**Devops project – Jenkins/K8s**

**Requirements:**

The organization needs to have a platform for automating deployment, scaling and operations of application containers across clusters of hosts. As a DevOps Engineer, I need to implement a DevOps lifecycle such that all the requirements are met without any change in the Docker containers in the testing environment.

Following are the specifications of the lifecycle:

1. Git workflow should be implemented.

2. CodeBuild should be triggered once the commits are made in the master branch.

3. The code should be containerized with the help of the Dockerfile. The Dockerfile should be built every time there is a push to GitHub. Create a custom Docker image using a Dockerfile.

4. As per the requirement, 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, deploy the configuration on the servers to install necessary softwares.

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

**Implementation:**

Machine1: Jenkins, Java

Machine2: Docker, Kubernetes

Machine3: Java, Docker, Kubernetes

Machine4: Docker, Kubernetes

A diagram of a machine

Description automatically generated

A diagram of a server

Description automatically generated

1. Launch an instance Machine-1 with instance type t2.medium and install terraform on it.

Sudo apt update

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

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

sudo apt update && sudo apt install terraform

A screenshot of a computer program

Description automatically generated

1. Create 3 more machines using terraform script for Kubernetes (1 K8s Master and 2 K8s slaves)

Sudo nano main.tf

A screenshot of a computer screen

Description automatically generated

1. Initialize terraform using terraform init

A screenshot of a computer

Description automatically generated

1. Use terraform plan to create an execution plan and to preview the changes

A screenshot of a computer program

Description automatically generated

1. Use terraform apply to apply the changes

A screenshot of a computer program

Description automatically generated

1. All 3 instances have been created

A screenshot of a computer

Description automatically generated

1. Install Ansible on machine1

A black background with white text

Description automatically generated

1. Configure and establish connection with slave nodes using ssh-keygen

A screenshot of a computer program

Description automatically generated

1. Paste the contents of id\_rsa.pub file into .ssh/authorized\_keys of all 3 machines
2. Update the /etc/ansible/hosts file to have the IP addresses of slave nodes

A screenshot of a computer

Description automatically generated

1. Verify using ansible -m ping all

A screenshot of a computer program

Description automatically generated

1. Create 3 ansible scripts to install the necessary softwares as below

Machine 1 – Jenkins, Java -> script1

Machine 3 – Java, docker, Kubernetes -> script2

Machine 2 & 4 – Docker, Kubernetes -> script3

Sudo nano 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 update

sudo apt-get install jenkins -y

sudo nano script2.sh

sudo apt update

sudo apt install openjdk-11-jdk -y

sudo apt install docker.io -y

sudo apt update

sudo apt upgrade -y

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

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo swapoff -a

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

sudo nano script3.sh

sudo apt update

sudo apt install docker.io -y

sudo apt update

sudo apt upgrade -y

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

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

sudo add-apt-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo swapoff -a

sudo apt update

sudo apt install -y kubelet kubeadm kubectl

1. Create an ansible playbook to use the above shell scripts

Sudo nano play.yaml

A screenshot of a computer program

Description automatically generated

1. Run the ansible playbook with syntax check to check for any errors

Ansible-playbook play.yaml –syntax-check

1. Do a dry run to see if it is successful

Ansible-playbook play.yaml –check

A screen shot of a computer

Description automatically generated

1. Execute the playbook to install the softwares on the respective machines

Ansible-playbook play.yaml

A screen shot of a computer

Description automatically generated

1. Manually execute kubeadm init on machine 3 (K8s-Master) which will generate token to join the cluster

Sudo kubeadm init

A screenshot of a computer program

Description automatically generated

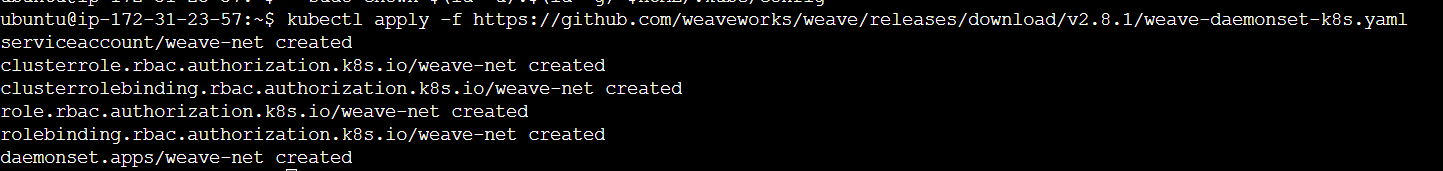
1. To start using cluster, provide permissions to the kube config file

mkdir -p $HOME/.kube

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

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

1. Execute below command to create the pod network



1. K8s master has been configured successfully

A screenshot of a computer

Description automatically generated

1. Paste the token on the slave nodes to join the cluster

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. Verify using kubectl get nodes

A screen shot of a computer

Description automatically generated

1. Configure the Jenkins dashboard and create a node for K8s Master using Manage Jenkins option. Here our new node (K8s\_M) will be the slave for Jenkins

A screenshot of a computer

Description automatically generated

1. Create a new node as K8s-Master and configure it

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

1. Create new credentials for dockerhub account in Jenkins

A screenshot of a computer

Description automatically generated

1. Go to GitHub and fork the given repository so we can create dockerfile and containerize it

A screenshot of a computer

Description automatically generated

1. Create a Dockerfile as below and commit changes

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Create a Jenkins pipeline job as below

A screenshot of a computer

Description automatically generated

The pipeline script will include 4 stages

* Test pipeline – Hello task
* Clone GitHub repository – Git task
* Build docker image from dockerfile and push it to DockerHub – Docker task
* Create deploy.yaml file for Kubernetes deployment and expose the website using Nodeport service -K8s task
* Test pipeline

A screenshot of a computer

Description automatically generated

* Clone GitHub repository

A screenshot of a computer

Description automatically generated

1. Execute the job and verify that Git files are created on Machine3 (K8s-Master) which will be the slave for Jenkins Master (Machine1)

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

1. Build docker image and push it to dockerhub

* Docker task

A computer screen with green text

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Pipeline job is successful and the image has been pushed to dockerhub

A screenshot of a computer

Description automatically generated

1. Next is to create the deployment file and service from the docker image above

* K8s task

A close-up of a text

Description automatically generated

1. Create deployment.yaml and service.yaml files in Github repository and commit them

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Execute the entire pipeline script and verify whether nginx deployment is successful on the slave nodes(Machine2 and 4) with port 30008

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Custom web page has been deployed successfully on the slave nodes using Jenkins

Slave2 (Machine4)

A cartoon of a cat

Description automatically generated

Slave1 (Machine 2)

A screenshot of a computer

Description automatically generated

1. **Requirement** – CodeBuild should be triggered once the commits are made in the master branch
2. To accomplish this, configure webhooks in Jenkins and Github

A screenshot of a computer

Description automatically generated

A screenshot of a webinar

Description automatically generated

1. Before we make changes to files in Github repo, we need to include a step to delete the existing nginx-deployment so that the changes are reflected in the webpage when pipeline is run.

A screenshot of a computer program

Description automatically generated

1. Now, any changes made in the master branch will be reflected in our webpages

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

Slave 1 output:

A cartoon of a cat

Description automatically generated

Slave2 output:

A cartoon of a cat

Description automatically generated