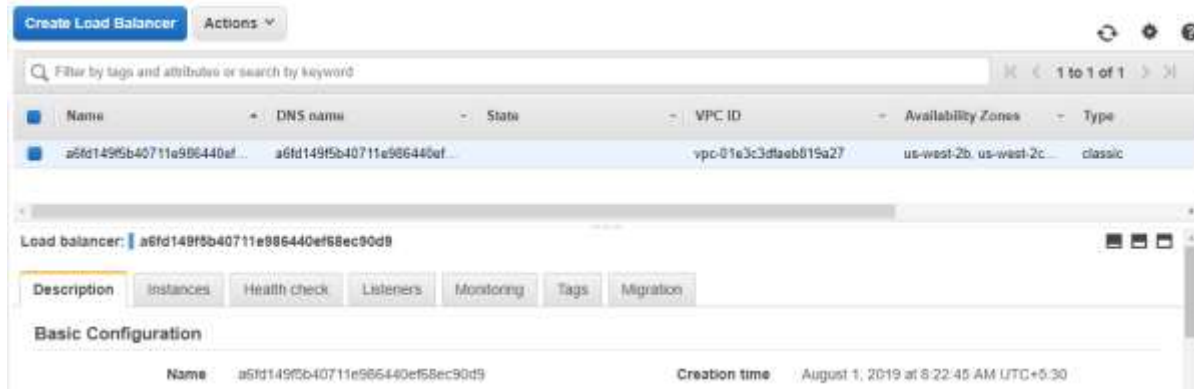
```
kubectl run springbootapp--image=anujsharma1990/springboot --port=8080
```

```
kubectl expose deployment/springbootapp --port=8080 --target-port=8080 --
type=LoadBalancer
```

```
root@ip-172-31-86-69:~# kubectl run springbootapp --image=anujsharma1990/springboot --port=8080
deployment.apps "springbootapp" created
root@ip-172-31-86-69:~# kubectl expose deployment/springbootapp --port=8080 --target-port=8080 --type=LoadBalancer
service "springbootapp" exposed
root@ip-172-31-86-69:~# kubectl get deployments
NAME            DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
springbootapp   1          1          1             1            11s
root@ip-172-31-86-69:~# kubectl get pods
NAME                                READY     STATUS    RESTARTS    AGE
springbootapp-b6f746b89-sj2sq       1/1      Running   0            16s
root@ip-172-31-86-69:~# kubectl get services
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes      ClusterIP   10.100.0.1    <none>         443/TCP        45m
springbootapp   LoadBalancer 10.100.132.0  a6fd149f5b407... 8080:31060/TCP 17s
root@ip-172-31-86-69:~#
```

**Please Note:** Once the pod is deployed, we can get the Load Balancer URL from springbootapp EKS Service. EKS will automatically configure the Load Balancer in AWS.

```
root@ip-172-31-86-69:~# kubectl describe svc springbootapp
Name:                springbootapp
Namespace:           default
Labels:              run=springbootapp
Annotations:         <none>
Selector:            run=springbootapp
Type:                LoadBalancer
IP:                  10.100.132.0
LoadBalancer Ingress: a6fd149f5b40711e986440ef68ec90d9-1889437699.us-west-2.elb.amazonaws.com
Port:                <unset> 8080/TCP
```



- To access the Spring Boot application, use the **Load Balancer URL** as shown below.

```
curl -w "\n" a6fd149f5b40711e986440ef68ec90d9-1889437699.us-west-2.elb.amazonaws.com:8080/greet/EKSSpringboot
```

```
root@ip-172-31-86-69:~# curl -w "\n" a6fd149f5b40711e986440ef68ec90d9-1889437699.us-west-2.elb.amazonaws.com:8080/greet/EKSSpringboot
Hi! EKSSpringboot
root@ip-172-31-86-69:~#
```

### Step 5.3.3: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

- Initialize your repository using the following command:

**git init**

- Add all the files to your git repository using the following command:

**git add .**

- Commit the changes using the following command:

**git commit . -m "Changes have been committed."**

- Push the files to the folder you initially created using the following command:

**git push -u origin master**

## Assisted Practice: 5.6 Deploy your Application

This section will guide you to:

- Configure Kubernetes cluster to push a random application to AWS EKS

This lab has mainly four subsections, namely:

5.4.1 Setting up EKS CTL command line and dependencies

5.4.2 Creating an EKS cluster using eksctl command line

5.4.3 Deploying an application to AWS EKS cluster

5.4.4 Pushing the code to GitHub repositories

### **Step 5.4.1:** Setting up EKS CTL command line and dependencies

**Please Note:** Amazon EKS clusters require **kubectl**, **kubelet** binaries, and AWS IAM Authenticator for Kubernetes to allow IAM authentication for Kubernetes cluster.

- Download the Amazon EKS-vended kubectl binary from Amazon S3:

Linux: <https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl>

- Follow the steps shown below in the screenshot.

```
wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
```

```
chmod +x kubectl
```

```
./kubectl
```

```

root@ip-172-31-17-73:~# wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
--2018-07-26 02:03:07-- https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
Resolving amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)... 52.210.253.65
Connecting to amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)|52.210.253.65|443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54146332 (52M) [binary/octet-stream]
Saving to: 'kubectl'

kubectl                                     100%[=====>] 51.44M  7.83MB/s
2018-07-26 02:03:14 (7.41 MB/s) - 'kubectl' saved [54146332/54146332]

root@ip-172-31-17-73:~# ./kubectl
-mv: ./kubectl: Permission denied
root@ip-172-31-17-73:~# chmod +x kubectl
root@ip-172-31-17-73:~# ./kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

```

- Configure **kubectl** in PATH variable to call **kubectl** command globally. Follow the set of commands given below to configure PATH variable:

**mkdir bin**

**cp ./kubectl \$HOME/bin/kubectl && export PATH=\$HOME/bin:\$PATH**

**kubectl version**

**kubectl version --short --client**

```

root@ip-172-31-17-73:~# mkdir bin
root@ip-172-31-17-73:~# cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
root@ip-172-31-17-73:~# kubectl version
Client Version: version.Info{Major:"1", Minor:"10", GitVersion:"v1.10.3", GitCommit:"2bba0
-26T20:40:11Z", GoVersion:"go1.9.3", Compiler:"gc", Platform:"linux/amd64"}

```

- Configure AWS CLI and aws-iam-authenticator. Follow the set of commands given below to install these command lines. Download the Amazon EKS-vended aws-iam-authenticator binary from Amazon S3:

Linux: <https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator>

**wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator**

**chmod +x ./aws-iam-authenticator**

**cp ./aws-iam-authenticator \$HOME/bin/aws-iam-authenticator && export PATH=\$HOME/bin:\$PATH**

**aws-iam-authenticator help**

```

root@ip-172-31-17-73:~# wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator
--2019-07-28 02:11:02-- https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator
Resolving amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)... 52.218.193.153
Connecting to amazon-eks.s3-us-west-2.amazonaws.com (amazon-eks.s3-us-west-2.amazonaws.com)|52.218.193.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26349462 (25M) [binary/octet-stream]
Saving to: 'aws-iam-authenticator'

aws-iam-authenticator 100%[=====>]
2019-07-28 02:11:05 (9.03 MB/s) - 'aws-iam-authenticator' saved [26349462/26349462]

root@ip-172-31-17-73:~# chmod +x ./aws-iam-authenticator
root@ip-172-31-17-73:~# cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH
root@ip-172-31-17-73:~# aws-iam-authenticator help
A tool to authenticate to Kubernetes using AWS IAM credentials

```

- Install **EKS CTL command line** to create an EKS cluster.

```

curl --silent --location
"https://github.com/weaveworks/eksctl/releases/download/latest_release/ek
sctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

mv /tmp/eksctl /usr/local/bin

eksctl version

```

```

root@ip-172-31-86-69:~# curl --silent --location "https://github.com/weaveworks/eksctl/1
p
root@ip-172-31-86-69:~# mv /tmp/eksctl /usr/local/bin
root@ip-172-31-86-69:~# eksctl version
[â
  '] version.Info{BuiltAt:"", GitCommit:"", GitTag:"0.2.1"}
root@ip-172-31-86-69:~# █

```

- Install AWS CLI using the sequence of commands given below.

```

apt install python-pip

pip install awscli

aws --version

```

- Configure AWS CLI. We need to create **Access Keys** in AWS IAM Console.

## Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. [Learn more](#)

Create access key

Access key ID	Created	Last used	Status	
AKIAVORWYFFGC3WVPNWC	2019-07-24 08:28 UTC+0530	2019-07-26 13:51 UTC+0530 with sts in us-east-1	Active	<a href="#">Make inactive</a> <a href="#">✕</a>

- Click on **Create Access key** and keep the keys safe with you.

Create access key

Access key ID	Created	Last used	Status	
AKIAVORWYFFGC3WVPNWC	2019-07-24 08:28 UTC+0530	2019-07-26 13:51 UTC+0530 with sts in us-east-1	Active	<a href="#">Make inactive</a> <a href="#">✕</a>
AKIAVORWYFFGE3YTFZFZ	2019-07-28 07:49 UTC+0530	N/A	Active	<a href="#">Make inactive</a> <a href="#">✕</a>

- Configure AWS CLI and provide **Access Keys and Secret Access Keys** while configuring AWS CLI.

```
root@ip-172-31-17-73:~# aws configure
AWS Access Key ID [None]: AKIAVORWYFFGE3YTFZFZ
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
root@ip-172-31-17-73:~#
```

## Step 5.4.2: Creating an EKS cluster using eksctl command line

- Create an EKS Cluster using the command below:

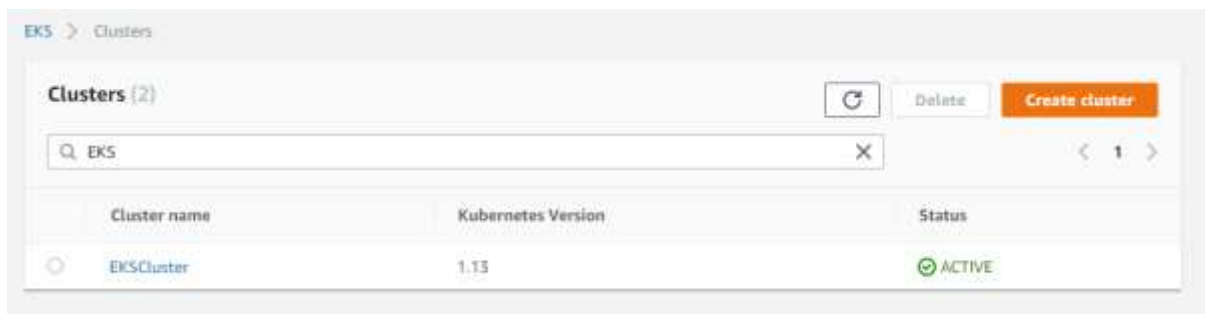
```
eksctl create cluster --name=EKScluster --nodes=2 --region=us-west-2
```

```
root@ip-172-31-06-69:~# eksctl create cluster --name=EKScluster --nodes=2 --region=us-west-2
{
  "using region us-west-2"
}
{
  "setting availability zones to [us-west-2c us-west-2d us-west-2b]"
}
{
  "subnets for us-west-2c - public:192.168.0.0/19 private:192.168.96.0/19"
}
{
  "subnets for us-west-2d - public:192.168.32.0/19 private:192.168.128.0/19"
}
{
  "subnets for us-west-2b - public:192.168.64.0/19 private:192.168.160.0/19"
}
{
  "nodegroup \"ng-c8e07a8f\" will use \"ami-0ba55127e61394e7\" [AmazonLinux2/1.19]"
}
{
  "using Kubernetes version 1.13"
}
{
  "creating EKS cluster \"EKScluster\" in \"us-west-2\" region"
}
{
  "will create 2 separate CloudFormation stacks for cluster itself and the initial nodegroup"
}
{
  "if you encounter any issues, check CloudFormation console or try \"eksctl utils describe-stacks --region=us-west-2 --name=EKScluster\""
}
{
  "2 sequential tasks: ( create cluster control plane \"EKScluster\", create nodegroup \"ng-c8e07a8f\" )"
}
{
  "building cluster stack \"eksctl-EKScluster-cluster\""
}
{
  "deploying stack \"eksctl-EKScluster-cluster\""
}
```

```
[4] all EKS cluster resources for "EKSCluster" had been created
[4] saved kubeconfig as "/root/.kube/config"
[4]
*) adding role "arn:aws:iam::130174662735:role/eksctl-EKSCluster-nodegroup-ng-c8-NodeInstanceRole-LFN1CH9HJUUHU" to auth ConfigMap
[4]
*) nodegroup "ng-c8e07a6f" has 0 node(s)
[4]
*) waiting for at least 1 node(s) to become ready in "ng-c8e07a6f"
[4]
*) nodegroup "ng-c8e07a6f" has 2 node(s)
[4]
*) node "ip-192-168-28-149.us-west-2.compute.internal" is ready
[4]
*) node "ip-192-168-76-186.us-west-2.compute.internal" is ready
[4]
*) kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
[4] EKS cluster "EKSCluster" in "us-west-2" region is ready
```

- Validate the cluster using **kubectl get node** command through AWS Console.

```
root@ip-172-31-86-69:~# kubectl get node
NAME
ip-192-168-28-149.us-west-2.compute.internal    STATUS    ROLES    AGE    VERSION
ip-192-168-76-186.us-west-2.compute.internal    Ready    <none>    5m     v1.13.7-eks-c57ff8
root@ip-172-31-86-69:~#
```



### Step 5.4.3: Deploying an application to AWS EKS cluster

- Create Kubernetes deployment and service using the set of commands mentioned below:

```
kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080
```

```
kubectl expose deployment/kubernetes-bootcamp --port=8080 --target-port=8080 --type=NodePort
```



```

root@ip-172-31-86-69:~# kubectl run kubernetes-bootcamp --image=docker.io/jocatalin/kubernetes-bootcamp:v1 --port=8080
deployment.apps "kubernetes-bootcamp" created
root@ip-172-31-86-69:~# kubectl expose deployment/kubernetes-bootcamp --port=8080 --target-port=8080 --type=NodePort
service "kubernetes-bootcamp" exposed
root@ip-172-31-86-69:~# kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
kubernetes-bootcamp-6c5cfd894b-9jqzr 0/1     ContainerCreating   0           6s
root@ip-172-31-86-69:~# kubectl get deployments
NAME                                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp                1         1         1             1           18s
root@ip-172-31-86-69:~# kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
kubernetes-bootcamp-6c5cfd894b-9jqzr 1/1     Running            0           19s
root@ip-172-31-86-69:~# kubectl get services
NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes                         ClusterIP   10.100.0.1    <none>        443/TCP          44m
kubernetes-bootcamp               NodePort    10.100.33.238 <none>        8080:30306/TCP   1m
root@ip-172-31-86-69:~#

```

#### Step 5.4.4: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

- Initialize your repository using the following command:

**git init**

- Add all the files to your git repository using the following command:

**git add .**

- Commit the changes using the following command:

**git commit . -m "Changes have been committed."**

- Push the files to the folder you initially created using the following command:

**git push -u origin master**