Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

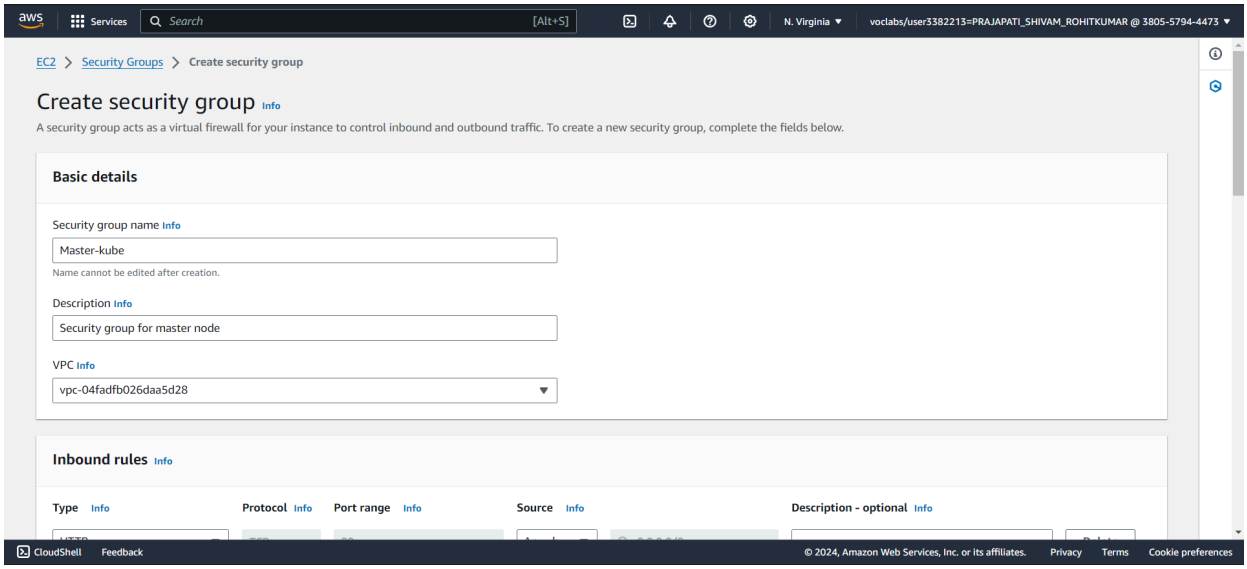
Experiment No : 3

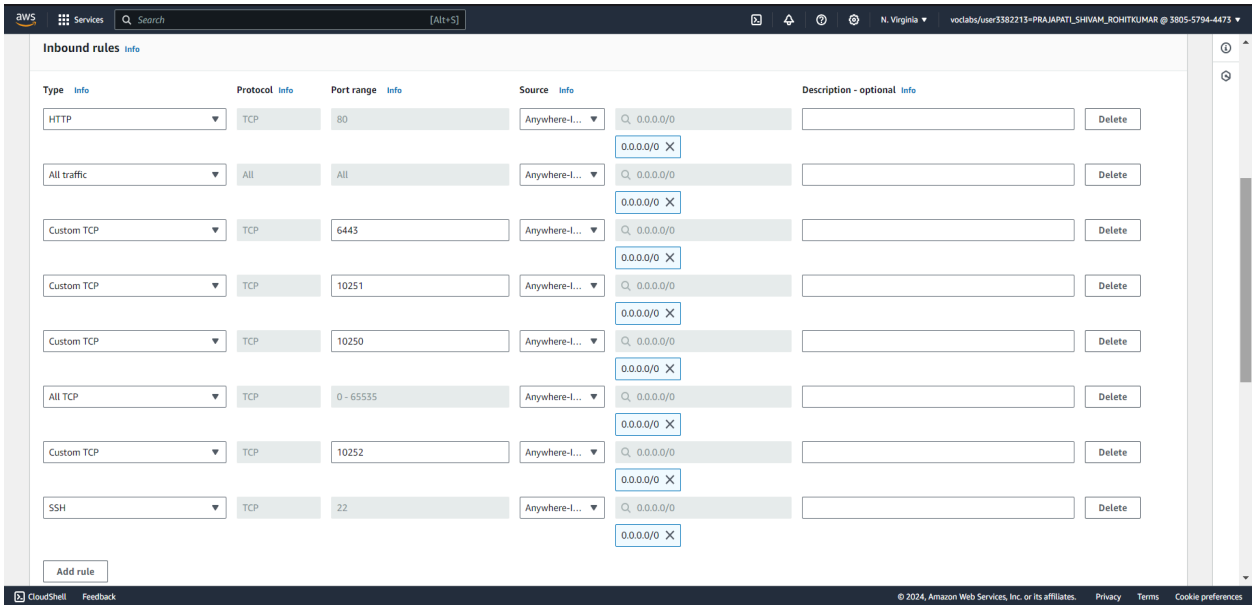
AIM: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernete Cluster on Linux Machines/Cloud Platforms.

PREREQUISITES:

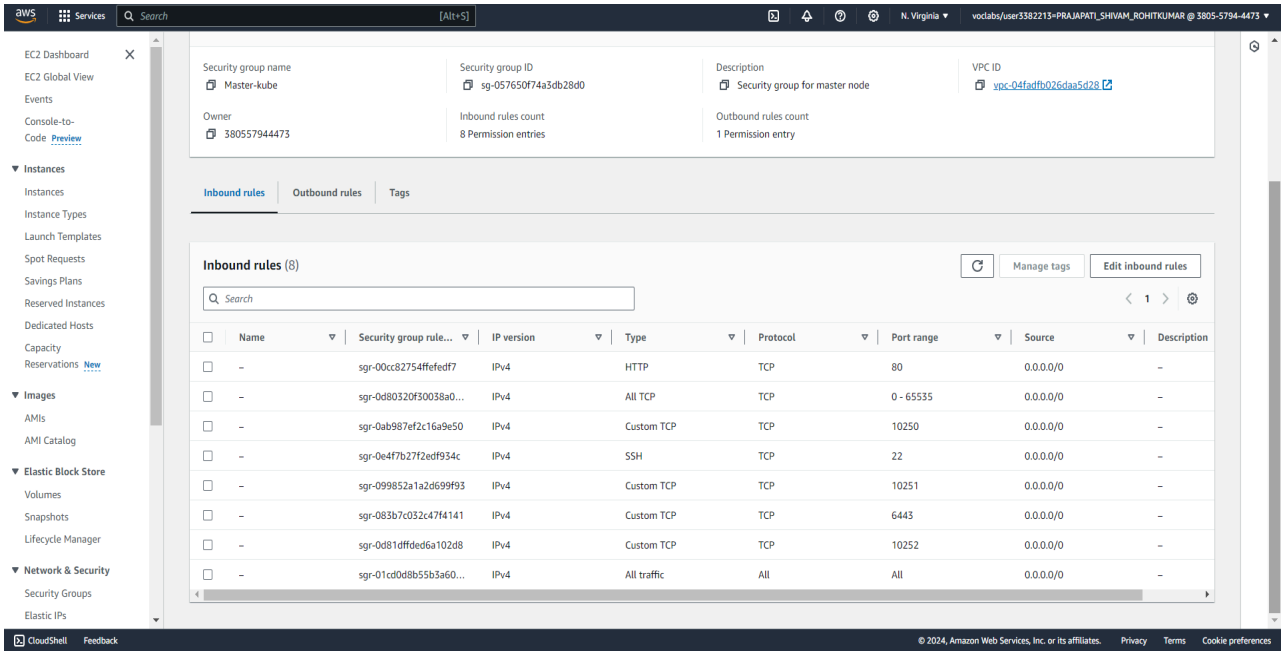
Create 2 Security Groups for Master Node and Worker Nodes and add the following inbound rules in those Groups.

Master Node Security Group:

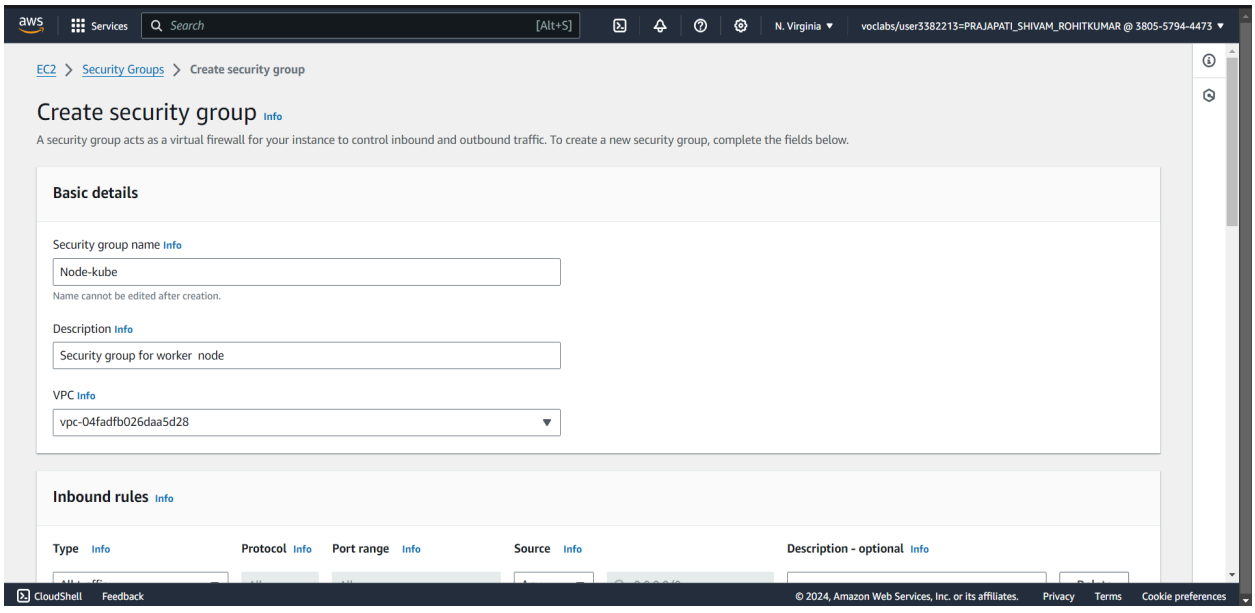


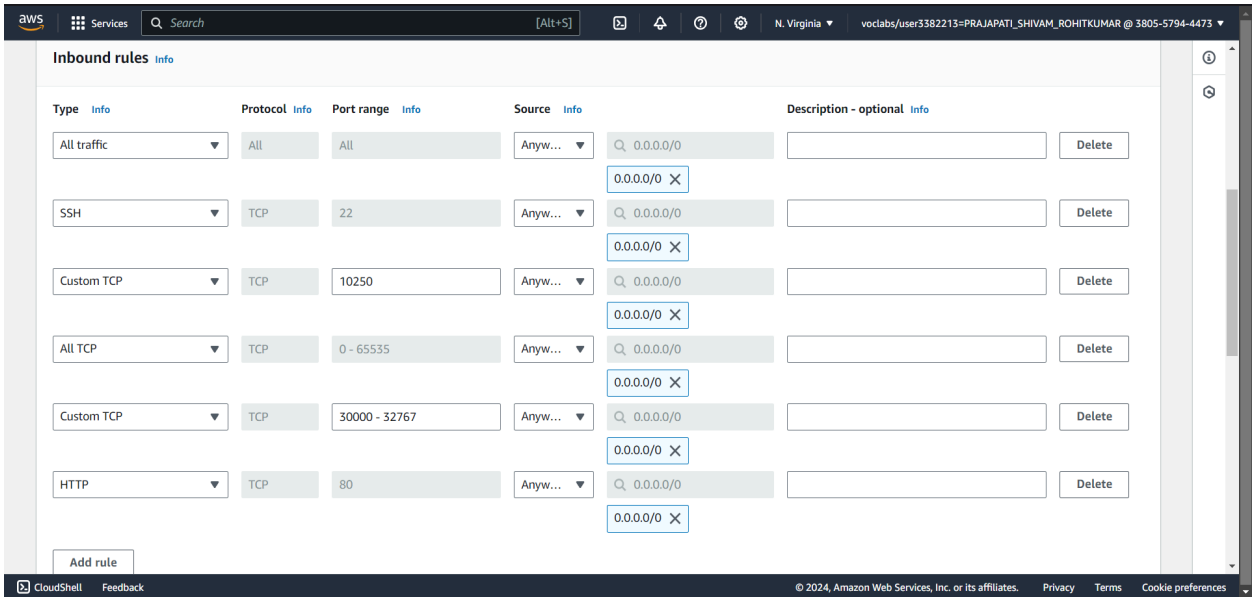


Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

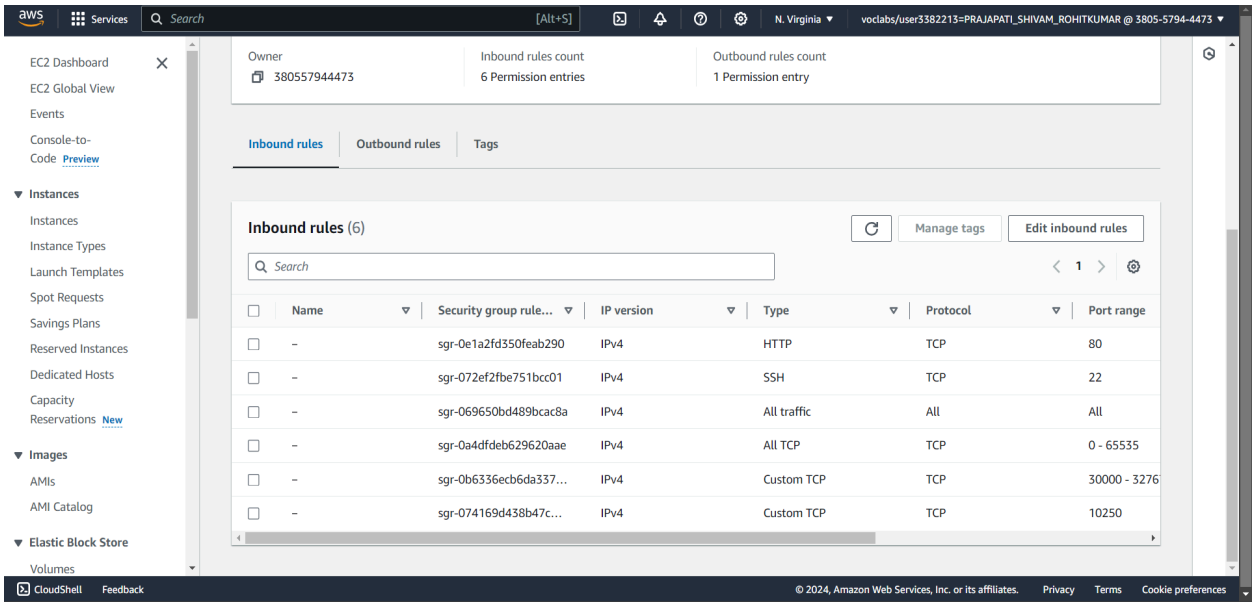


Worker Node Security Group :





Added Required Inbound rules for worker nodes

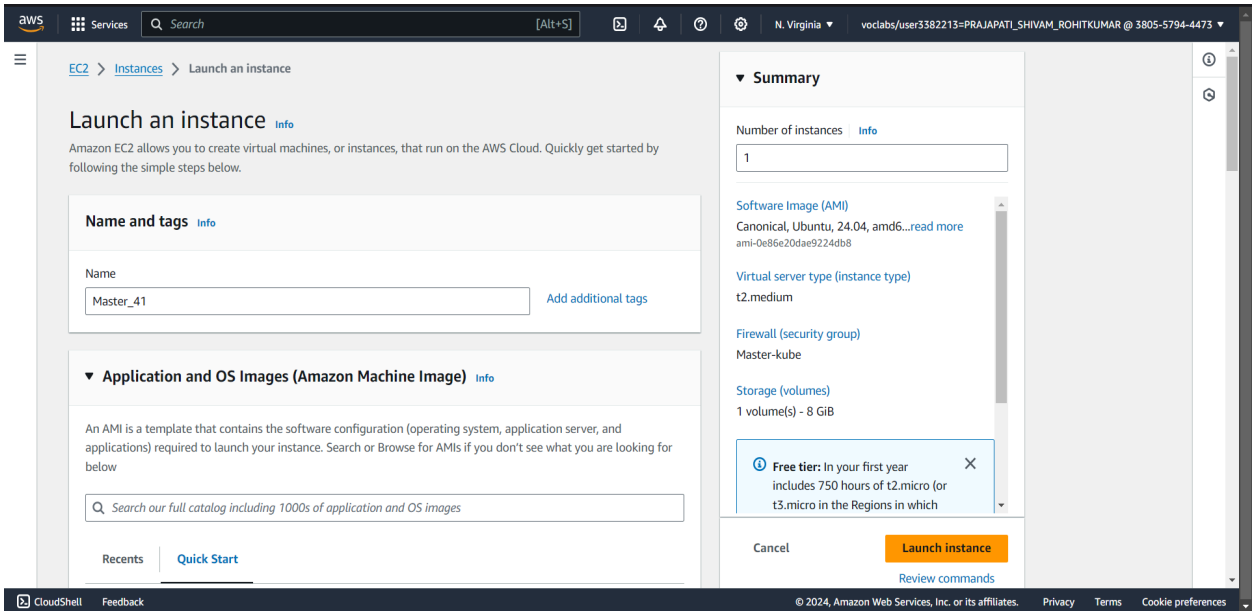


Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

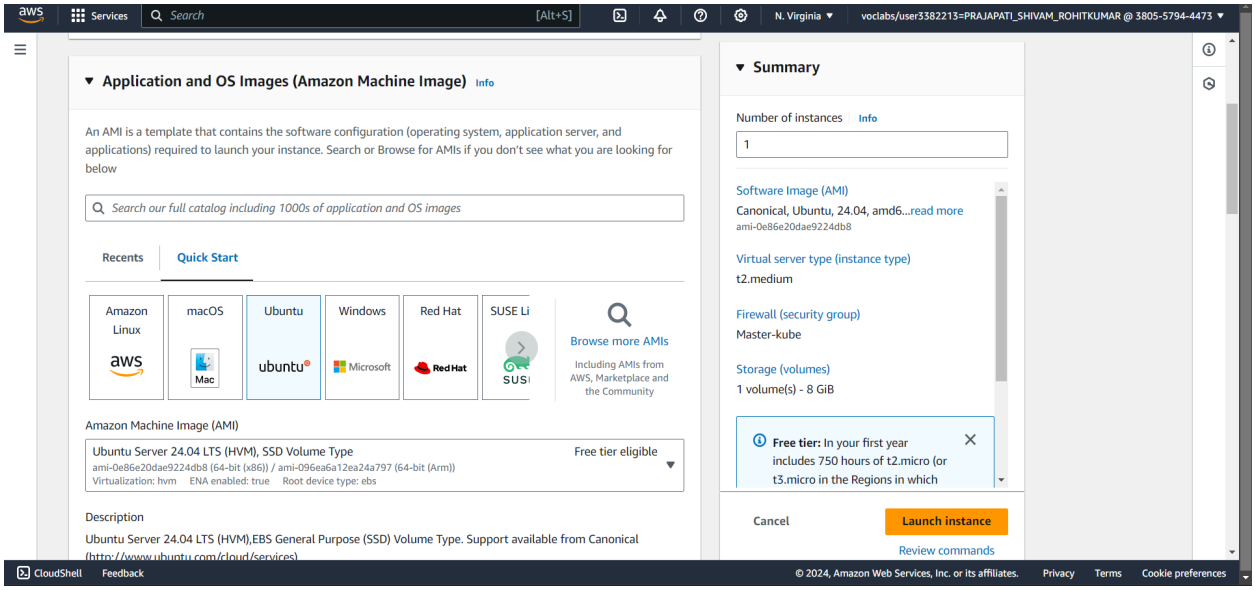
Step 1: Access your AWS Academy or personal account and create **three new EC2 instances**. For the AMI, choose **Ubuntu** and set the **instance type to t2.medium**. Generate an RSA key in .pem format and move the downloaded key to a new folder for **eg: Newfolder**; you can either use three separate keys or one shared key for all instances ,in my case I have made three different keys .

It's essential to select t2.medium, as it includes at least 2 CPUs, which are required for this setup.. Additionally, make sure to select security groups from the ones already available i.e master node and worker node will select their own security groups respectively

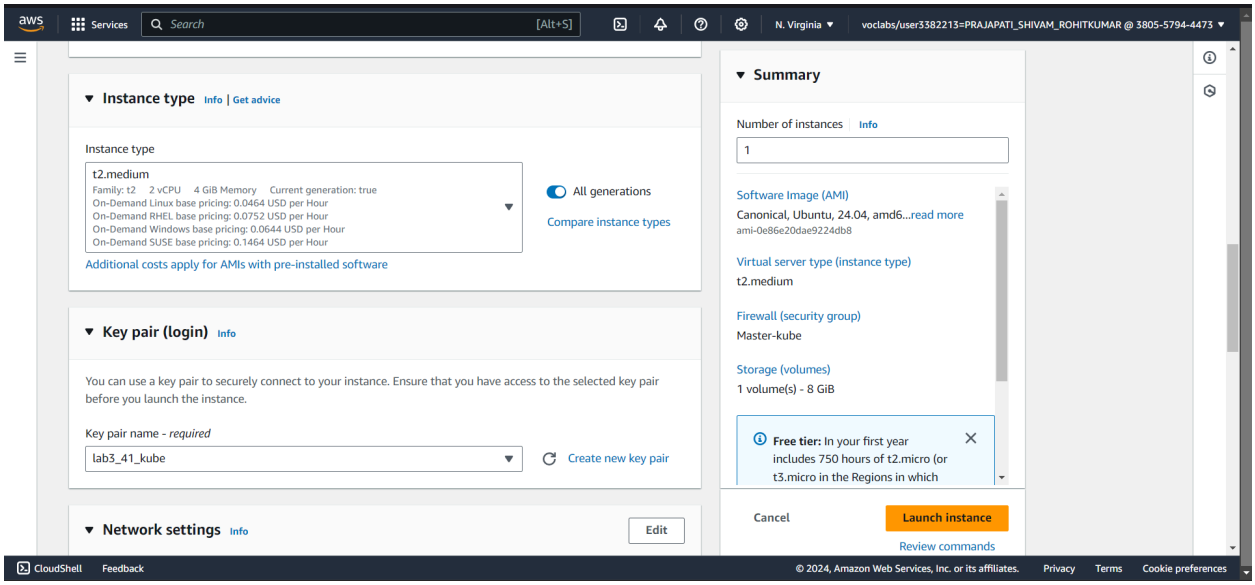
Master Node:



Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

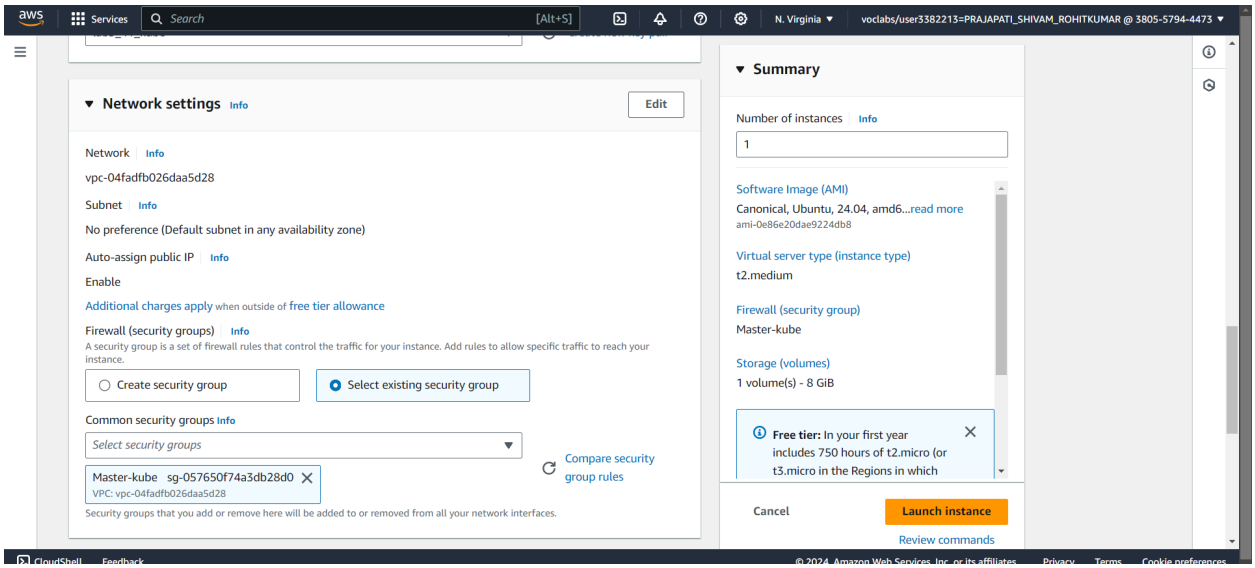


AMI as Ubuntu



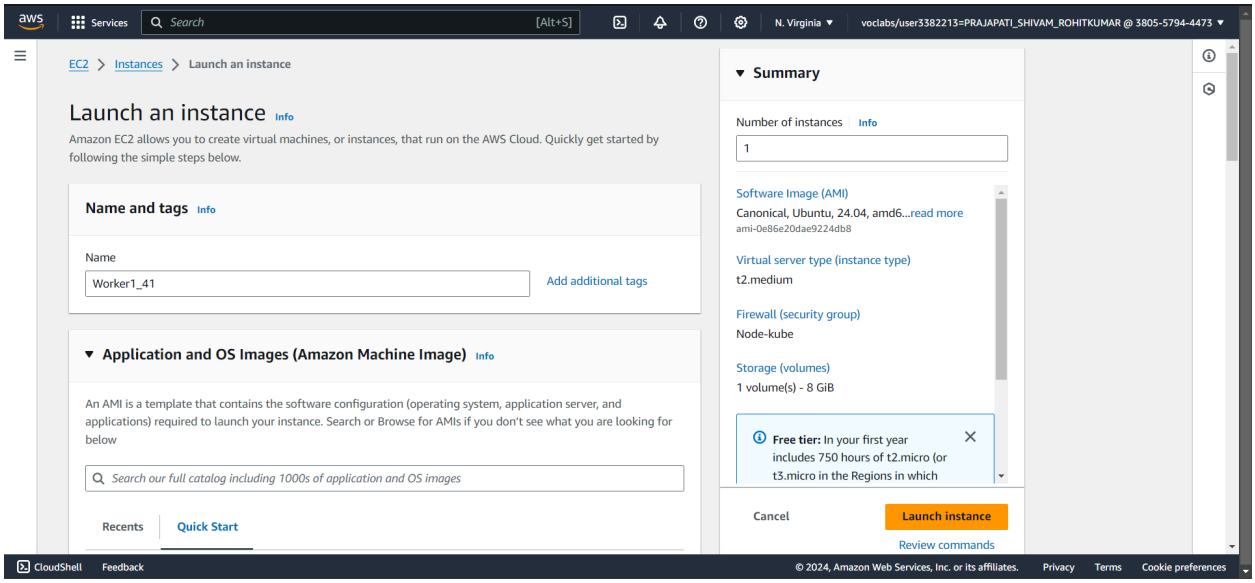
Generate Instance type and create key pair login

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

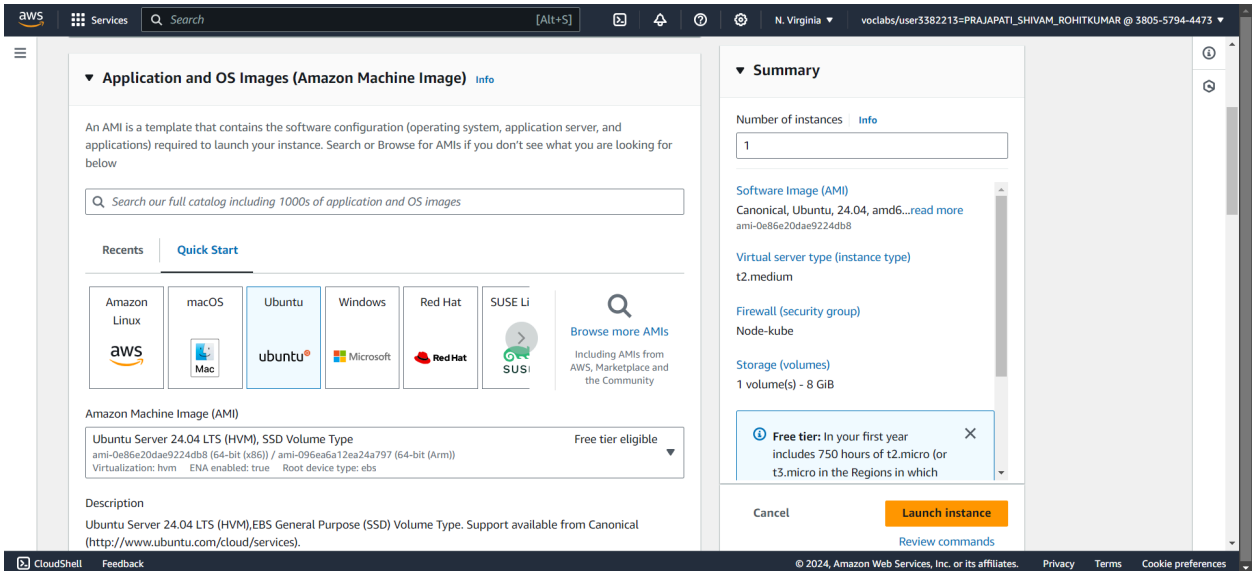


Select Required Security Group

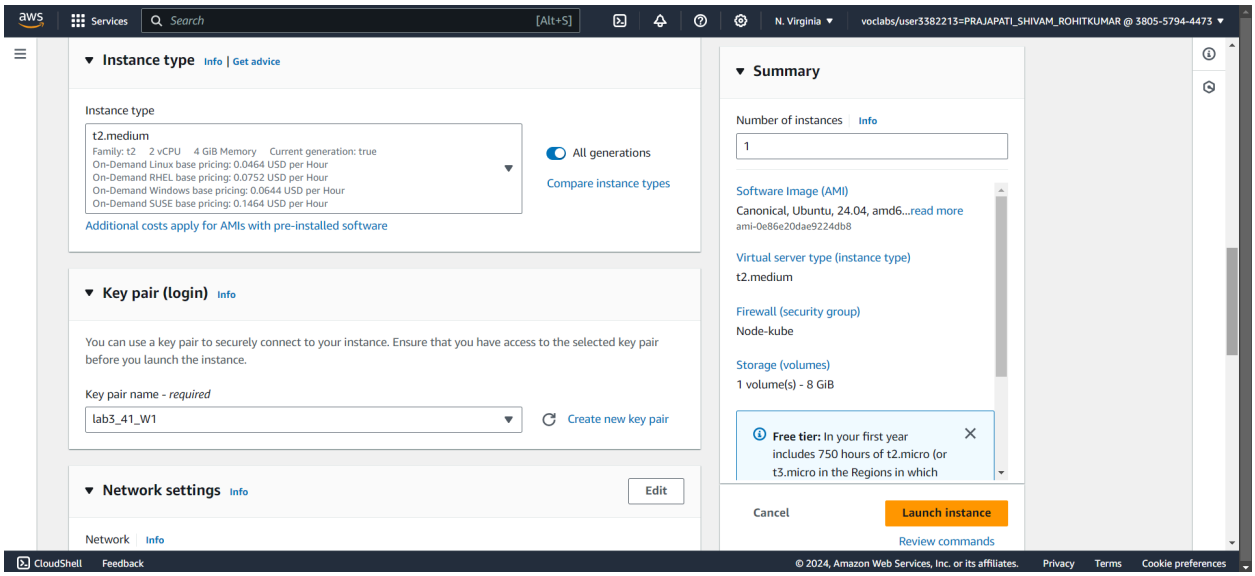
Worker Node 1:



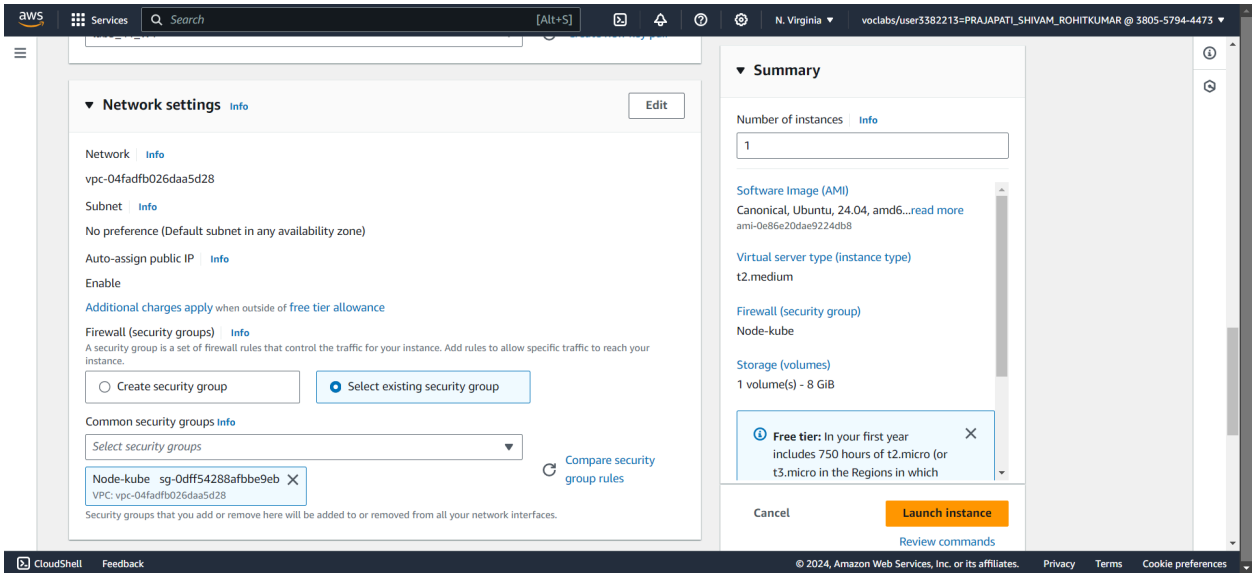
Give a name to your instance .



AMI as Ubuntu

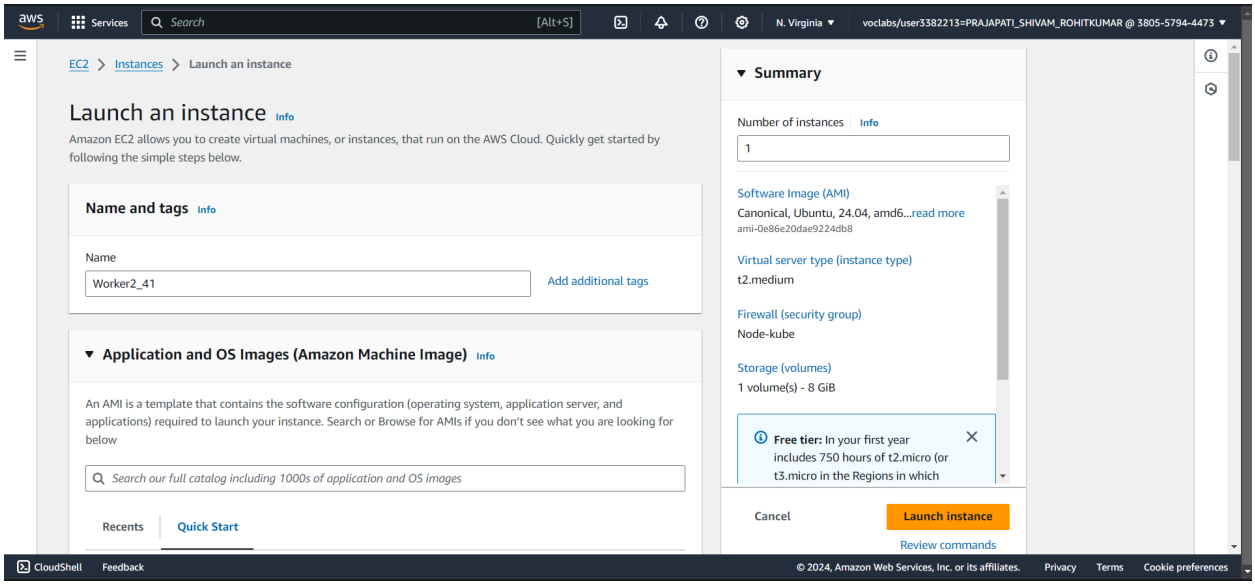


Created the Key pair login



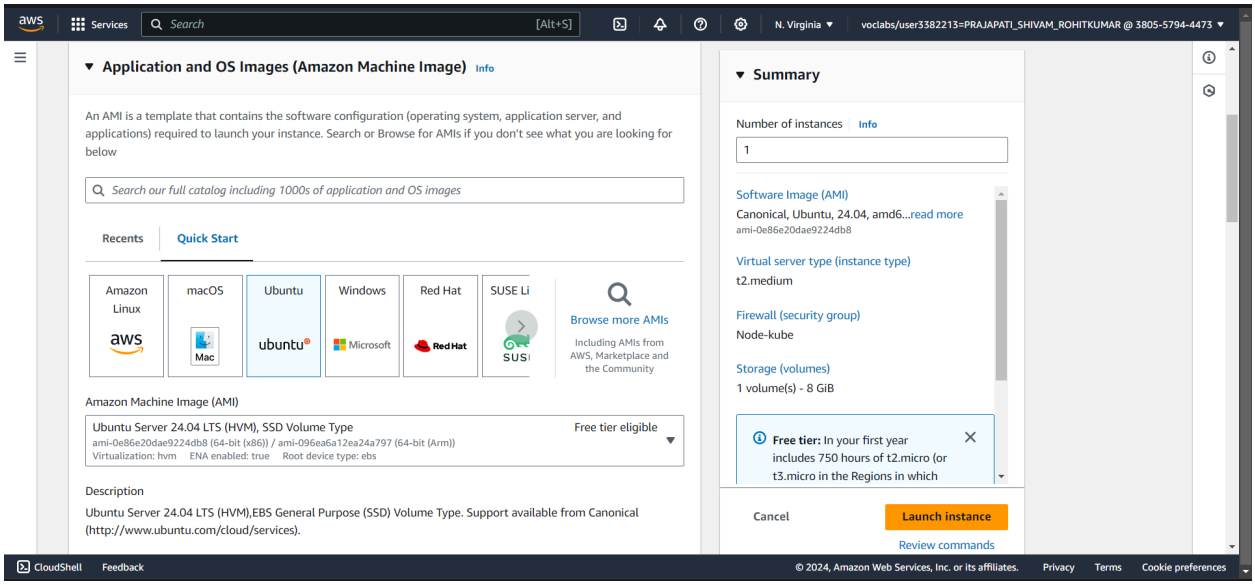
Select required Security Group

Worker Node 2:

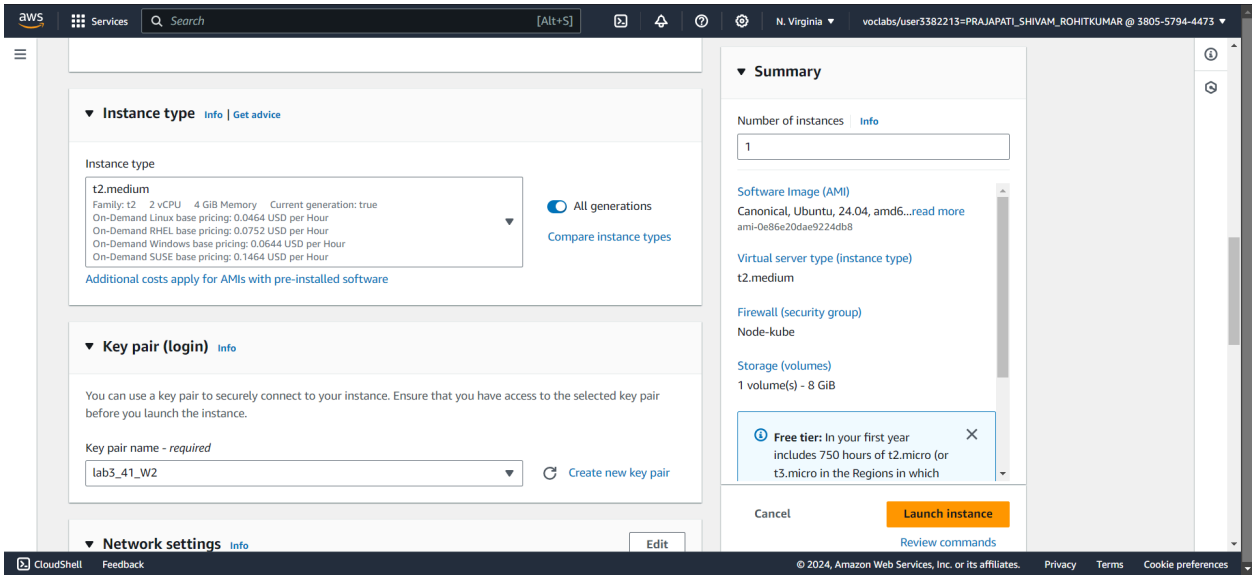


Give name to your Instance

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

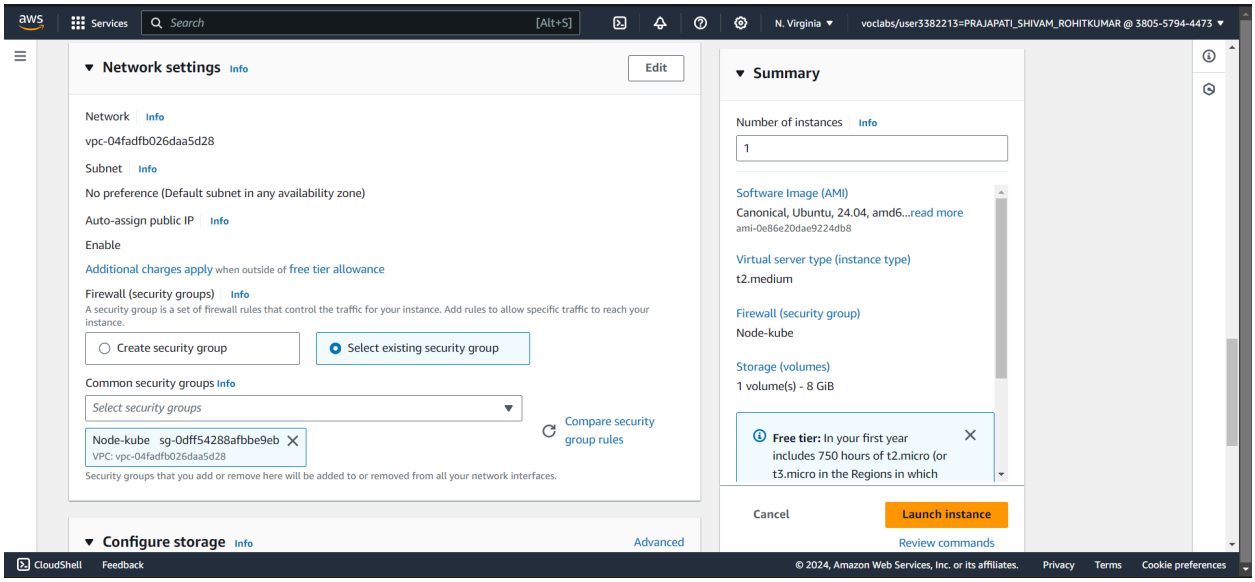


AMI as Ubuntu

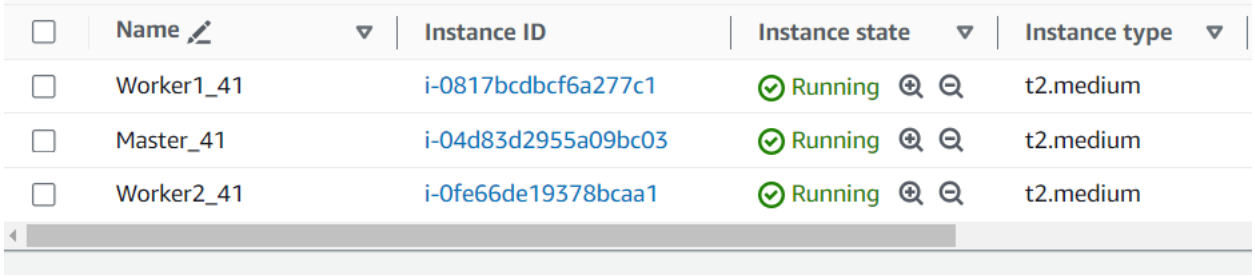


Select proper Instance Type and create key pair login

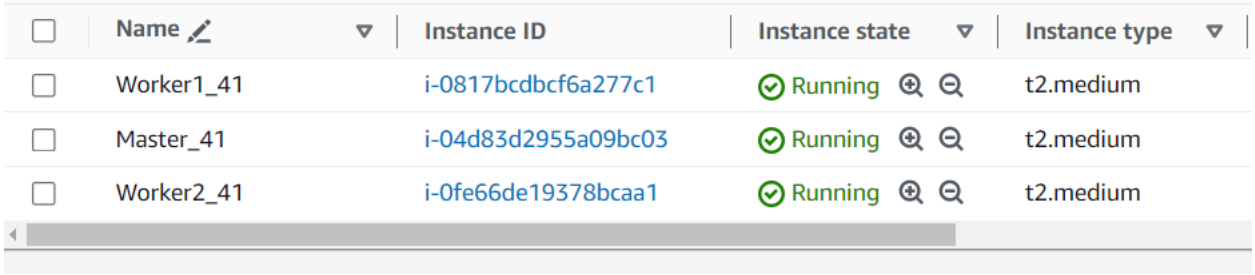
Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25



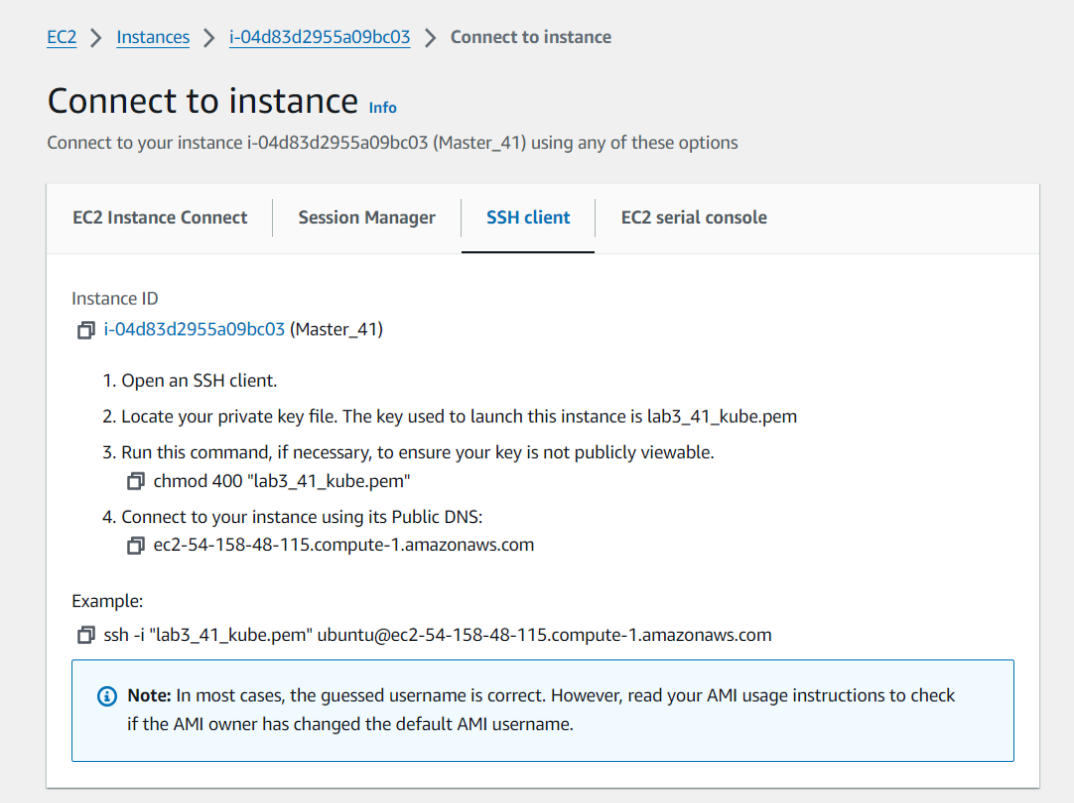
Instances are :



Step 2: After creating the instances click on Connect for all the instances one by one and navigate to the SSH Client section .



Master Node:

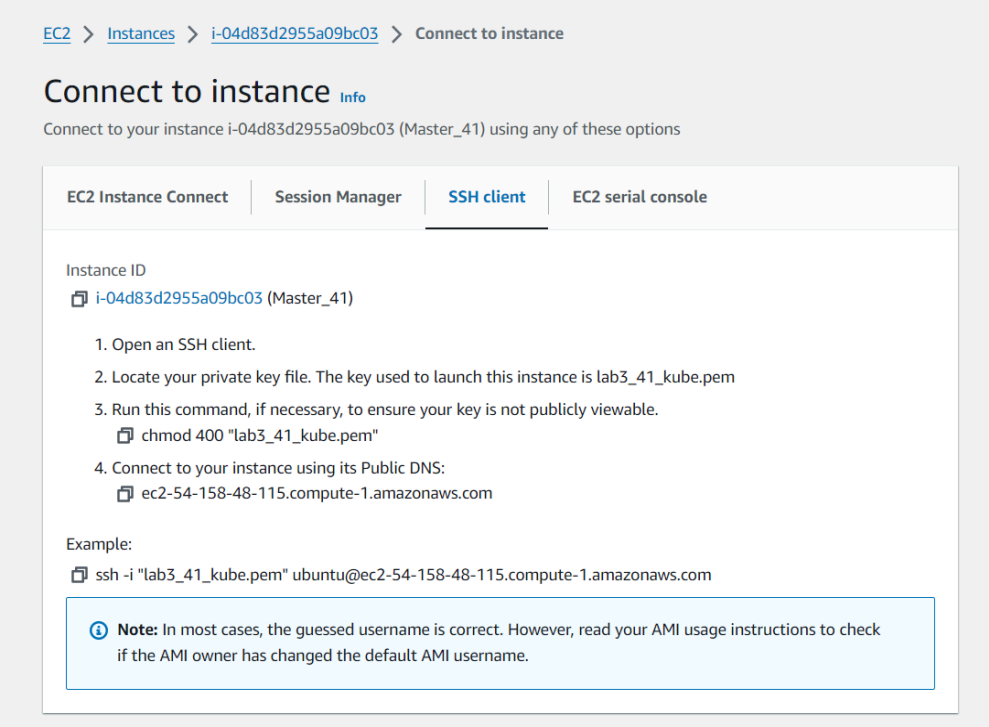


Step 3: Now open the folder in the terminal 3 times by right clicking on it for Master, Node1 & Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) from the ssh client section for instance in the terminal as shown above.

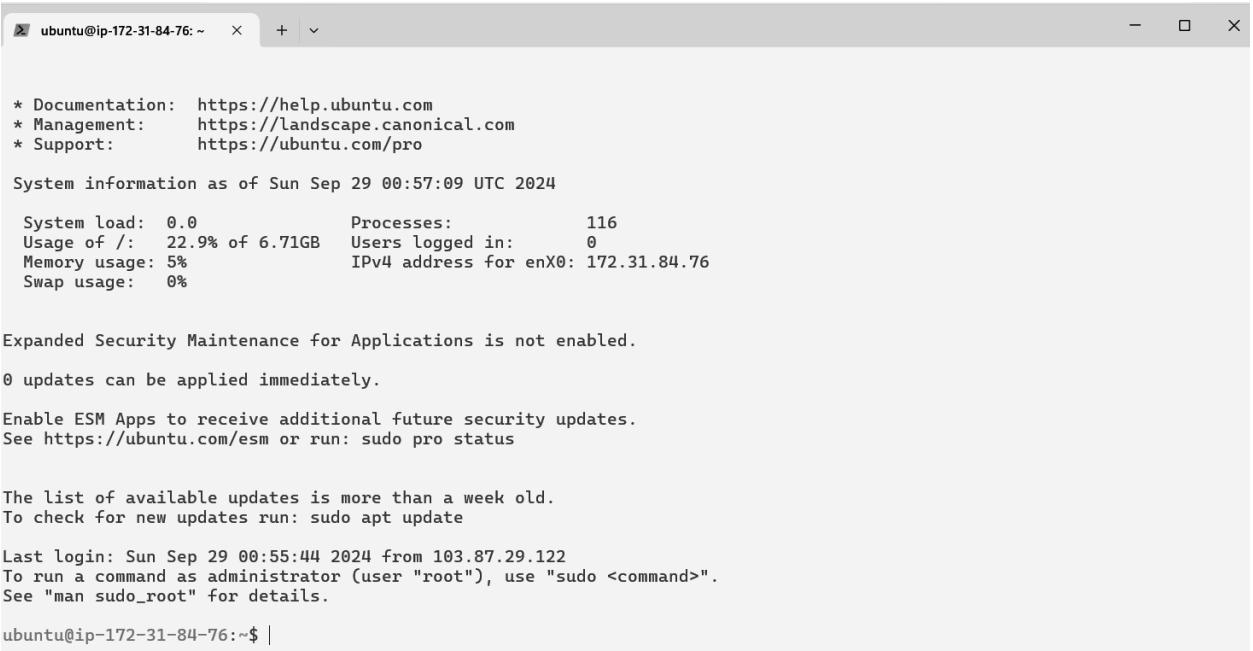
This will basically make our terminal to do remote login on our ec2 instance via SSH

Name: Shivam.R.Prajapati Div: D15C Roll No:41 Academic Year: 2024-25

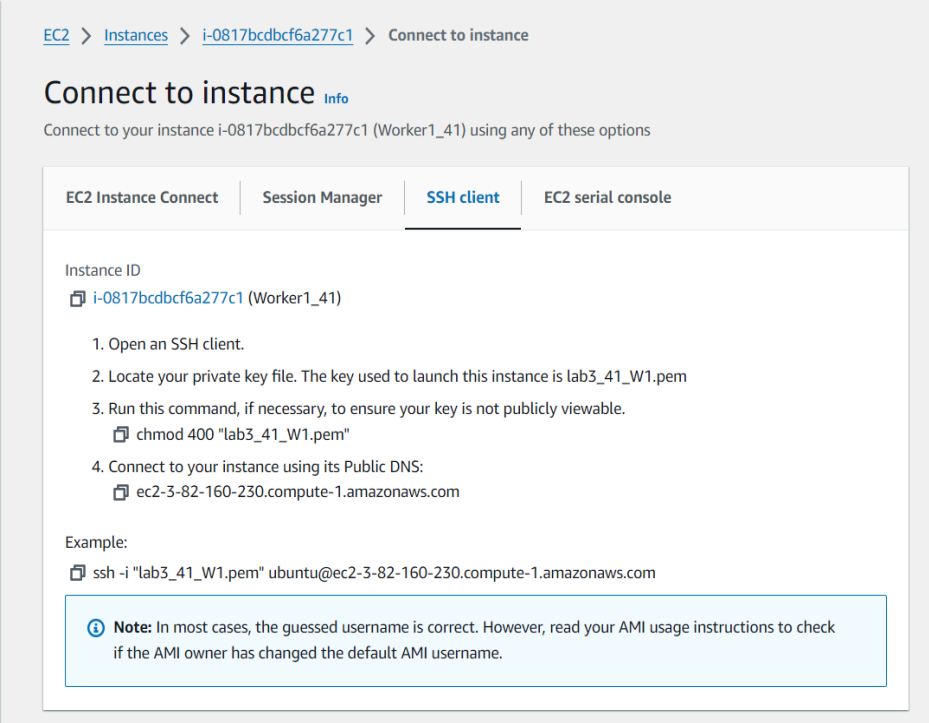
MasterNode:



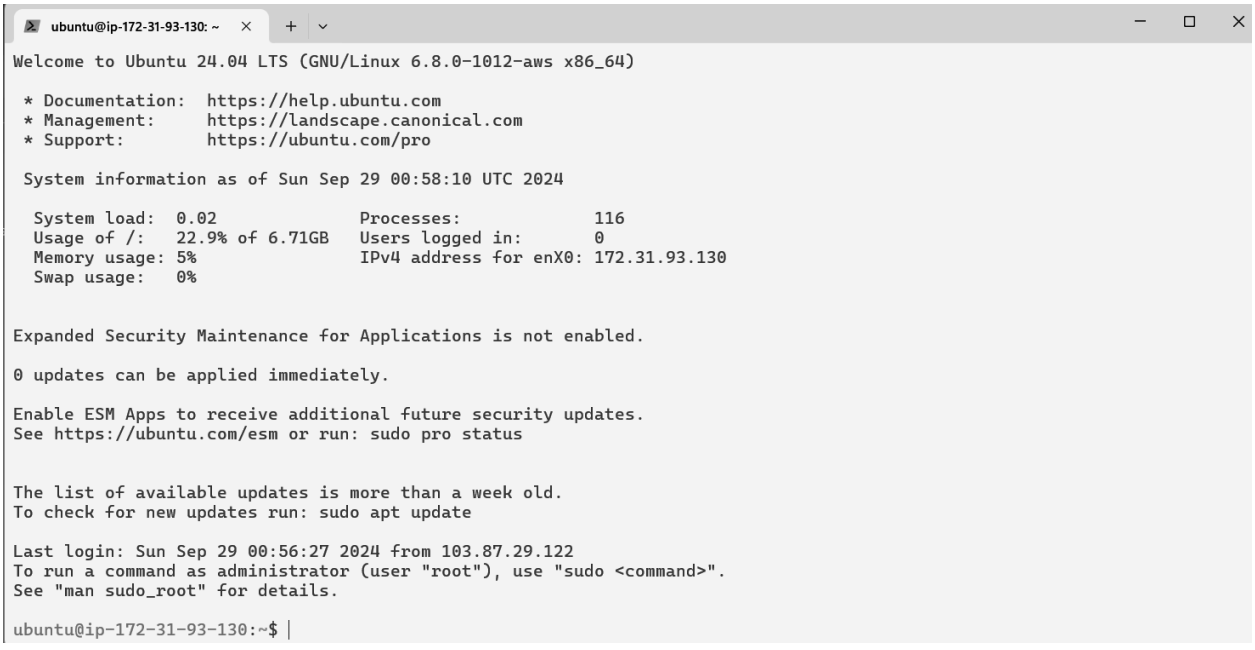
Successful Connection:



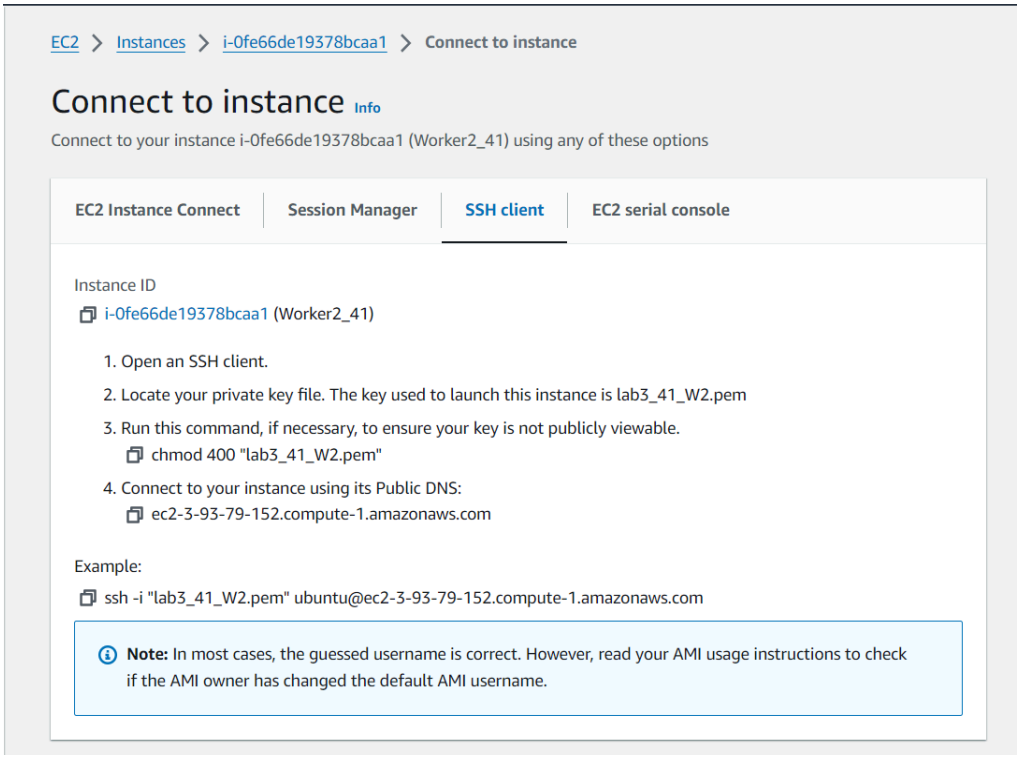
Worker Node 1:



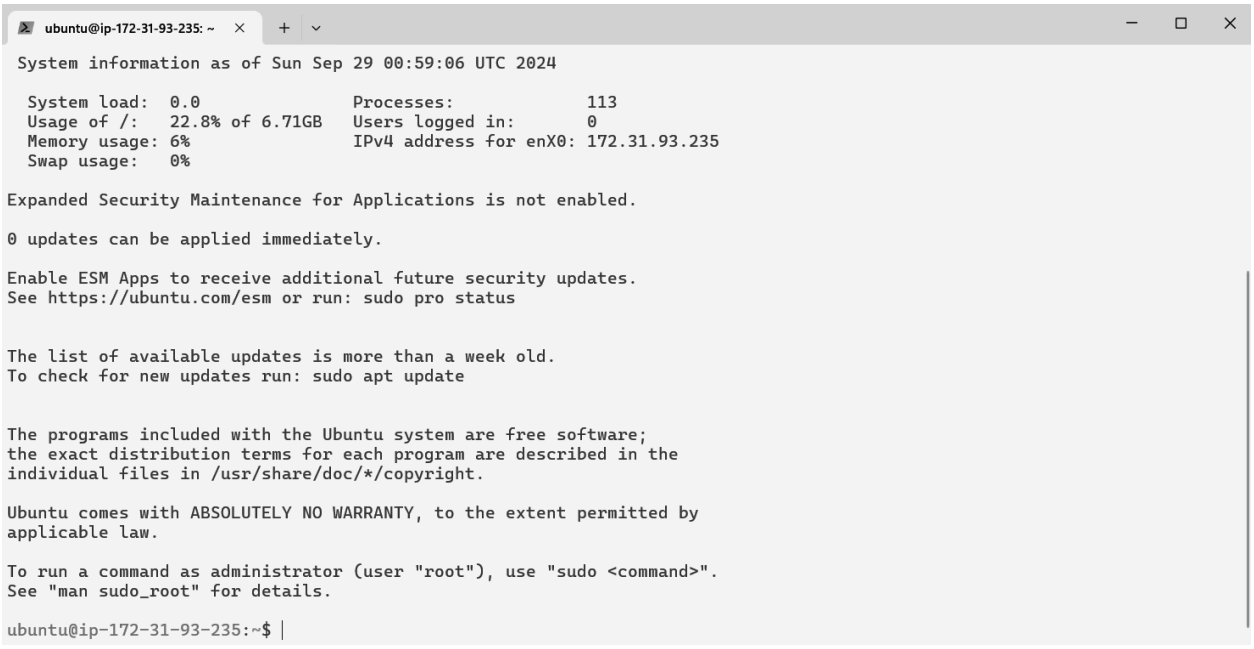
Successful Connection:



Worker Node 2:



Successful Connection:



Step 4: Run on Master,Worker Node 1,and Worker Node 2 the below commands to install and setup Docker in Master,Worker Node 1,and Worker Node 2.

a) curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-84-76:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtEb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
BBMBCgAhBQJYrefAAhsvBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEI2BgDwO
```

```
ubuntu@ip-172-31-84-76: ~
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:39 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.9 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4560 B]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [272 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.3 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 4s (6810 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-84-76:~$
```

b) sudo apt-get update
sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-84-76:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 143 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
```

```
ubuntu@ip-172-31-84-76: ~
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.17.1-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.serv
ice.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-84-76:~$
```

c)sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]

}
EOF

```
ubuntu@ip-172-31-84-76:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-84-76:~$
```

d) sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-84-76:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-84-76:~$
```

Step 5: Run the below command to **install Kubernetes** on all the three terminals one by one

a) sudo rm -f /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-90-144:~$ sudo rm -f /etc/apt/sources.list.d/kubernetes.list
```

b) sudo apt-get update

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored
ection in apt-key(8) for details.
```

c) sudo apt-get install -y apt-transport-https ca-certificates curl gpg

```
ubuntu@ip-172-31-90-144:~$ sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
```

```
Fetched 904 kB in 0s (29.7 MB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68007 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Preparing to unpack .../curl_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl4t64_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl4t64:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Preparing to unpack .../libcurl3t64-gnutls_8.5.0-2ubuntu10.4_amd64.deb ...
Unpacking libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) over (8.5.0-2ubuntu10.1) ...
Setting up apt-transport-https (2.7.14build2) ...
Setting up libcurl4t64:amd64 (8.5.0-2ubuntu10.4) ...
Setting up libcurl3t64-gnutls:amd64 (8.5.0-2ubuntu10.4) ...
Setting up curl (8.5.0-2ubuntu10.4) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart packagekit.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

d) sudo mkdir -p -m 755 /etc/apt/keyrings

```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p -m 755 /etc/apt/keyrings
```

f) curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

```
ubuntu@ip-172-31-90-144:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key
ng.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
```

g) echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-90-144:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
c/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

f) sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

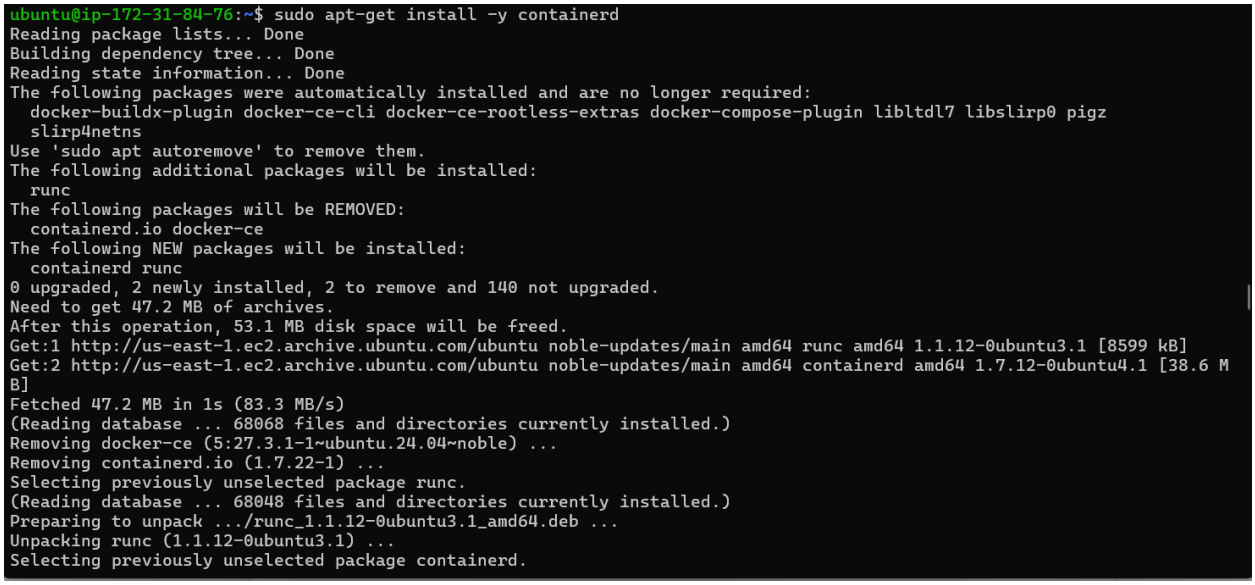
sudo apt-mark hold kubelet kubeadm kubectl

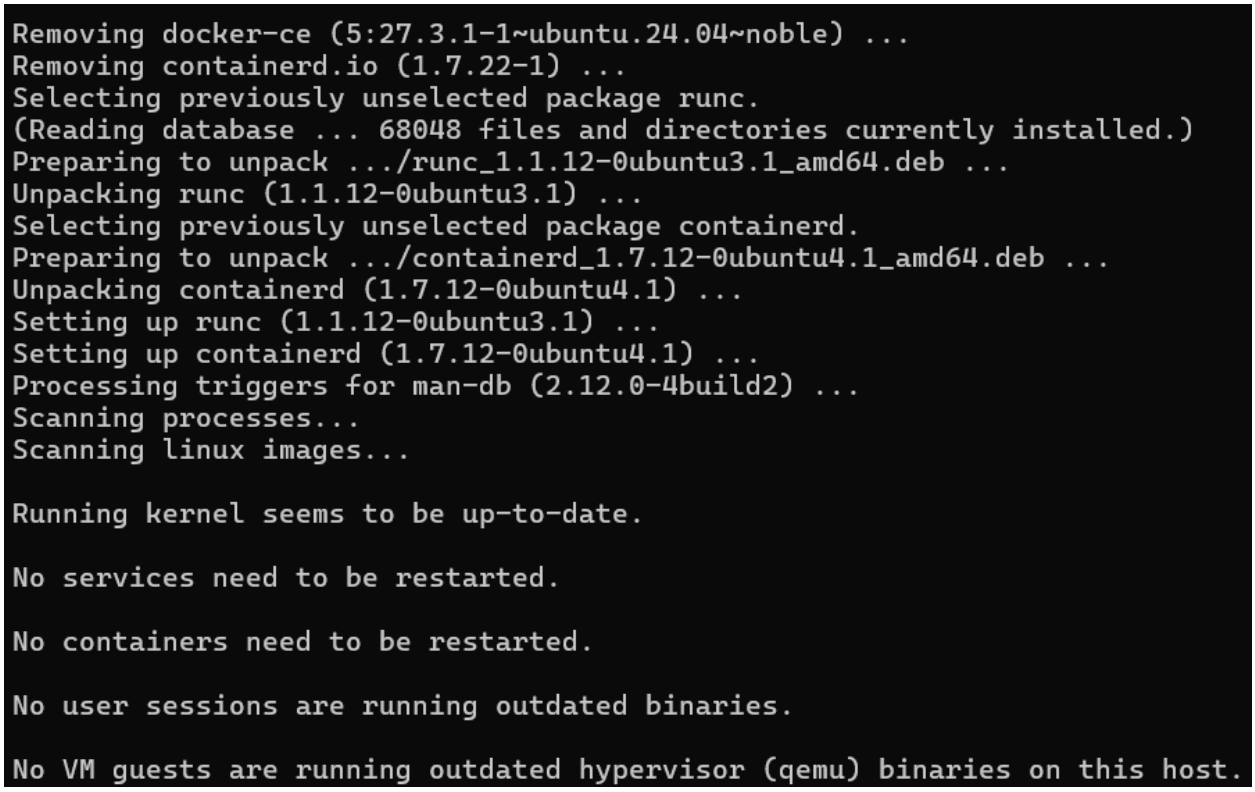
```
ubuntu@ip-172-31-90-144:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (9987 B/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.
ection in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 140 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (70.6 MB/s)
Selecting previously unselected package conntrack.
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

g) sudo systemctl enable --now kubelet

h) sudo apt-get install -y containerd





i) sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-90-144:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0

[metrics]
  address = ""
  grpc_histogram = false

[plugins]
```

j) sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

```
ubuntu@ip-172-31-84-76: ~
ubuntu@ip-172-31-84-76:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-29 01:34:39 UTC; 229ms ago
       Docs: https://containerd.io
   Main PID: 5324 (containerd)
      Tasks: 8
     Memory: 13.1M (peak: 14.3M)
        CPU: 59ms
     CGroup: /system.slice/containerd.service
             └─5324 /usr/bin/containerd

Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934118179Z" level=info msg="Start subscribi>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934140135Z" level=info msg=serving... addre>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934165273Z" level=info msg="Start recoverin>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934187499Z" level=info msg=serving... addre>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934214174Z" level=info msg="Start event mon>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934226888Z" level=info msg="Start snapshots>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934234364Z" level=info msg="Start cni netwo>
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.934240935Z" level=info msg="Start streaming>
Sep 29 01:34:39 ip-172-31-84-76 systemd[1]: Started containerd.service - containerd container runtime.
Sep 29 01:34:39 ip-172-31-84-76 containerd[5324]: time="2024-09-29T01:34:39.936194445Z" level=info msg="containerd succ>
ubuntu@ip-172-31-84-76:~$ sudo apt-get install -y socat
```

k) sudo apt-get install -y socat

```
ubuntu@ip-172-31-84-76:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (15.2 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68112 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-84-76:~$
```

Step 6: Set up the Kubernetes cluster by running the following command exclusively on the master node:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16. This command initializes the Kubernetes control plane and specifies the pod network range to be used within the cluster i.e kubeadm will initialize the kubernetes cluster and cidr is Classless Inter Domain Range which decides the range of network range of the pods

```
ubuntu@ip-172-31-84-76:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[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'
W0929 01:52:20.346395    6244 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox 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-172-31-84-76 kubernetes kubernetes.default kubernetes.default
.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.84.76]
[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-172-31-84-76 localhost] and IPs [172.31.84.76 127.0.0.1 ::1
]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-84-76 localhost] and IPs [172.31.84.76 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
```

```
ubuntu@ip-172-31-84-76: ~
certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Boots
trap 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 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
ubuntu@ip-172-31-84-76:~$
```

Run this command on master:

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

These commands create a .kube directory in your home folder, copy the Kubernetes admin configuration file into it, and change the ownership of that configuration file to the current user to ensure proper access.

```
ubuntu@ip-172-31-84-76:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-76:~$
```

Step 7: Now Run the command **kubectl get nodes** to see the nodes before executing Join command on nodes. kubectl is a command-line tool used to interact with and manage Kubernetes clusters. It allows users to deploy applications, inspect and manage cluster resources, and view logs, among other tasks, by sending commands to the Kubernetes API server.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-76    NotReady   control-plane   6m42s   v1.31.1
ubuntu@ip-172-31-84-76:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
For this from the kubeadm init command output for the master node copy and paste this on both the node

```
[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 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
ubuntu@ip-172-31-84-76:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE     VERSION
ip-172-31-84-76   NotReady   control-plane   6m42s   v1.31.1
ubuntu@ip-172-31-84-76:~$
```

Copy and paste: ***sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t ***
--discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88

Worker Node 1:

```
ubuntu@ip-172-31-93-130:~$ sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.001865005s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Worker Node 2:

```
ubuntu@ip-172-31-93-235:~$ sudo kubeadm join 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
        --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fcb453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.000833351s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step 9: Run the command **kubectl get nodes** to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-76    NotReady   control-plane   30m     v1.31.1
ip-172-31-93-130   NotReady   <none>          4m23s   v1.31.1
ip-172-31-93-235   NotReady   <none>          37s     v1.31.1
ubuntu@ip-172-31-84-76:~$
```

Step 10: Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml .This command applies the Calico network plugin configuration, which enables networking capabilities in the Kubernetes cluster .It enhances the effective communication between pods in network

```
ubuntu@ip-172-31-84-76:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-84-76:~$
```

Now perform **sudo systemctl status kubelet**

```
ubuntu@ip-172-31-84-76:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Sun 2024-09-29 01:52:38 UTC; 35min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 6918 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 33.6M (peak: 34.0M)
        CPU: 26.546s
     CGroup: /system.slice/kubelet.service
             └─6918 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/ku>

Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         : unknown
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:  > pod="kube-system/coredns-7c65d6cfc9-7nrwx" podUID="1b227dfc-524d-4b11>
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.874236    6918 log.go:32] "StopPodSandbox from runtime se>
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         rpc error: code = Unknown desc = failed to stop container "1daf0>
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:         : unknown
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]:  > podSandboxID="8ce61ac60e9c44c62ac2ac734ef78143faf196ccf04c67427bc23a9>
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.874271    6918 kuberuntime_manager.go:1479] "Failed to st>
Sep 29 02:27:45 ip-172-31-84-76 kubelet[6918]: E0929 02:27:45.960413    6918 kubelet.go:1865] "KillPod failed" err="[fa>
Sep 29 02:27:49 ip-172-31-84-76 kubelet[6918]: I0929 02:27:49.223548    6918 scope.go:117] "RemoveContainer" containerI>
Sep 29 02:27:49 ip-172-31-84-76 kubelet[6918]: E0929 02:27:49.223664    6918 pod_workers.go:1301] "Error syncing pod, s>
ubuntu@ip-172-31-84-76:~$
```

Now Run command **kubectl get nodes -o wide we can see Status is ready**.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   39m     v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    <none>          12m     v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    <none>          8m45s   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

Now to Rename run this command
Rename to Worker_Node1_41: kubectl label node ip-172-31-93-138 kubernetes.io/role=Worker_Node1_41
Rename to Worker_Node2_41 : kubectl label node ip-172-31-93-235 kubernetes.io/role=Worker_Node2_41

Make Sure to give Correct corresponding IP while renaming which can be seen from previous output also.

```
ubuntu@ip-172-31-84-76:~$ kubectl label node ip-172-31-93-130 kubernetes.io/role=Worker_Node1_41
node/ip-172-31-93-130 labeled
ubuntu@ip-172-31-84-76:~$ kubectl label node ip-172-31-93-235 kubernetes.io/role=Worker_Node2_41
node/ip-172-31-93-235 labeled
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   43m   v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    Worker_Node1_41 16m   v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    Worker_Node2_41 12m   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

Step 11: Run command **kubectl get nodes -o wide** . And Hence we can see we have Successfully connected both the Worker Nodes to the Master Node.

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   43m   v1.31.1   172.31.84.76    <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-130  Ready    Worker_Node1_41 16m   v1.31.1   172.31.93.130   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-93-235  Ready    Worker_Node2_41 12m   v1.31.1   172.31.93.235   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-76:~$
```

CONCLUSION:

In this experiment, we established a Kubernetes cluster on AWS EC2 instances, comprising one master node and two worker nodes. After setting up Docker and the required Kubernetes tools (kubelet, kubeadm, kubectl, and containerd) on all nodes, we initialized the master and incorporated the worker nodes into the cluster. Initially, the nodes displayed a NotReady status, which we corrected by installing the Calico network plugin. We also tagged the nodes with their respective roles (control-plane and worker). In the end, all nodes changed to the Ready state, showcasing that we successfully set up and managed the Kubernetes cluster.