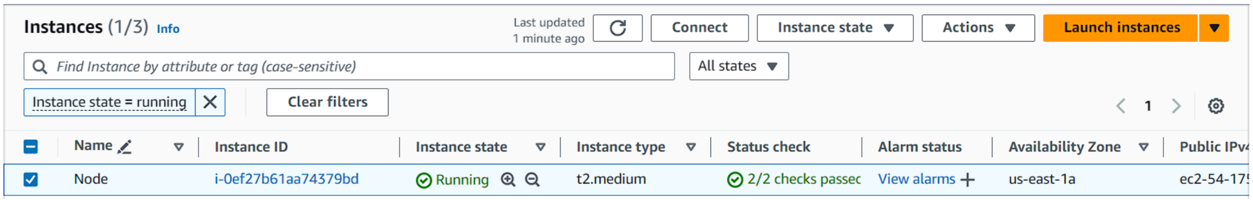
Name - Brijesh Sharma **Div - D15C** **Roll No. 48**

Experiment - 4

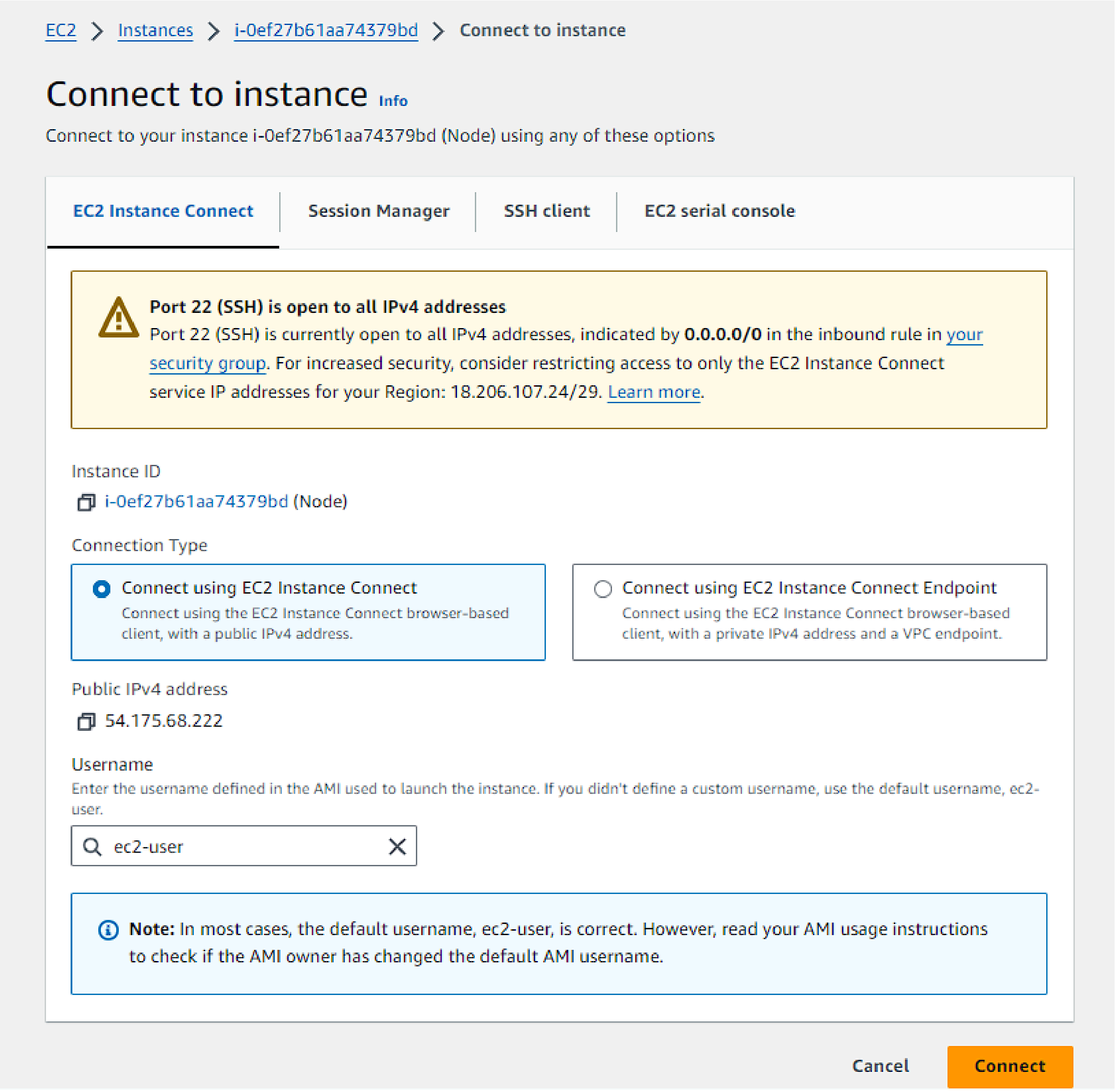
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes Application.

1) Create an EC2 instance and allow SSH traffic to connect.


Instances (1/3) Info
Last updated 1 minute ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
Instance state = running ✕
Clear filters
< 1 >

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
☑	Node	i-0ef27b61aa74379bd	✓ Running	t2.medium	✓ 2/2 checks passec	View alarms +	us-east-1a	ec2-54-175

2) Select the instance and click on Connect and go to Connect.



Connect to instance Info

Connect to your instance i-0ef27b61aa74379bd (Node) using any of these options

Port 22 (SSH) is open to all IPv4 addresses
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. Learn more.

Instance ID
i-0ef27b61aa74379bd (Node)

Connection Type

● Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

○ Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

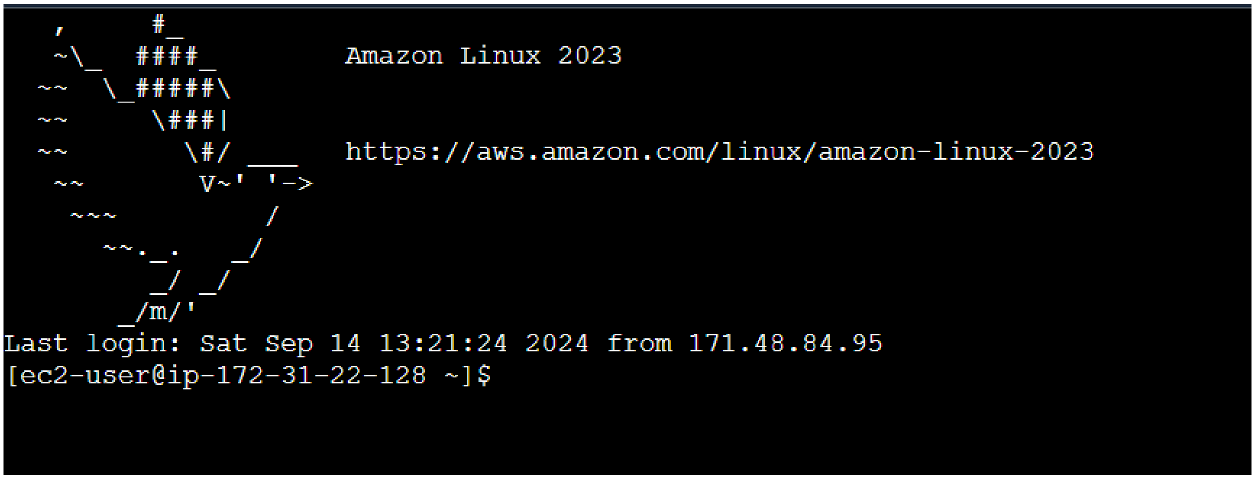
Public IPv4 address
54.175.68.222

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

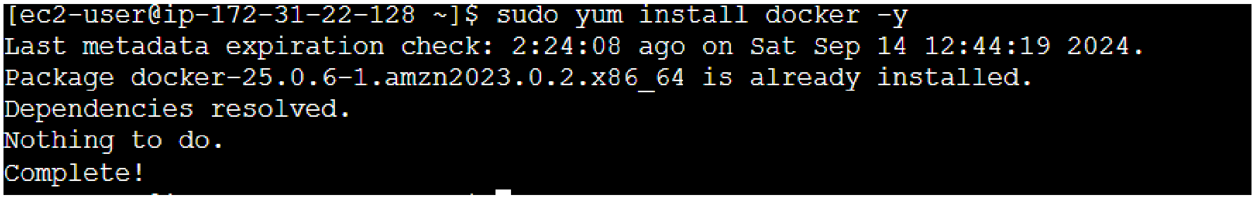
3) EC2 server is connected.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Sep 14 13:21:24 2024 from 171.48.84.95
[ec2-user@ip-172-31-22-128 ~]$
```

4) To install docker run the following command:

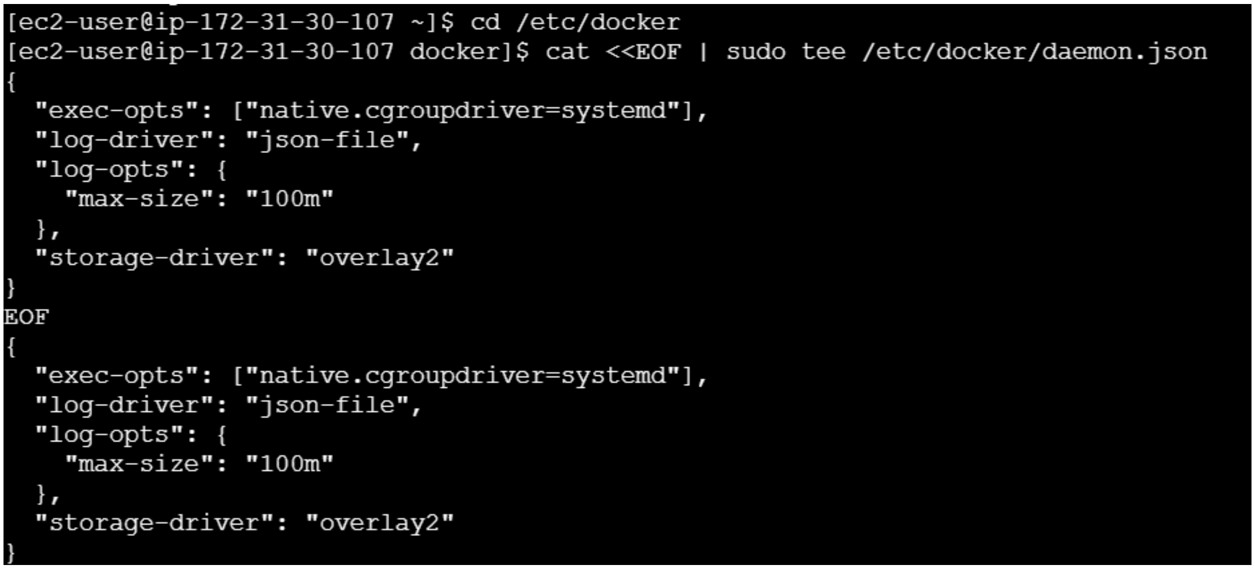
   sudo yum install docker -y



```
[ec2-user@ip-172-31-22-128 ~]$ sudo yum install docker -y
Last metadata expiration check: 2:24:08 ago on Sat Sep 14 12:44:19 2024.
Package docker-25.0.6-1.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

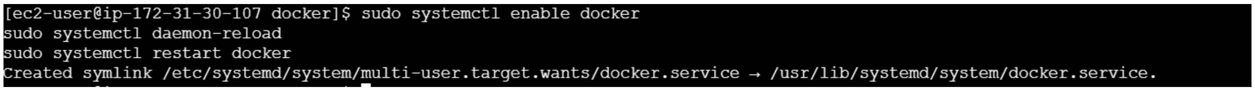
5) Configure cgroup in daemon.json file using the following commands

```
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
```



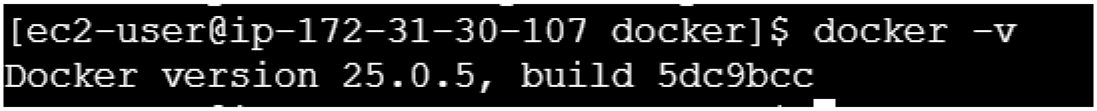
```
[ec2-user@ip-172-31-30-107 ~]$ cd /etc/docker
[ec2-user@ip-172-31-30-107 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

6) Run the following command after this:
   sudo systemctl enable docker
   sudo systemctl daemon-reload
   sudo systemctl restart docker



```
[ec2-user@ip-172-31-30-107 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
```

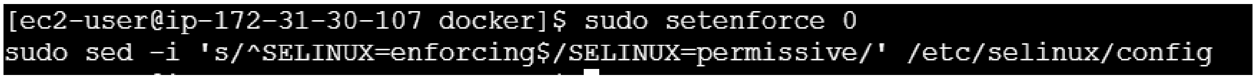
7) Check for installation of docker by docker -v command.



```
[ec2-user@ip-172-31-30-107 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

Install Kubernetes

8) Disable SELinux before configuring kubelet
   - sudo setenforce 0
   - sudo sed -i 's/^SELINUX-enforcing$/SELINUX=permissive/' /etc/selinux/config

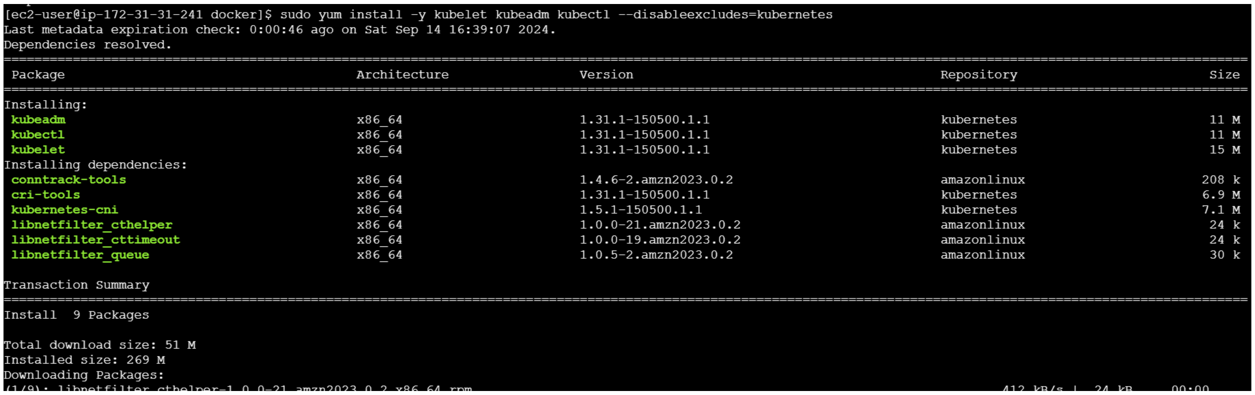


```
[ec2-user@ip-172-31-30-107 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

9) Add kubernetes repository
   cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
   [kubernetes]
   name=Kubernetes
   baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
   enabled=1

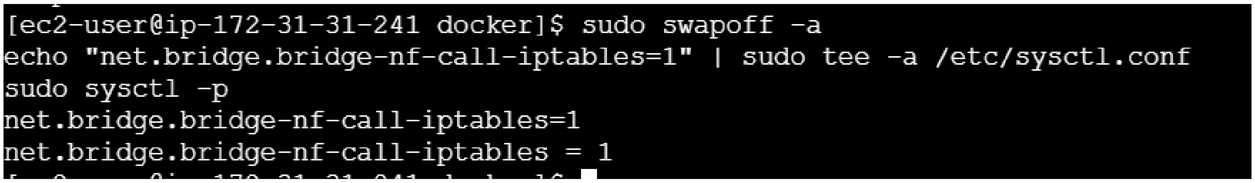
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

10) Run the commands to update and install kubernetes packages
sudo yum update
sudo yum install -y kubelet kubeadm kubectl --disableexcludes kubernetes



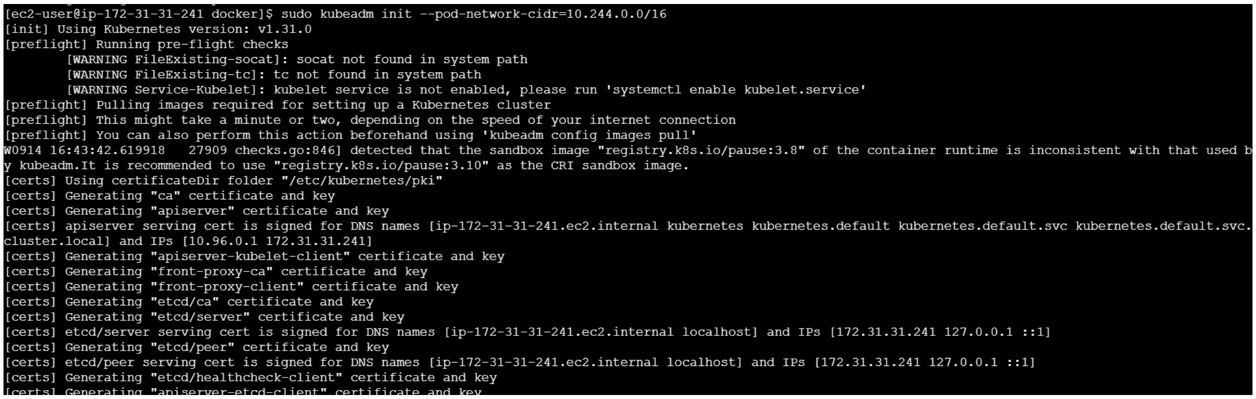
```
[ec2-user@ip-172-31-31-241 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:46 ago on Sat Sep 14 16:39:07 2024.
Dependencies resolved.
================================================================================================================================
 Package                        Architecture         Version                                Repository                    Size
================================================================================================================================
Installing:
 kubeadm                        x86_64               1.31.1-150500.1.1                      kubernetes                    11 M
 kubectl                        x86_64               1.31.1-150500.1.1                      kubernetes                    11 M
 kubelet                        x86_64               1.31.1-150500.1.1                      kubernetes                    15 M
Installing dependencies:
 conntrack-tools                x86_64               1.4.6-2.amzn2023.0.2                   amazonlinux                  208 k
 cri-tools                      x86_64               1.31.1-150500.1.1                      kubernetes                   6.9 M
 kubernetes-cni                 x86_64               1.5.1-150500.1.1                       kubernetes                   7.1 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

Transaction Summary
================================================================================================================================
Install  9 Packages

Total download size: 51 M
Installed size: 269 M
Downloading Packages:
(1/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm                               412 kB/s |  24 kB     00:00
```

11) 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



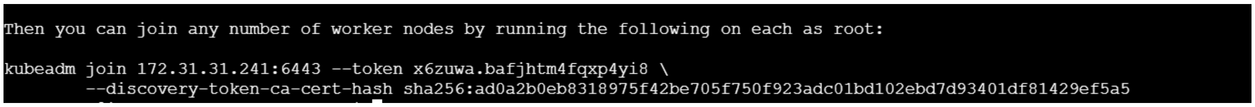
```
[ec2-user@ip-172-31-31-241 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-31-241 docker]$
```

12) Initialize the kubecluster
sudo kubeadm init --pod-network-cidr-10.244.0.0/16



```
[ec2-user@ip-172-31-31-241 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0914 16:43:42.619918   27909 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used b
y kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-31-241.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.
cluster.local] and IPs [10.96.0.1 172.31.31.241]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-31-241.ec2.internal localhost] and IPs [172.31.31.241 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-31-241.ec2.internal localhost] and IPs [172.31.31.241 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
```

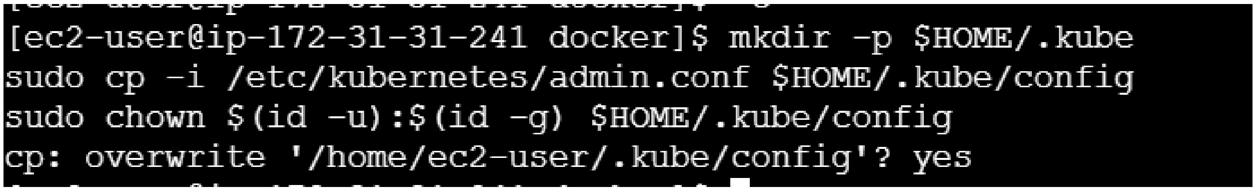
13) Copy the Join Command-



```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.31.241:6443 --token x6zuwa.bafjhtm4fqxp4yi8 \
        --discovery-token-ca-cert-hash sha256:ad0a2b0eb8318975f42be705f750f923adc01bd102ebd7d93401df81429ef5a5
```

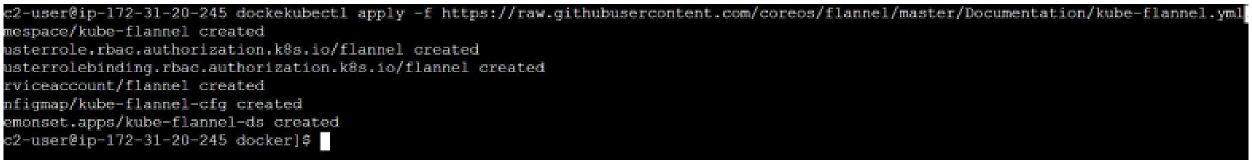
kubeadm join 172.31.31.241:6443 --token x6zuwa.bafjhtm4fqxp4yi8 \
--discovery-token-ca-cert-hash
sha256:ad0a2b0eb8318975f42be705f750f923adc01bd102ebd7d93401df81429ef5a5

14) Run the following commands



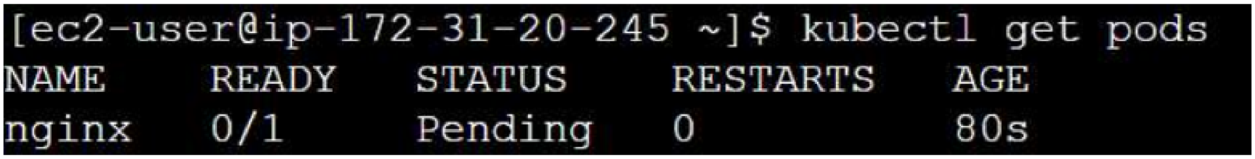
```
[ec2-user@ip-172-31-31-241 docker]$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
cp: overwrite '/home/ec2-user/.kube/config'? yes
```

15) Add a common network plugin called Flannel as mentioned in the code below:
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



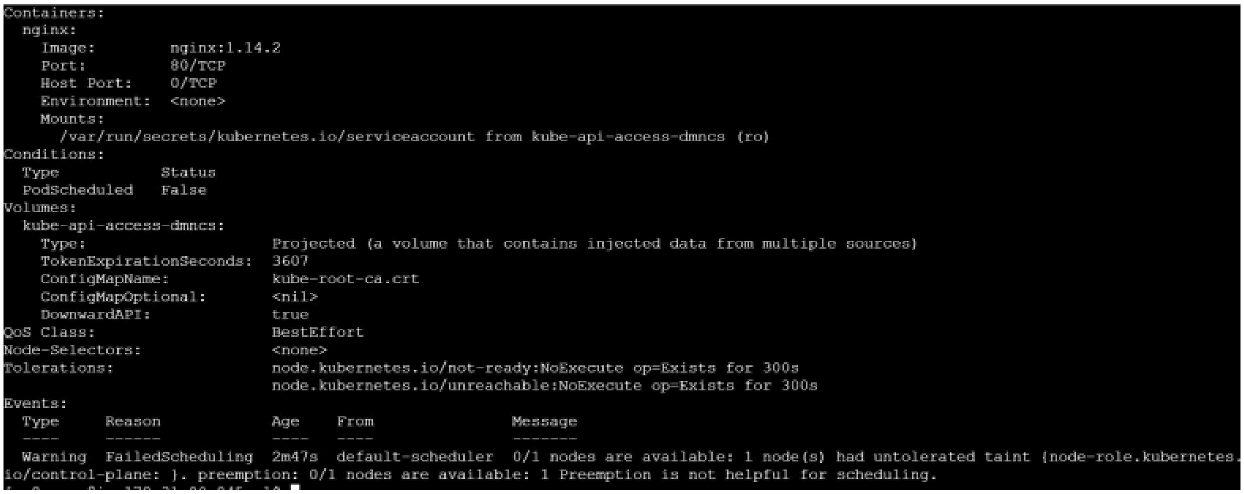
```
c2-user@ip-172-31-20-245 dockekubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
mespace/kube-flannel created
usterrole.rbac.authorization.k8s.io/flannel created
usterrolebinding.rbac.authorization.k8s.io/flannel created
rviceaccount/flannel created
nfigmap/kube-flannel-cfg created
emonset.apps/kube-flannel-ds created
c2-user@ip-172-31-20-245 docker]$
```

16) Check for creation of pods

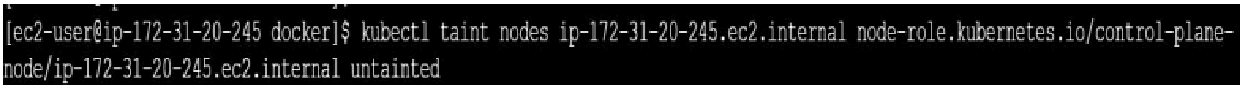


```
[ec2-user@ip-172-31-20-245 ~]$ kubectl get pods
NAME     READY    STATUS     RESTARTS    AGE
nginx    0/1      Pending    0           80s
```

17) To change the state from pending to running, use the following command kubectl describe pod nginx
This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.

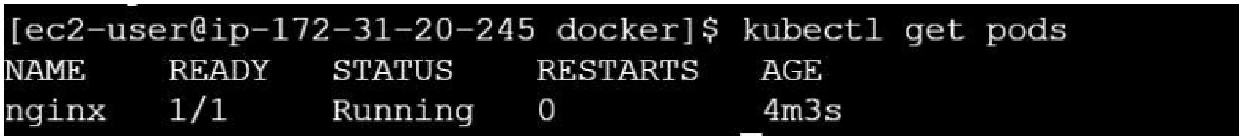


```
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dmncs (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-dmncs:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  2m47s  default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.
io/control-plane: }. preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

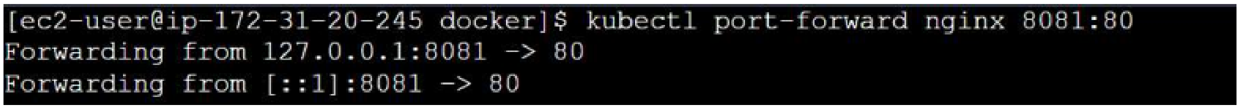


```
[ec2-user@ip-172-31-20-245 docker]$ kubectl taint nodes ip-172-31-20-245.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-20-245.ec2.internal untainted
```

18) Check the status of pods.



```
[ec2-user@ip-172-31-20-245 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          4m3s
```



```
[ec2-user@ip-172-31-20-245 docker]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

Conclusion:
We successfully configured Kubernetes environment on Amazon Linux EC2 instance. We installed Docker and adjusted its settings to use systemd for cgroup management. Then, we installed Kubernetes. We desabled SELinux, and added the Kubernetes repository. Then we installed the necessary components. Then added a network plugin and deployed on Nginx server. We also addressed issues related to pod scheduling and port forwarding, ensuring the Nginx pod.