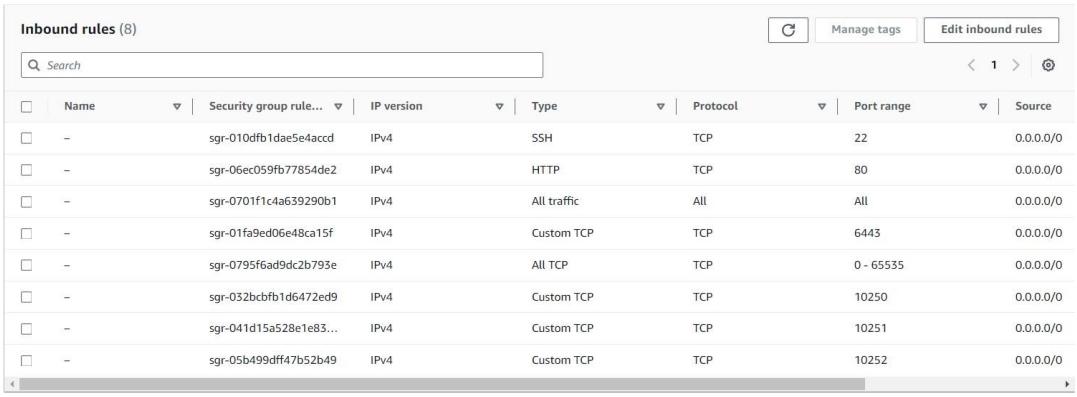
Master:



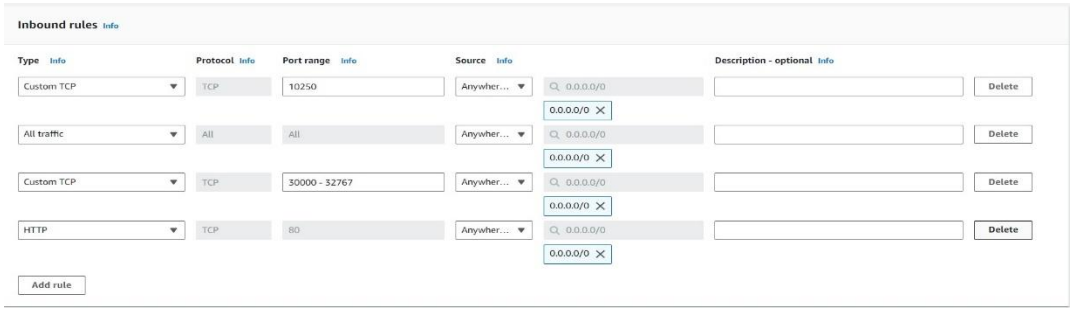
Inbound rules (8)

Manage tags | Edit inbound rules

Search

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-010dfb1dae5e4accd	IPv4	SSH	TCP	22	0.0.0.0/0
–	sgr-06ec059fb77854de2	IPv4	HTTP	TCP	80	0.0.0.0/0
–	sgr-0701f1c4a639290b1	IPv4	All traffic	All	All	0.0.0.0/0
–	sgr-01fa9ed06e48ca15f	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
–	sgr-0795f6ad9dc2b793e	IPv4	All TCP	TCP	0 - 65535	0.0.0.0/0
–	sgr-032bcbfb1d6472ed9	IPv4	Custom TCP	TCP	10250	0.0.0.0/0
–	sgr-041d15a528e1e83...	IPv4	Custom TCP	TCP	10251	0.0.0.0/0
–	sgr-05b499dff47b52b49	IPv4	Custom TCP	TCP	10252	0.0.0.0/0

Node :

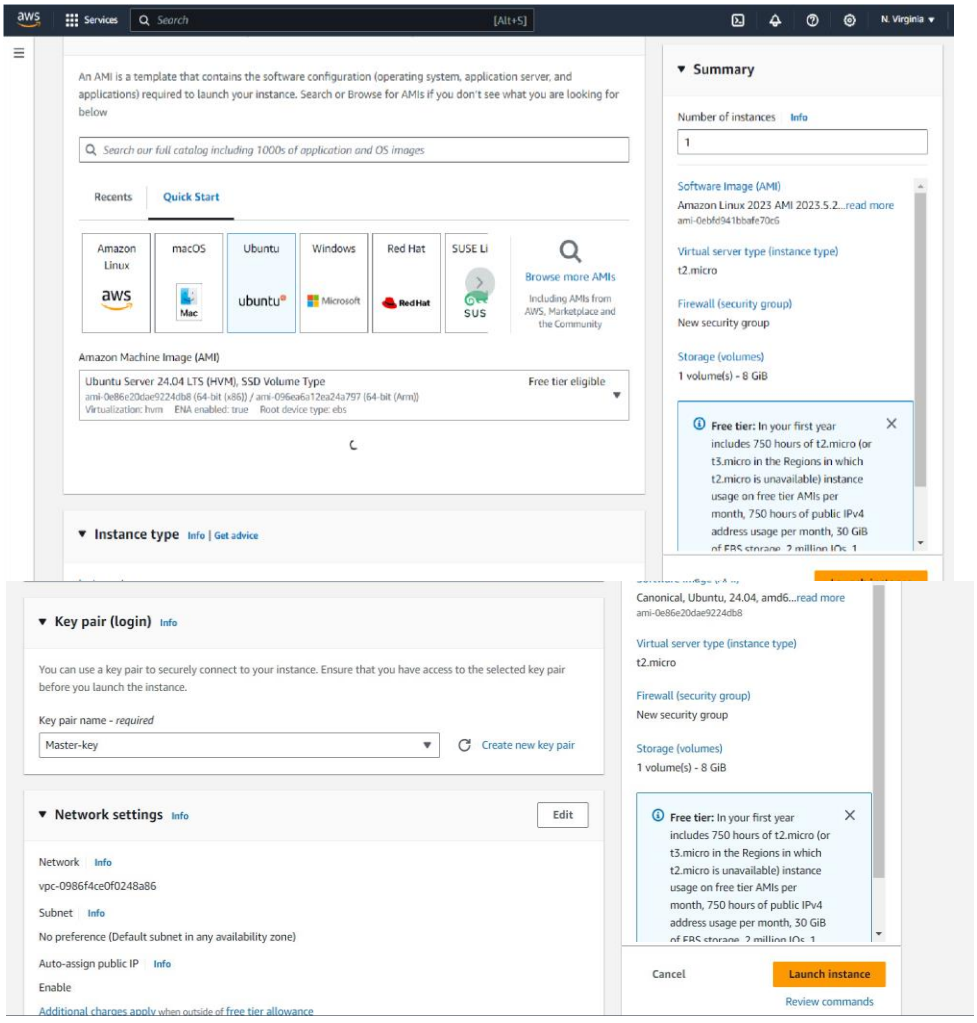


Inbound rules Info

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
Custom TCP	TCP	10250	Anywher... 0.0.0.0/0		Delete
All traffic	All	All	Anywher... 0.0.0.0/0		Delete
Custom TCP	TCP	30000 - 32767	Anywher... 0.0.0.0/0		Delete
HTTP	TCP	80	Anywher... 0.0.0.0/0		Delete

Add rule

Master:



aws | Services | Search [Alt+S] | N. Virginia

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

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

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible

ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

▼ Instance type Info | Get advice

▼ Summary

Number of instances Info: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...read more

ami-0ebfd941bbafe70c6

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

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

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*: Master-key | Create new key pair

▼ Network settings Info | Edit

Network Info: vpc-0986f4ce0f0248a86

Subnet Info: No preference (Default subnet in any availability zone)

Auto-assign public IP Info: Enable

Additional charges apply when outside of free tier allowance

Canonical, Ubuntu, 24.04, amd6...read more

ami-0e86e20dae9224db8

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

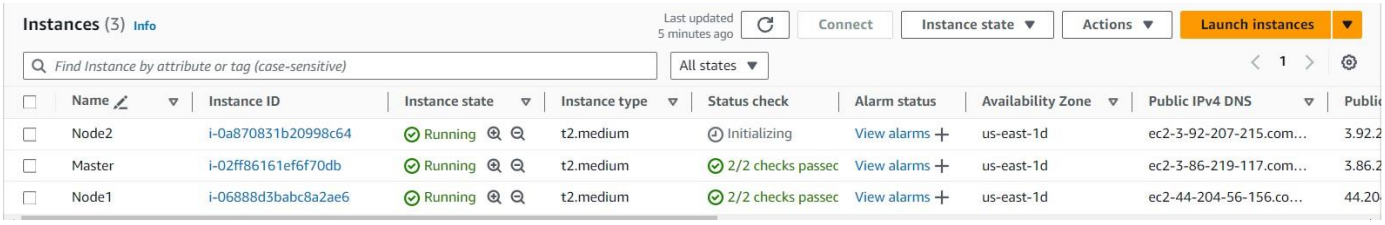
Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1

Cancel | Launch instance | Review commands

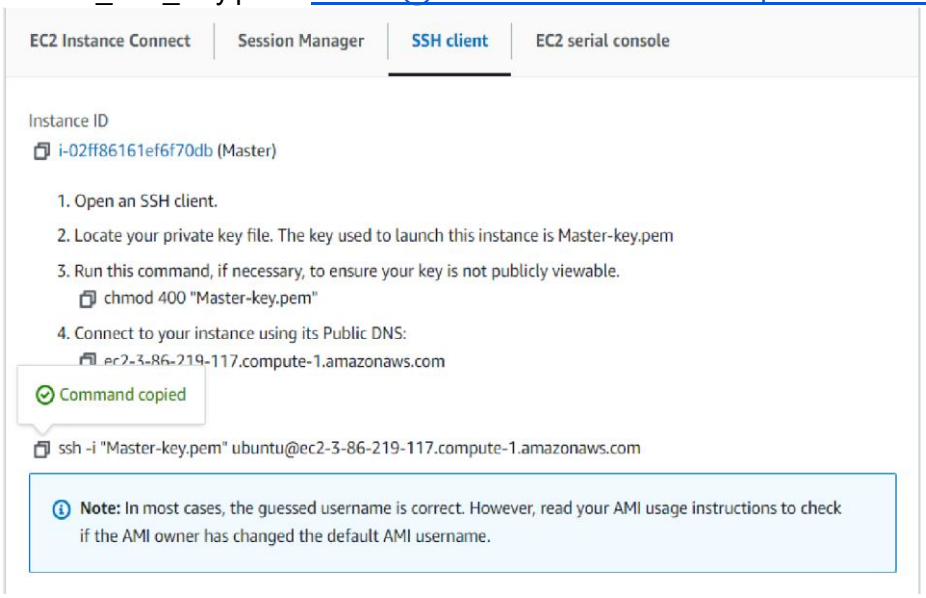
Do Same for 2 Nodes and use security groups of Node for that.

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

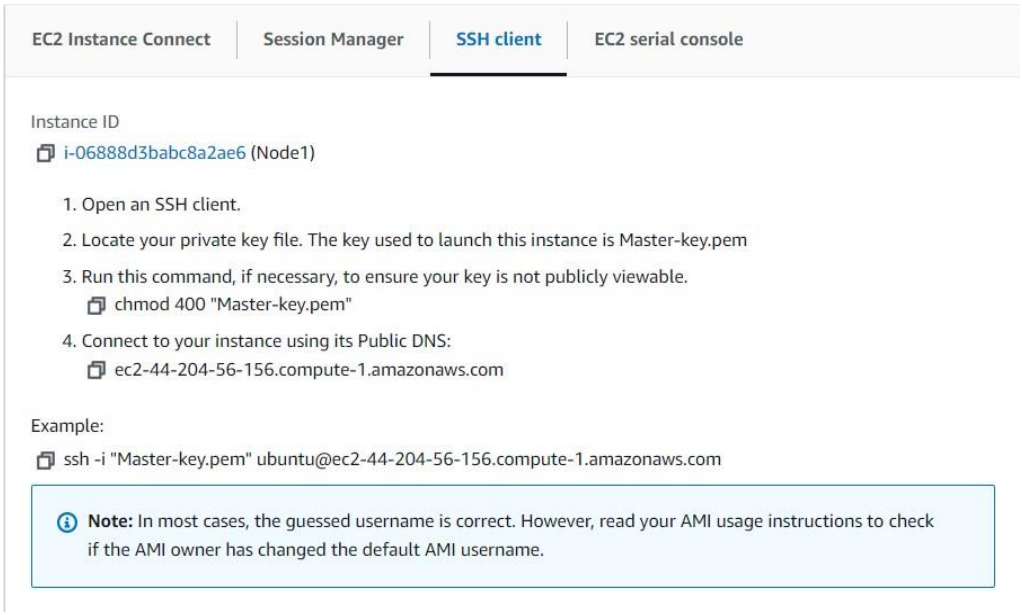


(Download Key)

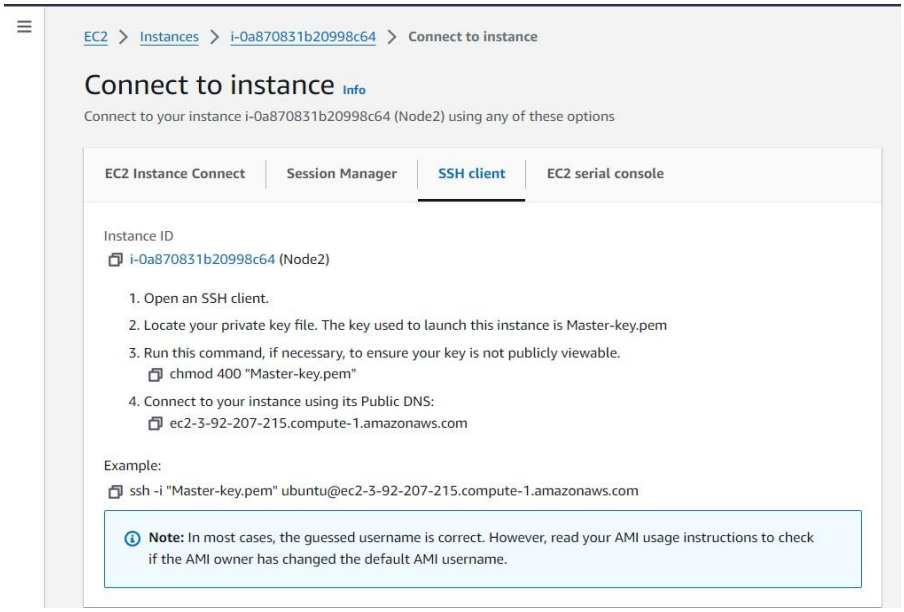
Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com) Master:



Node 1:

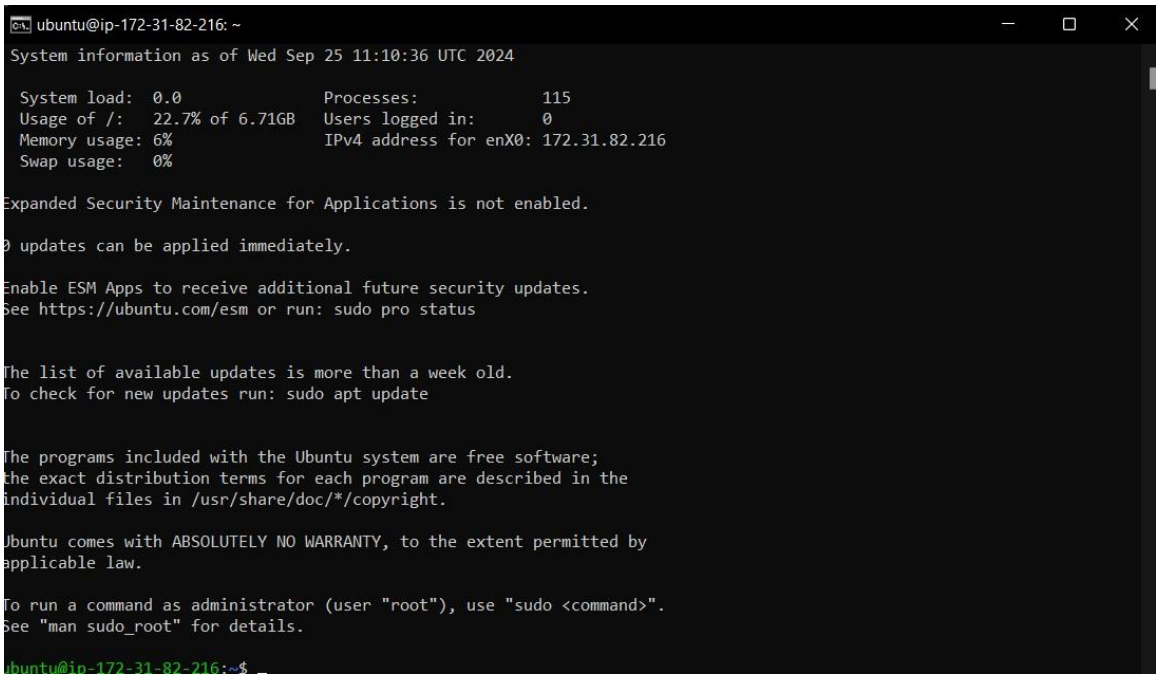


Node 2:



Here I have use 2 keys 1 for master and 1 for 2 node so I have to run open 3 terminals. In master key folder 1 terminal and 2 terminals in node 1 key folder.
If you use 1 Key only, you can open 3 terminal in one folder only.

Successful Connection:

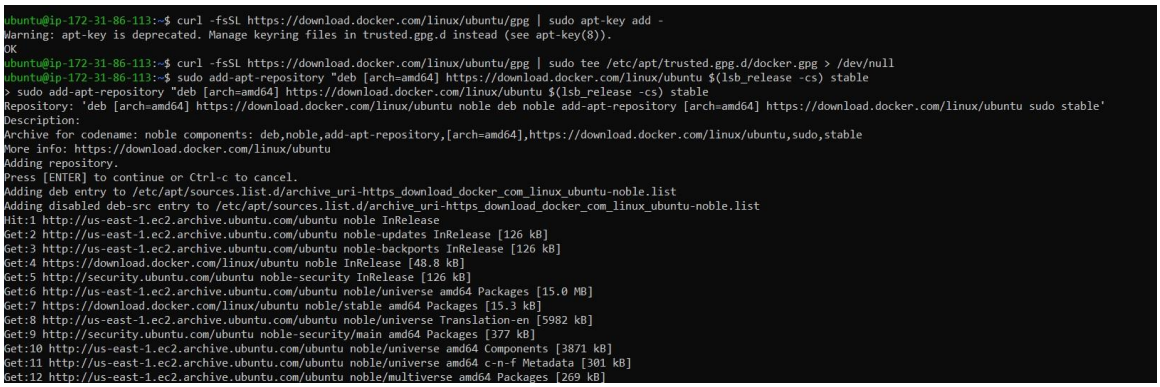


Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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" sudo



```
W: Skipping acquire of configured file 'sudo/cnf/Commands-amd64' as repository 'https://download.docker.com/linux/ubuntu noble
ubuntu@ip-172-31-86-113:~$ 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 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 a
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted
W: Skipping acquire of configured file 'deb/binary-amd64/Packages' as repository 'https://download.docker.com/linux/ubuntu noble
W: Skipping acquire of configured file 'deb/i18n/Translation-en' as repository 'https://download.docker.com/linux/ubuntu noble
W: Skipping acquire of configured file 'deb/dep11/Components-amd64.yml' as repository 'https://download.docker.com/linux/ubuntu
st?)
W: Skipping acquire of configured file 'deb/cnf/Commands-amd64' as repository 'https://download.docker.com/linux/ubuntu noble In
W: Skipping acquire of configured file 'noble/binary-amd64/Packages' as repository 'https://download.docker.com/linux/ubuntu nob
t?)
```

```
W: Skipping acquire of configured file 'sudo/cnf/Commands-amd64' as repository 'https://download.docker.com/linux/ubuntu noble InRelease' doesn't ha
ubuntu@ip-172-31-86-113:~$ sudo apt-get install -y docker-ce
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 139 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 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.24.04~noble [30.3 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.24.04~noble [15.0 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.24.04~noble [25.6 MB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:27.3.1-1~ubuntu.24.04~noble [9588 kB]
Get:10 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.29.7-1~ubuntu.24.04~noble [12.7 MB]
Fetched 123 MB in 2s (66.1 MB/s)
Selecting previously unselected package pigz.
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.serv
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.
```

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 host.
ubuntu@ip-172-31-86-113:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-86-113:~$ cat <<EOF | sudo tee /etc/docker/daemon.json{
> cat <<EOF | sudo tee /etc/docker/daemon.json{"exec-opts": ["native.cgroupdriver=systemd"]}
> EOF
cat <<EOF | sudo tee /etc/docker/daemon.json{"exec-opts": ["native.cgroupdriver=systemd"]}
ubuntu@ip-172-31-86-113:~$
```

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

```
ubuntu@ip-172-31-86-113:~$ sudo systemctl enable 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-172-31-86-113:~$ sudo sustemctl daemon-reload
sudo: sustemctl: command not found
ubuntu@ip-172-31-86-113:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-86-113:~$ sudo systemctl restart docker
ubuntu@ip-172-31-86-113:~$
```

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-172-31-86-113:~$ sudo systemctl restart docker
ubuntu@ip-172-31-86-113:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key |
> 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-172-31-86-113:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-86-113:~$
```

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

```
Last login: Wed Sep 25 11:02:35 2024 from 103.187.228.87
ubuntu@ip-172-31-86-113:~$ sudo apt-get update
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 (9934 B/s)
Reading package lists... Done
```

```
list?)
W: Skipping acquire of configured file 'sudo/cnf/Commands-amd64' as repository 'https://download.docker.com/linux/ubuntu noble InRelease'
ubuntu@ip-172-31-86-113:~$ sudo apt-get install -y kubelet kubeadm kubectl
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 139 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.
```

```
Setting up kubectl (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
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.
ubuntu@ip-172-31-86-113:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-86-113:~$
```

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

```
ubuntu@ip-172-31-86-113:~$ 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 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 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 139 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 amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (35.3 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68044 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb
```

sudo mkdir -p /etc/containerd

sudo containerd config default | sudo tee /etc/containerd/config.toml

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-86-113:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-86-113:~$ 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
```

```
  [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_keyprovider.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_keyprovider.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
```

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

```
ubuntu@ip-172-31-86-113:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-86-113:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-86-113:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-25 12:21:56 UTC; 22s ago
       Docs: https://containerd.io
   Main PID: 5564 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 14.0M)
        CPU: 147ms
     CGroup: /system.slice/containerd.service
             └─5564 /usr/bin/containerd

Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034051286Z" level=info msg=serving... address=/run/containerd/containerd.sock.t
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034058833Z" level=info msg="Start subscribing containerd event"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034138988Z" level=info msg="Start recovering state"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034189447Z" level=info msg="Start event monitor"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034204446Z" level=info msg="Start snapshots syncer"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034212888Z" level=info msg="Start cni network conf syncer for default"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034219382Z" level=info msg="Start streaming server"
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034106837Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 25 12:21:56 ip-172-31-86-113 containerd[5564]: time="2024-09-25T12:21:56.034331653Z" level=info msg="containerd successfully booted in 0.025705s"
Sep 25 12:21:56 ip-172-31-86-113 systemd[1]: Started containerd.service - containerd container runtime.
ubuntu@ip-172-31-86-113:~$
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-86-113:~$ 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 139 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 (13.9 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-172-31-86-113:~$
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-86-113:~$ 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'
W0925 12:23:38.236882    5795 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that
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-172-31-86-113 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
[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-172-31-86-113 localhost] and IPs [172.31.86.113 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-86-113 localhost] and IPs [172.31.86.113 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[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
[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 172.31.86.113:6443 --token td441j.fcvzp0pysi3k6i1a \
        --discovery-token-ca-cert-hash sha256:dda15d6076f45caff2b27dfb12d73e5bc7fa10b545c4330da9773df0007f5f2c
ubuntu@ip-172-31-86-113:~$
```

Run this command on master and also copy and save the Join command from above.

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

```
kubeadm join 172.31.86.113:6443 --token td441j.fcvzp0pysi3k6i1a \
        --discovery-token-ca-cert-hash sha256:dda15d6076f45caff2b27dfb12d73e5bc7fa10b545c4330da9773df0007f5f2c
ubuntu@ip-172-31-86-113:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-86-113:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-86-113:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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

```
ubuntu@ip-172-31-86-113:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-86-113   NotReady   control-plane   20m   v1.31.1
```

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

**sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 **
--discovery-token-ca-cert-hash

sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

Node 1:

```
ubuntu@ip-172-31-28-117:~$ sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
        --discovery-token-ca-cert-hash sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

[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 501.396793ms
[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-172-31-28-117:~$
```

Node 2:

```
ubuntu@ip-172-31-18-135:~$ sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
        --discovery-token-ca-cert-hash sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6
[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 1.001003808s
[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-172-31-18-135:~$
```

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

```
ubuntu@ip-172-31-27-176:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-18-135   NotReady   <none>          88s     v1.31.1
ip-172-31-27-176   NotReady   control-plane   10m     v1.31.1
ip-172-31-28-117   NotReady   <none>          2m58s   v1.31.1
```

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-172-31-27-176:~$ 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
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-86-113:~$ 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 Wed 2024-09-25 13:31:59 UTC; 11min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 853 (kubelet)
      Tasks: 12 (limit: 4676)
     Memory: 42.7M (peak: 43.7M)
        CPU: 9.350s
     CGroup: /system.slice/kubelet.service
             └─853 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --container-runtime-endpoint=u

Sep 25 13:42:38 ip-172-31-86-113 kubelet[853]:         rpc error: code = Unknown desc = failed to stop container "11e91c64db074d0c0fc3812656b9bd62f65db9c95b7107f966b9964a94ea08be": failed to kill container "1
Sep 25 13:42:38 ip-172-31-86-113 kubelet[853]:         : unknown
Sep 25 13:42:38 ip-172-31-86-113 kubelet[853]:  > podSandboxID="c8e5b58be88685dcf5129a2dccde2ed0f267e3e812abdb9f6609ab796c1aa8f9"
Sep 25 13:42:38 ip-172-31-86-113 kubelet[853]: E0925 13:42:38.167282     853 kuberuntime_manager.go:1479] "Failed to stop sandbox" podSandboxID={"Type":"containerd","ID":"c8e5b58be88685dcf5129a2dccde2ed0f267e
Sep 25 13:42:38 ip-172-31-86-113 kubelet[853]: E0925 13:42:38.167786     853 kubelet.go:1865] "KillPod failed" err="[failed to \"KillContainer\" for \"kube-apiserver\" with KillContainerError: \"rpc error: co
Sep 25 13:42:40 ip-172-31-86-113 kubelet[853]: E0925 13:42:40.289409     853 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady message:Networ
Sep 25 13:42:45 ip-172-31-86-113 kubelet[853]: E0925 13:42:45.290316     853 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady message:Networ
Sep 25 13:42:50 ip-172-31-86-113 kubelet[853]: E0925 13:42:50.291192     853 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady message:Networ
Sep 25 13:42:55 ip-172-31-86-113 kubelet[853]: E0925 13:42:55.291969     853 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady message:Networ
Sep 25 13:43:00 ip-172-31-86-113 kubelet[853]: E0925 13:43:00.292745     853 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkReady=false reason:NetworkPluginNotReady message:Networ
```

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

```
ubuntu@ip-172-31-27-176:~$ ubuntu@ip-172-31-27-176:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-18-135  Ready    <none>          6m19s   v1.31.1   172.31.18.135   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-27-176  Ready    control-plane   15m     v1.31.1   172.31.27.176   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-28-117  Ready    <none>          7m49s   v1.31.1   172.31.28.117   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
```

Now to Rename run this command kubectl label node ip-172-

31-18-135 kubernetes.io/role=worker Rename to Node

1:kubectl label node ip-172-31-28-117

kubernetes.io/role=Node1

Rename to Node 2:kubectl label node ip-172-31-18-135 kubernetes.io/role=Node2

```
ubuntu@ip-172-31-27-176:~$ kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1
node/ip-172-31-28-117 labeled
ubuntu@ip-172-31-27-176:~$ kubectl label node ip-172-31-18-135 kubernetes.io/role=Node2
node/ip-172-31-18-135 labeled
```

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-172-31-27-176:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES          AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-18-135  Ready    Node2          12m   v1.31.1   172.31.18.135   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-27-176  Ready    control-plane  21m   v1.31.1   172.31.27.176   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-28-117  Ready    Node1          13m   v1.31.1   172.31.28.117   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-27-176:~$
```

Or run kubectl get nodes

```
ubuntu@ip-172-31-27-176:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE   VERSION
ip-172-31-18-135    Ready    Node2           24m   v1.31.1
ip-172-31-27-176    Ready    control-plane   33m   v1.31.1
ip-172-31-28-117    Ready    Node1           25m   v1.31.1
ubuntu@ip-172-31-27-176:~$
```