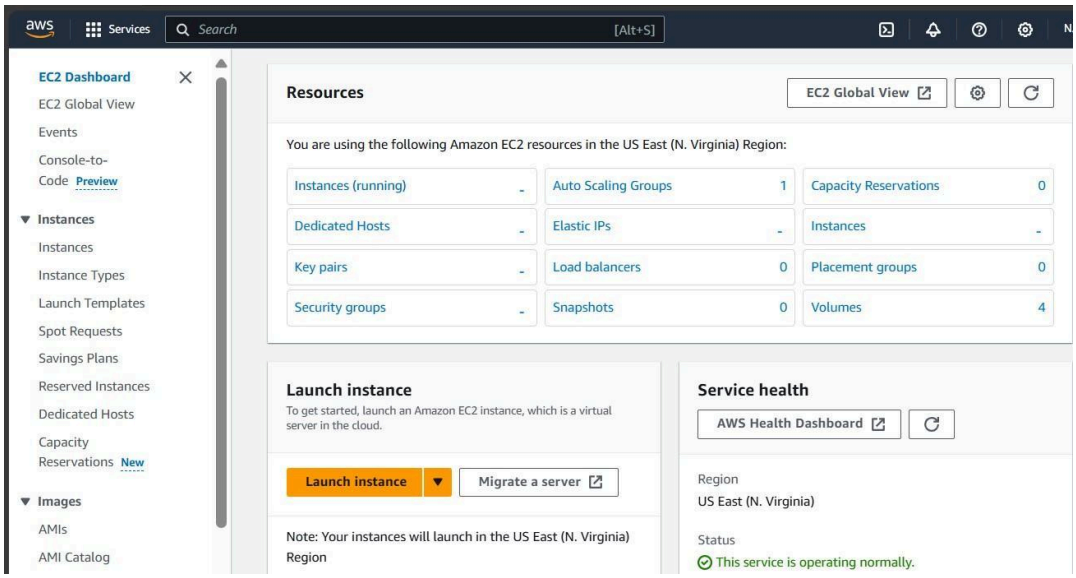
EXPERIMENT NO. 3

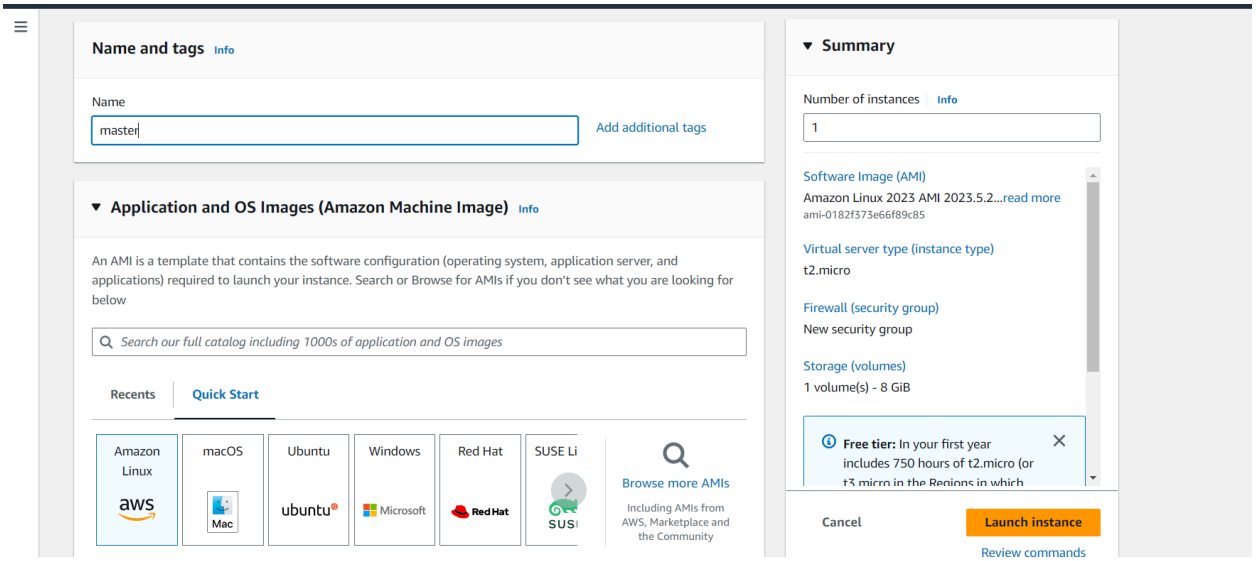
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud.

Procedure:

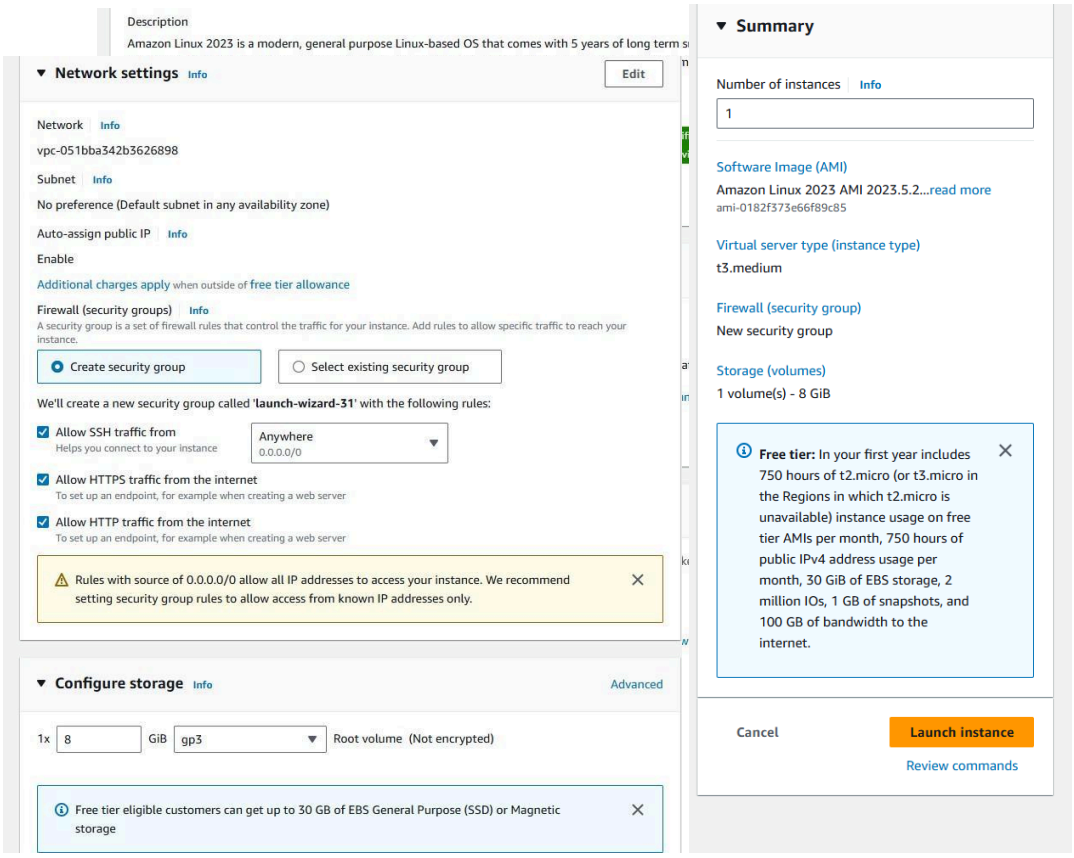
1. **Creation Of Instance**

- 

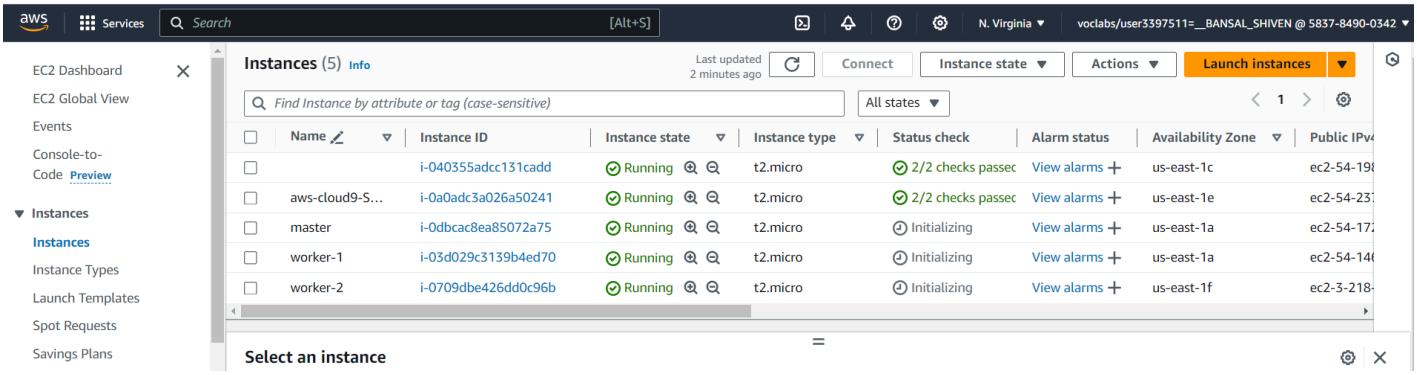
Search EC-2 instance. Then create three EC-2 instances and choose Amazon Linux as OS and also allow ssh traffic from anywhere.



NAME: SHIVEN BANSAL **ROLL NO : 03** **CLASS: D15C**



To efficiently run kubernetes cluster select instance type of at least t3.medium as kubernetes recommends at least 2 vCPU to run smoothly on it.



- Then for making connection through SSH into all 3 machines each in separate terminal Use this following command:
  ssh -i <keyname>.pem ubuntu@<public_ip_address> where keyname is name of the key you created here i created key server.pem and use public IP address.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

```
ADMIN@DESKTOP-LPV2RP5 MINGW64 ~
$ cd Downloads/

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ chmod 400 "server.pem"

ADMIN@DESKTOP-LPV2RP5 MINGW64 ~/Downloads
$ ssh -i "server.pem" ec2-user@ec2-44-192-73-91.compute-1.amazonaws.com
The authenticity of host 'ec2-44-192-73-91.compute-1.amazonaws.com (44.192.73.91
)' can't be established.
ED25519 key fingerprint is SHA256:xIIIlRCIBX1LUOJsm9WHO/YIrDSioPOJfOPEoRKtYJk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-192-73-91.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-172-31-78-148 ~]$
```

2. Installation Of Docker on three machines

- For installation of Docker into all three machines run the following command:
  sudo yum install docker -y

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:20:45 ago on Sat Sep 14 06:53:06 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     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_ 5.7 MB/s | 401 kB     00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_6 6.2 MB/s | 183 kB     00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm 2.2 MB/s |  75 kB     00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023 1.9 MB/s |  58 kB     00:00
(5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_ 1.2 MB/s |  30 kB     00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rp 2.2 MB/s |  84 kB     00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm      1.7 MB/s |  83 kB     00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm    22 MB/s | 3.2 MB     00:00
(9/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm  32 MB/s |  44 MB     00:01
(10/10): containerd-1.7.20-1.amzn2023.0.1.x86_6  23 MB/s |  35 MB     00:01
--------------------------------------------------------------------------------
Total                                            53 MB/s |  84 MB     00:01
Running transaction check
Transaction check succeeded.
  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-172-31-81-216 ~]$ client_loop: send disconnect: Connection reset by
 peer
```

- 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-172-31-81-216 ~]$ cd /etc/docker
[ec2-user@ip-172-31-81-216 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"
}
[ec2-user@ip-172-31-81-216 docker]$
```

- Then 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

```
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /us
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-81-216 docker]$ sudo systemctl restart docker
```

- Then check the version of docker installed. docker -v

```
[ec2-user@ip-172-31-81-216 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

3. Installation Of Kubernetes on three machines

- SELinux needs to be disable before configuring kubelet thus run the following command
  sudo setenforce 0
  sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-81-216 ~]$ sudo setenforce 0
[ec2-user@ip-172-31-81-216 ~]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permis
sive/' /etc/selinux/config
```

- Here 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.xml.key
  exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
  EOF

```
[ec2-user@ip-172-31-81-216 ~]$ 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-172-31-81-216 ~]$
```

- After that Run following command to make the updation and also to install kubelet ,kubeadm, kubectl: sudo yum update

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum update
Kubernetes                                      114 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
```

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

```
[ec2-user@ip-172-31-81-216 ~]$ sudo yum install -y kubelet kubeadm kubectl --dis
ableexcludes=kubernetes
Last metadata expiration check: 0:00:35 ago on Sat Sep 14 07:38:02 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     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_cthelper-1.0.0-21.amzn2023. 436 kB/s |  24 kB     00:00
(2/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2. 1.5 MB/s |  30 kB     00:00
(3/9): libnetfilter_cttimeout-1.0.0-19.amzn2023 309 kB/s |  24 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2.x86 2.3 MB/s | 208 kB     00:00
(5/9): cri-tools-1.30.1-150500.1.1.x86_64.rpm    34 MB/s | 8.6 MB     00:00
(6/9): kubectl-1.30.5-150500.1.1.x86_64.rpm      21 MB/s |  10 MB     00:00
(7/9): kubeadm-1.30.5-150500.1.1.x86_64.rpm      17 MB/s |  10 MB     00:00
(8/9): kubelet-1.30.5-150500.1.1.x86_64.rpm      32 MB/s |  17 MB     00:00
(9/9): kubernetes-cni-1.4.0-150500.1.1.x86_64.r  21 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
  Verifying        : kubectl-1.30.5-150500.1.1.x86_64                      7/9
  Verifying        : kubelet-1.30.5-150500.1.1.x86_64                      8/9
  Verifying        : kubernetes-cni-1.4.0-150500.1.1.x86_64                9/9

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-172-31-81-216 ~]$
```

- 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

```
[ec2-user@ip-172-31-81-216 ~]$ sudo swapoff -a
[ec2-user@ip-172-31-81-216 ~]$ echo "net.bridge.bridge-nf-call-iptables=1" | sud
o tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-81-216 ~]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
```

4. Perform this ONLY on the Master machine

- Initialize kubernetes by typing below command

sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

```
[ec2-user@ip-172-31-81-216 ~]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/1
6 --ignore-preflight-errors=all
I0914 07:46:39.398298   28873 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the requir
ed 2

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 172.31.81.216:6443 --token f35qak.rrjyqp89bqsomqrn \
        --discovery-token-ca-cert-hash sha256:593693973f6b40e7bc61dec2c73c617ba2
26caafee64bc2776ee360c42b6f29c
```

- So after initialization you will get token at the end for joining master and worker. Like here I got this :(save this token as it is required later.Then you can join any number of worker nodes by running the following on each as root.)

kubeadm join 172.31.81.216:6443 --token f35qak.rrjyqp89bqsomqrn \
    --discovery-token-ca-cert-hash
sha256:593693973f6b40e7bc61dec2c73c617ba226caafee64bc2776ee360c42b6f29c

- Also,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-172-31-81-216 ~]$ mkdir -p $HOME/.kube
[ec2-user@ip-172-31-81-216 ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube
/config
[ec2-user@ip-172-31-81-216 ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Then, add a common networking plugin called flammel file as mentioned in the code. kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
[ec2-user@ip-172-31-81-216 ~]$ kubectl apply -f https://raw.githubusercontent.co
m/coreos/flannel/master/Documentation/kube-flannel.yml
error: error validating "https://raw.githubusercontent.com/coreos/flannel/master
/Documentation/kube-flannel.yml": error validating data: failed to download open
api: Get "https://172.31.81.216:6443/openapi/v2?timeout=32s": dial tcp 172.31.81
.216:6443: connect: connection refused; if you choose to ignore these errors, tu
rn validation off with --validate=false
```

This step gives an error

Conclusion:

In this experiment, we successfully set up a Kubernetes cluster across three Amazon Linux EC2 instances, each equipped with Kubernetes components. The master node was initialized using kubeadm, and pod networking was configured with the Flannel network plugin. Worker nodes were integrated into the cluster through the join command generated during the master node's initialization. The process provided a comprehensive understanding of Kubernetes cluster setup on EC2. However, there was a noticeable delay in the worker nodes connecting to the master node, which may have been caused by network connectivity issues or configuration discrepancies.