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

Theory:
Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.
Today, both large and small software companies are deploying thousands of container instances daily. Managing this level of complexity at scale requires advanced tools. Enter Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat. Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

- Managing resource consumption by applications or teams
- Distributing application load evenly across the infrastructure
- Automatically load balancing requests across multiple instances of an application
- Monitoring resource usage to prevent applications from exceeding resource limits and automatically restarting them if needed
- Moving application instances between hosts when resources are low or if a host fails
- Automatically utilizing additional resources when new hosts are added to the cluster
- Facilitating canary deployments and rollbacks with ease

Necessary Requirements:
- EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.

- **Minimum Requirements:**
  - Instance Type: t2.medium
  - CPUs: 2
  - Memory: Adequate for container orchestration.

This ensured that the Kubernetes cluster had the necessary resources to function smoothly

Sign in to AWS Management Console:

Go to AWS Management Console.

Log in with your account credentials.

Navigate to EC2 Service:

In the AWS Console, search for EC2 and select it to open the EC2 dashboard.

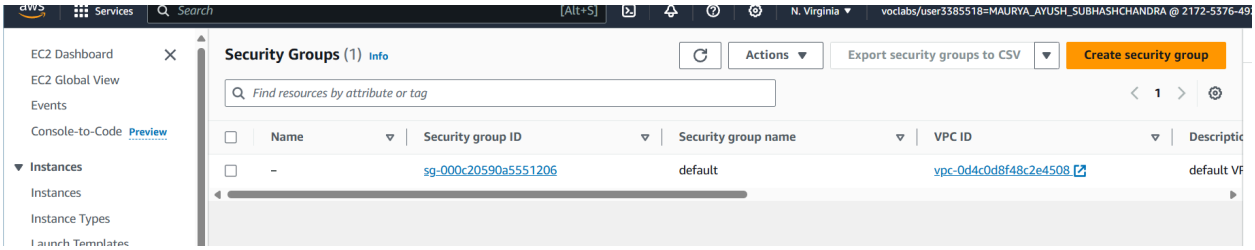
Go to "Security Groups":

On the left-hand navigation pane, under Network & Security, click on Security Groups.

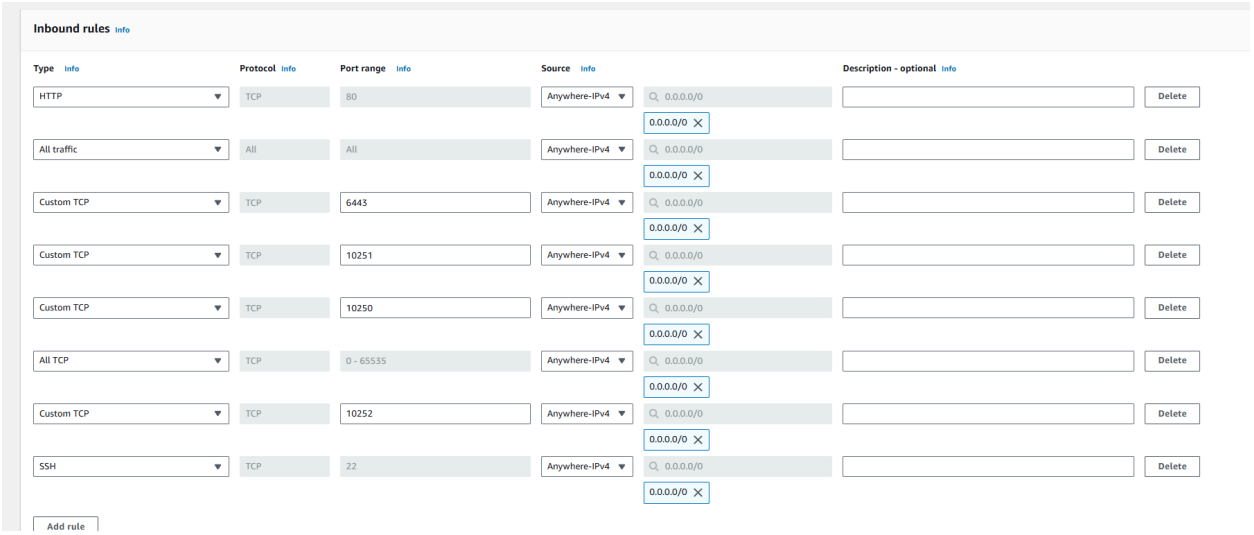
Create Security Group:

Click the Create Security Group button at the top.

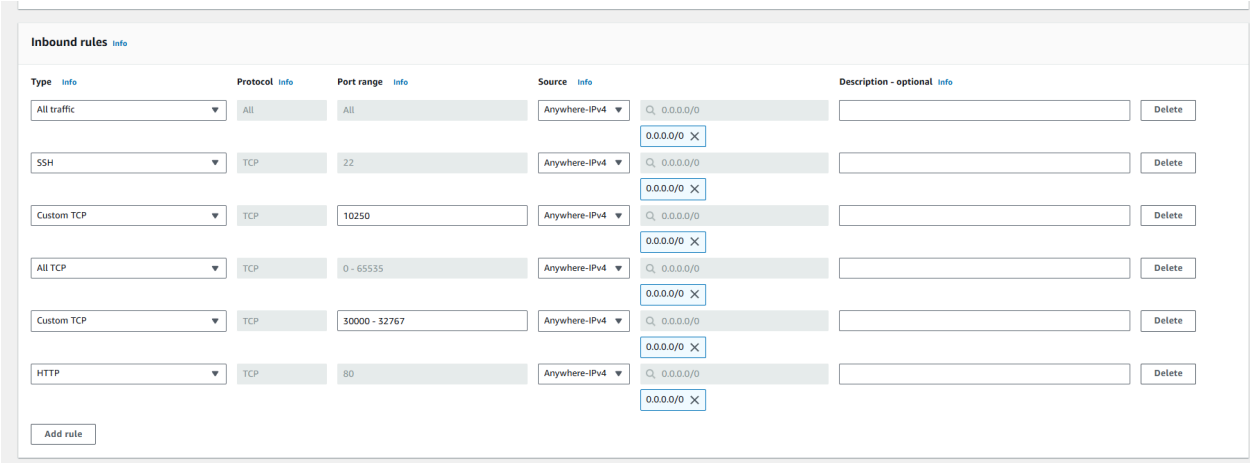
Create 2 Security Groups for Master and Nodes and add the following rules inbound rules in those Groups.



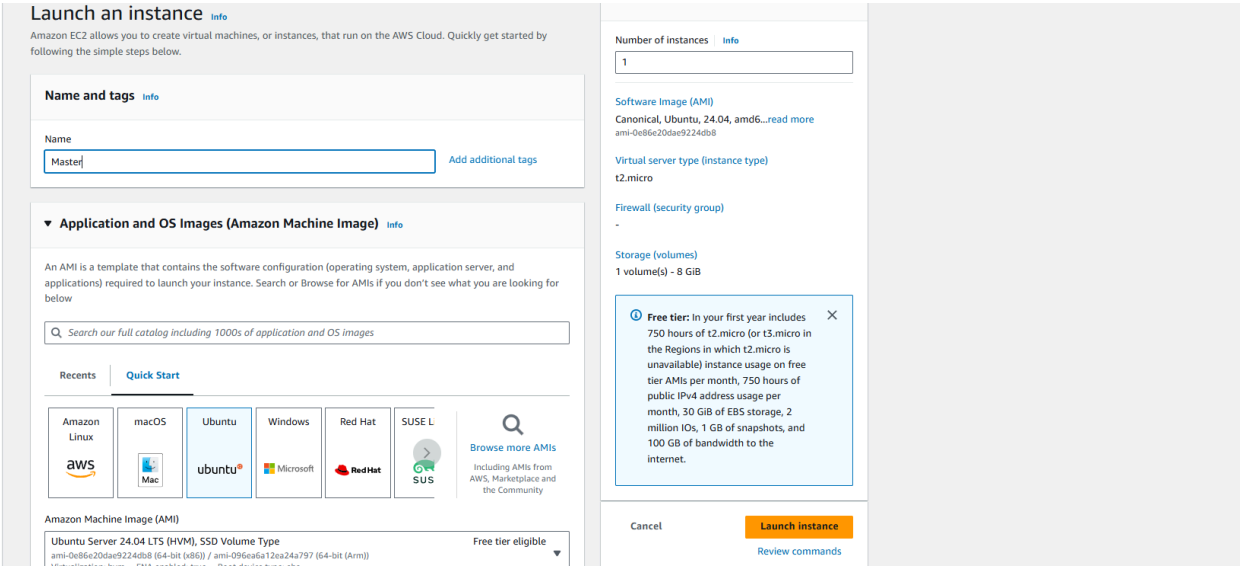
MASTER:

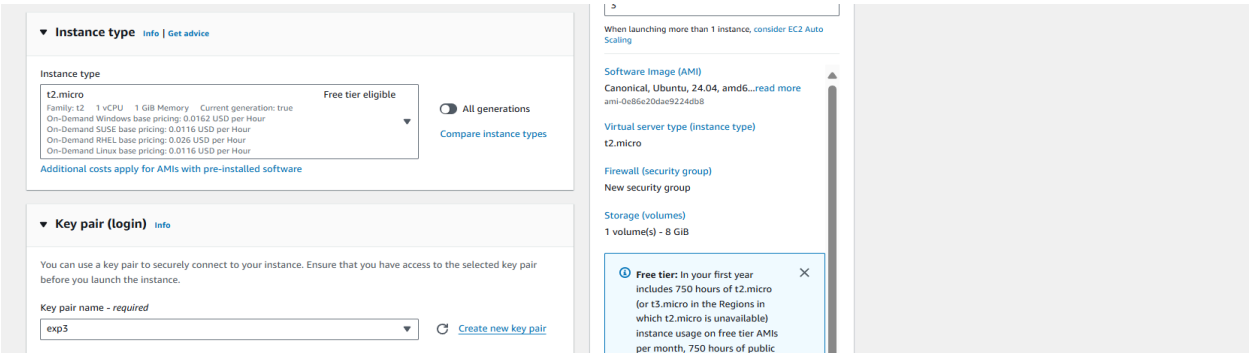


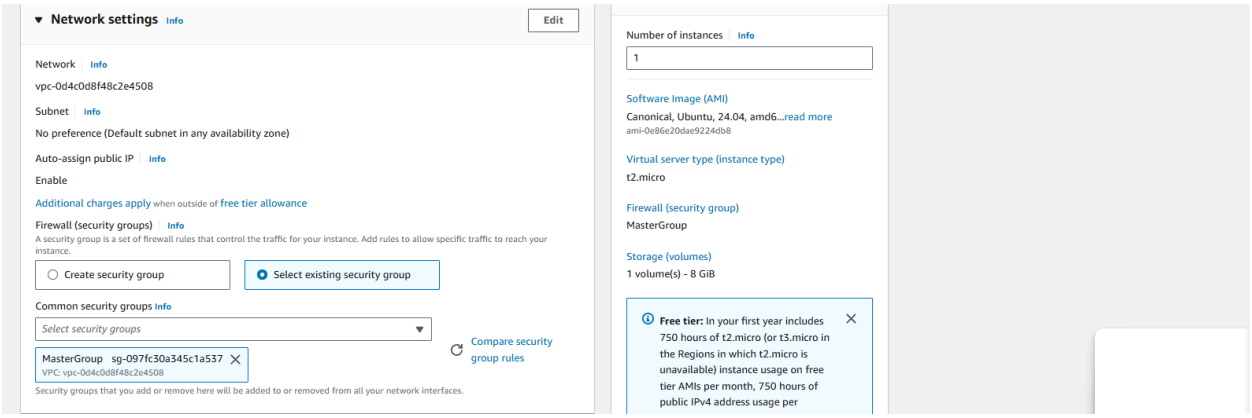
NODE:



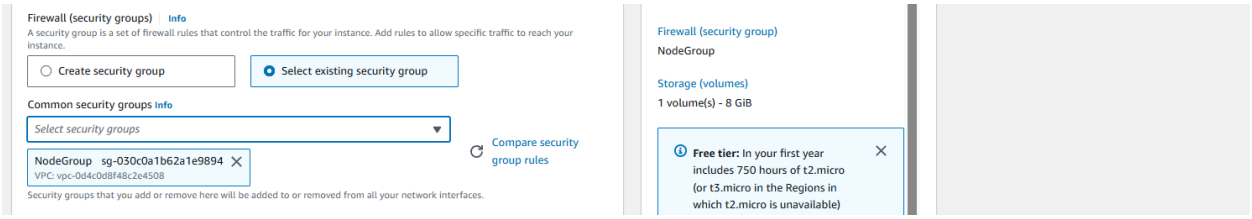
Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.micro (because in academic account only t2.micro is present) as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common key also.



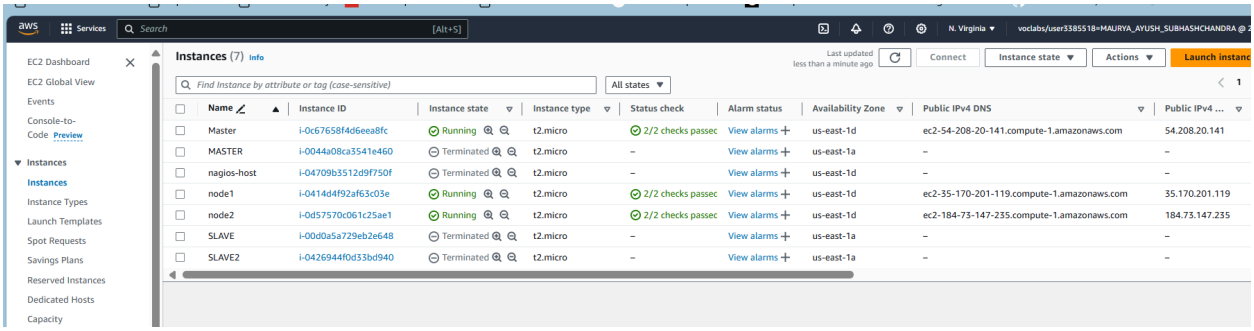




Do Same for 2 Nodes and use security groups of Node for that.



Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH Client.



Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i .....) in the terminal.
ssh -i "<PATH TO FILE>exp3.pem" ubuntu@ec2-54-208-20-141.compute-1.amazonaws.com

MASTER:

Connect to your instance i-0c67658f4d6eea8fc (Master) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-0c67658f4d6eea8fc (Master)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp3.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "exp3.pem"
4. Connect to your instance using its Public DNS:
   ec2-54-208-20-141.compute-1.amazonaws.com

Example:
ssh -i "exp3.pem" ubuntu@ec2-54-208-20-141.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

NODE 1&2:

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-0414d4f92af63c03e (node1)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp3.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "exp3.pem"
4. Connect to your instance using its Public DNS:
   ec2-35-170-201-119.compute-1.amazonaws.com

Example:
ssh -i "exp3.pem" ubuntu@ec2-35-170-201-119.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-0d57570c061c25ae1 (node2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp3.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "exp3.pem"
4. Connect to your instance using its Public DNS:
   ec2-184-73-147-235.compute-1.amazonaws.com

Example:
ssh -i "exp3.pem" ubuntu@ec2-184-73-147-235.compute-1.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

```
ubuntu@ip-172-31-82-192: ~
s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-208-20-141.compute-1.amazonaws.com' (ED25
519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Sep 26 15:21:26 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.82.192
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-82-192:~$
```

```
ubuntu@ip-172-31-86-84: ~
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-86-84:~$
```

```
ubuntu@ip-172-31-86-8: ~
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-86-8:~$
```

Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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-172-31-82-192:~$ curl -fsSL https://download.docker.com/linux/ubun
tu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instea
d (see apt-key(8)).
OK
```

```
ubuntu@ip-172-31-82-192:~$ curl -fsSL https://download.docker.com/linux/ubun
tu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-82-192:~$ sudo add-apt-repository "deb [arch=amd64] https:/
/download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble
 stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docke
r_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_d
ownload_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
 [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelea
se [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Pa
```

```
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 Components [212 B]
Get:51 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 c-n-f Metadata [116 B]
Get:52 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages
[15.3 kB]
Fetched 29.1 MB in 6s (4873 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s
) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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 InRelea
se
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: The key(s
) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON 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
r/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-172-31-82-192:~$
```

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

```
No VM guests are running outdated hypervisor (qemu) binaries on this ho
ubuntu@ip-172-31-82-192:~$ 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-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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-172-31-82-192:~$
```

Step 5: Run the below command to install Kubernets.

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
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-172-31-82-192:~$ # Add the Kubernetes GPG key and save it to the K
eyring
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gp
g --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# Add the Kubernetes repository to your APT sources list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://p
kgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/ku
bernetes.list

# Update package lists
sudo apt update
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N)
 y
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8
s.io/core:/stable:/v1.31/deb/ /
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 InRelea
se
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  Packages [4865 B]
Fetched 6051 B in 1s (6276 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
142 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s
) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-82-192:~$
```

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-82-192:~$ 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 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 InRelea
se
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/v1.31/deb  InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s
) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON 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:
  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 142 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 conntr
ack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
table:/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:/s
table:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/s
Processing triggers for man-db (2.12.0-4build2) ...
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.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-82-192:~$
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

```
ubuntu@ip-172-31-82-192:~$ sudo systemctl enable --now kubelet
sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer requir
ed:
  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 additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd6
```

```
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
```

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

```
ubuntu@ip-172-31-82-192:~$ 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 = ""
```

```
[proxy_plugins]

[stream_processors]

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_ke
yprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar"

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_ke
yprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar+gzip"

[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
ubuntu@ip-172-31-82-192:~$
```

sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd

```
ubuntu@ip-172-31-82-192:~$ 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; p>
     Active: active (running) since Thu 2024-09-26 16:19:13 UTC; 499ms ago
       Docs: https://containerd.io
   Main PID: 5387 (containerd)
      Tasks: 6
     Memory: 15.0M (peak: 15.4M)
        CPU: 78ms
     CGroup: /system.slice/containerd.service
             └─5387 /usr/bin/containerd

Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
Sep 26 16:19:13 ip-172-31-82-192 systemd[1]: Started containerd.service - c>
Sep 26 16:19:13 ip-172-31-82-192 containerd[5387]: time="2024-09-26T16:19:1>
lines 1-21/21 (END)
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-82-192:~$ 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 requir
ed:
  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 142 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 (15.7 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 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.

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-172-31-82-192:~$
```

Step 6: Initialize the Kubercluster. Now perform this on Master Instance.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-82-192:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[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 (957 MB) is less than the minimum 1700 MB
[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'
W0926 16:55:13.630110    7239 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by 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-172-31-82-192 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.82.192]
[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-172-31-82-192 localhost] and IPs [172.31.82.192 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-82-192 localhost] and IPs [172.31.82.192 127.0.0.1 ::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
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
[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 1.533243039s
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 11.505850128s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node ip-172-31-82-192 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node ip-172-31-82-192 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: jrhztc.eyi07duk03zq4eqr
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[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 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr \
        --discovery-token-ca-cert-hash sha256:a037b3c6608d5fdadd8dfd100793d17c759dbeb046a5ccd908a90f2e52d2055
```

From this command we get token and ca-

Run this command on master and

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-172-31-82-192:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-82-192:~$
```

Step 7: Now Run the command **kubectl get nodes** to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE     VERSION
ip-172-31-82-192  NotReady   control-plane   7m21s   v1.31.1
ubuntu@ip-172-31-82-192:~$
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.

sudo kubeadm join <your-master-node-ip>:6443 --token <your-token>
--discovery-token-ca-cert-hash sha256:<your-ca-cert-hash>

*kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr *
--discovery-token-ca-cert-hash
sha256:a6037b3c6608d5fdadd8dfd100793d17c759dbeb046a5ccd908a90f2e52d2055
(SLASH '\' MIGHT GIVE ERROR IF IT GIVES ERROR THEN TRY WITHOUT ERROR)

NODE1:

```
ubuntu@ip-172-31-86-84:~$ sudo kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr --discovery-token-ca-cert-hash sha256:a6037b3c6608d5fdadd8dfd
100793d17c759dbeb046a5ccd908a90f2e52d2055
[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 1.004592587s
[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.

ubuntu@ip-172-31-86-84:~$
```

NODE2:

```
ubuntu@ip-172-31-86-8:~$ sudo kubeadm join 172.31.82.192:6443 --token jrhztc.eyi07duk03zq4eqr --discovery-token-ca-cert-hash sha256:a6037b3c6608d5fdadd8dfd1
00793d17c759dbeb046a5ccd908a90f2e52d2055
[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 1.003588565s
[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: Now Run the command on Master **kubectl get nodes** to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-82-192   NotReady   control-plane   55m     v1.31.1
ip-172-31-86-8     NotReady   <none>          7m44s   v1.31.1
ip-172-31-86-84    NotReady   <none>          7m5s    v1.31.1
ubuntu@ip-172-31-82-192:~$
```

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

```
ubuntu@ip-172-31-82-192:~$ 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-172-31-82-192:~$
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-82-192:~$ 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 Thu 2024-09-26 16:55:46 UTC; 57min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 7822 (kubelet)
      Tasks: 10 (limit: 1130)
     Memory: 49.2M (peak: 72.1M)
        CPU: 44.924s
     CGroup: /system.slice/kubelet.service
             └─7822 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kube>

Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.304654    7822 kuberuntime_>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         rpc error: code = Unknown desc = f>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         : unknown
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:  > pod="kube-system/kube-scheduler-ip-172->
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393676    7822 log.go:32] "R>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         rpc error: code = Unknown desc = f>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:         : unknown
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]:  > podSandboxID="af90c21fb06b79773120a5ffc>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393747    7822 kuberuntime_>
Sep 26 17:52:55 ip-172-31-82-192 kubelet[7822]: E0926 17:52:55.393869    7822 kubelet.go:1>
lines 1-23/23 (END)
```

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

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-82-192  Ready    control-plane   58m     v1.31.1   172.31.82.192   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-8    Ready    <none>          10m     v1.31.1   172.31.86.8     <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-84   Ready    <none>          9m42s   v1.31.1   172.31.86.84    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-82-192:~$
```

Now to Rename run this command

kubectl label node <node-ip> kubernetes.io/role=worker

Rename to Node 1:**kubectl label node ip-172-31-86-8 kubernetes.io/role=Node1**

Rename to Node 2:**kubectl label node ip-172-31-86-84 kubernetes.io/role=Node2**

```
ubuntu@ip-172-31-82-192:~$ kubectl label node ip-172-31-86-8 kubernetes.io/role=Node1
^[[A^[[Anode/ip-172-31-86-8 labeled
ubuntu@ip-172-31-82-192:~$ kubectl label node ip-172-31-86-84 kubernetes.io/role=Node2
node/ip-172-31-86-84 labeled
ubuntu@ip-172-31-82-192:~$
```

Step 11: Run command **kubectl get nodes -o wide** . And Hence we can see we have Successfully connected Node 1 and Node 2 to the Master.

```
ubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-82-192  Ready    control-plane   83m   v1.31.1   172.31.82.192   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-8    Ready    Node1           35m   v1.31.1   172.31.86.8     <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-86-84   Ready    Node2           34m   v1.31.1   172.31.86.84    <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
^[[Aubuntu@ip-172-31-82-192:~$ kubectl get nodes -o wide
```

Or run **kubectl get nodes**

```
^[[Aubuntu@ip-172-31-82-192kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-82-192  Ready    control-plane   85m   v1.31.1
ip-172-31-86-8    Ready    Node1           37m   v1.31.1
ip-172-31-86-84   Ready    Node2           36m   v1.31.1
ubuntu@ip-172-31-82-192:~$
```

Conclusion:

In this Advanced DevOps Lab experiment, we began by setting up three EC2 Ubuntu instances on AWS, designating one as the Master node and the others as Worker nodes. We then installed Docker and Kubernetes on all instances, ensuring Docker was properly configured. The Kubernetes cluster was initialized on the Master node, and the Flannel networking plugin was applied to facilitate communication between nodes. Finally, we joined the Worker nodes to the cluster using the provided token and hash, resulting in a fully operational Kubernetes cluster ready for managing and scaling containerized applications.