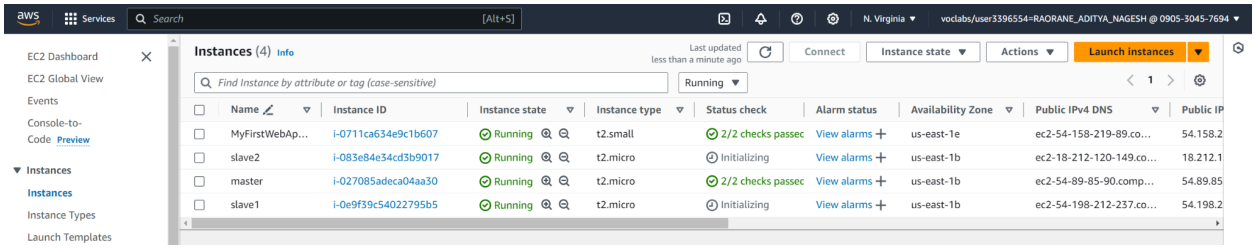
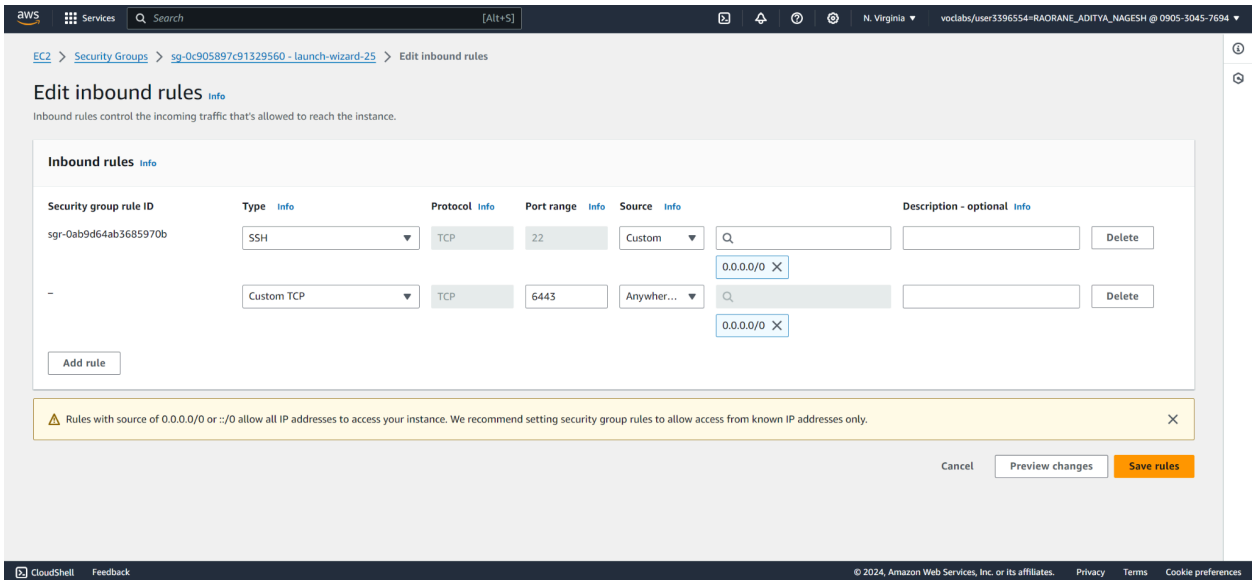
Name: Aditya Nagesh Raorane Class: D15C/ Batch B Roll No: 44

Aim: To understand the Kubernetes Cluster Architecture, install and spin up a Kubernetes Cluster on Linux Machines/Cloud.

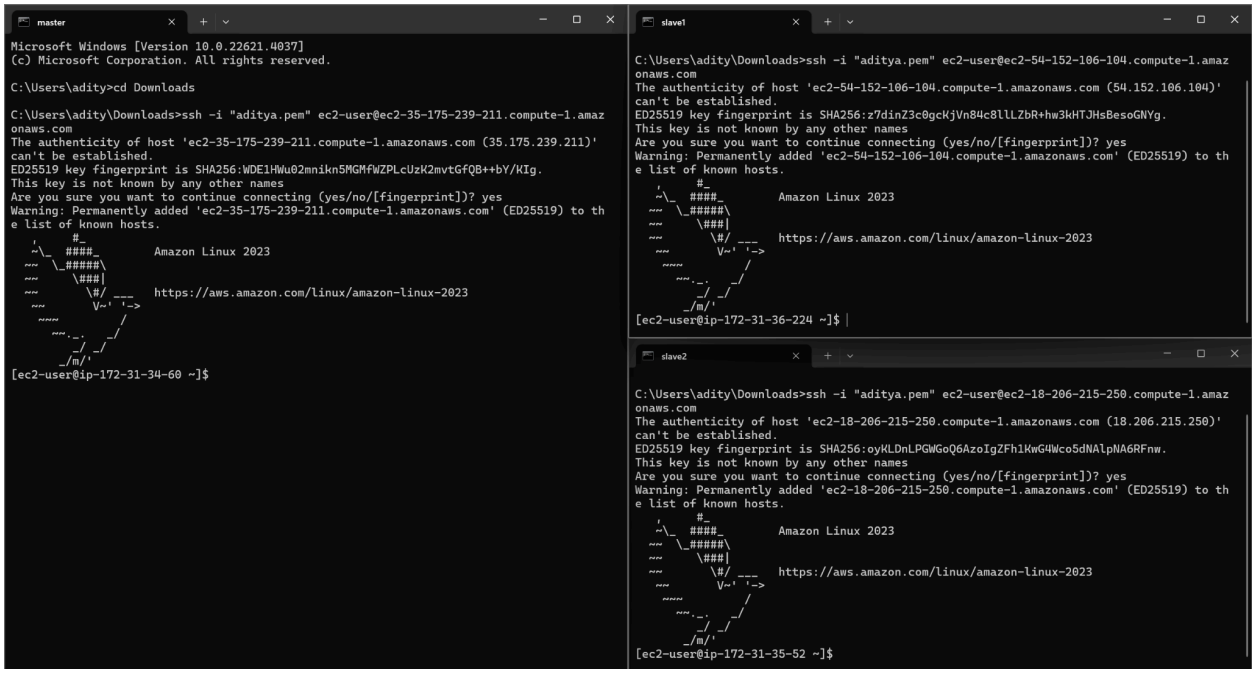
1] Create 3 EC2 instances with all running on Amazon Linux as OS with inbound SSH allowed. To efficiently run the machines it is advised to select the instance type as t2.medium which comes with 4 GiB of memory and 2 vCPU's.



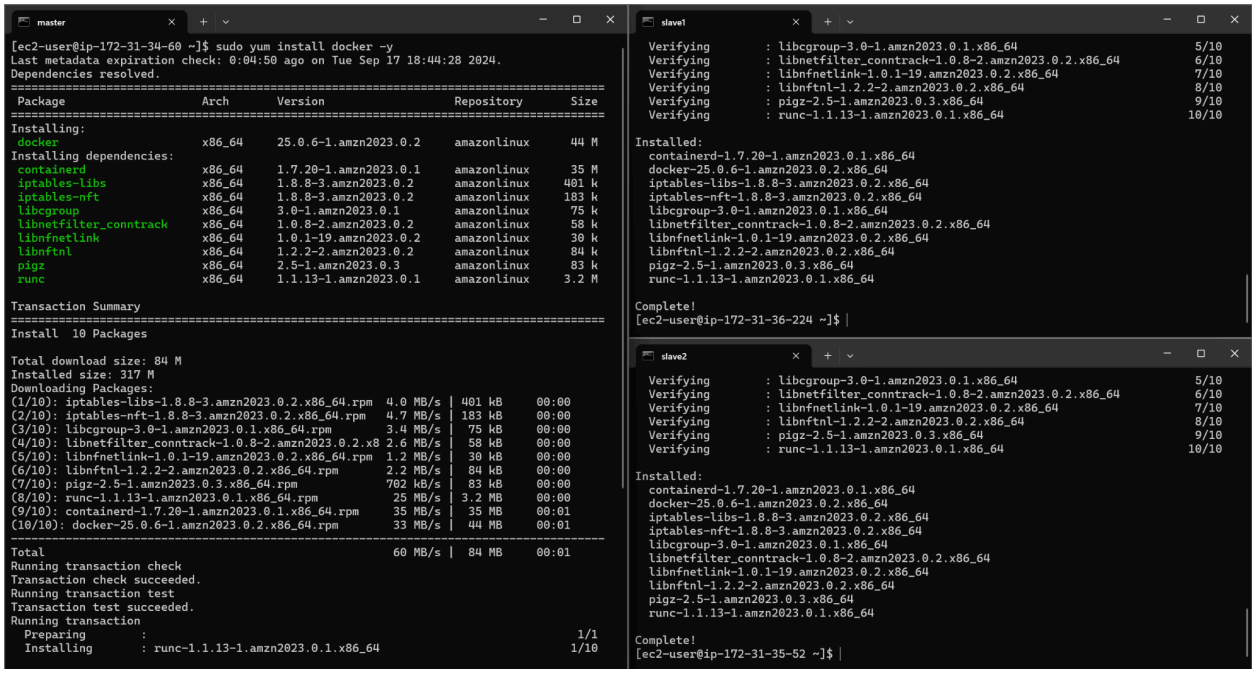
2]Update the inbound rules for the security groups on all three machines to allow TCP traffic on port 6443 from source 0.0.0.0/0.



3] SSH into all 3 machines each in a separate terminal.



4]To install Docker, use the following command:
sudo yum install docker -y

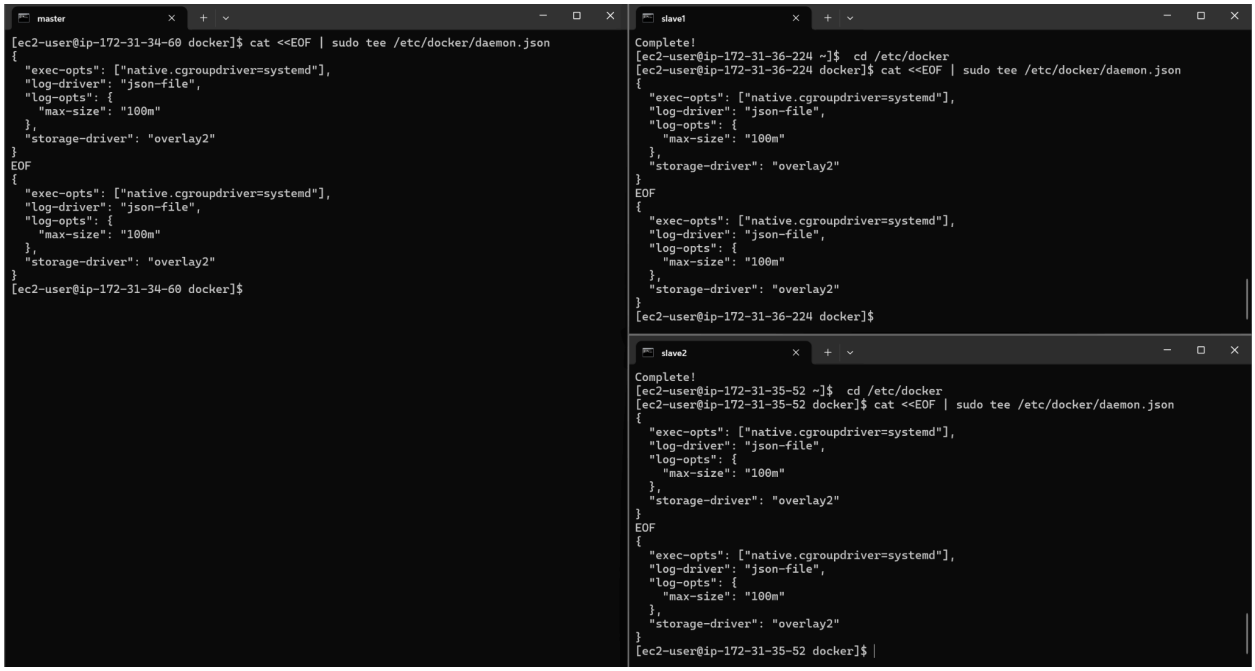


5] Then, configure cgroup in a daemon.json file by using the following commands. This allows kubernetes to manage host more efficiently

● cd /etc/docker

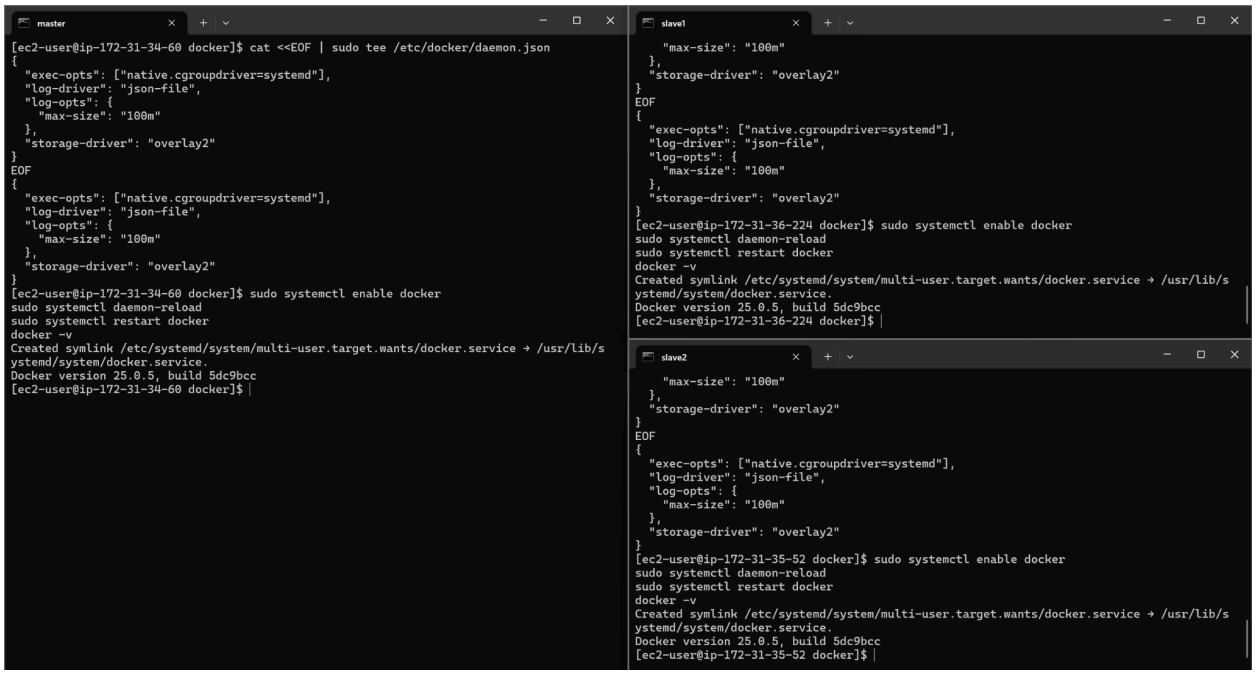
● cat <<EOF | sudo tee /etc/docker/daemon.json

```
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
```



6] After configuring restart docker service service :

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

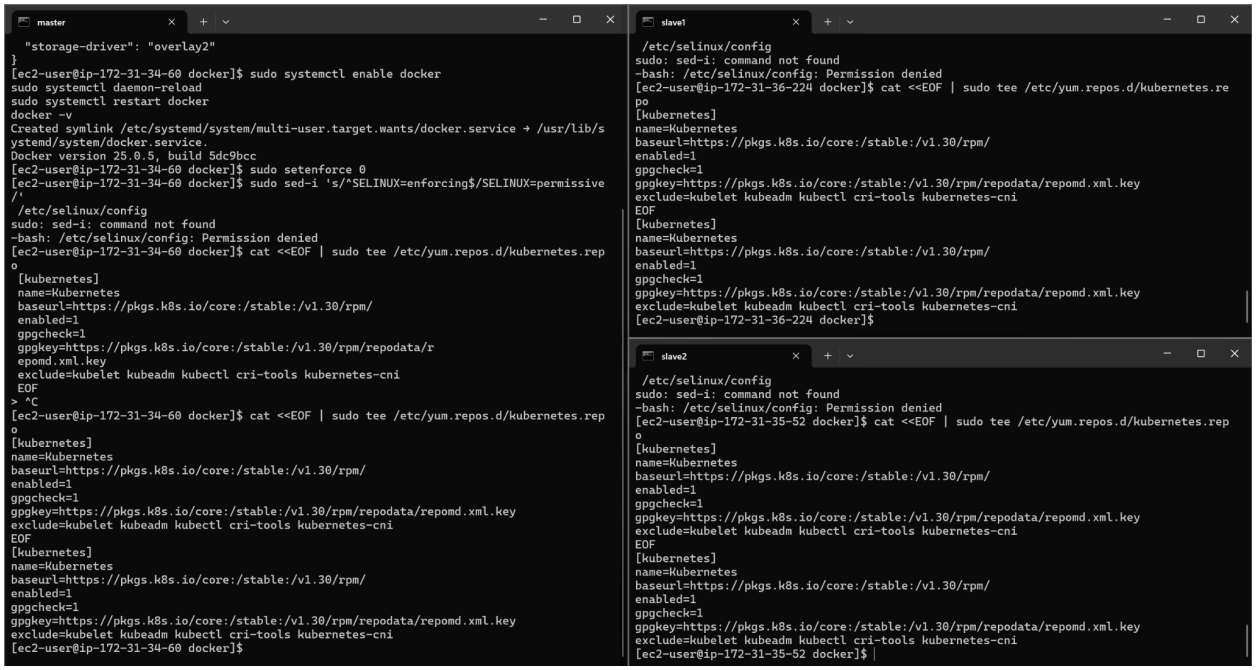


7]SELinux needs to be disabled before configuring kubelet to avoid interference with kubernetes api server

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

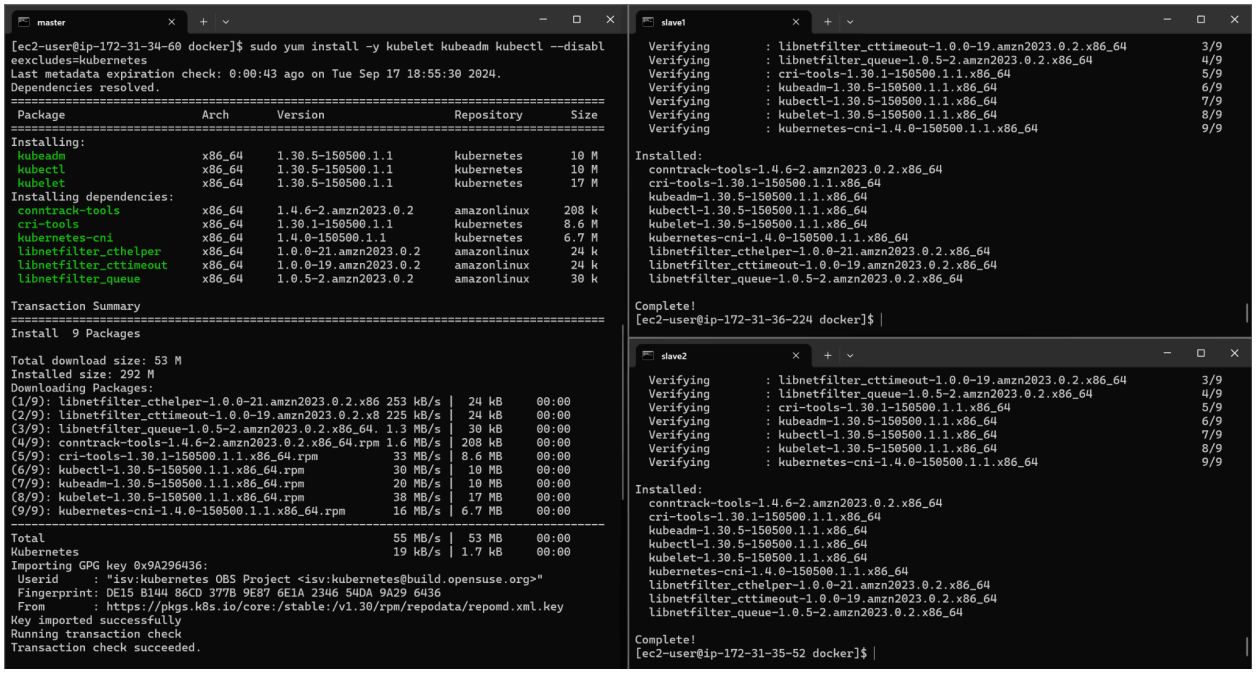
Add kubernetes repository (paste in terminal)

```
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/r
epomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```



8] Typefollowingcommandstoinstallsetofkubernetespackages:

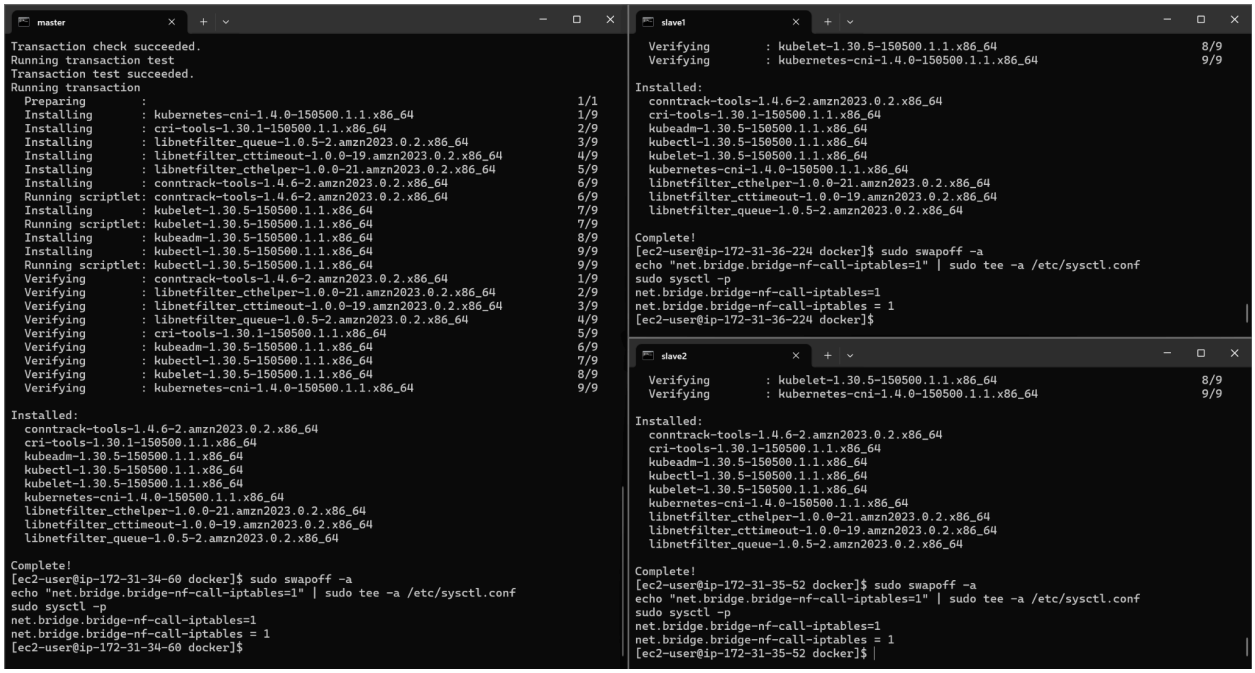
- sudo yum update
- sudo yum install - y kubelet kubeadm kubectl --disableexcludes=kubernetes



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

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

● sudo sysctl -p



10] **Perform this ONLY on the MASTER machine**:-

a)Initialize Kubernetes By Typing Below Command:

- sudokubeadminit--pod-network-cidr=10.244.0.0/16--ignore-preflight-errors=all

b)Copy the mkdir and chown commands from 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)Copy the Kubernetes join command from your output to the clipboard

- kubeadm join 172.31.34.60:6443 --token f70o2w.perwi1ygbt7qu724 \
  --discovery-token-ca-cert-hash sha256:df9aa1a237965f949c055717ba438799f890a8647b8cd10386fa7ed864e59096

d)Then, add a common networking plugin called flammel file as mentioned code.

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

```
[ec2-user@ip-172-31-34-60 docker]$ kubectl apply -f https://raw.githubusercontent.com/c
oreos/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-172-31-34-60 docker]$
```

e)Check the created node using this command

- kubectl get nodes

```
[ec2-user@ip-172-31-34-60 docker]$ kubectl get nodes
NAME                           STATUS   ROLES           AGE     VERSION
ip-172-31-34-60.ec2.internal   Ready    control-plane   5m41s   v1.30.5
```

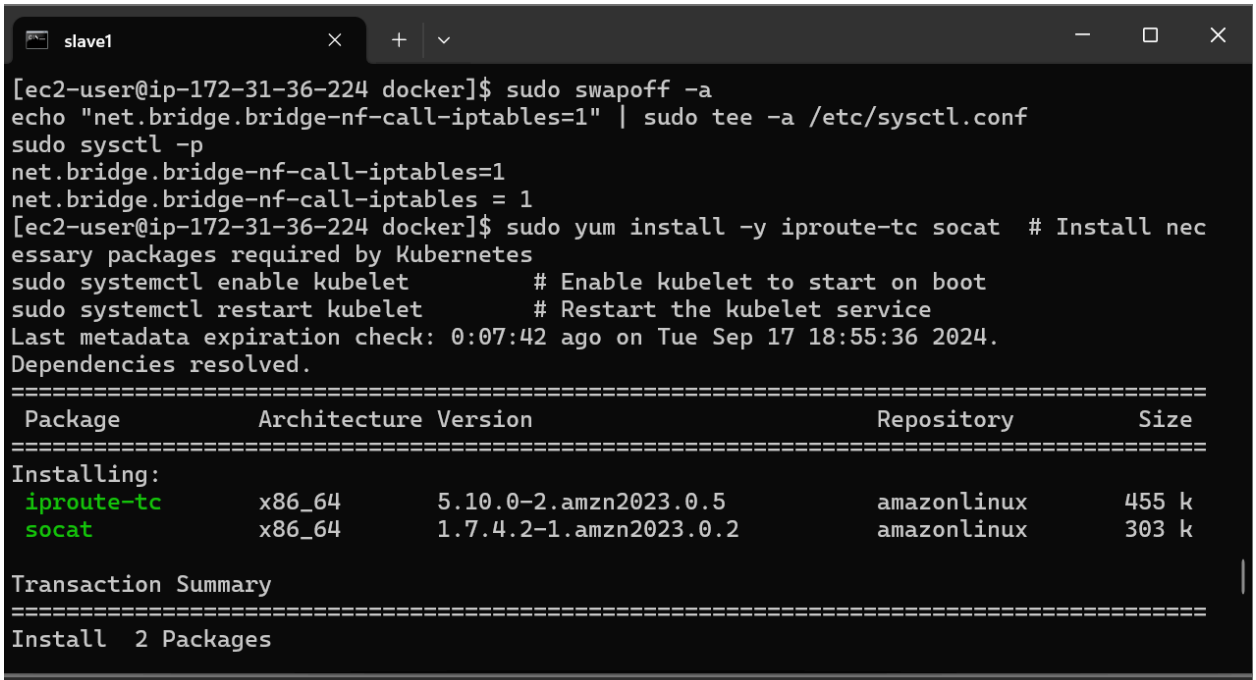
11] **Perform this ONLY on the WORKER machines:-**

a)Paste the below command on all 2 worker machines

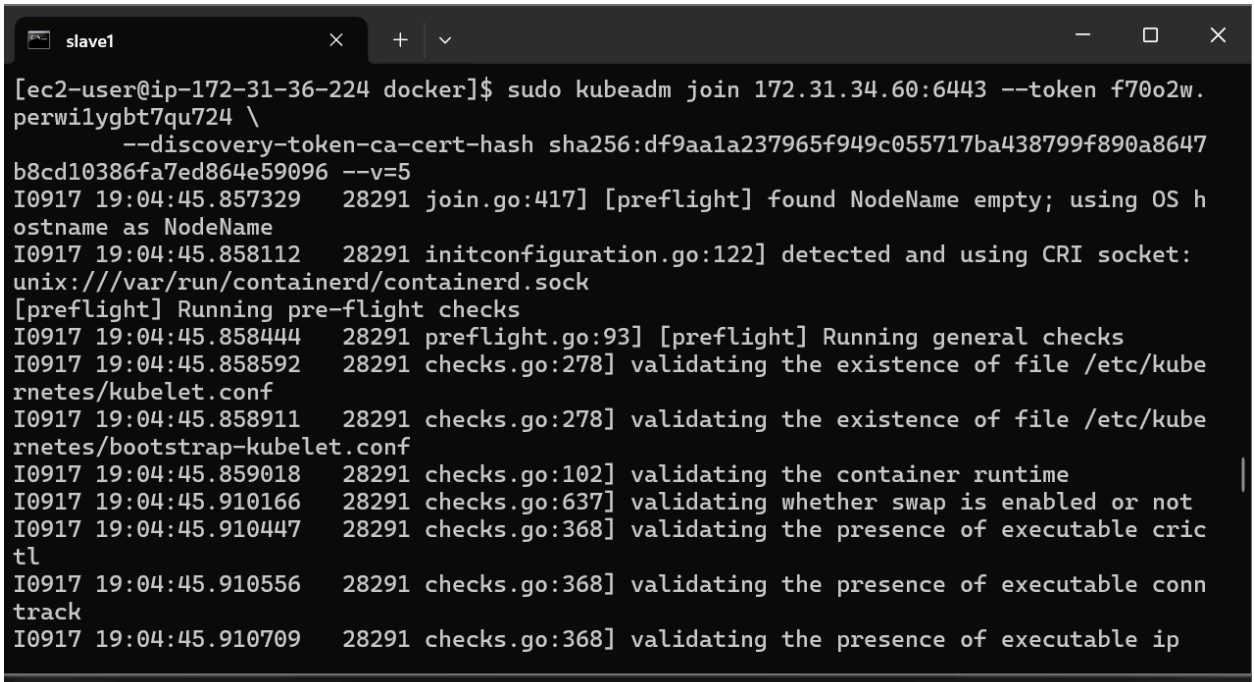
● sudo yum install iproute-tc socat-y (necessary packages required by kubernetes)

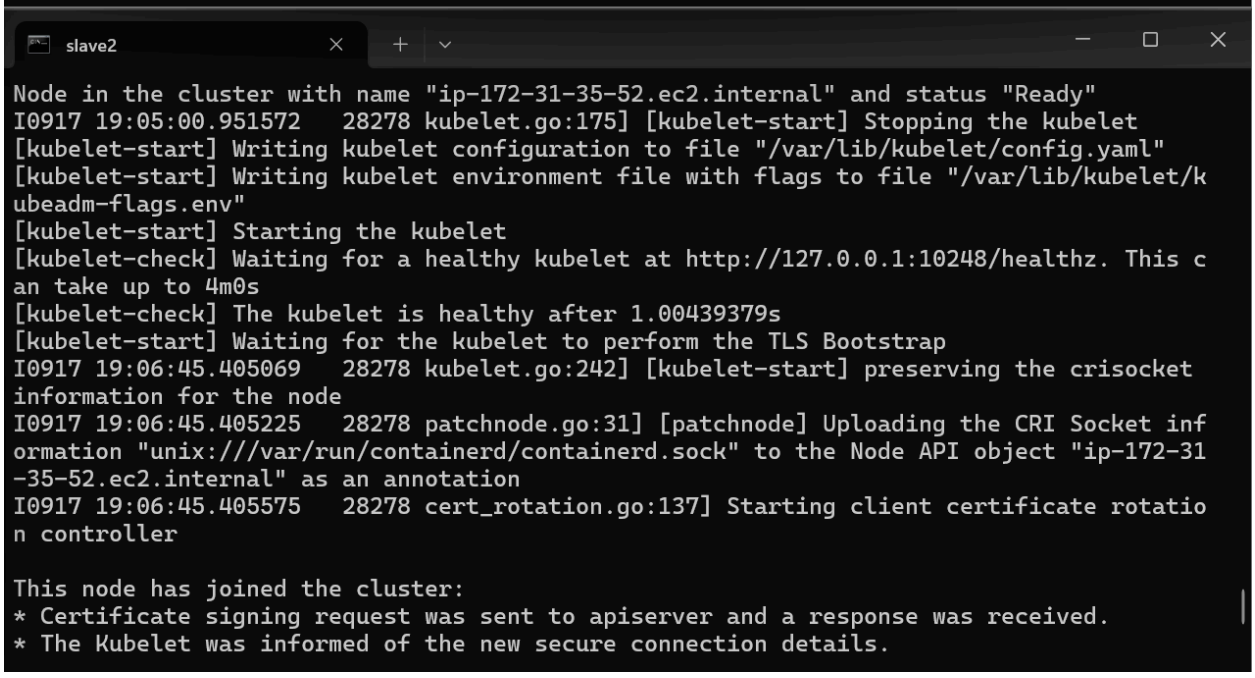
● sudo systemctl enable kubelet

● sudo systemctl restart kubelet



b)kubeadm join 172.31.34.60:6443 --token f70o2w.perwi1ygbt7qu724 \
--discovery-token-ca-cert-hash sha256:df9aa1a237965f949c055717ba438799f890a8647b8cd10386fa7ed864e59096





c) With the help of command the worker nodes are connected to the master node and are ready to do tasks assigned by the master node.

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.

```
[ec2-user@ip-172-31-34-60 docker]$ kubectl get nodes
NAME                            STATUS   ROLES           AGE     VERSION
ip-172-31-34-60.ec2.internal    Ready    control-plane   9m18s   v1.30.5
ip-172-31-35-52.ec2.internal    Ready    <none>          112s    v1.30.5
ip-172-31-36-224.ec2.internal   Ready    <none>          110s    v1.30.5
[ec2-user@ip-172-31-34-60 docker]$
```

Conclusion: We commenced the installation and configuration of the essential Kubernetes packages. While some were accessible via the default Linux repositories, others required the addition of external repositories for proper installation. During the setup, we encountered an issue where the nodes were tainted, resulting in the Kubernetes API server crashing. This was mitigated by removing the taints from the nodes. Additionally, SELinux was disabled to prevent potential conflicts with Kubernetes operations. In conclusion, we successfully integrated the worker nodes with the Kubernetes master node, achieving a fully operational cluster setup.