**Date 19.10.2024**

DevOps Project-2

You are hired as a DevOps engineer for Analytics Pvt Ltd. This company is a product based organization which uses Docker for their containerization needs within the company. The final product received a lot of traction in the first few weeks of launch. Now with the increasing demand, the organization needs to have a platform for automating deployment, scaling, and operations of application containers across clusters of hosts, As a DevOps engineer, you need implement a DevOps life cycle, such that all the requirements are implemented without any change in the Docker containers in the testing environment. Up until now, this organization used to follow a monolithic architecture with just 2 developers.

The product is present on <https://github.com/hshar/website.git>

Following are the specifications of life-cycle:

1. Git workflow should be implemented. Since the company follows monolithic architecture of Development you need to take care of version control. The release should happen only on 25th of every month.

2. Code build should be triggered once the commits are made in the master Branch.

3. The code should be containerized with the help of the Docker file, The Dockerfile should be built every time if there is a push to Git-Hub. Create a custom Docker image using a Dockerfile.

4. As per the requirement in the production server, you need to use the Kubernetes cluster and the containerized code from Docker hub should be deployed with 2 replicas. Create a NodePort service and configure the same for port 30008

5. Create a Jenkins pipeline script to accomplish the above task.

6. For configuration management of the infrastructure, you need to deploy the configuration on the servers to install necessary software and configurations.

Using Terraform accomplish the task of infrastructure creation in the AWS cloud provider.

Architectural Advice Software’s to be installed on the respective machines using configuration management.

Worker1: Jenkins, Java.

Worker2: Docker, Kubernetes.

Worker3: Java, Docker, Kubernetes

Worker4: Docker, Kubernetes.

A diagram of a machine

Description automatically generated

A diagram of a server

Description automatically generated

The product is present on <https://github.com/hshar/website.git>

Launch EC2 Jenkins Master Machine and update with below command

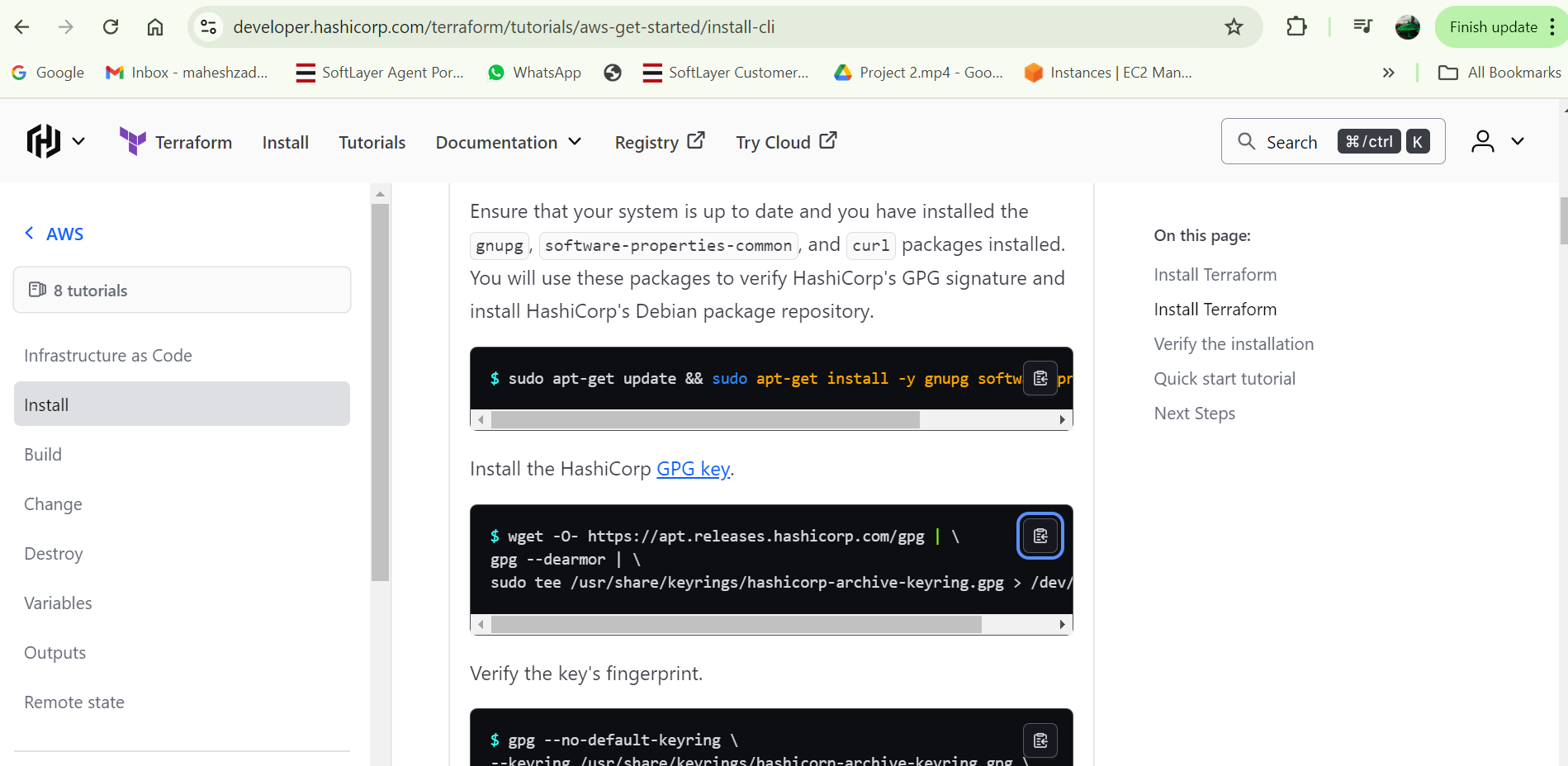
A screenshot of a computer

Description automatically generated

sudo apt update

now Install Terraform from official website

<https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli>



Run the below commands

sudo apt-get update && sudo apt-get install -y gnupg software-properties-common

Install the HashiCorp [GPG key](https://apt.releases.hashicorp.com/gpg).

wget -O- https://apt.releases.hashicorp.com/gpg | \

gpg --dearmor | \

sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null

Verify the key's fingerprint.

gpg --no-default-keyring \

--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \

--fingerprint

Add the official HashiCorp repository to your system. The lsb\_release -cs command finds the distribution release codename for your current system, such as buster, groovy, or sid.

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \

https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | \

sudo tee /etc/apt/sources.list.d/hashicorp.list

Download the package information from HashiCorp.

sudo apt update

Install Terraform from the new repository.

sudo apt-get install terraform

verify the installation

terraform –version

A screen shot of a computer

Description automatically generated

Now Install the ansible on ubuntu

<https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu>

create the script file a.sh

A screenshot of a computer

Description automatically generated

sudo apt update

sudo apt install software-properties-common sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible -y

Now run the command after saving the file

Bash a.sh

Verify ansible

Ansible –version

A screenshot of a computer program

Description automatically generated

Now Next Step to launch the infrastructure.

Go to IAM user and create the new user

create the access and secret key .

|  |  |
| --- | --- |
| Access key ID | Secret access key |
| AKIAQ3EGU2AALBKEUC43 | 0MNPKRalFor6NRtfvFNPvlEaGqvIa5y1aDHm5qPj |

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Create the main.tf

A screenshot of a computer

Description automatically generated

provider "aws" {

access\_key = "AKIAQ3EGU2AALBKEUC43"

secret\_key = "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"

region = "us-east-1"

}

resource "aws\_instance" "K8-M" {

ami = "ami-005fc0f236362e99f"

instance\_type = "t2.medium"

key\_name = "nvirginia"

tags = {

Name = "k8s-M/J-s"

}

}

resource "aws\_instance" "K8-S1" {

ami = "ami-005fc0f236362e99f"

instance\_type = "t2.micro"

key\_name = "nvirginia"

tags = {

Name = "k8s-S1"

}

}

resource "aws\_instance" "K8-S2" {

ami = "ami-005fc0f236362e99f"

instance\_type = "t2.micro"

key\_name = "nvirginia"

tags = {

Name = "k8s-s2"

}

}

Now initialize Teraform

A screenshot of a computer

Description automatically generated

Run the command

Terraform plan

A screenshot of a computer

Description automatically generated

Now type the command

Terraform apply

A black screen with a white border

Description automatically generated

Type yes

A screenshot of a computer

Description automatically generated

Three resources added

A screenshot of a computer

Description automatically generated

Pls connect all three instances and update by the below command

Sudo apt update

A screenshot of a computer

Description automatically generated

Now we have create the ansible cluster

Now we have to create password less connection.

A screenshot of a computer

Description automatically generated

Created keypair ssh-keygen ( ansible master node)

Move to cd .ssh

A screenshot of a computer

Description automatically generated

Copy the key and paste in each of the slave ansible machines

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICPt9eLnnwqK81L5JnN74jP8ObdsEWCvCw6Lv2ltiEn1 ubuntu@ip-172-31-37-160

A screen shot of a computer

Description automatically generated

Save it

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now ping all the slave machine in ansible master machine

Go to host file

Cd /etc/ansible

A screenshot of a computer

Description automatically generated

Save it

Copy all the private Ip of all ansible slave machine in the hosts file path

cd / tec/ansible

Type the below command

Ansible -m ping all

All the machines are pining now

A screenshot of a computer

Description automatically generated

Now create the palybook file

Now create the playbook

Sudo nano play.yaml

---

- name: executing scripts for Jenkins and Java on master

become: true

hosts: localhost

tasks:

- script: localhost.sh

- name: executing scripts for installation of Java,docker.k8s on k8s- master

become: true

hosts: k8s-m

tasks:

- script: master.sh

- name: executing scripts for k8s installation slave1

become: true

hosts: k8s-s1

tasks:

- script: slaves.sh

- name: executing scripts for k8s installation on slaves

become: true

hosts: k8s-s2

tasks:

- script: slaves.sh

A screenshot of a computer

Description automatically generated

Save it

Create the localhost.shA screenshot of a computer

Description automatically generated

GNU nano 6.2 localhost.sh \*

sudo apt update

sudo apt install openjdk-17-jdk -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins -y

A screen shot of a computer

Description automatically generated

* Jenkins installation link

<https://www.jenkins.io/doc/book/installing/linux/>

A screenshot of a computer

Description automatically generated

now create master.sh

sudo apt update

sudo apt install openjdk-17-jdk -y

# disable swap

sudo swapoff -a

# Create the .conf file to load the modules at bootup

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

# sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ipétables = 1

net.ipv4.ip\_forward = 1

EOF

# Apply sysctl params without reboot

sudo sysctl --system

## Install CRIO Runtime

sudo apt-get update -y

sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y

sudo apt-get install -y cri-o

sudo systemctl daemon-reload

sudo systemctl enable crio --now

sudo systemctl start crio.service

echo "CRI runtime installed successfully"

#Add Kubernetes APT repository and install required packages

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update -y

sudo apt-get install -y kubelet="1.29.0-\*" kubectl="1.29.0-\*" kubeadm="1.29.0-\*"

sudo apt-get update -y

sudo apt-get install -y jq

sudo systemctl enable now kubelet

sudo systemctl start kubelet

create slaves.sh

sudo apt update

sudo apt install openjdk-17-jdk -y

# disable swap

sudo swapoff -a

# Create the .conf file to load the modules at bootup

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

# sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ipétables = 1

net.ipv4.ip\_forward = 1

EOF

# Apply sysctl params without reboot

sudo sysctl --system

## Install CRIO Runtime

sudo apt-get update -y

sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y

sudo apt-get install -y cri-o

sudo systemctl daemon-reload

sudo systemctl enable crio --now

sudo systemctl start crio.service

echo "CRI runtime installed successfully"

#Add Kubernetes APT repository and install required packages

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update -y

sudo apt-get install -y kubelet="1.29.0-\*" kubectl="1.29.0-\*" kubeadm="1.29.0-\*"

sudo apt-get update -y

sudo apt-get install -y jq

sudo systemctl enable now kubelet

sudo systemctl start kubelet

A screenshot of a computer

Description automatically generated

now run the below to dry run the playbook

ansible-playbook play.yaml –-check

A screenshot of a computer

Description automatically generated

Playbook run successfully

A screenshot of a computer

Description automatically generated

Run the below command in Kubernetes master

A screenshot of a computer

Description automatically generated

ubuntu@ip-172-31-44-233:~$ sudo kubeadm init

A screenshot of a computer

Description automatically generated

kubeadm join 172.31.44.233:6443 --token jux2lk.ug7ryywnhbqwb59c \

--discovery-token-ca-cert-hash sha256:d218546d4d9f3946543f0548b3e25770317e5c5b720305bb49818e720bb148dd

Run these three commands

A screen shot of a computer

Description automatically generated

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

A screenshot of a computer

Description automatically generated

controllers created

# Install Claico Network Plugin Network

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

## Execute on ALL of your Worker Node's

1. Perform pre-flight checks

sudo kubeadm reset pre-flight checks

2. Paste the join command you got from the master node and append `--v=5` at the end but first use sudo su command to become root (avoid using sudo your-token).

sudo su

run the command in both k8s slaves machines

kubeadm join 172.31.44.233:6443 --token jux2lk.ug7ryywnhbqwb59c --discovery-token-ca-cert-hash sha256:d218546d4d9f3946543f0548b3e25770317e5c5b720305bb49818e720bb148dd --v=5

A screenshot of a computer

Description automatically generated

Now Setting up the Jenkins master

Start the Jenkins Service

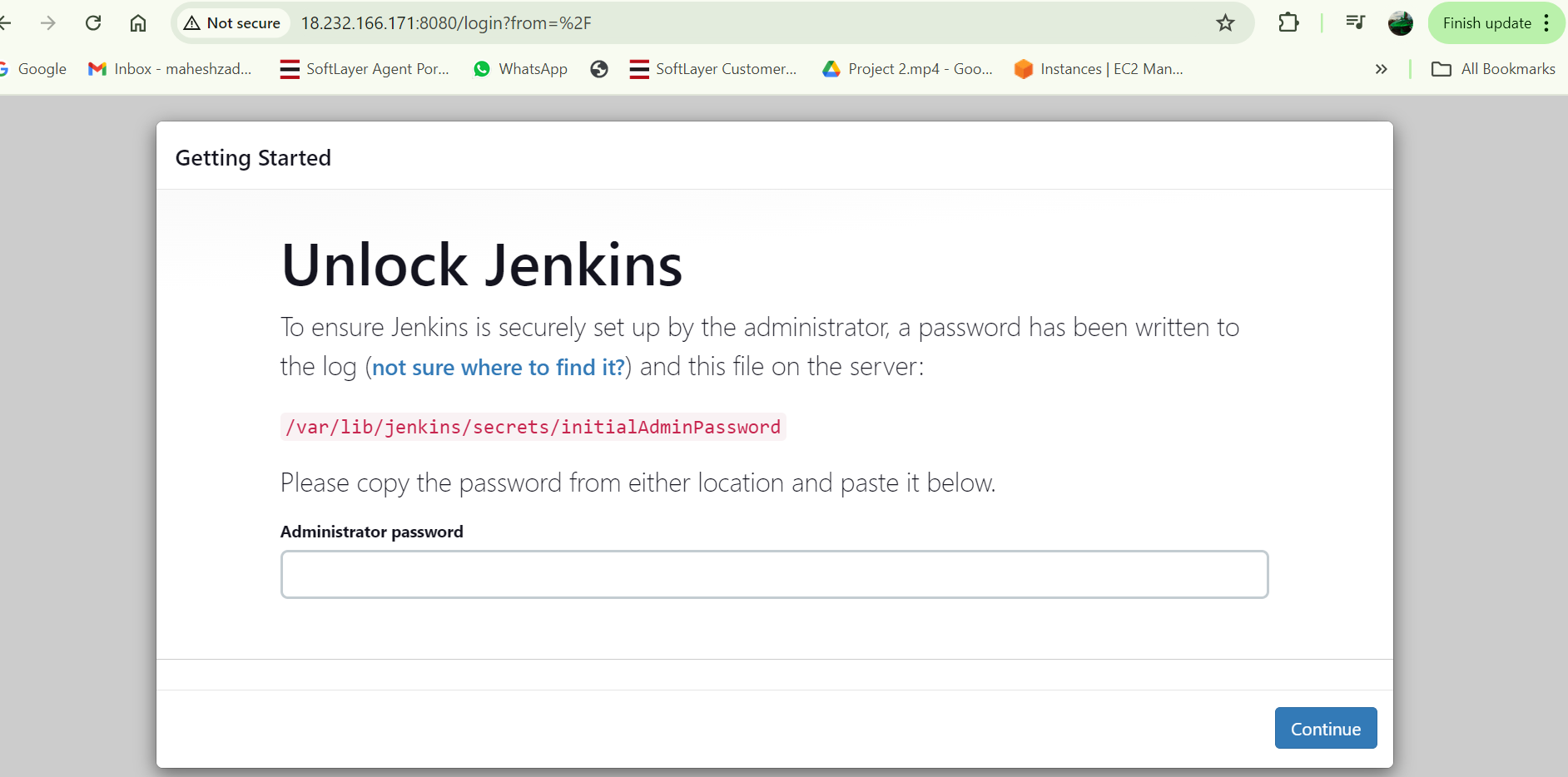
sudo systemctl start Jenkins

sudo systemctl status jenkins

Copy the public IP and paste in browser

A screenshot of a computer

Description automatically generated



Run the below command in Jenkins master server

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

5d1a8e22dd284b63b84ba87098ca67a3

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now create the node

A screenshot of a computer

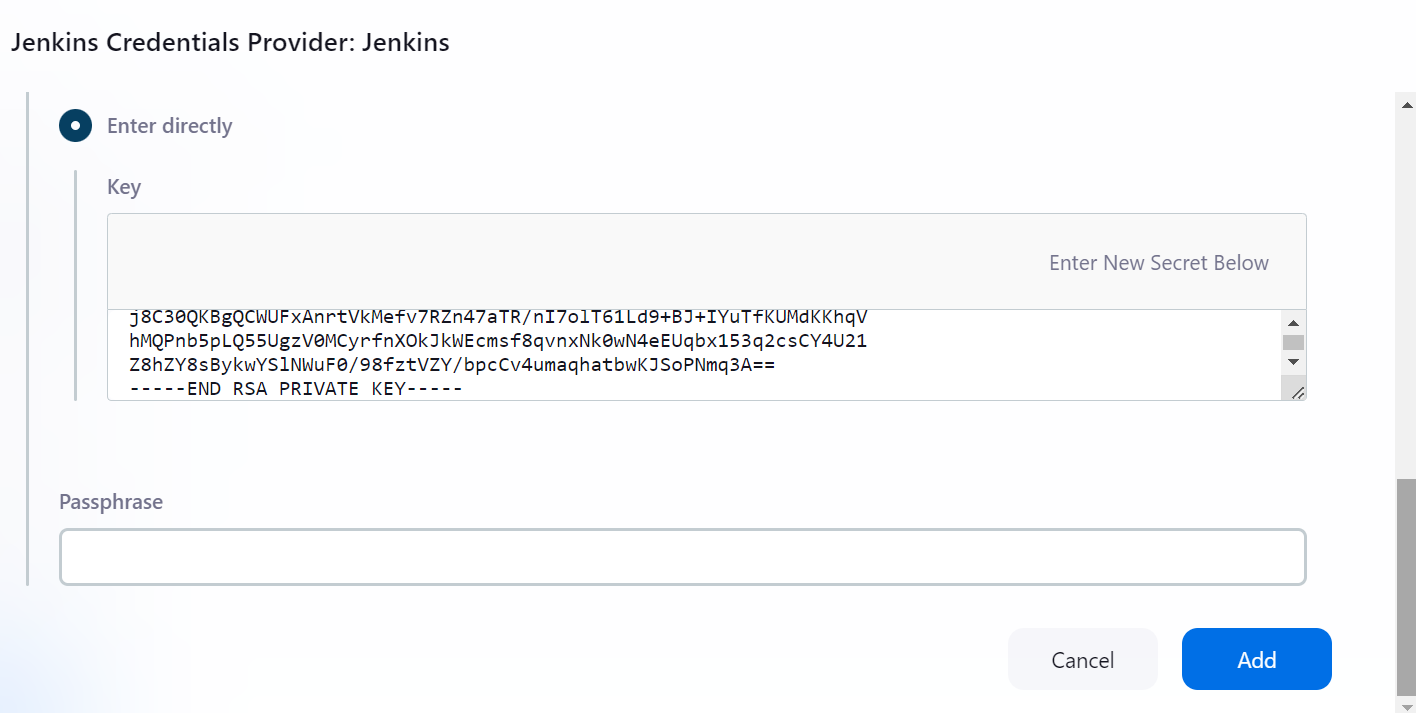
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



Click on save

A screenshot of a computer

Description automatically generated

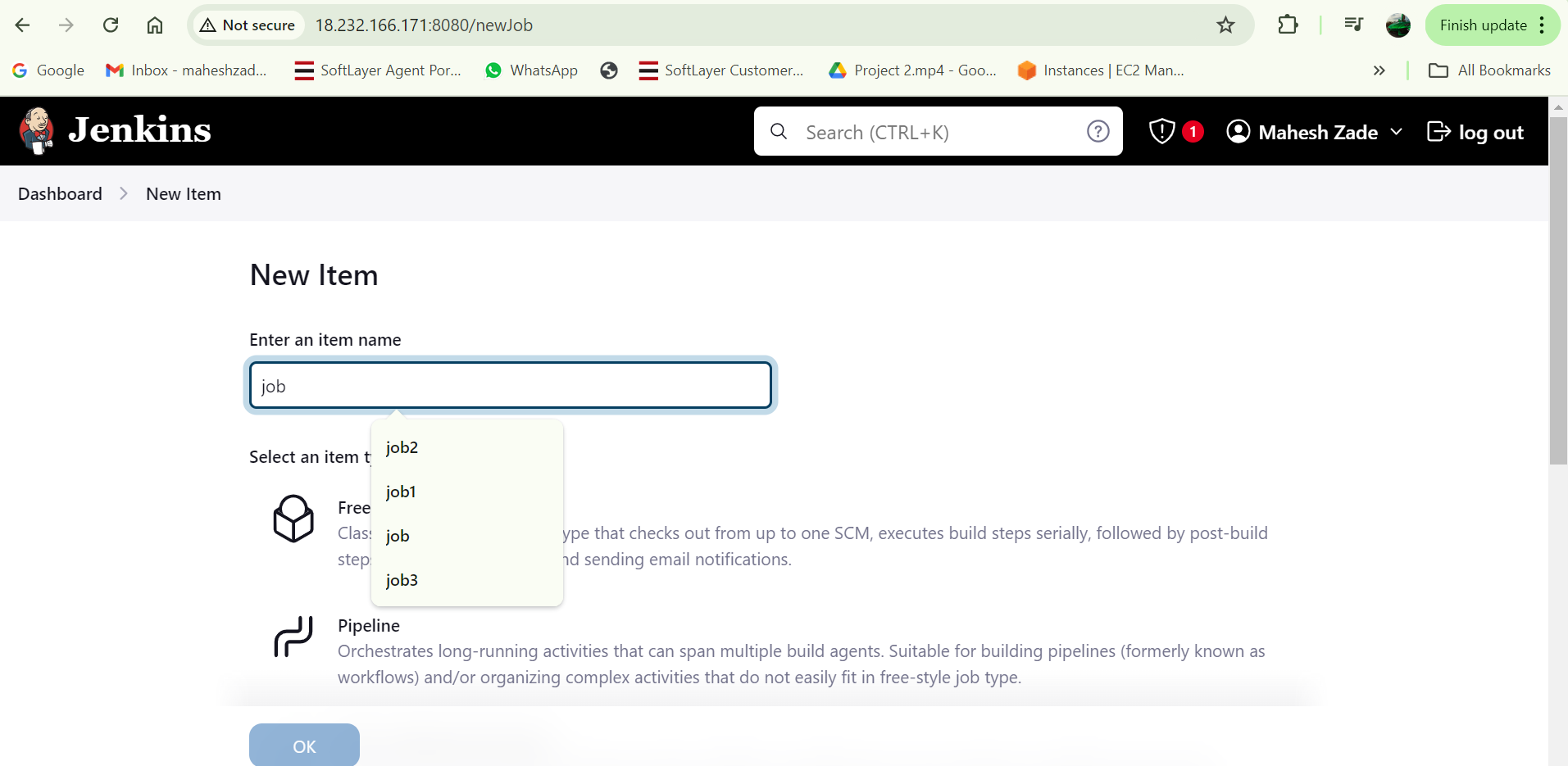
Node is created

A screenshot of a computer

Description automatically generated

Next step is create a script and job

Go to Dashboard and create ajob



Click on hook trigger

A screenshot of a computer

Description automatically generated

Copy the playload URL for Jenkins

<http://18.232.166.171:8080/>

<https://github.com/hshar/website.git>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

I have fork this repository

Create webhooks now

Go to Settings and click on webhook

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Click on add webhook

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Go to jon and select hellow world pipeline

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Save script

Create docker hub credentials

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Copy this docker hub credentials to the Jenkins job

A screenshot of a computer

Description automatically generated

Check in Kubernetes master server

A screenshot of a computer

Description automatically generated

Run the above commands

Now we will the docker image

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Pipeline run successfully

pipeline{

agent none

environment {

DOCKERHUB\_CREDENTIALS=credentials('1c8e77f5-c40d-4ec2-aebb-792967cc39c6')

}

stages{

stage('Hello'){

agent{

label 'K8s-Master'

}

steps{

echo 'Hello World'

}

}

stage('git'){

agent{

label 'K8s-Master'

}

steps{

git'https://github.com/maheshiitm/capstoneproject2.git'

}

}

stage('docker') {

agent {

label 'K8s-Master'

}

steps {

sh 'sudo docker build /home/ubuntu/jenkins/workspace/job -t maheshz/image2'

sh 'sudo echo $DOCKERHUB\_CREDENTIALS\_PSW | sudo docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

sh 'sudo docker push maheshz/image2'

}

}

stage('Kubernetes') {

agent {

label 'K8s-Master'

}

steps {

sh 'kubectl create -f deploment.yaml'

sh 'kubectl create -f service.yaml'

}

}

}

}

A screenshot of a computer

Description automatically generated

Service.yaml

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Check docker image in k8s Server

A screenshot of a computer

Description automatically generated

3 min again docker image created

Check Deployment file in the k8 server

A screenshot of a computer

Description automatically generated

Deployment is created

A screenshot of a computer screen

Description automatically generated

Check the command

Kubectl get deploymnet

Kubectl get service

Capstone Project-2 Done Successfully

Thank you