NAME:ATHARVA SHINDE ROLL.NO:54 CLASS:D15C

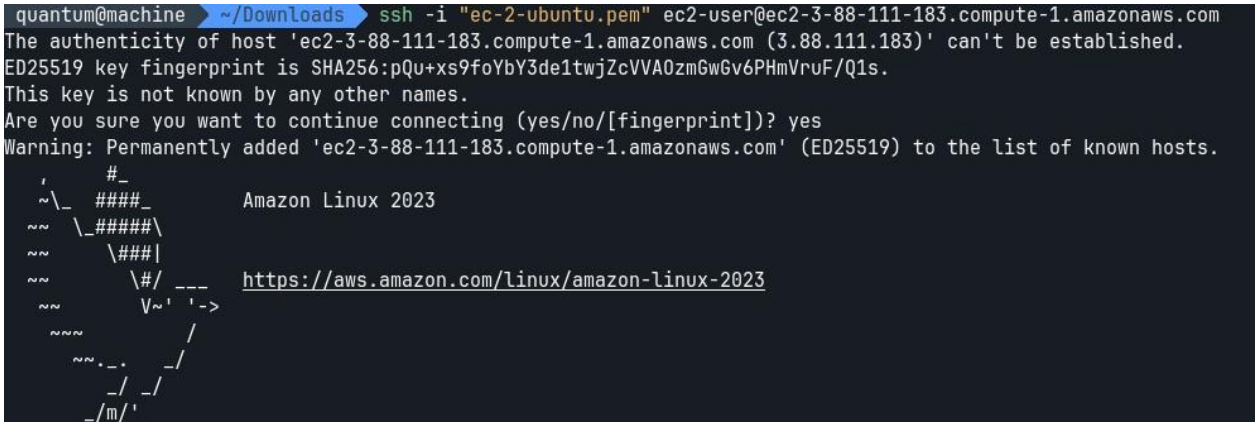
EXPERIMENT NO. 3

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud

1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed
   To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly

Name	Instance ID	Instance state	Instance type
kube-master	i-00aa79ac09d7462c0	Running	t2.medium
kube-worker1	i-0bab86cd3fbfcb40a	Running	t2.medium
kube-worker2	i-00dcfd302ffd80dda	Running	t2.medium

2. SSH into all 3 machines each in separate terminal



3. From now on, until mentioned, perform these steps on all 3 machines.

Install Docker

sudo yum install docker -y

```
[ec2-user@ip-172-31-92-18 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:09:56 ago on Wed Sep 11 15:19:39 2024.
Dependencies resolved.
================================================================================
 Package                                                    Architecture
================================================================================
Installing:
 docker                                                     x86_64
Installing dependencies:
 containerd                                                 x86_64
 iptables-libs                                              x86_64
 iptables-nft                                               x86_64
 libcgroup                                                  x86_64
 libnetfilter_conntrack                                     x86_64
 libnfnetlink                                               x86_64
 libnftnl                                                   x86_64
 pigz                                                       x86_64
 runc                                                       x86_64

Transaction Summary
```

Then, configure cgroup in a daemon.json file by using following commands

- cd /etc/docker

- cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"
  }
  EOF

- sudo systemctl enable docker

- sudo systemctl daemon-reload
- sudo systemctl restart docker
- docker -v

```
[ec2-user@ip-172-31-81-63 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-81-63 docker]$
```

4. Install Kubernetes on all 3 machines

SELinux needs to be disable before configuring kubelet

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/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:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
[ec2-user@ip-172-31-81-63 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-81-63 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

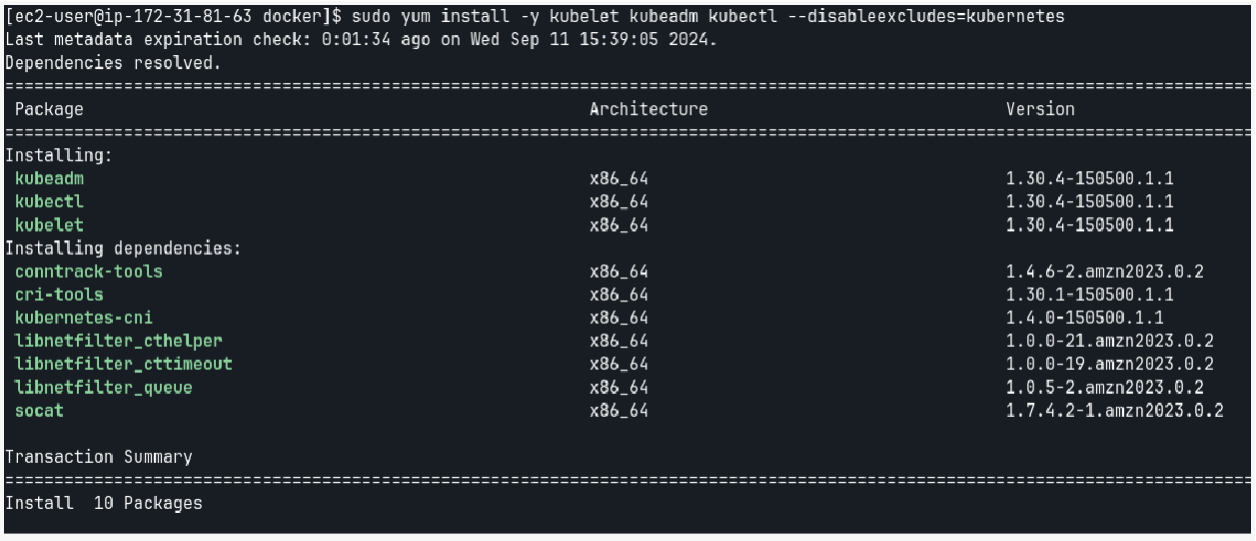
Add kubernetes repository (paste in terminal)

cat<<EOF|sudotee/etc/yum.repos.d/kubernetes.repo[kubernetes]
name=Kubernetes baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/enabled=1
gpgcheck=1 gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubeletkubeadmkubectlcri-toolskubernetes-cniEOF

Type following commands:

- sudo yum update

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



```
[ec2-user@ip-172-31-81-63 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:01:34 ago on Wed Sep 11 15:39:05 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture          Version
================================================================================
Installing:
 kubeadm                 x86_64                1.30.4-150500.1.1
 kubectl                 x86_64                1.30.4-150500.1.1
 kubelet                 x86_64                1.30.4-150500.1.1
Installing dependencies:
 conntrack-tools         x86_64                1.4.6-2.amzn2023.0.2
 cri-tools               x86_64                1.30.1-150500.1.1
 kubernetes-cni          x86_64                1.4.0-150500.1.1
 libnetfilter_cthelper   x86_64                1.0.0-21.amzn2023.0.2
 libnetfilter_cttimeout  x86_64                1.0.0-19.amzn2023.0.2
 libnetfilter_queue      x86_64                1.0.5-2.amzn2023.0.2
 socat                   x86_64                1.7.4.2-1.amzn2023.0.2

Transaction Summary
================================================================================
Install  10 Packages
```

After installing Kubernetes, we need to configure internetoptions to allow bridging.
sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee
-a /etc/sysctl.conf
sudo sysctl -p

5. **Perform this ONLY on the Master machine**

Initialize kubernetes by typing below command

sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=all

```
[addons] Applied essential addon: kube-proxy

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
        --discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8c5
[ec2-user@ip-172-31-81-63 docker]$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-81-63 docker]$
```

Copy the mkdir and chown commands from the top and executethem

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

Copy this join link and save it in clipboard (copy from youroutput as it different for each instance)

kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
--discovery-token-ca-cert-hash
sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6c d2d2f6f8c5

Then, add a common networking plugin called flammel file as mentioned in the code.

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Docum entation/kube-flannel.yml

```
[ec2-user@ip-172-31-81-63 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Check the created pod using this command

- kubectl get pods

6. **Perform this ONLY on the worker machines**

   Paste the below command on all 2 worker machines
   - Sudo yum install iproute-tc -y
   - sudo systemctl enable kubelet
   - sudo systemctl restart kubelet

- kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
  --discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8c5

Now we can see in the master/control node of kubernetes that worker nodes are connected by typing **watch kubectl get nodes** in the **master node instance**

NAME:ATHARVA SHINDE ROLL.NO:54 CLASS:D15C

```
Every 2.0s: kubectl get nodes

NAME                              STATUS   ROLES           AGE     VERSION
ip-172-31-81-63.ec2.internal      Ready    control-plane   29m     v1.30.4
ip-172-31-87-137.ec2.internal     Ready    <none>          5m58s   v1.30.4
ip-172-31-92-18.ec2.internal      Ready    <none>          5m53s   v1.30.4
```