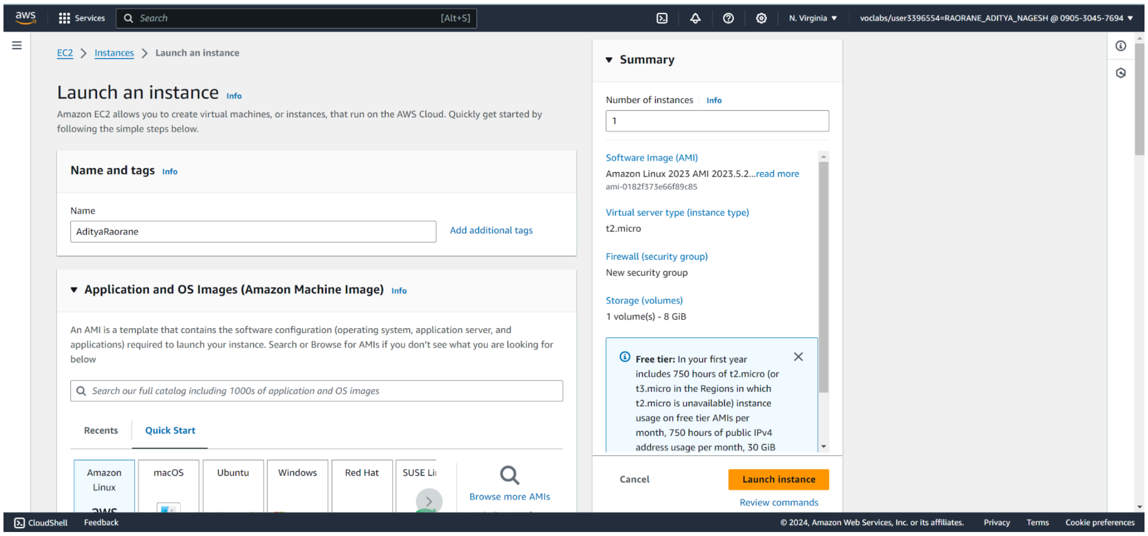
Experiment 4

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

1. **A] Creation Of EC-2 instance →**

   Launch an AWS EC2 instance named AdityaRaorane using an AWS Linux AMI. Configure the Security Group's Inbound Rules to allow SSH access, then choose the t2.micro instance type.



Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: AdityaRaorane

Add additional tags

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Summary

Number of instances: 1

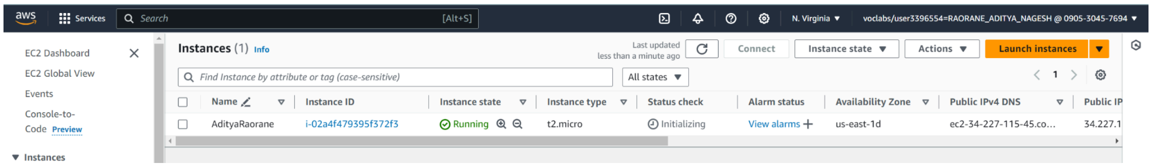
Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more ami-0182f373e66f89c85

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB



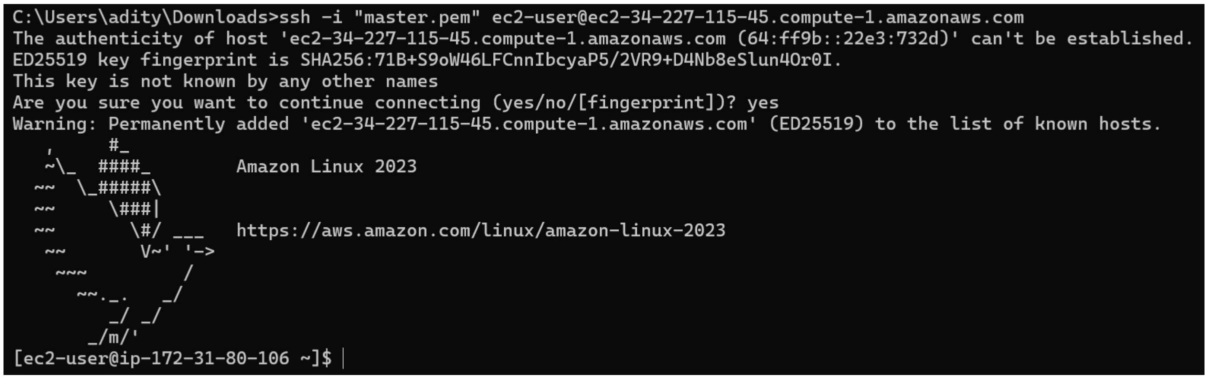
Instances (1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
AdityaRaorane	i-02a4f479395f372f3	Running	t2.micro	Initializing	View alarms +	us-east-1d	ec2-34-227-115-45.co...	34.227.1

B] Connecting to an AWS EC2 Instance via SSH →

To connect to an AWS EC2 instance via SSH, change the key file's permission using chmod 400 "keyname.pem", then run ssh -i <keyname>.pem ubuntu@<public_ip_address> to establish the connection.

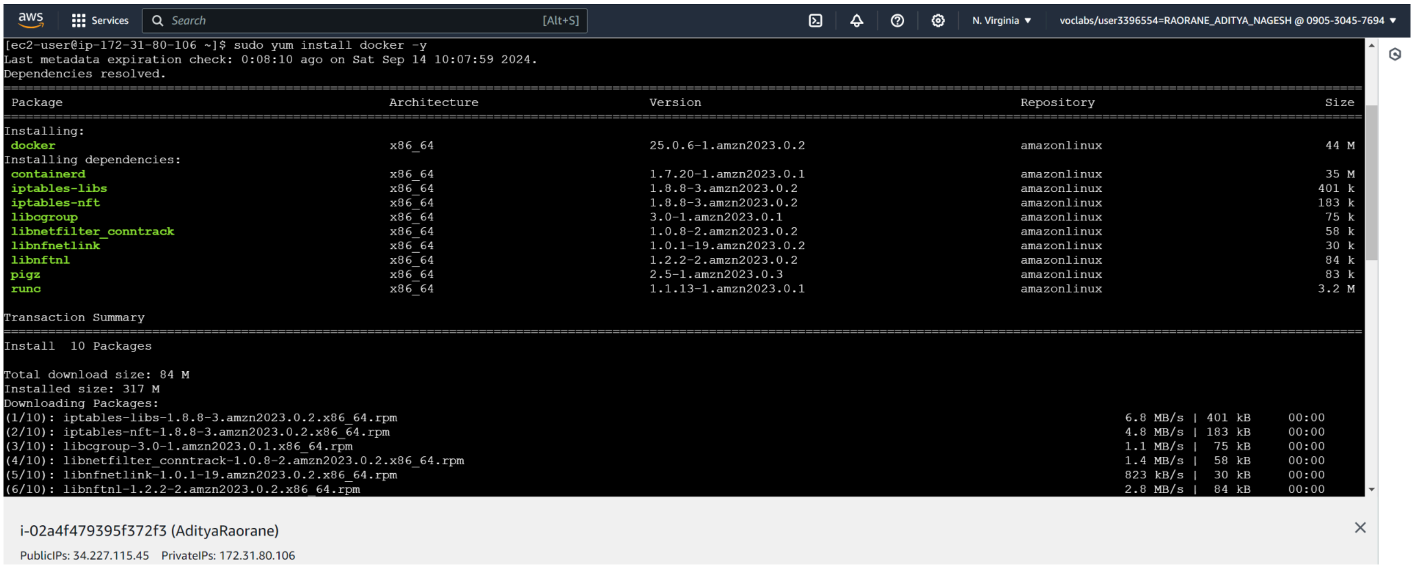
ssh -i "master.pem" ec2-user@ec2-34-227-115-45.compute-1.amazonaws.com



```
C:\Users\adity\Downloads>ssh -i "master.pem" ec2-user@ec2-34-227-115-45.compute-1.amazonaws.com
The authenticity of host 'ec2-34-227-115-45.compute-1.amazonaws.com (64:ff9b::22e3:732d)' can't be established.
ED25519 key fingerprint is SHA256:71B+S9oW46LFCnnIbcyaP5/2VR9+D4Nb8eSlun4Or0I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-227-115-45.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-80-106 ~]$
```

2. Installation of Docker →

Run the following command: sudo yum install docker -y



```
[ec2-user@ip-172-31-80-106 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:08:10 ago on Sat Sep 14 10:07:59 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
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                                    6.8 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                                     4.8 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                                          1.1 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm                           1.4 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm                                    823 kB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                                         2.8 MB/s |  84 kB     00:00
```

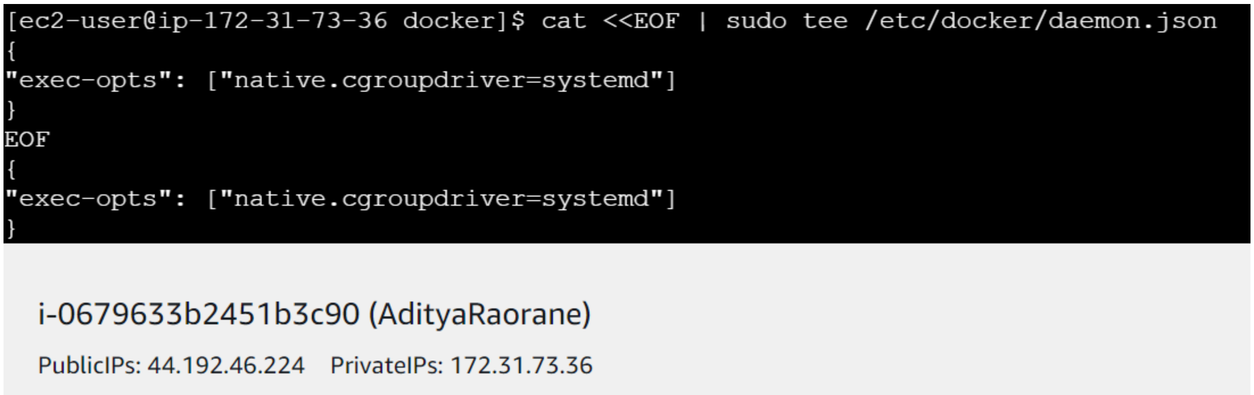
i-02a4f479395f372f3 (AdityaRaorane)

PublicIPs: 34.227.115.45 PrivateIPs: 172.31.80.106

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
```



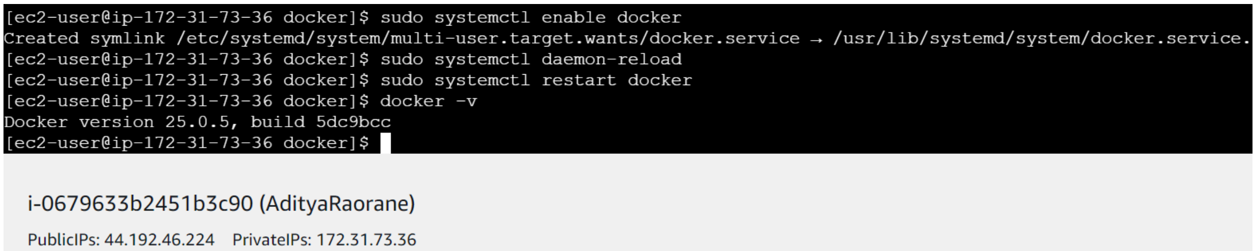
```
[ec2-user@ip-172-31-73-36 docker]$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

After this run the following command to enable and start docker and also to load the daemon.json file.

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



```
[ec2-user@ip-172-31-73-36 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-73-36 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-73-36 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-73-36 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

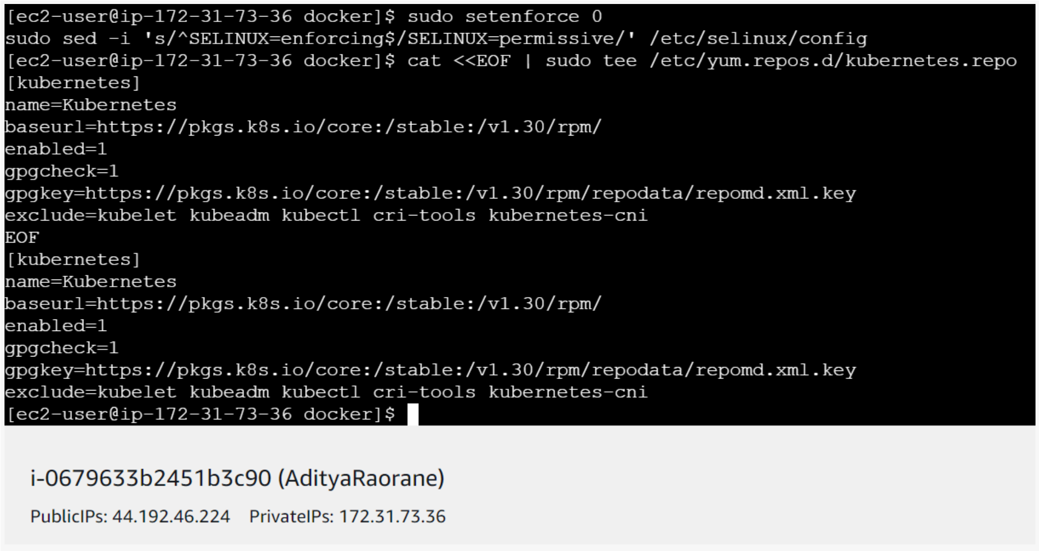
3. Install Kubernetes →

a]SELinux needs to be disabled before configuring kubelet Run the following command

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

b] We are adding kubernetes using the repository whose command is given below.

```
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.x
ml.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```



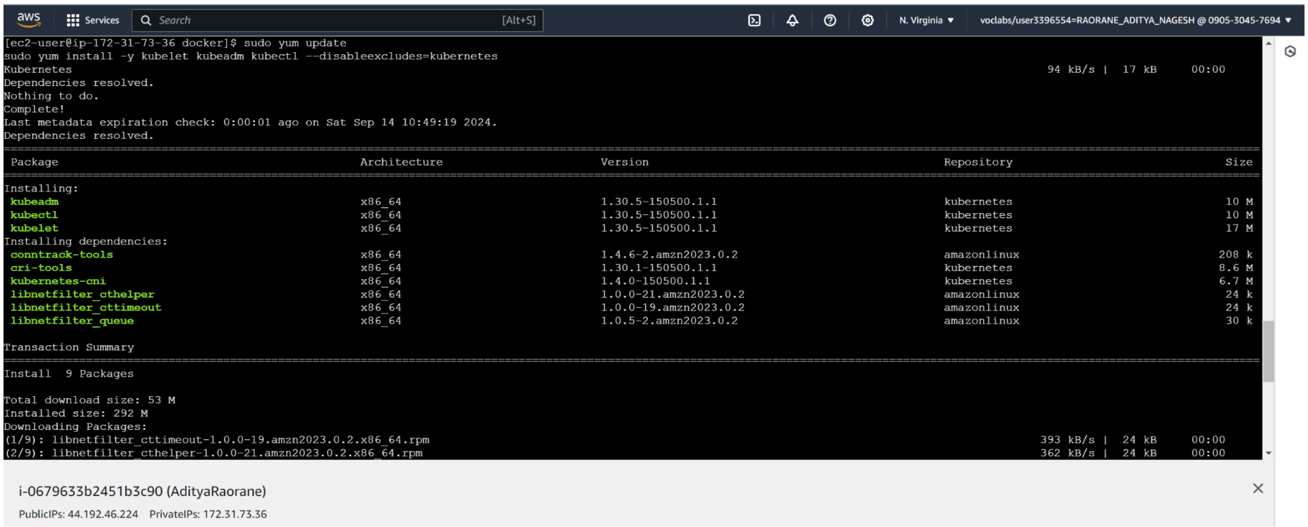
```
[ec2-user@ip-172-31-73-36 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-73-36 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
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-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

c] After that Run following command to make the updation and also to install kubelet, kubeadm, kubectl:

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


aws
Services
Search
[Alt+S]
N. Virginia ▾
voclabs/user3396554=RAORANE_ADITYA_NAGESH @ 0905-3045-7694 ▾

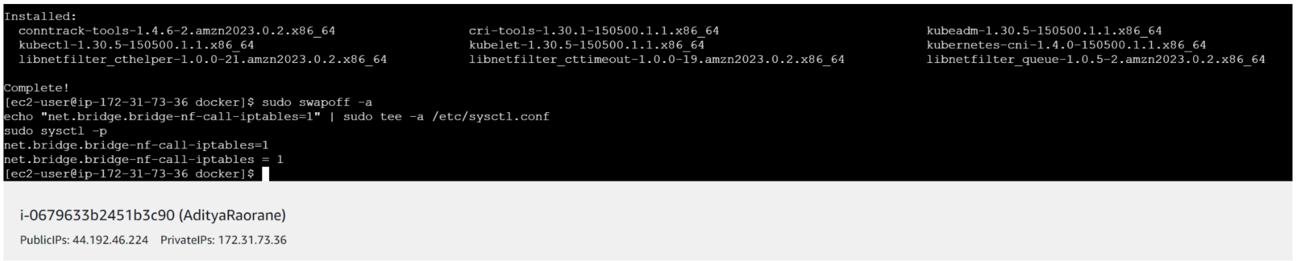
```
[ec2-user@ip-172-31-73-36 docker]$ sudo yum update
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes                                                                                      94 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:01 ago on Sat Sep 14 10:49:19 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

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

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm                                 393 kB/s |  24 kB     00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm                                  362 kB/s |  24 kB     00:00
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

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

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



```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64               cri-tools-1.30.1-150500.1.1.x86_64                  kubeadm-1.30.5-150500.1.1.x86_64
  kubectl-1.30.5-150500.1.1.x86_64                          kubelet-1.30.5-150500.1.1.x86_64                    kubernetes-cni-1.4.0-150500.1.1.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-172-31-73-36 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-172-31-73-36 docker]$
```

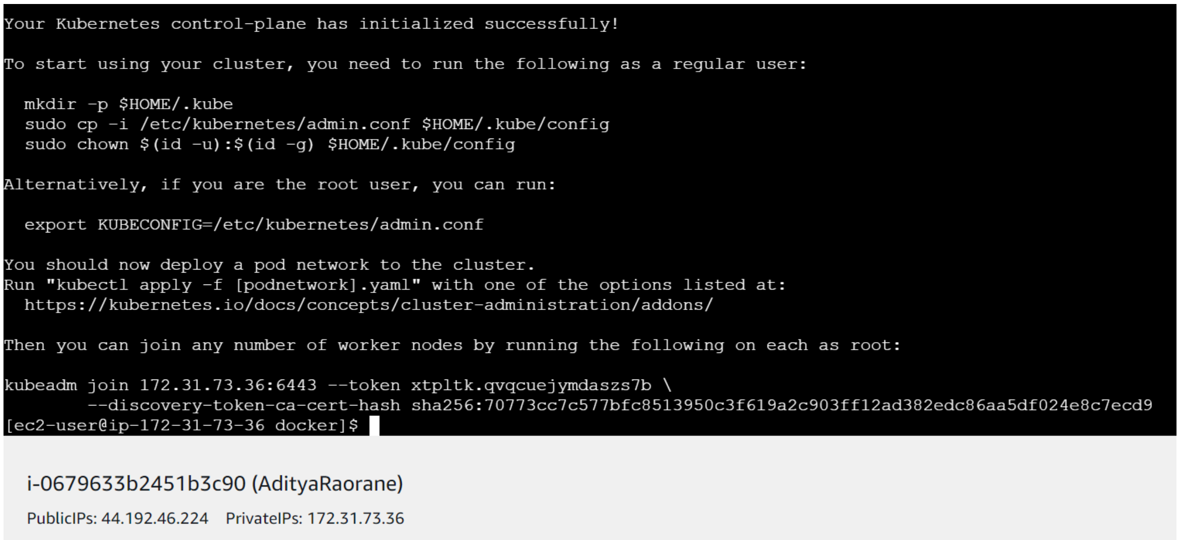
i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

4. Initialize the Kubecluster →

a] Initializes a Kubernetes cluster with `kubeadm`, sets up the pod network CIDR to `10.244.0.0/16` for network communication, and ignores preflight checks for CPU and memory requirements.

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=NumCPU,Mem
```



```
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 172.31.73.36:6443 --token xtpltk.qvqcuejymdaszs7b \
        --discovery-token-ca-cert-hash sha256:70773cc7c577bfc8513950c3f619a2c903ff12ad382edc86aa5df024e8c7ecd9
[ec2-user@ip-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

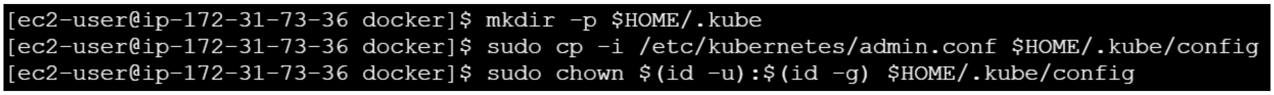
PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

b] copy the token and save for future use .

```
kubeadm join 172.31.73.36:6443 --token xtpltk.qvqcuejymdaszs7b \
        --discovery-token-ca-cert-hash
sha256:70773cc7c577bfc8513950c3f619a2c903ff12ad382edc86aa5df024e8
c7ecd9
```

c] Copy the mkdir and chown commands from the top and execute them

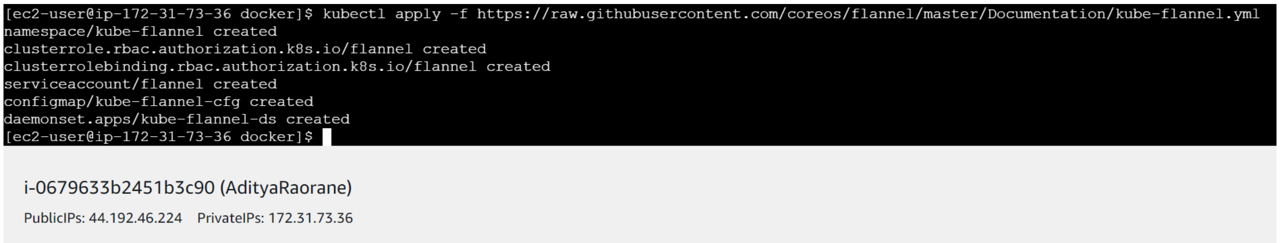
- mkdir -p $HOME/.kube
- sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
- sudo chown $(id -u):$(id -g) $HOME/.kube/config



```
[ec2-user@ip-172-31-73-36 docker]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-73-36 docker]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[ec2-user@ip-172-31-73-36 docker]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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

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



```
[ec2-user@ip-172-31-73-36 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-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

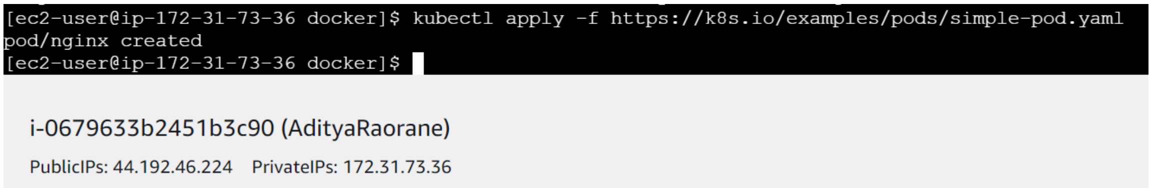
PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

5. Deploying an NGINX Server on Your Kubernetes Cluster →

a]Now that the cluster is up and running,we can deploy our nginx server on this cluster. Apply deployment using this following command:

`kubectl apply -f`

https://k8s.io/examples/pods/simple-pod.yaml

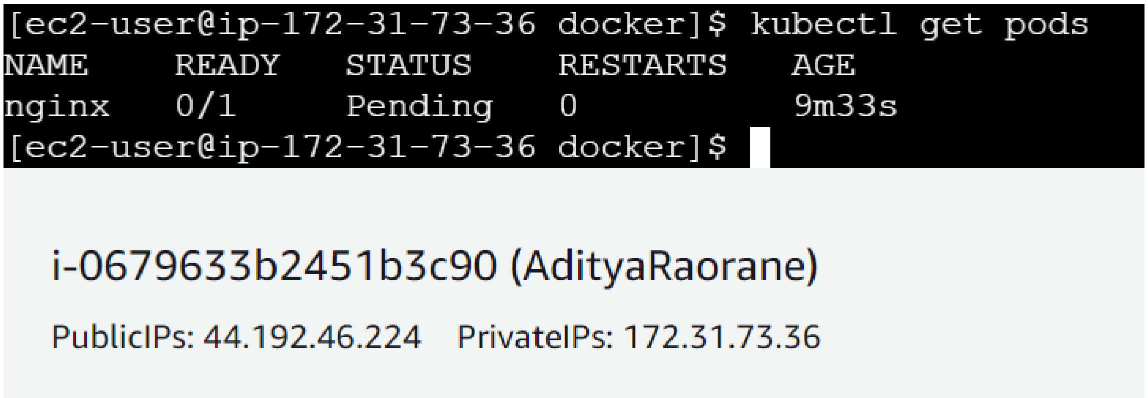


```
[ec2-user@ip-172-31-73-36 docker]$ kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
[ec2-user@ip-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

b]Then use **kubectl get pods** to check whether the pod gets created or not.



```
[ec2-user@ip-172-31-73-36 docker]$ kubectl get pods
NAME     READY    STATUS     RESTARTS    AGE
nginx    0/1      Pending    0           9m33s
[ec2-user@ip-172-31-73-36 docker]$
```

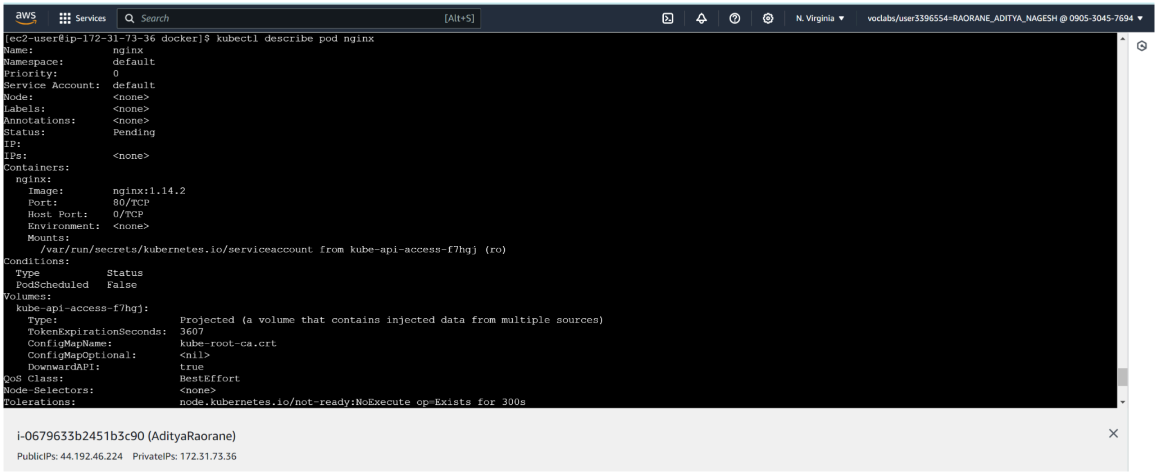
i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

c]To convert state from pending to running use 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.

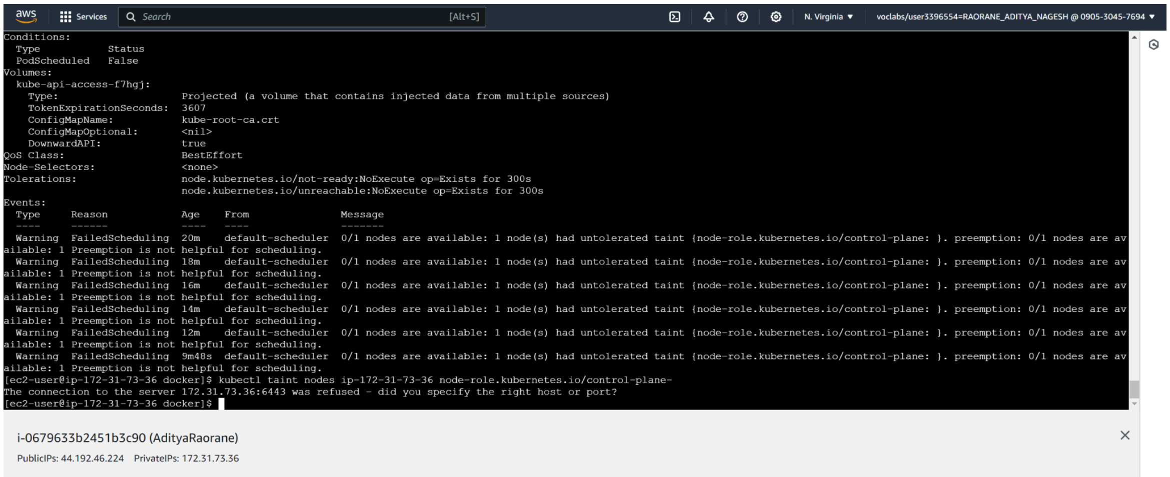


aws
Services
Search [Alt+S]
N. Virginia ▾
voclabs/user3396554=RAORANE_ADITYA_NAGESH @ 0905-3045-7694 ▾

```
[ec2-user@ip-172-31-73-36 docker]$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
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-f7hgj (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-f7hgj:
    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
```

i-0679633b2451b3c90 (AdityaRaorane)

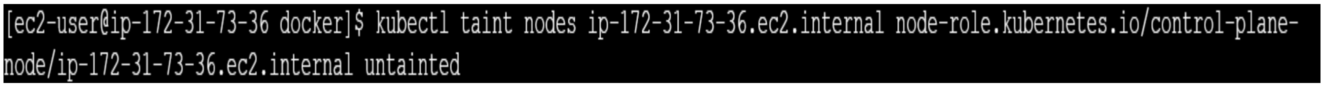
PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36



```
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-f7hgj:
    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  20m    default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  18m    default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  16m    default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  14m    default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  12m    default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
  Warning  FailedScheduling  9m48s  default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }. preemption: 0/1 nodes are av
ailable: 1 Preemption is not helpful for scheduling.
[ec2-user@ip-172-31-73-36 docker]$ kubectl taint nodes ip-172-31-73-36 node-role.kubernetes.io/control-plane-
The connection to the server 172.31.73.36:6443 was refused - did you specify the right host or port?
[ec2-user@ip-172-31-73-36 docker]$
```

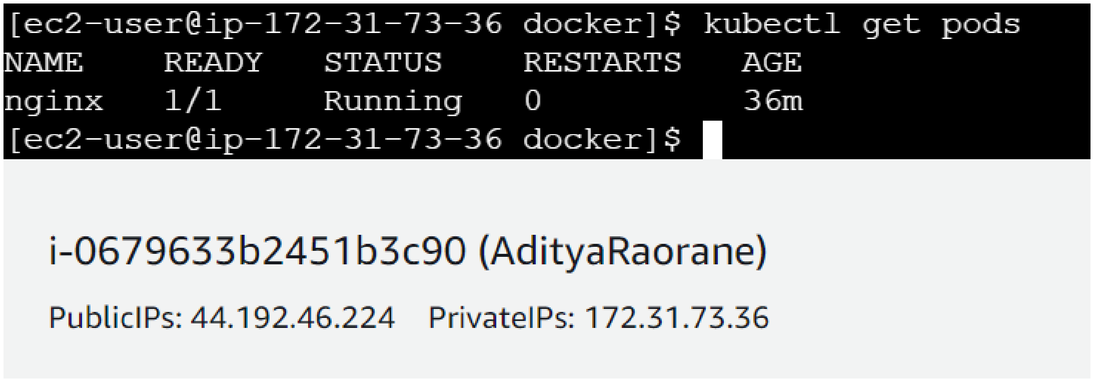
i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36



```
[ec2-user@ip-172-31-73-36 docker]$ kubectl taint nodes ip-172-31-73-36.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-73-36.ec2.internal untainted
```

6. Now check pod status is is running

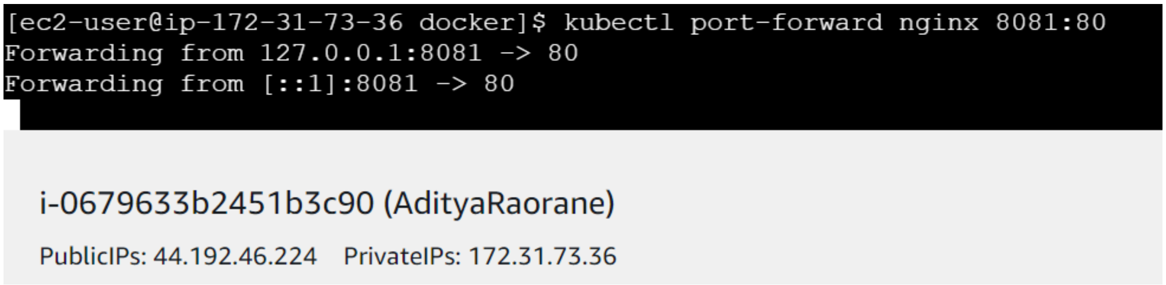


```
[ec2-user@ip-172-31-73-36 docker]$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          36m
[ec2-user@ip-172-31-73-36 docker]$
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

7. 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-172-31-73-36 docker]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

i-0679633b2451b3c90 (AdityaRaorane)

PublicIPs: 44.192.46.224 PrivateIPs: 172.31.73.36

8. **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
```

If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

Conclusion:

Thus we established a Kubernetes cluster on AWS EC2, configured Docker with cgroup settings, and set up Kubernetes components. We applied Flannel for networking and deployed an NGINX server. Next, we exposed the NGINX server using a Kubernetes service to allow external access. Finally, we implemented autoscaling based on CPU usage to dynamically manage traffic load on the NGINX pods.