**DevOps Project**

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.

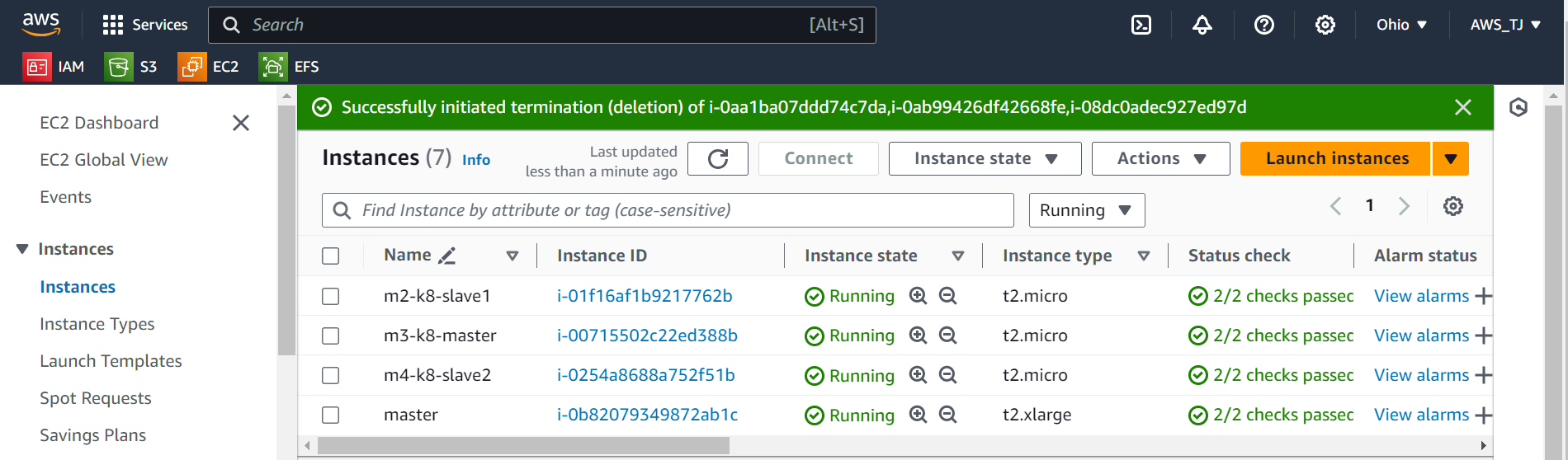
4. As per the requirement in the production server, you need to use the Kubernetes cluster and the containerized code from Docker hub should be deployed with 2 replicas. Create a NodePort service and configure the same for port 30008

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.

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

**Answer –**

****

**On master** installed terraform –

Terraform script

provider "aws" {

region = "us-east-2"

access\_key = ""

secret\_key = ""

}

resource "aws\_instance" "K8s-slave2" {

ami = "ami-0f30a9c3a48f3fa79"

instance\_type = "t2.medium"

key\_name = "sagarohio"

tags = {

Name = "m2-slave"

}

}

resource "aws\_instance" "K8s-master" {

ami = "ami-0f30a9c3a48f3fa79"

instance\_type = "t2.medium"

key\_name = "sagarohio"

tags = {

Name = "m3-master"

}

}

resource "aws\_instance" "K8s-Node2" {

ami = "ami-0f30a9c3a48f3fa79"

instance\_type = "t2.medium"

key\_name = "sagarohio"

tags = {

Name = "m4-slave"

}

}

**installed ansible on main machine, connect all the other three instances with ssh keygen method**

**then use the scripts and playbook to install the required softwares in the machines**

script1.sh

sudo apt update

sudo apt install openjdk-11-jdk -y

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

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

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

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

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

sudo apt-get install jenkins -y

sudo apt install docker.io -y

script2.sh

sudo apt update

sudo apt install docker.io -y

sudo apt install openjdk-11-jdk -y

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

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sou>

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet

script3.sh

sudo apt update

sudo apt install docker.io -y

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

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sou>

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet

History -

1 sudo apt update

2 sudo nano t.sh

3 bash t.sh

4 sudo nano main.tf

5 terraform init

6 terraform plan

7 terraform apply

8 sudo apt install software-properties-common

9 sudo add-apt-repository --yes --update ppa:ansible/ansible

10 sudo apt install ansible

11 cd .ssh

12 ls

13 ssh-keygen

14 ls

15 cat id\_rsa.pub

16 cd /etc/ansible

17 ls

18 sudo nano hosts

19 ansible -m ping all

20 sudo nano hosts

21 ansible -m ping all

22 sudo nano script1.sh

23 sudo nano script2.sh

24 sudo nano script3.sh

25 sudo nano play.yaml

26 ansible-playbook play.yaml --syntax-check

27 ansible-playbook play.yaml --check

28 ansible-playbook play.yaml

29 sudo nano script1.sh

30 sudo nano script2.sh

31 ansible-playbook play.yaml --syntax-check

32 ansible-playbook play.yaml --check

33 ansible-playbook play.yaml

34 sudo apt update

35 ls

36 ansible --version

37 cd /etc/ansible

38 ls

39 sudo nano script1.sh

40 sudo nano script2.sh

41 ansible-playbook play.yaml --syntax-check

42 ansible-playbook play.yaml --check

43 ansible-playbook play.yaml

44 sudo apt update

45 history

Play.yaml

---

- name: install Jenkins and Java on host

become: true

hosts: localhost

tasks:

- name: running script to install tools on host

script: script1.sh

- name: install docker, K8s and Java on main

become: true

hosts: master

tasks:

- name: running script to install tools on main

script: script2.sh

- name: install docker, K8s on nodes

become: true

hosts: slaves

tasks:

- name: running script to install tools on node

script: script3.sh

after the playbook is executed

run this command on your master node

sudo kubeadm init --apiserver-advertise-address=privateipofmaster

after this is done, copy the join command and run it on the slaves with superuser permissions

after the token is pasted, come back to master

run these

mkdir -p $HOME/.kube

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

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

then download calico yaml package

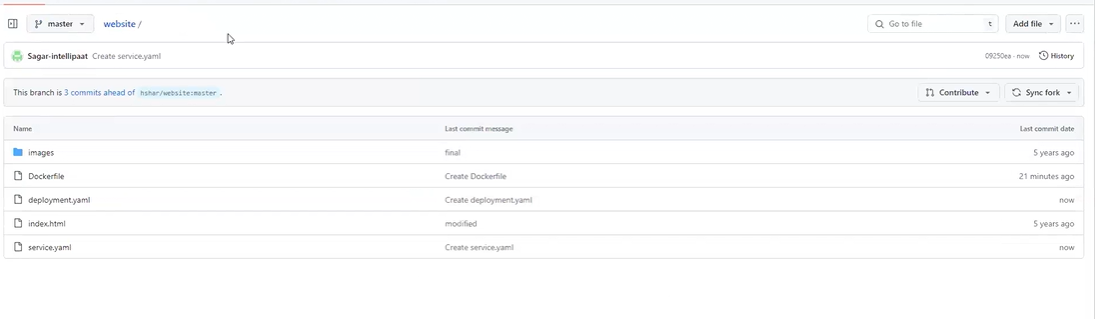
curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.2/manifests/calico.yaml -O

kubectl apply -f calico.yaml

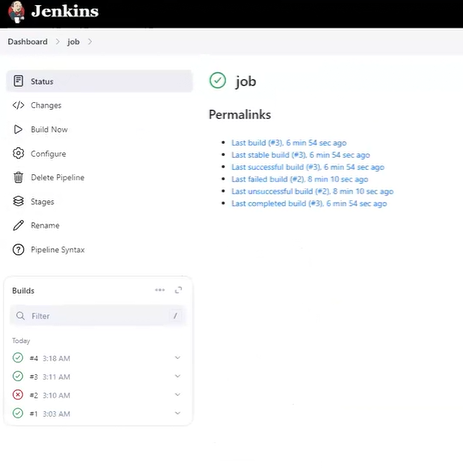
kubectl get nodes

after this, open the Jenkins dashboard, install ssh agent plug in

git files –



create the node



create the pipeline (Full pipeline syntax)

pipeline {

agent none

environment {

DOCKERHUB\_CREDENTIALS=credentials('96f7b437-094a-4fd6-ad97-9414e556f678')

}

stages {

stage('Hello') {

steps {

echo 'Hello World'

}

}

stage('git') {

agent {

label 'k8node'

}

steps {

git 'https://github.com/Sagar-intellipaat/duplicate.git'

}

}

stage('docker') {

agent {

label 'k8node'

}

steps {

sh 'sudo docker build /home/ubuntu/jenkins/workspace/newjob -t sagarintellipaat/project2'

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

sh 'sudo docker push sagarintellipaat/project2'

}

}

stage('kubernetes') {

agent {

label 'k8node'

}

steps {

sh 'kubectl delete deploy nginx-deployment'

sh 'kubectl apply -f deployment.yaml'

sh 'kubectl delete service my-service'

sh 'kubectl apply -f service.yaml'

}

}

}

}

**deployment.yaml file**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: myproj

image: sagarintellipaat/project2

ports:

- containerPort: 80

**service.yaml**

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

type: NodePort

selector:

app: nginx

ports:

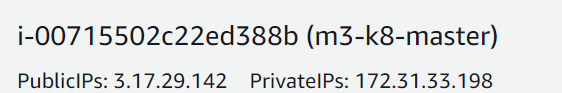
- port: 80

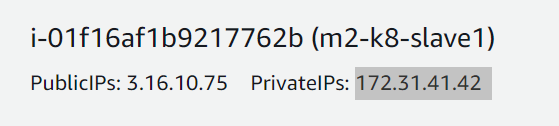
targetPort: 80

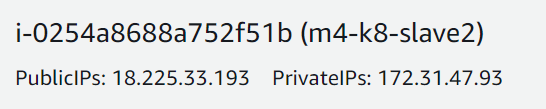
nodePort: 30008

**Output –**









MASTER

