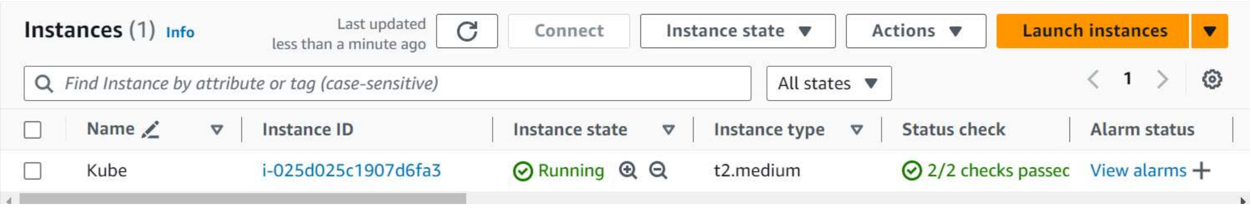
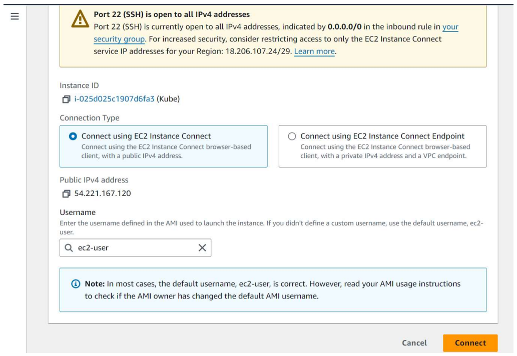
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

1. Create an EC2 instance with OS as Amazon Linux and make sure to allow SSH traffic.


Instances (1) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
□	Kube	i-025d025c1907d6fa3	✓ Running	t2.medium	✓ 2/2 checks passec	View alarms +

2. Select the instance and click on connect. On the page scroll down and click on connect to open the command line.



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.

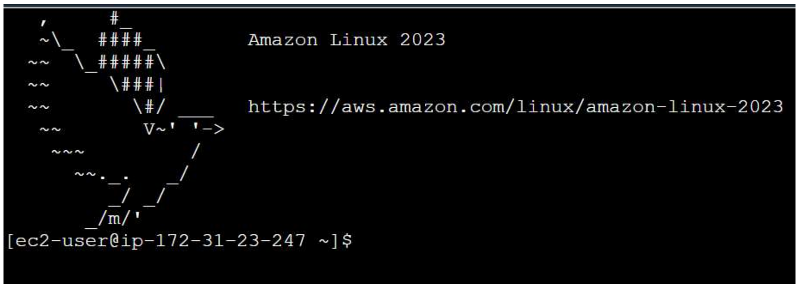
Instance ID
i-025d025c1907d6fa3 (Kube)

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
54.221.167.120

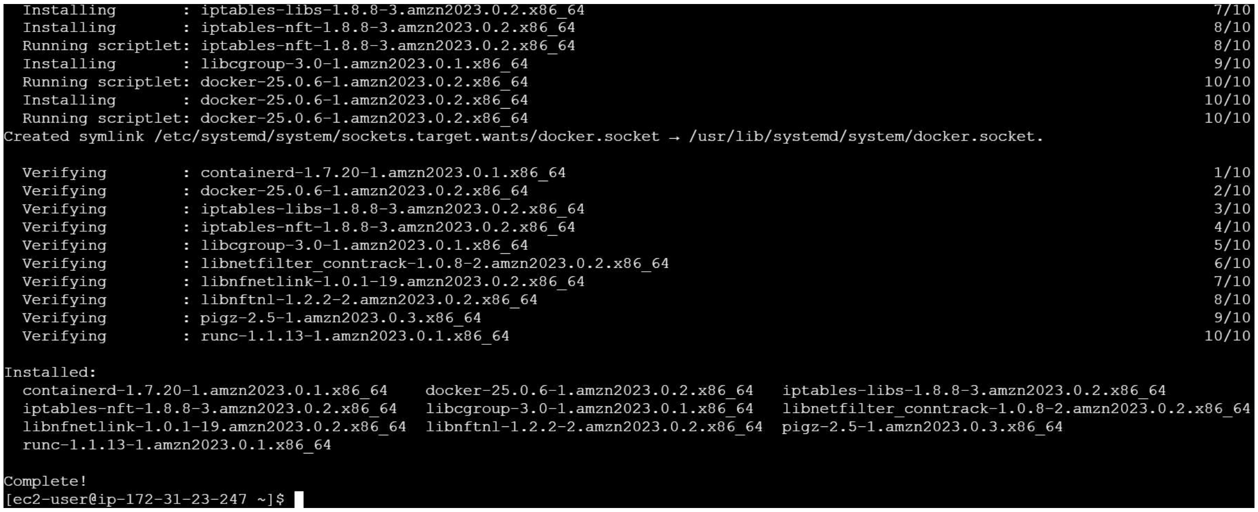
Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
ec2-user

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.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-23-247 ~]$
```

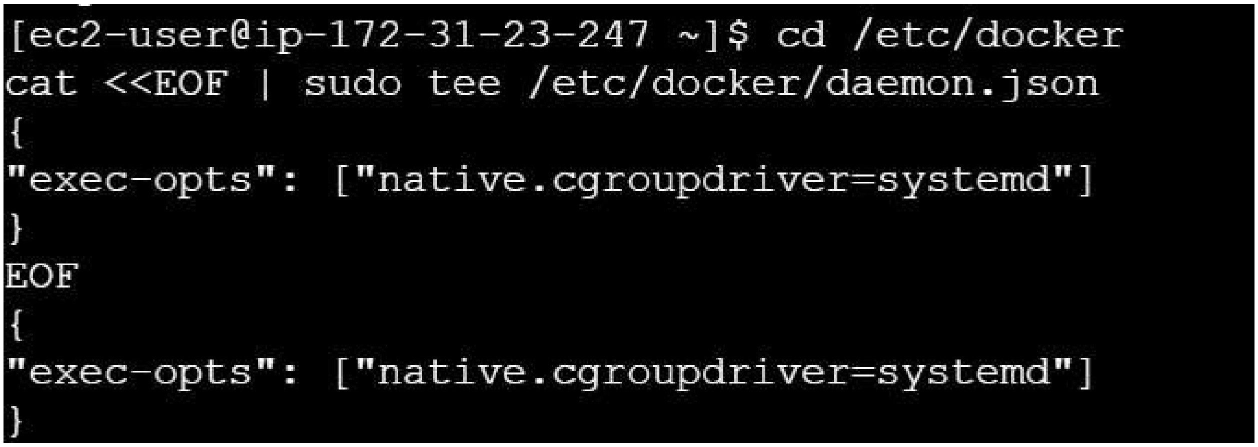
3. To install docker run the following command:
sudo yum install docker -y



```
  Installing       : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                                       7/10
  Installing       : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                        8/10
  Running scriptlet: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                        8/10
  Installing       : libcgroup-3.0-1.amzn2023.0.1.x86_64                                             9/10
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64                                            10/10
  Installing       : docker-25.0.6-1.amzn2023.0.2.x86_64                                            10/10
  Running scriptlet: docker-25.0.6-1.amzn2023.0.2.x86_64                                            10/10
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

  Verifying        : containerd-1.7.20-1.amzn2023.0.1.x86_64                                         1/10
  Verifying        : docker-25.0.6-1.amzn2023.0.2.x86_64                                             2/10
  Verifying        : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                                       3/10
  Verifying        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                                        4/10
  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-172-31-23-247 ~]$
```

4. Configure cgroup in daemon.json file using the following commands:

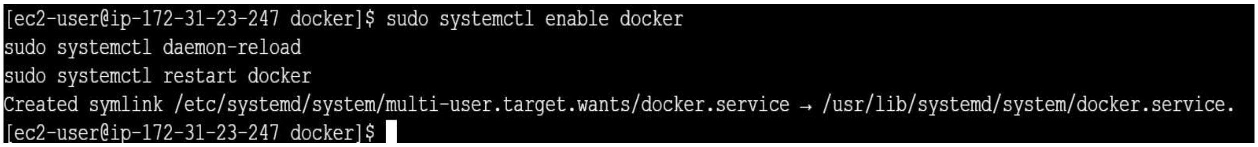
```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
```



```
[ec2-user@ip-172-31-23-247 ~]$ cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

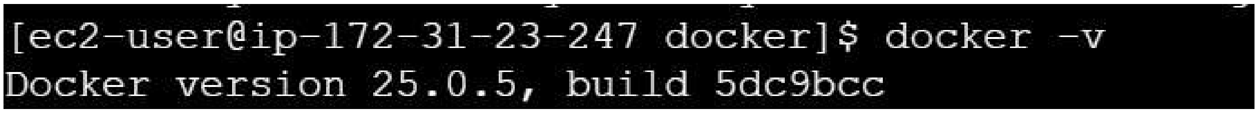
5. Run the following command after this:

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```



```
[ec2-user@ip-172-31-23-247 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.
[ec2-user@ip-172-31-23-247 docker]$
```

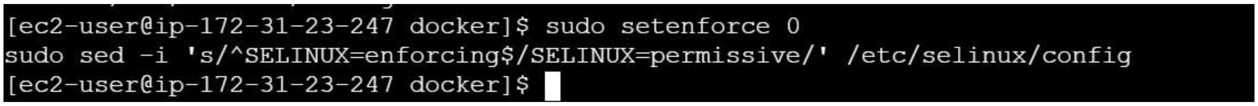
6. Verify installation using docker -v command



```
[ec2-user@ip-172-31-23-247 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

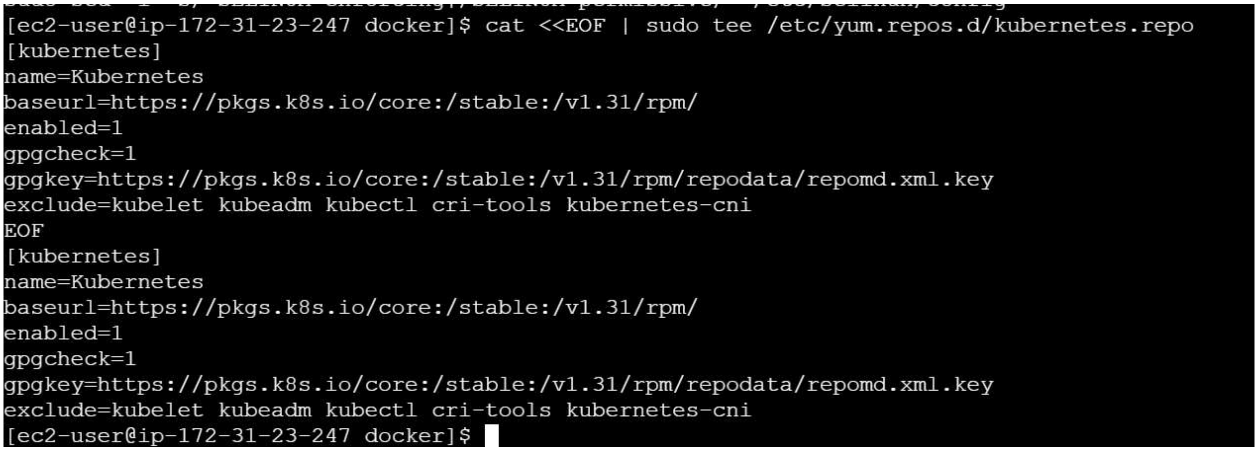
7. Install Kubernetes

I. Disable SELinux before configuring kubelet
sudo setenforce 0
sudo sed -i ‘s/^SELINUX=enforcing$/SELINUX=permissive/’ /etc/selinux/config



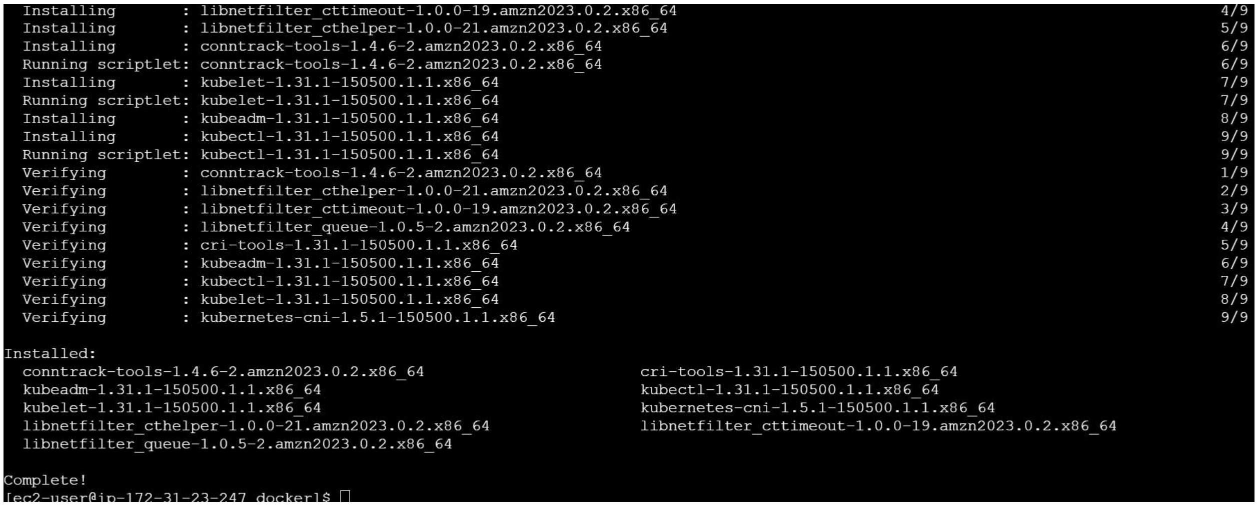
```
[ec2-user@ip-172-31-23-247 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-23-247 docker]$
```

II. 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



```
[ec2-user@ip-172-31-23-247 docker]$ 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
[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
[ec2-user@ip-172-31-23-247 docker]$
```

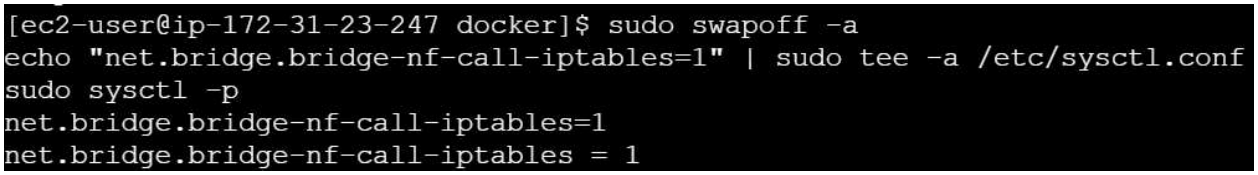
III. Run the commands to update and install kubernetes packages
sudo yum update
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes



```
  Installing       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                                           4/9
  Installing       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                                            5/9
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                                   6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                                   6/9
  Installing       : kubelet-1.31.1-150500.1.1.x86_64                                                                              7/9
  Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                                                                              7/9
  Installing       : kubeadm-1.31.1-150500.1.1.x86_64                                                                              8/9
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                                              9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64                                                                              9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                                                   1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                                            2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                                           3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                                                4/9
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64                                                                            5/9
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64                                                                              6/9
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                                              7/9
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64                                                                              8/9
  Verifying        : kubernetes-cni-1.5.1-150500.1.1.x86_64                                                                        9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                        cri-tools-1.31.1-150500.1.1.x86_64
  kubeadm-1.31.1-150500.1.1.x86_64                                   kubectl-1.31.1-150500.1.1.x86_64
  kubelet-1.31.1-150500.1.1.x86_64                                   kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                 libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-23-247 docker]$ 
```

IV. Configure internet options to allow bridging

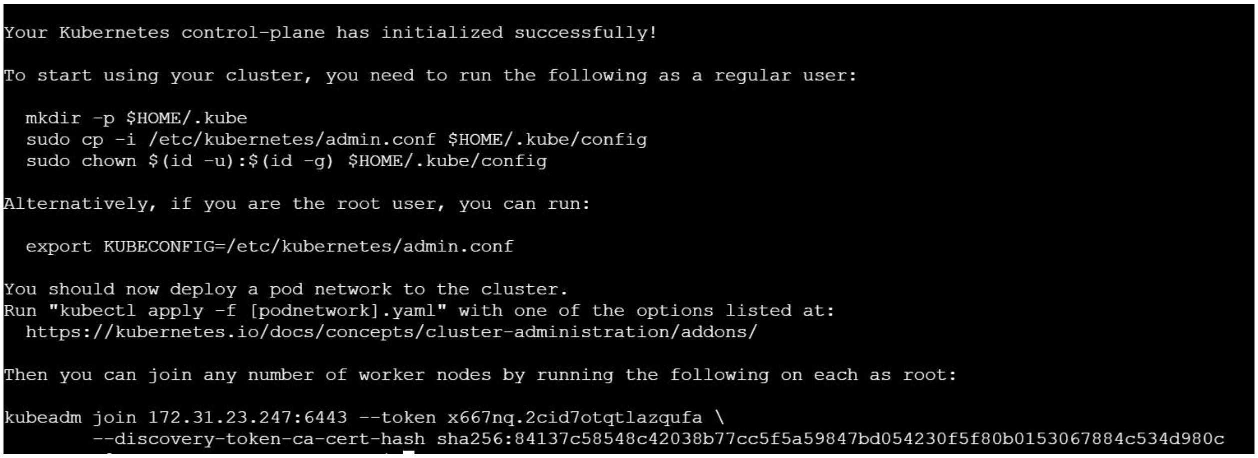
- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p



```
[ec2-user@ip-172-31-23-247 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
```

8. Initialize the kubecluster

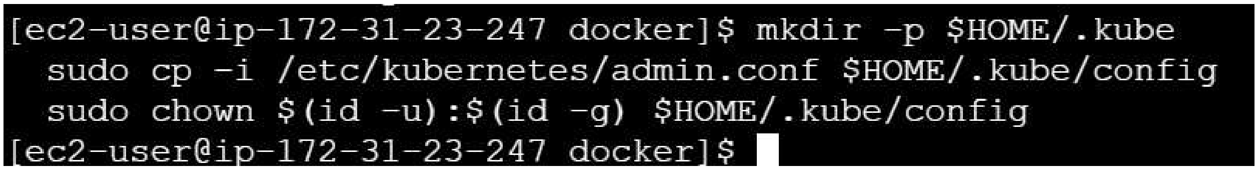
   sudo kubeadm init --pod-network-cidr=10.244.0.0/16



```
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 172.31.23.247:6443 --token x667nq.2cid7otqtlazqufa \
        --discovery-token-ca-cert-hash sha256:84137c58548c42038b77cc5f5a59847bd054230f5f80b0153067884c534d980c
```

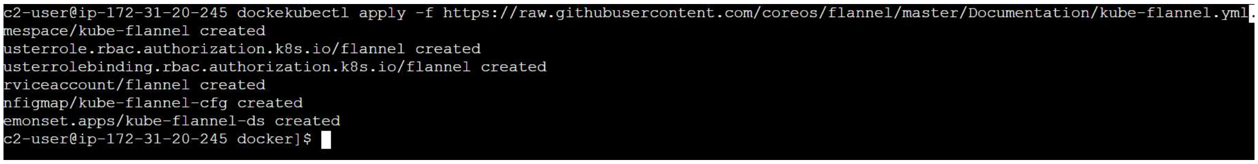
Save the join command in notepad as it will be used later.

Run the 3 commands starting from mkdir given above.



```
[ec2-user@ip-172-31-23-247 docker]$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-23-247 docker]$
```

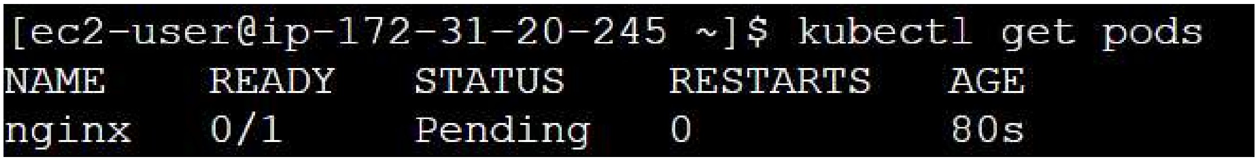
Add a common network plugin called Flannel as mentioned in the code below:
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



```
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]$
```

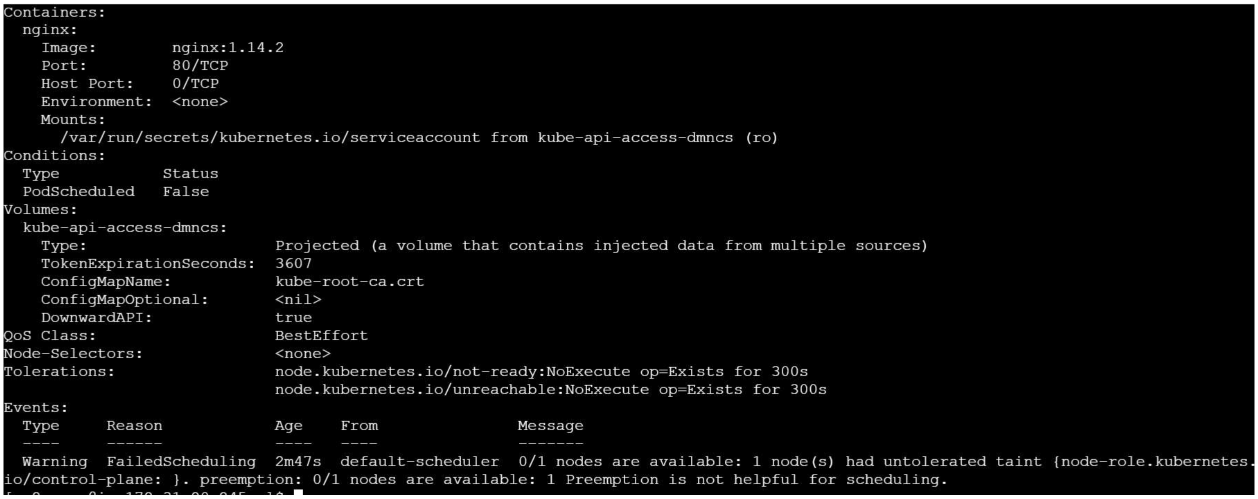
Cluster is up and running

9. Deploy nginx server on this cluster using the command
   kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
   Also run kubectl get pods to check creation of pod

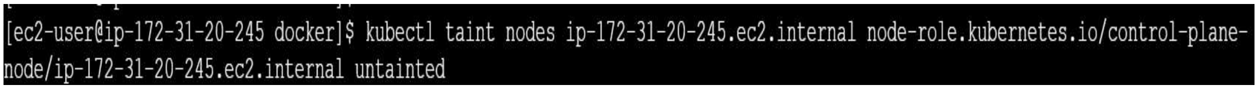


```
[ec2-user@ip-172-31-20-245 ~]$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    0/1     Pending   0          80s
```

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.

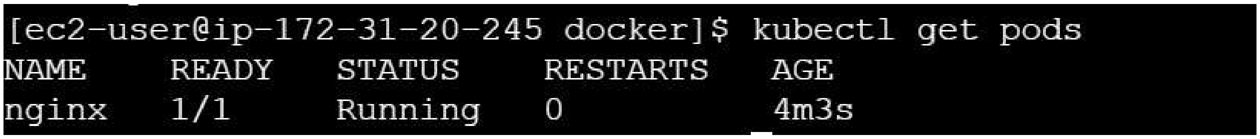


```
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dmncs (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-dmncs:
    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.
```



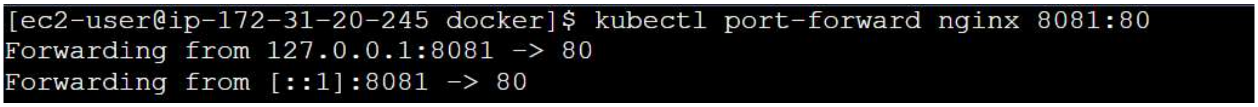
```
[ec2-user@ip-172-31-20-245 docker]$ kubectl taint nodes ip-172-31-20-245.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-20-245.ec2.internal untainted
```

10. Check the status of pod



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

11. Lastly, mention the port you want to host. Here I have used localhost 8081 then check it. kubectl port-forward nginx 8081:80



```
[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:

In this experiment, we successfully configured a Kubernetes environment on an Amazon Linux EC2 instance. We installed Docker and adjusted its settings to use `systemd` for cgroup management. We then set up Kubernetes by disabling SELinux, configuring the Kubernetes repository, and installing necessary components. After initializing the cluster and applying the Flannel network plugin, we deployed an Nginx server. We also addressed issues related to pod scheduling and port forwarding, ensuring the Nginx pod was accessible via port 8081 on local machine.