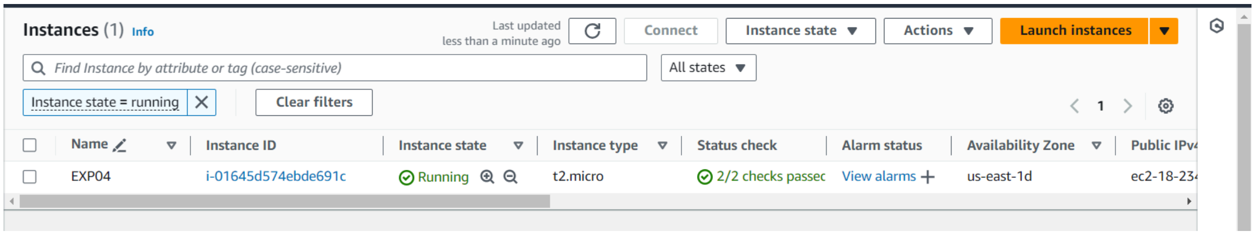
Name-Ishan Kiran Joshi Div-D15C Roll No-21

Experiment 4

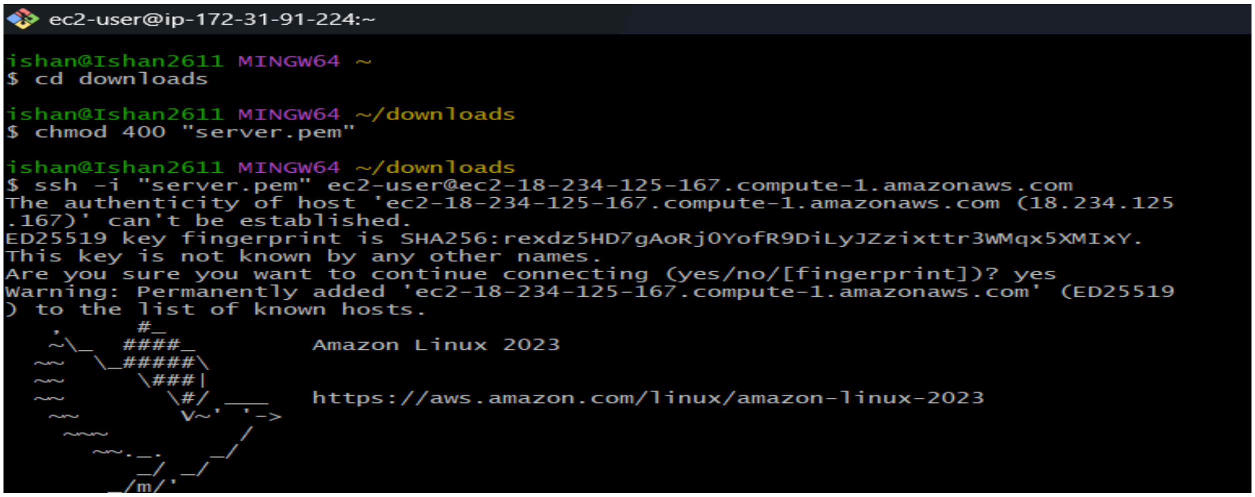
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

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



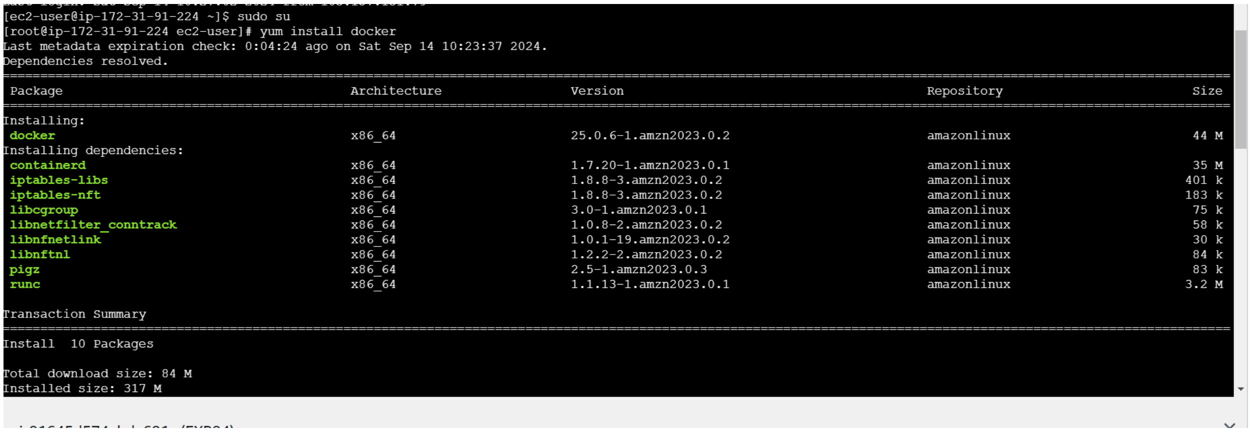
□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
□	EXP04	i-01645d574ebde691c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1d	ec2-18-234

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

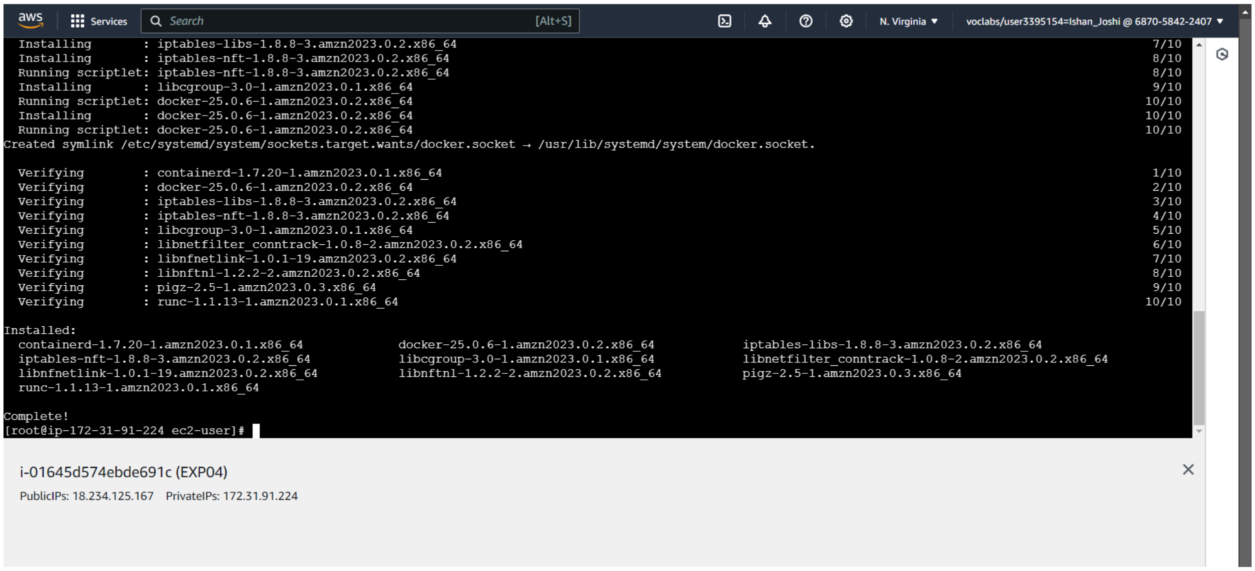


```
ec2-user@ip-172-31-91-224:~

ishan@Ishan2611 MINGW64 ~
$ cd downloads

ishan@Ishan2611 MINGW64 ~/downloads
$ chmod 400 "server.pem"

ishan@Ishan2611 MINGW64 ~/downloads
$ ssh -i "server.pem" ec2-user@ec2-18-234-125-167.compute-1.amazonaws.com
The authenticity of host 'ec2-18-234-125-167.compute-1.amazonaws.com (18.234.125
.167)' can't be established.
ED25519 key fingerprint is SHA256:rexdz5HD7gAoRj0YofR9DiLyJZzixttr3WMqx5XMIxY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-234-125-167.compute-1.amazonaws.com' (ED25519
) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

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



```
[ec2-user@ip-172-31-91-224 ~]$ sudo su
[root@ip-172-31-91-224 ec2-user]# yum install docker
Last metadata expiration check: 0:04:24 ago on Sat Sep 14 10:23:37 2024.
Dependencies resolved.
================================================================================================================================
 Package                          Architecture           Version                                Repository                 Size
================================================================================================================================
Installing:
 docker                           x86_64                 25.0.6-1.amzn2023.0.2                  amazonlinux                44 M
Installing dependencies:
 containerd                       x86_64                 1.7.20-1.amzn2023.0.1                  amazonlinux                35 M
 iptables-libs                    x86_64                 1.8.8-3.amzn2023.0.2                   amazonlinux               401 k
 iptables-nft                     x86_64                 1.8.8-3.amzn2023.0.2                   amazonlinux               183 k
 libcgroup                        x86_64                 3.0-1.amzn2023.0.1                     amazonlinux                75 k
 libnetfilter_conntrack           x86_64                 1.0.8-2.amzn2023.0.2                   amazonlinux                58 k
 libnfnetlink                     x86_64                 1.0.1-19.amzn2023.0.2                  amazonlinux                30 k
 libnftnl                         x86_64                 1.2.2-2.amzn2023.0.2                   amazonlinux                84 k
 pigz                             x86_64                 2.5-1.amzn2023.0.3                     amazonlinux                83 k
 runc                             x86_64                 1.1.13-1.amzn2023.0.1                  amazonlinux               3.2 M

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

Total download size: 84 M
Installed size: 317 M
```



```
  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!
[root@ip-172-31-91-224 ec2-user]#
```

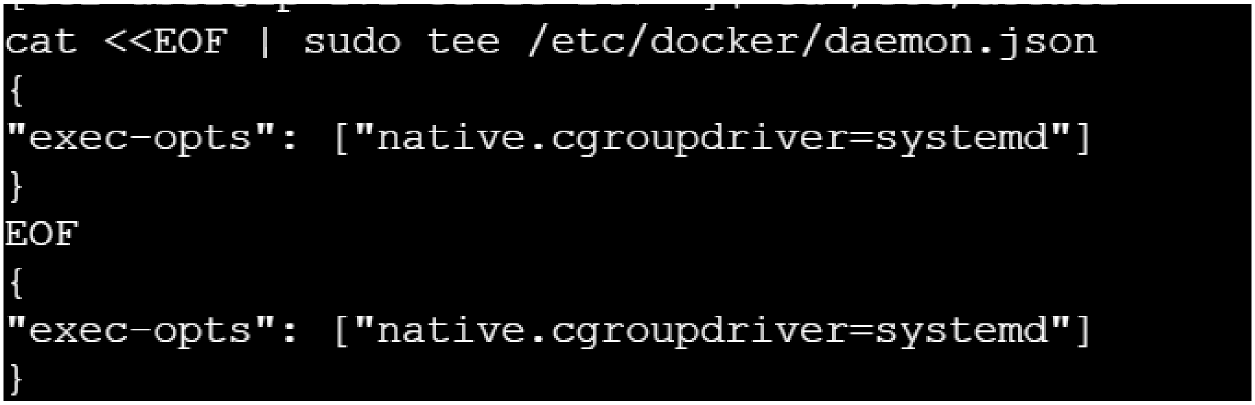
i-01645d574ebde691c (EXP04)

PublicIPs: 18.234.125.167   PrivateIPs: 172.31.91.224

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"
}
```

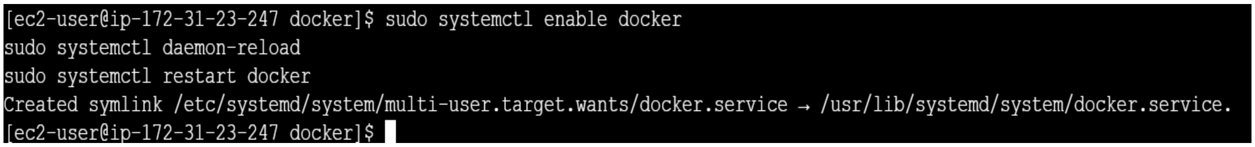
EOF



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

   II. Add kubernetes repository
      cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
      [kubernetes]
      name=Kubernetes

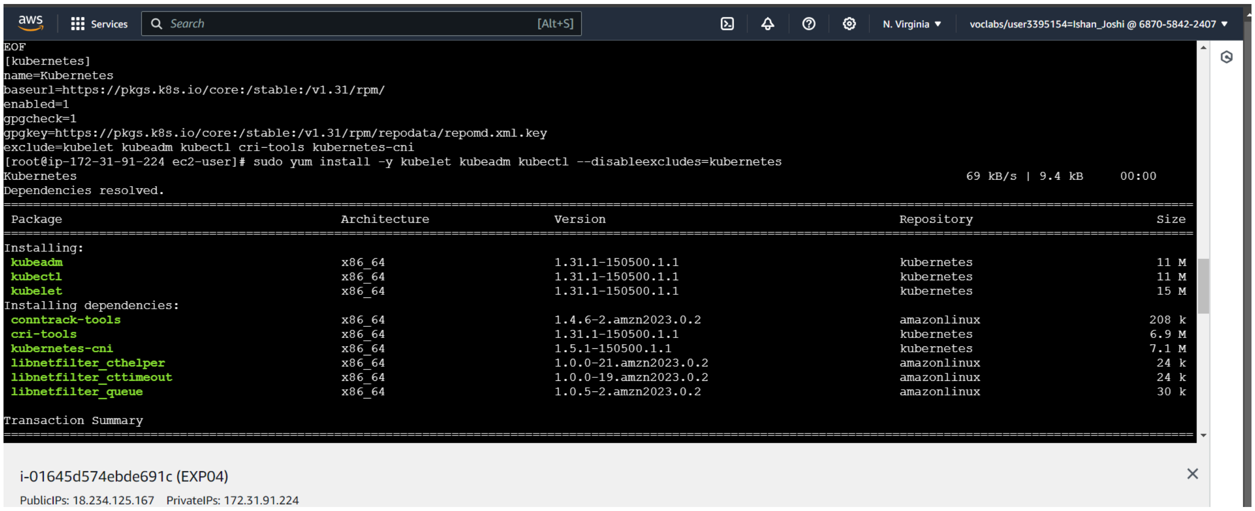
```
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

III. Run the commands to update and install kubernetes packages

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

IV. Configure internet options to allow bridging

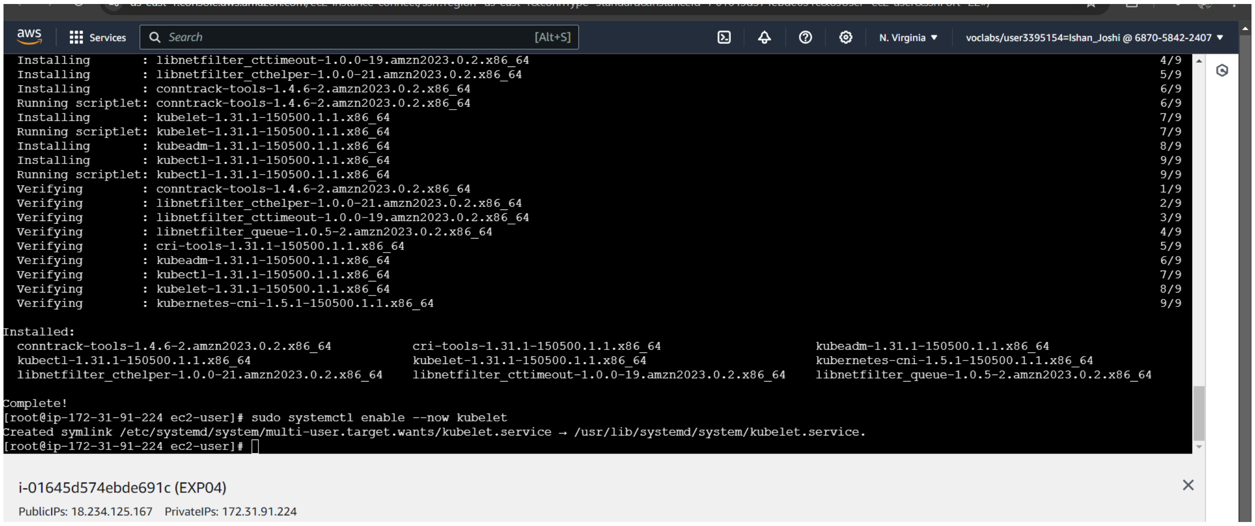
- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p



```
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
[root@ip-172-31-91-224 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                                   69 kB/s | 9.4 kB     00:00
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
===========================================================================================================================
```

i-01645d574ebde691c (EXP04)

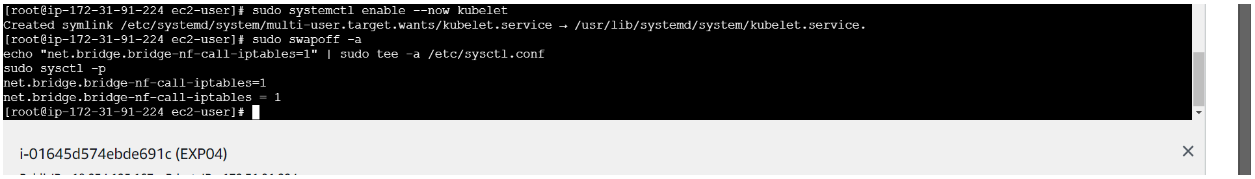
PublicIPs: 18.234.125.167 PrivateIPs: 172.31.91.224



```
  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!
[root@ip-172-31-91-224 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-91-224 ec2-user]#
```

i-01645d574ebde691c (EXP04)

PublicIPs: 18.234.125.167   PrivateIPs: 172.31.91.224

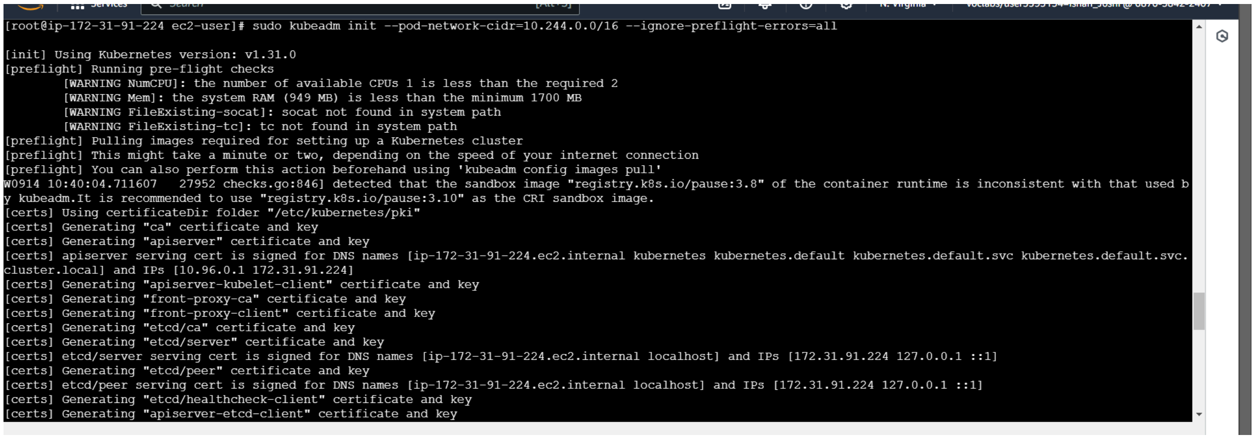


```
[root@ip-172-31-91-224 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-91-224 ec2-user]# sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-91-224 ec2-user]#
```

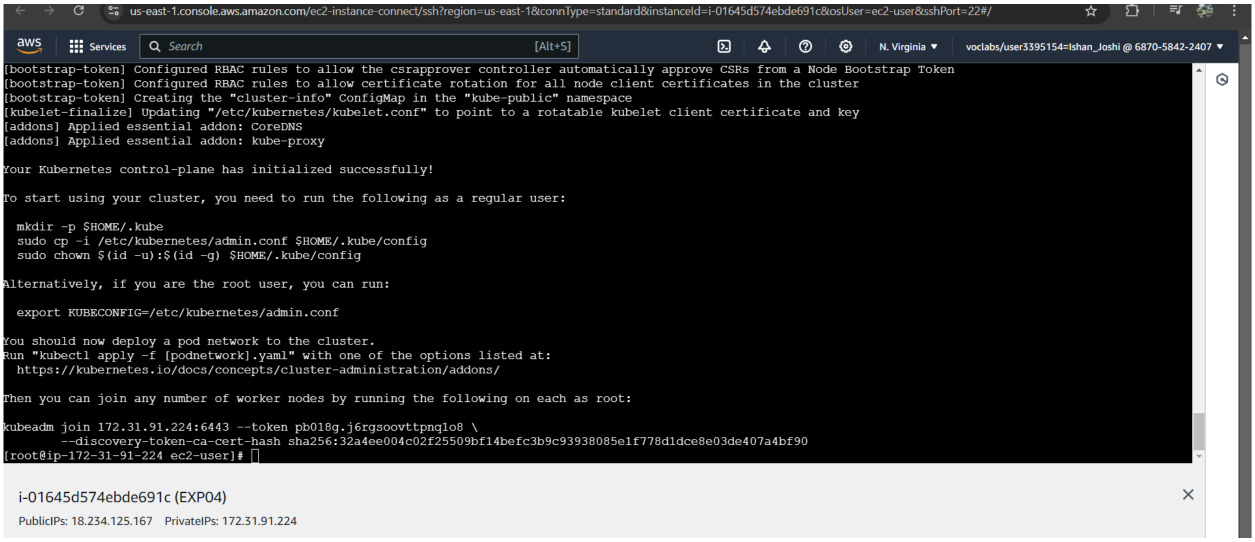
i-01645d574ebde691c (EXP04)

8. Initialize the kubecluster

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



```
[root@ip-172-31-91-224 ec2-user]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[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 10:40:04.711607   27952 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-172-31-91-224.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.
cluster.local] and IPs [10.96.0.1 172.31.91.224]
[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-172-31-91-224.ec2.internal localhost] and IPs [172.31.91.224 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-91-224.ec2.internal localhost] and IPs [172.31.91.224 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
```

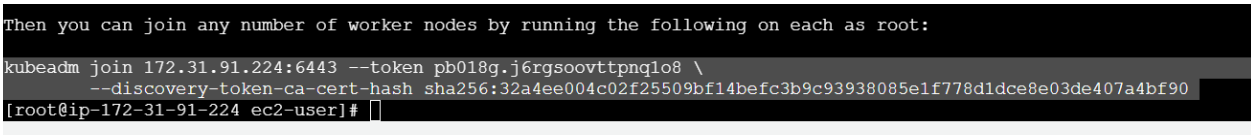

us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-01645d574ebde691c&osUser=ec2-user&sshPort=22#/
aws
Services
Search
[Alt+S]
N. Virginia ▼
voclabs/user3395154=Ishan_Joshi @ 6870-5842-2407 ▼

```
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[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 172.31.91.224:6443 --token pb018g.j6rgsoovttpnq1o8 \
        --discovery-token-ca-cert-hash sha256:32a4ee004c02f25509bf14befc3b9c93938085e1f778d1dce8e03de407a4bf90
[root@ip-172-31-91-224 ec2-user]# 
```

i-01645d574ebde691c (EXP04)

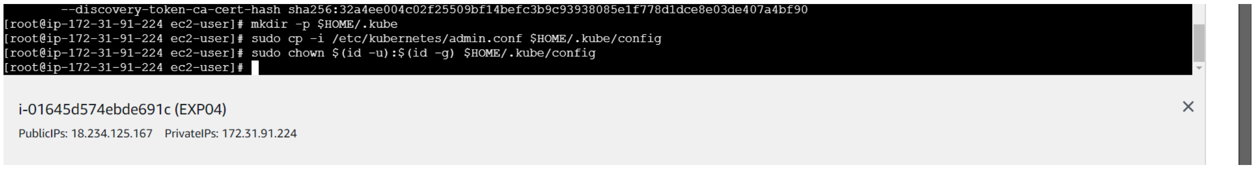
PublicIPs: 18.234.125.167 PrivateIPs: 172.31.91.224

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



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

kubeadm join 172.31.91.224:6443 --token pb018g.j6rgsoovttpnq1o8 \
        --discovery-token-ca-cert-hash sha256:32a4ee004c02f25509bf14befc3b9c93938085e1f778d1dce8e03de407a4bf90
[root@ip-172-31-91-224 ec2-user]# 
```

Run the 3 commands starting from mkdir given above.



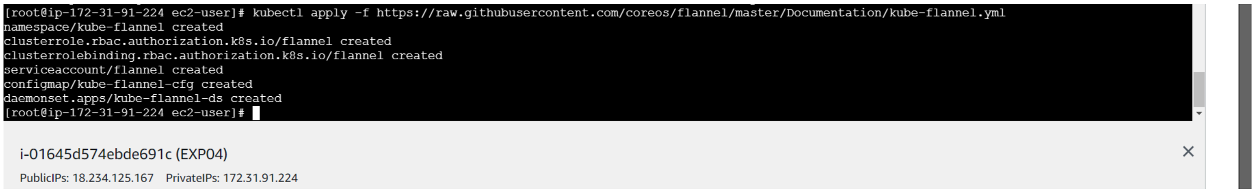
```
        --discovery-token-ca-cert-hash sha256:32a4ee004c02f25509bf14befc3b9c93938085e1f778d1dce8e03de407a4bf90
[root@ip-172-31-91-224 ec2-user]# mkdir -p $HOME/.kube
[root@ip-172-31-91-224 ec2-user]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@ip-172-31-91-224 ec2-user]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@ip-172-31-91-224 ec2-user]# 
```

i-01645d574ebde691c (EXP04)

PublicIPs: 18.234.125.167 PrivateIPs: 172.31.91.224

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



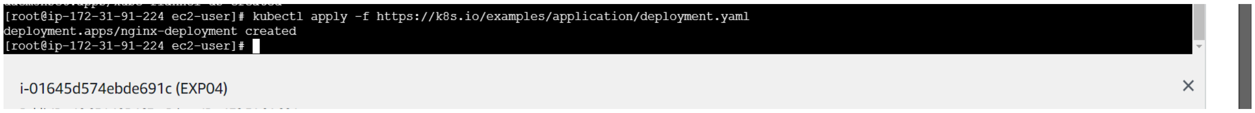
```
[root@ip-172-31-91-224 ec2-user]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[root@ip-172-31-91-224 ec2-user]# 
```

i-01645d574ebde691c (EXP04)

PublicIPs: 18.234.125.167 PrivateIPs: 172.31.91.224

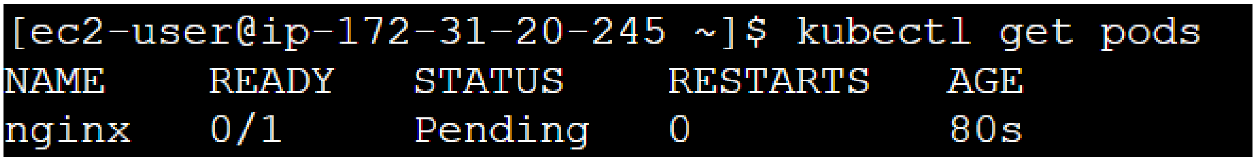
Cluster is up and running

9. Deploy nginx server on this cluster using the command
   kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml



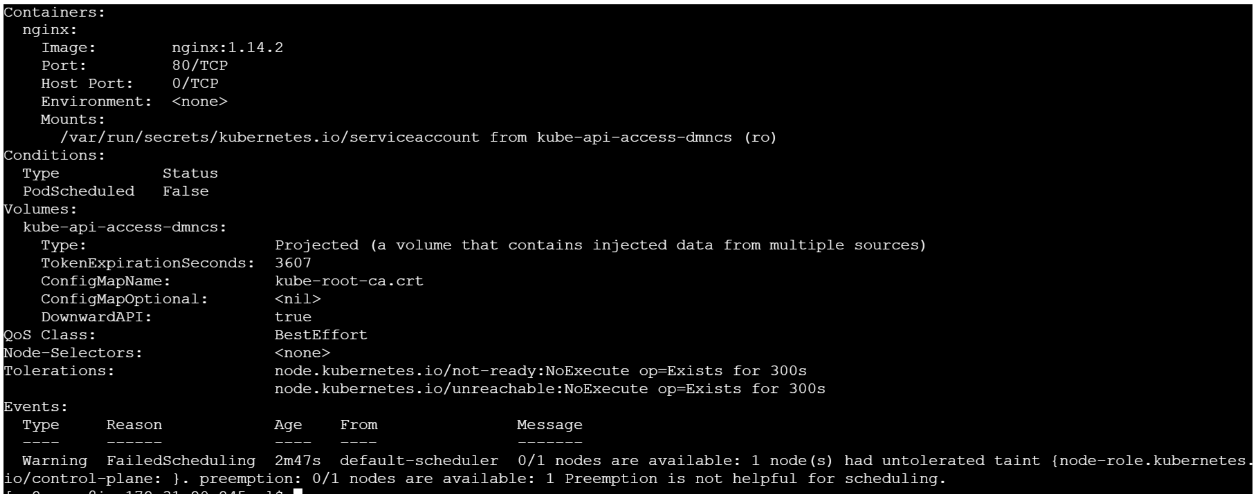
```
[root@ip-172-31-91-224 ec2-user]# kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[root@ip-172-31-91-224 ec2-user]#
```

   Also run kubectl get pods to check creation of pod

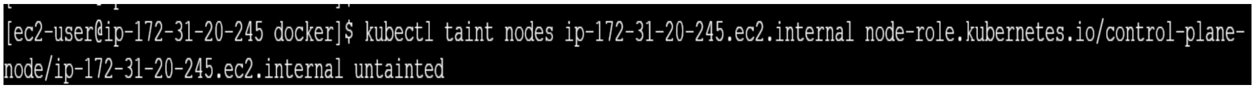


```
[ec2-user@ip-172-31-20-245 ~]$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    0/1     Pending   0          80s
```

To change the state from pending to running, use thecommand
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.

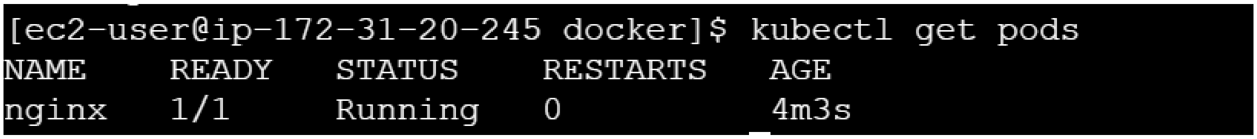


```
Containers:
  nginx:
    Image:        nginx:1.14.2
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dmncs (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-dmncs: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



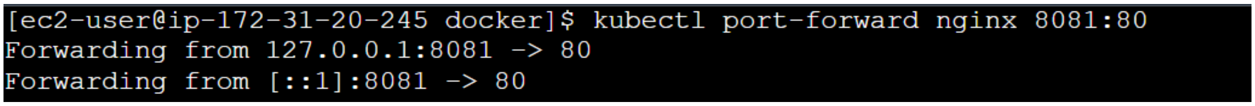
```
[ec2-user@ip-172-31-20-245 docker]$ kubectl taint nodes ip-172-31-20-245.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-20-245.ec2.internal untainted
```

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



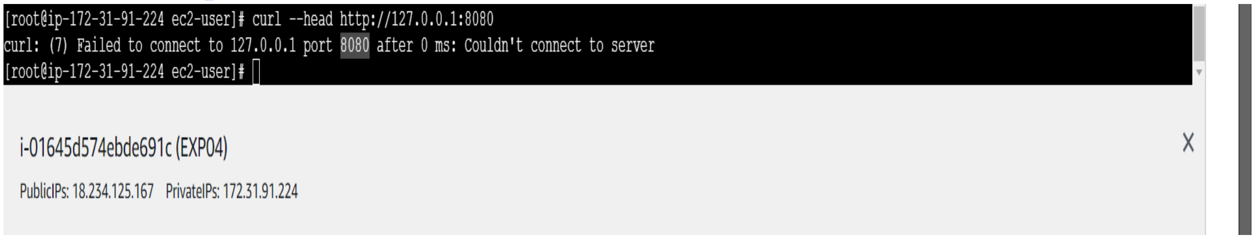
```
[ec2-user@ip-172-31-20-245 docker]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

12.. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

curl --head http://127.0.0.1:8080



```
[root@ip-172-31-91-224 ec2-user]# curl --head http://127.0.0.1:8080
curl: (7) Failed to connect to 127.0.0.1 port 8080 after 0 ms: Couldn't connect to server
[root@ip-172-31-91-224 ec2-user]# 
```

i-01645d574ebde691c (EXP04)

PublicIPs: 18.234.125.167   PrivateIPs: 172.31.91.224

Due to issues in the server it shows error

Conclusion:

In this experiment, we successfully set up a Kubernetes environment on an Amazon Linux EC2 instance. Docker was installed and configured to use systemd for cgroup management. Kubernetes was then installed by disabling SELinux, configuring the repository, and installing the required components. After initializing the cluster and deploying the Flannel network plugin, we launched an Nginx server. Additionally, we resolved issues with pod scheduling and port forwarding, ensuring the Nginx pod could be accessed through port 8081 on the local machine.In the last we could not verify as it was showing server errors