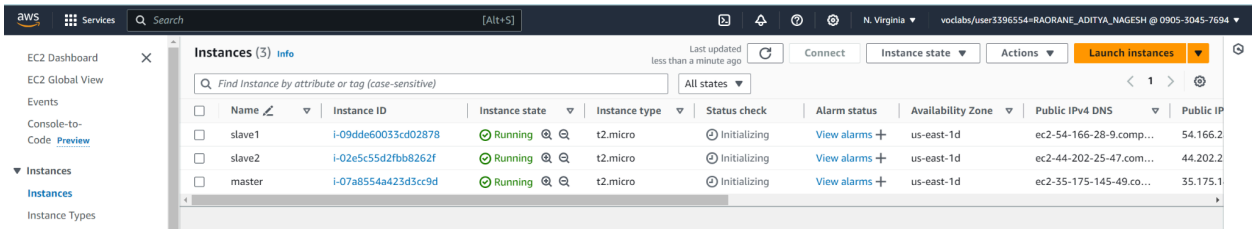
EXPERIMENT NO. 3

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.micro

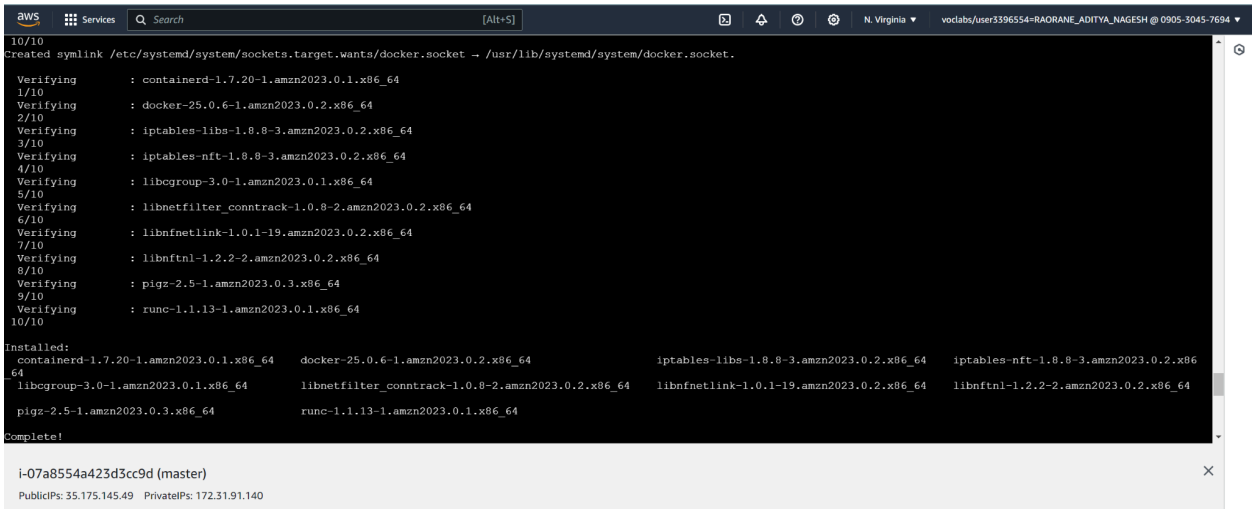


2. Perform these steps on all 3 machines.

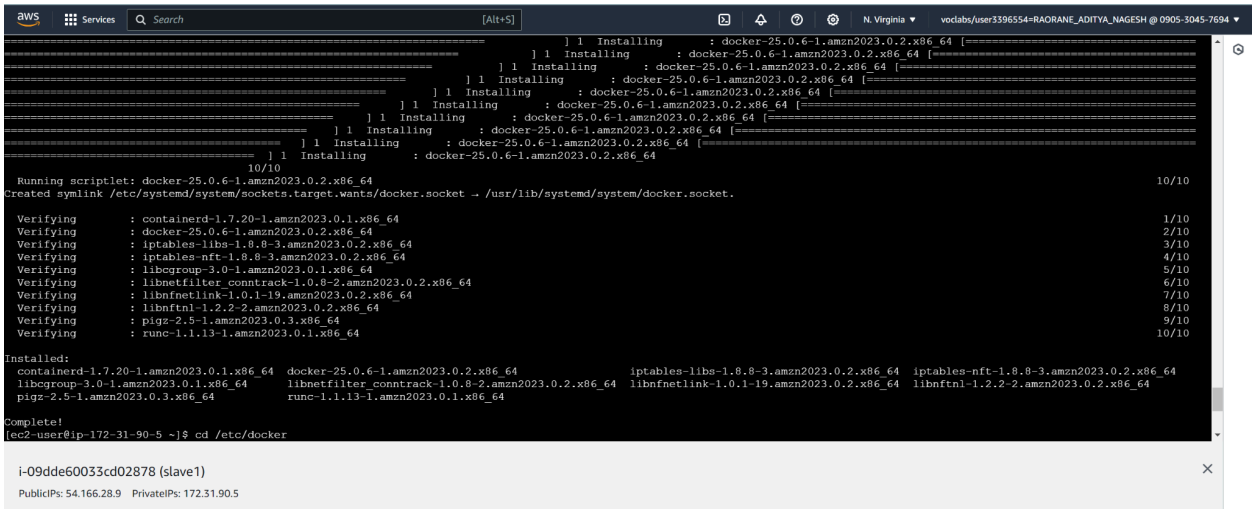
a]Install Docker

Command : sudo yum install docker -y

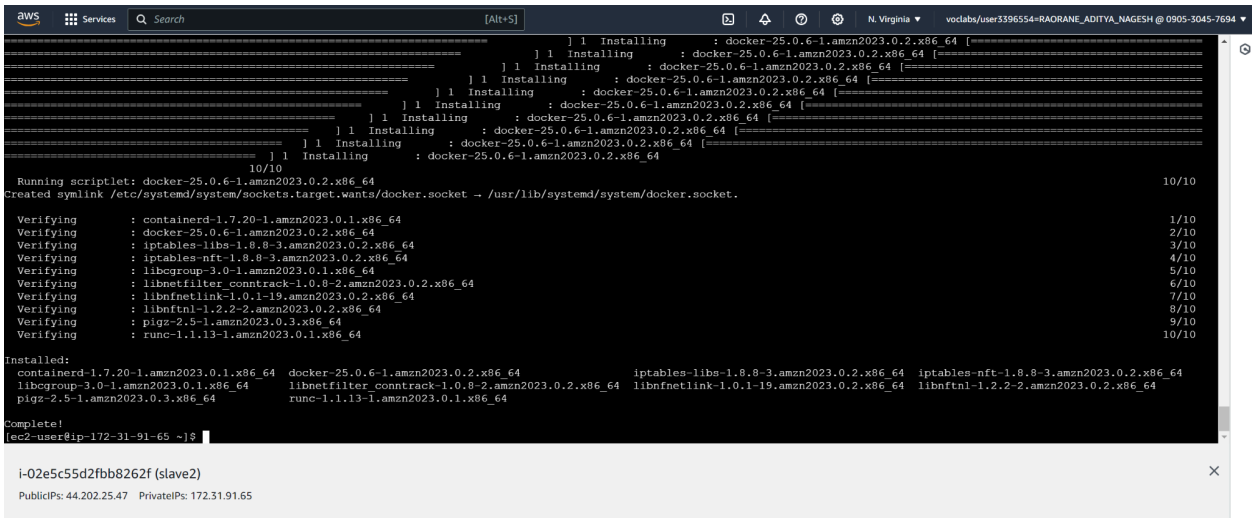
Master:-



Slave1:-



Slave2:-



b]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
```

Master:-

```
[ec2-user@ip-172-31-91-140 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
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

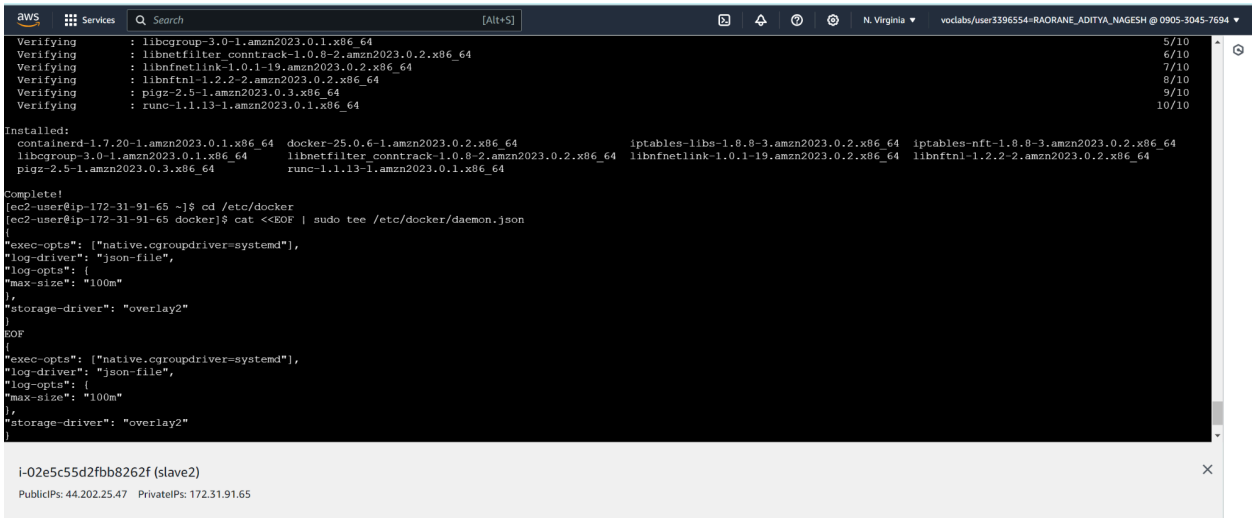
Slave1:-

```
  Verifying        : libcgroup-3.0-1.amzn2023.0.1.x86_64                                                                  5/10
  Verifying        : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64                                                   6/10
  Verifying        : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64                                                            7/10
  Verifying        : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                                                                 8/10
  Verifying        : pigz-2.5-1.amzn2023.0.3.x86_64                                                                       9/10
  Verifying        : runc-1.1.13-1.amzn2023.0.1.x86_64                                                                   10/10

Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64  docker-25.0.6-1.amzn2023.0.2.x86_64              iptables-libs-1.8.8-3.amzn2023.0.2.x86_64  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
  libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64           runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-90-5 ~]$ cd /etc/docker
[ec2-user@ip-172-31-90-5 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
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
```

i-09dde60033cd02878 (slave1)

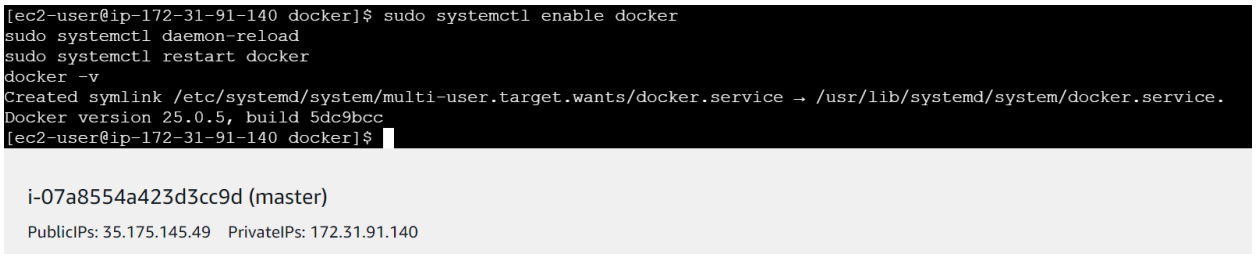
PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

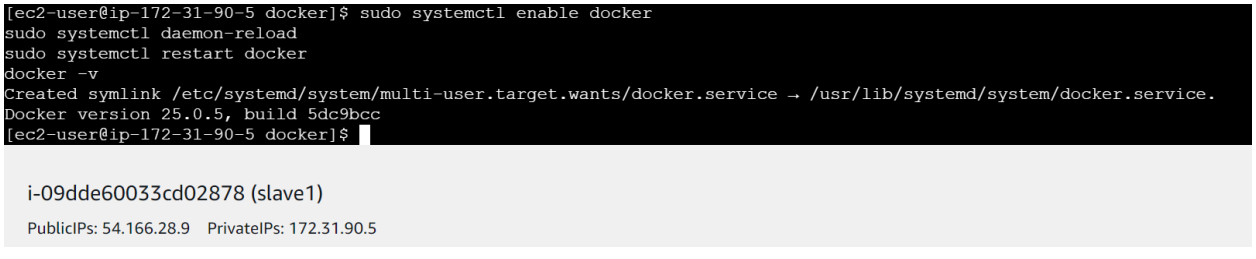
Slave2:-

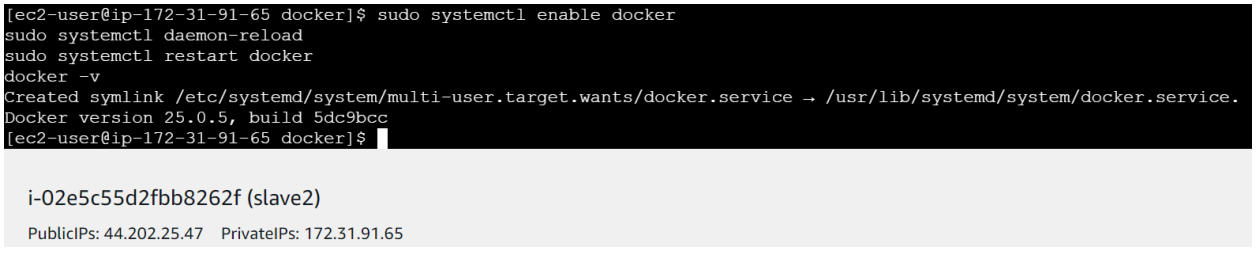


c] Run these commands to enable docker.

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







3. Install Kubernetes on all 3 machines

a]SELinux needs to be disabled before configuring kubelet.

- sudo setenforce 0
- sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-91-140 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

b] Add kubernetes repository :-

```
cat <<EOF | sudo tee /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=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

```
[ec2-user@ip-172-31-91-140 docker]$ cat <<EOF | sudo tee /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=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[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=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ cat <<EOF | sudo tee /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=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[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=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ cat <<EOF | sudo tee /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=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[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=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

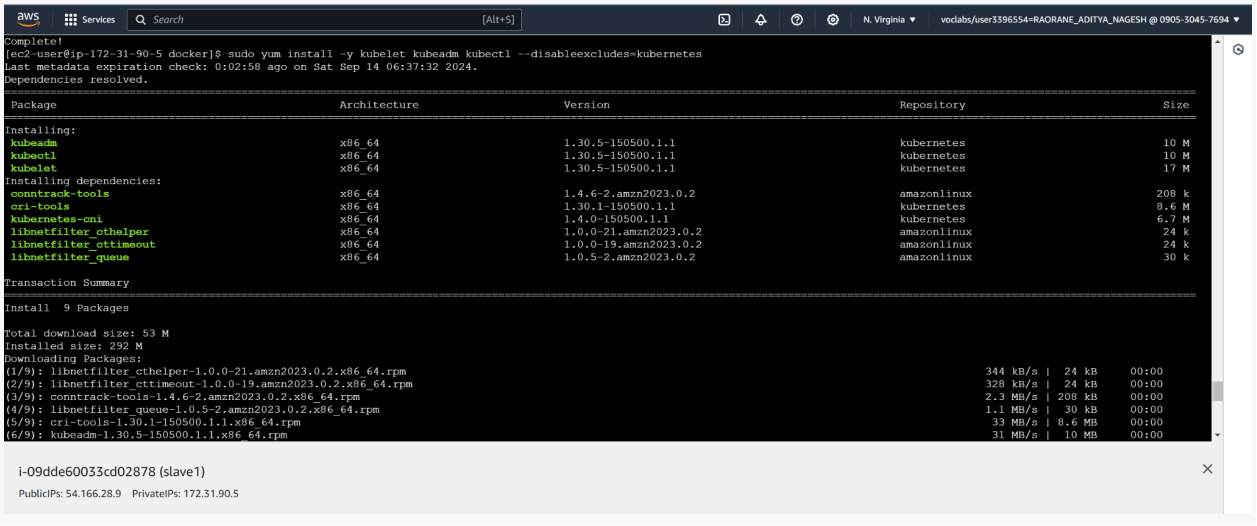
c] Type following commands to install set of kubernetes packages:

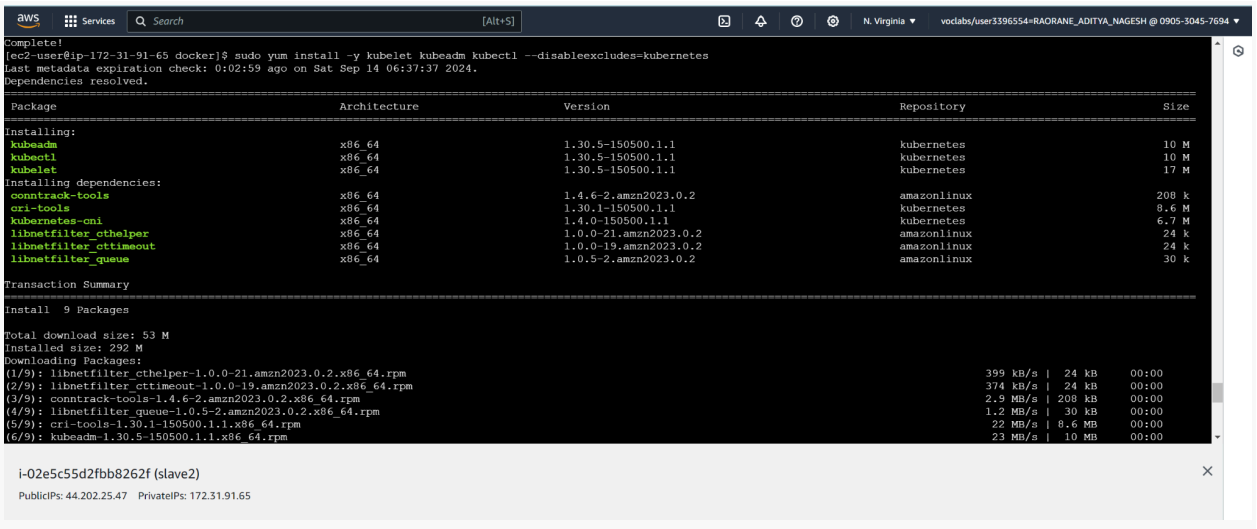
- sudo yum update
- sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

```
[ec2-user@ip-172-31-91-140 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:02:56 ago on Sat Sep 14 06:37:25 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture            Version                 Repository                Size
================================================================================
Installing:
 kubeadm                 x86_64                  1.30.5-150500.1.1       kubernetes                10 M
 kubectl                 x86_64                  1.30.5-150500.1.1       kubernetes                10 M
 kubelet                 x86_64                  1.30.5-150500.1.1       kubernetes                17 M
Installing dependencies:
 conntrack-tools         x86_64                  1.4.6-2.amzn2023.0.2    amazonlinux              208 k
 cri-tools               x86_64                  1.30.1-150500.1.1       kubernetes               8.6 M
 kubernetes-cni          x86_64                  1.4.0-150500.1.1        kubernetes               6.7 M
 libnetfilter_cthelper   x86_64                  1.0.0-21.amzn2023.0.2   amazonlinux               24 k
 libnetfilter_cttimeout  x86_64                  1.0.0-19.amzn2023.0.2   amazonlinux               24 k
 libnetfilter_queue      x86_64                  1.0.5-2.amzn2023.0.2    amazonlinux               30 k
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140





d] After installing Kubernetes, we need to configure internet options to allow bridging.

- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p

```
[ec2-user@ip-172-31-91-140 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

```
[ec2-user@ip-172-31-90-5 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-90-5 docker]$
```

i-09dde60033cd02878 (slave1)

PublicIPs: 54.166.28.9 PrivateIPs: 172.31.90.5

```
[ec2-user@ip-172-31-91-65 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-91-65 docker]$
```

i-02e5c55d2fbb8262f (slave2)

PublicIPs: 44.202.25.47 PrivateIPs: 172.31.91.65

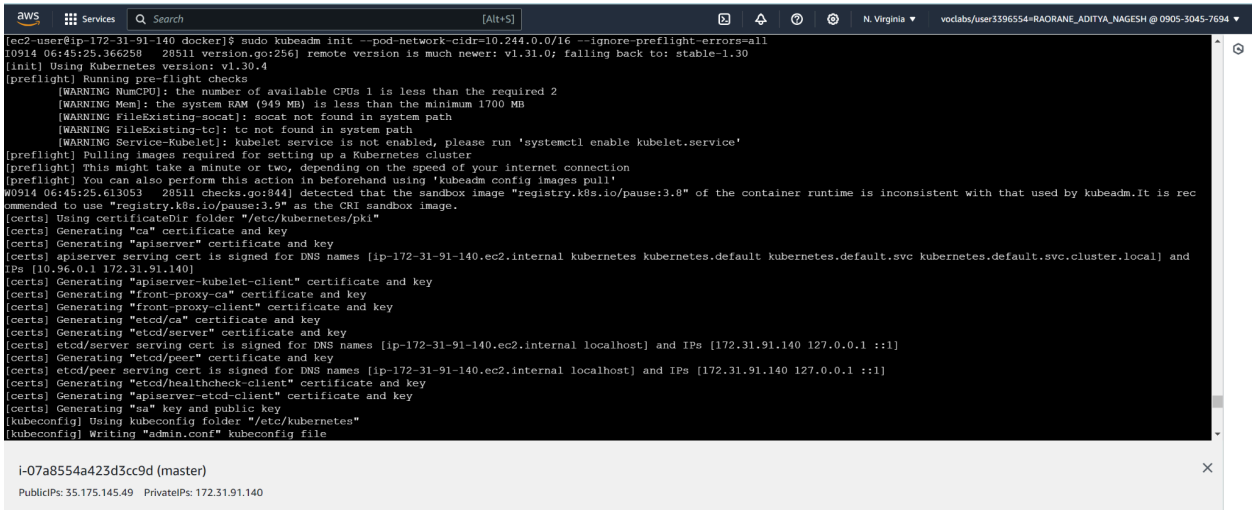
4. **Perform this ONLY on the Master machine**

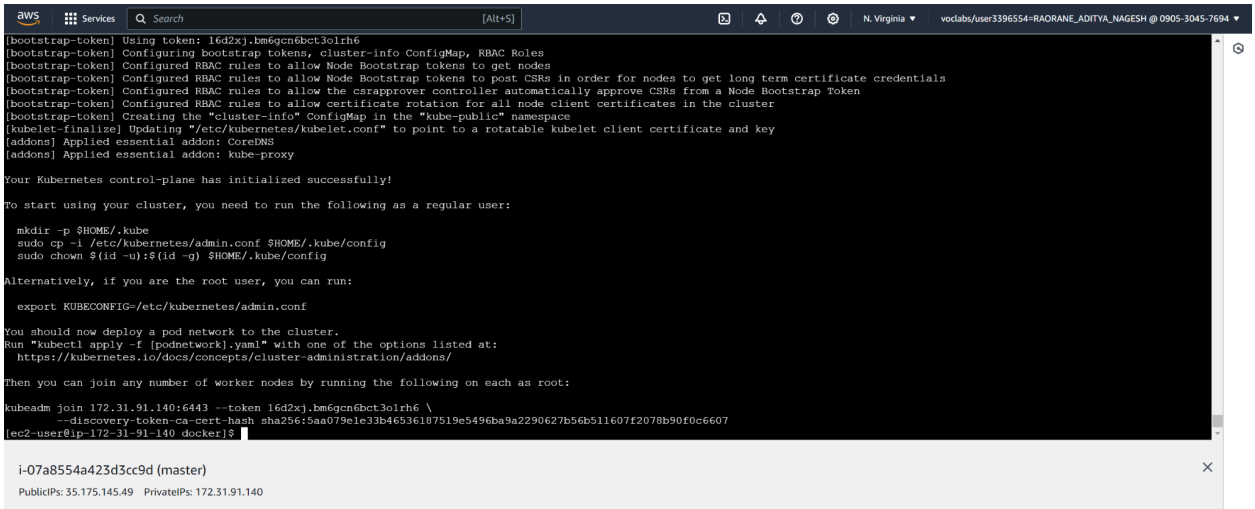
a] Initialize kubernetes by typing below command

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

b] Copy the mkdir and chown commands from the top and execute them.

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





c] Use this link to join the other nodes :-

kubeadm join 172.31.91.140:6443 --token 16d2xj.bm6gcn6bct3o1rh6 \
--discovery-token-ca-cert-hash sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607

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

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

```
[ec2-user@ip-172-31-91-140 docker]$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-91-140 docker]$ ^C
[ec2-user@ip-172-31-91-140 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
[ec2-user@ip-172-31-91-140 docker]$ kubectl get pods
No resources found in default namespace.
[ec2-user@ip-172-31-91-140 docker]$
```

i-07a8554a423d3cc9d (master)

PublicIPs: 35.175.145.49 PrivateIPs: 172.31.91.140

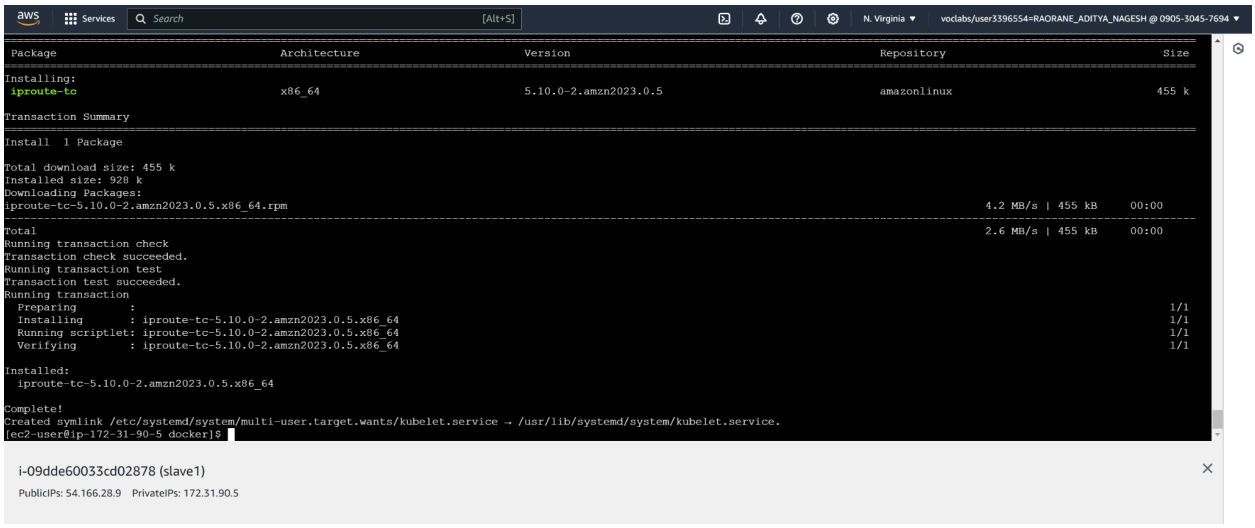
e] Check the created pod using this command

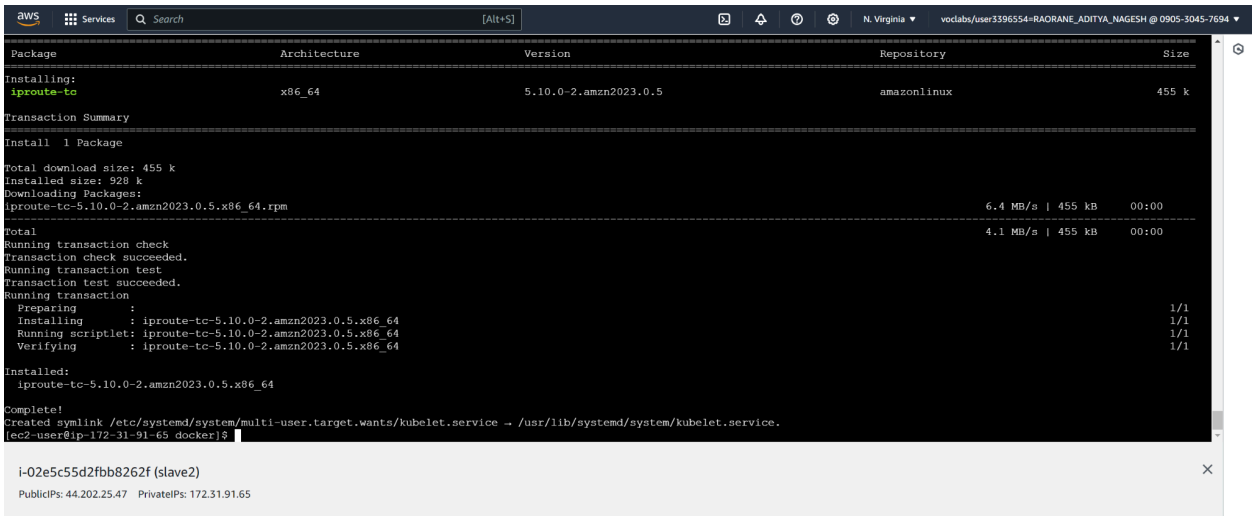
- kubectl get pods

5. **Perform this ONLY on the slave machines**

   a] Paste the below command on all 2 salve machines

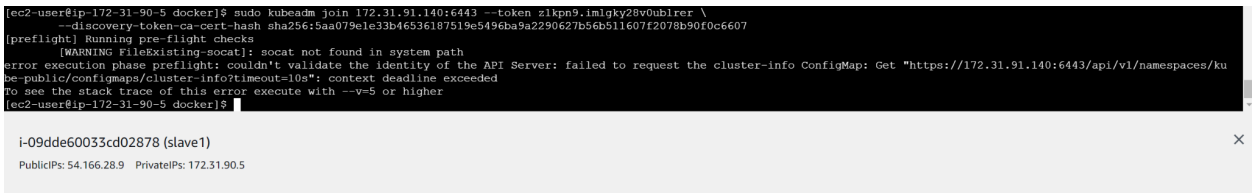
   - sudo yum install iproute-tc -y
   - sudo systemctl enable kubelet
   - sudo systemctl restart kubelet

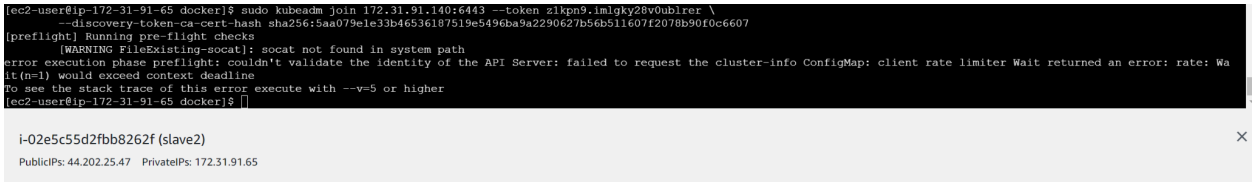




b] Run this command to join the master machine.

- kubeadm join 172.31.91.140:6443 --token 16d2xj.bm6gcn6bct3o1rh6 \
  --discovery-token-ca-cert-hash sha256:5aa079e1e33b46536187519e5496ba9a2290627b56b511607f2078b90f0c6607





c] 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.**

Note:-
The "deadline exceeded" error likely occurred due to connectivity issues, firewall restrictions, expired tokens, or misconfigurations preventing the node from properly communicating with the Kubernetes control plane.

Conclusion:
We explored Kubernetes Cluster Architecture by setting up and configuring a Kubernetes cluster on AWS EC2 Linux instances. After launching EC2 machines, we installed Docker as the container runtime and deployed Kubernetes using `kubeadm, kubelet, kubectl`. During the node-joining process, we encountered common challenges such as network connectivity issues and token expiration. Overcoming these highlighted the importance of proper network configuration, firewall settings, and token management, ensuring smooth communication between nodes and the control plane in the cluster.