**Assignment 1**

You have been asked to:

● Deploy a Kubernetes Cluster for 3 nodes

● Create a nginx deployment of 3 replicas

**Solution Approach**

**Step 1: Create ec2 instances and Deploy Kubernetes Cluster of 3 nodes**

* We will create 3 ec2 instances (t2medium with 20G storage)– 1 Master and 2 Worker Nodes
* Update the security group to allow all traffic
* Run the following codes to set up Kubernetes on all nodes:
  + Create *kubernetes.sh* file to include the following content inside each node

# Execute the below script in both the master and slave servers

sudo vi kubernetes.sh

sudo yum install -y docker

sudo systemctl enable docker

sudo systemctl start docker

sudo setenforce 0

sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/

enabled=1

gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.28/rpm/repodata/repomd.xml.key

exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

EOF

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

sudo systemctl enable --now kubelet

VER=$(curl -s https://api.github.com/repos/Mirantis/cri-dockerd/releases/latest|grep tag\_name | cut -d '"' -f 4|sed 's/v//g')

wget https://github.com/Mirantis/cri-dockerd/releases/download/v${VER}/cri-dockerd-${VER}.amd64.tgz

tar xvf cri-dockerd-${VER}.amd64.tgz

sudo mv cri-dockerd/cri-dockerd /usr/local/bin/

wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-docker.service

wget https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/packaging/systemd/cri-docker.socket

sudo mv cri-docker.socket cri-docker.service /etc/systemd/system/

sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

sudo systemctl daemon-reload

sudo systemctl enable cri-docker.service

sudo systemctl enable --now cri-docker.socket

sudo kubeadm config images pull --cri-socket unix:///var/run/cri-dockerd.sock

* Running the *kubernetes.sh* script on all nodes

sudo chmod +x kubernetes.sh # make the kubernetes.sh file an executable

./kubernetes.sh # executing the kubernetes.sh file

* Start the kubernetes cluster only in the master

sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=172.31.9.106 --cri-socket unix:///var/run/cri-dockerd.sock

* On executing the above command, the cluster gets created, and it returns a command that needs to be run on the remaining slave nodes to connect them to the cluster

sudo kubeadm join 172.31.34.62:6443 --token 9lgeqo.yeiqjaoy8hgi2mdn --discovery-token-ca-cert-hash sha256:11bb8e8168f8d1f060a9d1d74ab532515079fec2e7a1c448d33d2369db8224b4 --cri-socket unix:///var/run/cri-dockerd.sock

* Next, in the master node, we need to add all worker nodes present in admin.conf file to .kube/config so that they can use the kubectl commands

cd ~

# create empty directory .kube

mkdir -p $HOME/.kube

ls -al

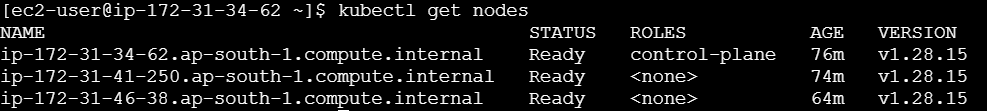
# copies admin.conf into a new file config in new location

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

# provides permission to the config file

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

* Kubernetes cluster of 3 nodes is deployed:



**Step 2: Create nginx deployment of 3 replicas**

1. Create a deployment file *nginxdeploy.yml*

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3

selector:

matchLabels:

app: mydemoapp

template:

metadata:

labels:

app: mydemoapp

spec:

containers:

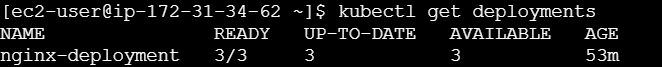
- name: mydemocontainer

image: nginx

1. Run the deployment file

kubectl apply -f nginxdeploy.yml

kubectl get deployments



**Assignment 2**

You have been asked to:

● Use the previous deployment

● Create a service of type NodePort for nginx deployment

● Check the nodeport service on a browser to verify

**Solution Approach**

**Create a service of type NodePort for nginx deployment**

1. Create file *nodeport-service.yml* as follows:

apiVersion: v1

kind: Service

metadata:

name: nodeport-service

labels:

app: nginx

spec:

type: NodePort

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

1. Run the service

kubectl apply -f nodeport-service.yml

kubectl get service nodeport-service



**Assignment 3**

You have been asked to:

● Use the previous deployment

● Change the replicas to 5 for the deployment

**Solution Approach**

**Change the number of replicas to 5**

Edit the deployment file *nginxdeploy.yml* and change replicas to 5

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 5

selector:

matchLabels:

app: mydemoapp

template:

metadata:

labels:

app: mydemoapp

spec:

containers:

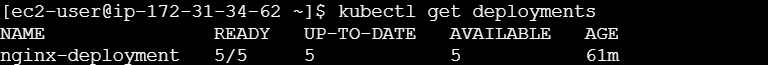
- name: mydemocontainer

image: nginx

1. Run the deployment file

kubectl apply -f nginxdeploy.yml

kubectl get deployments



**Assignment 4**

You have been asked to:

1. Use the previous deployment

2. Change the service type to ClusterIP

**Solution Approach**

**Change the service type of the NodePort service above to ClusterIP**

apiVersion: v1

kind: Service

metadata:

name: clusterip-service

labels:

app: nginx

spec:

type: ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

**Assignment 5**

You have been asked to:

● Use the previous deployment

● Deploy an nginx deployment of 3 replicas

● Create an nginx service of type clusterip

● Create an ingress service /apache to apache service /nginx to nginx service

**Solution Approach**

**Step 1: Deploy an nginx deployment of 3 replicas**

Create a deployment file *nginxdeploy.yml*

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3

selector:

matchLabels:

app: mydemoapp

template:

metadata:

labels:

app: mydemoapp

spec:

containers:

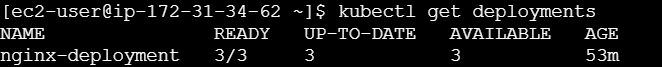
- name: mydemocontainer

image: nginx

Run the deployment file

kubectl apply -f nginxdeploy.yml

kubectl get deployments



**Step 2: Create an nginx service of type ClusterIP**

Create file *clusterip-service.yml* as follows:

apiVersion: v1

kind: Service

metadata:

name: clusterip-service

labels:

app: nginx

spec:

type: ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

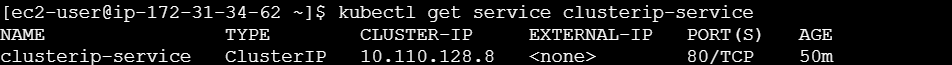
port: 80

targetPort: 80

Run the service

kubectl apply -f clusterip-service.yml

kubectl get service clusterip-service



**Step 3: Create an ingress service /apache to apache service /nginx to nginx service**

1. Create Apache deployment and service

apiVersion: apps/v1

kind: Deployment

metadata:

name: apache-deployment

labels:

app: apache

spec:

replicas: 1

selector:

matchLabels:

app: apache

template:

metadata:

labels:

app: apache

spec:

containers:

- name: apache

image: httpd

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: apache-service

spec:

selector:

app: apache

ports:

- protocol: TCP

port: 80

targetPort: 80

1. Create Nginx deployment and service

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 1

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

1. Create an ingress.yml resource

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: apache-nginx-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /apache

pathType: Prefix

backend:

service:

name: apache-service

port:

number: 80

- path: /nginx

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

1. Apply the ingress configuration

kubectl apply -f ingress.yaml