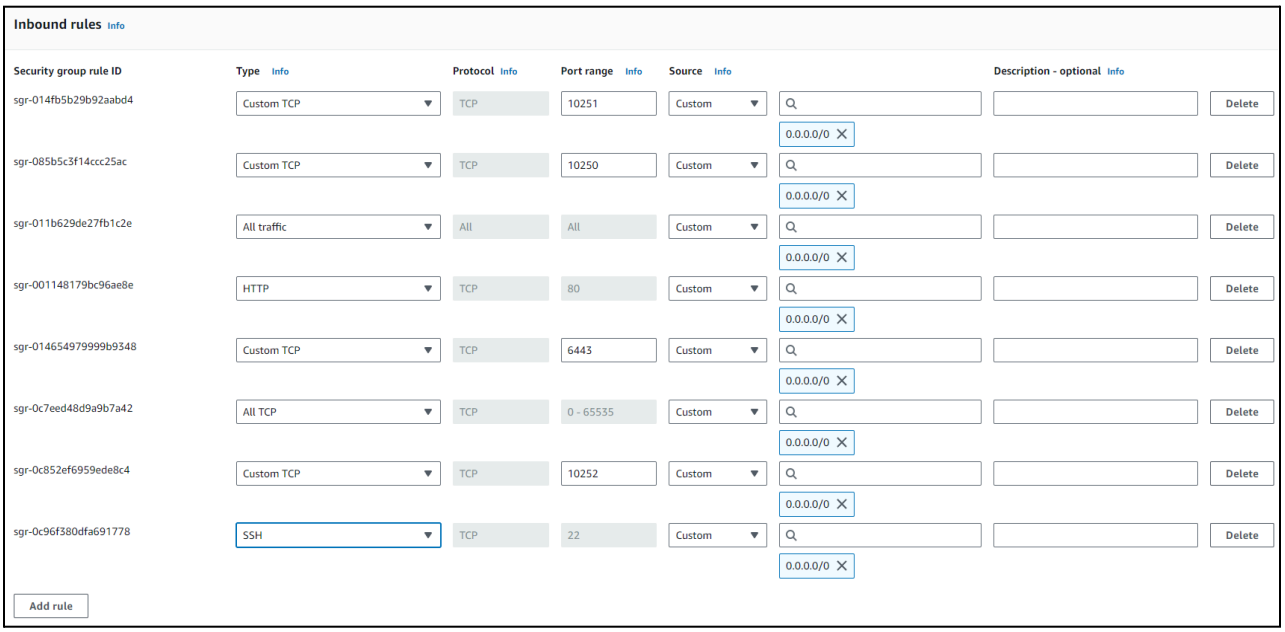
ADVANCE DEVOPS EXP-3

ANSH SARFARE **D15A/50**

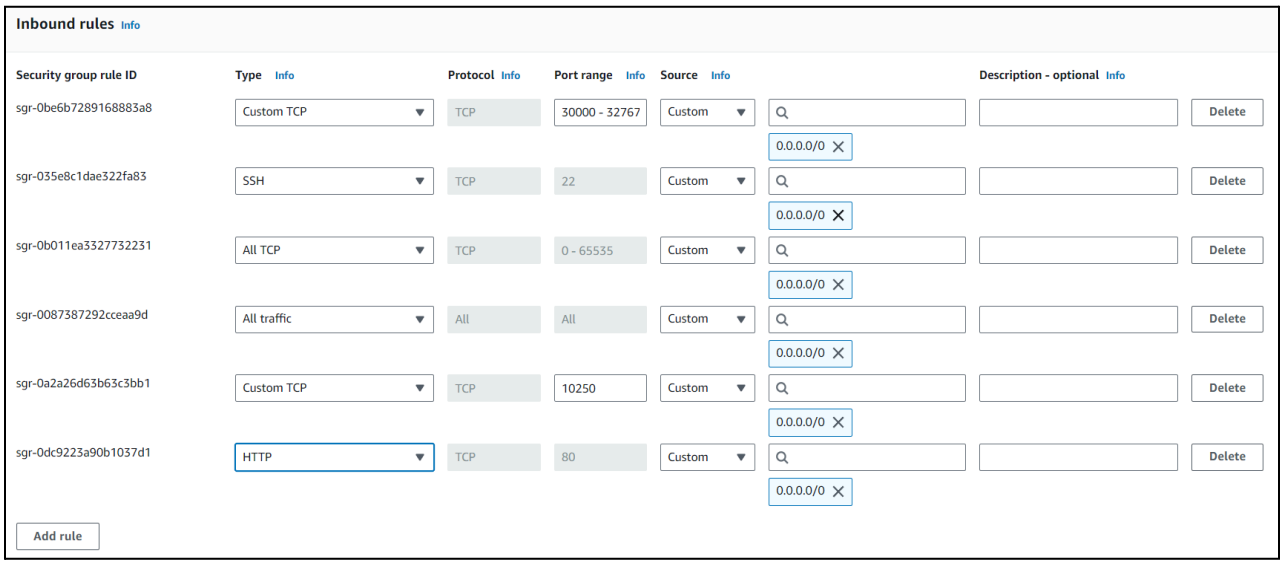
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Create 2 Security Groups for Master and Nodes and add the following inbound rules in those groups

Master:

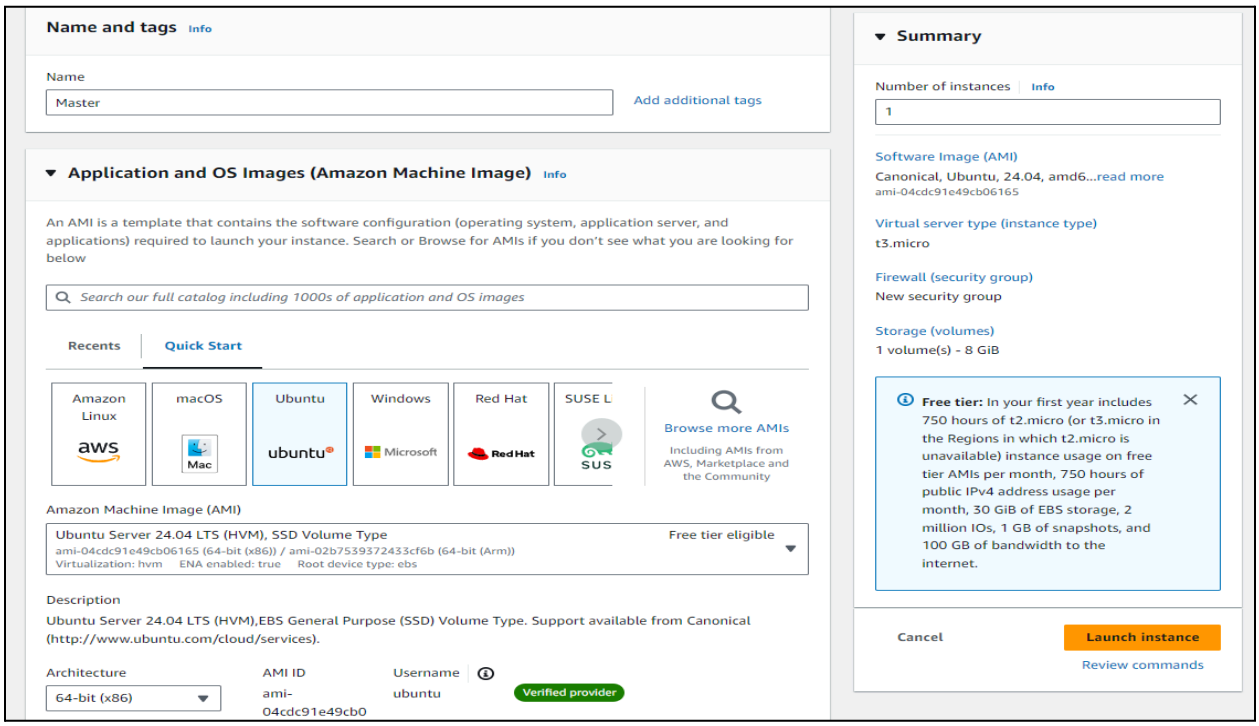


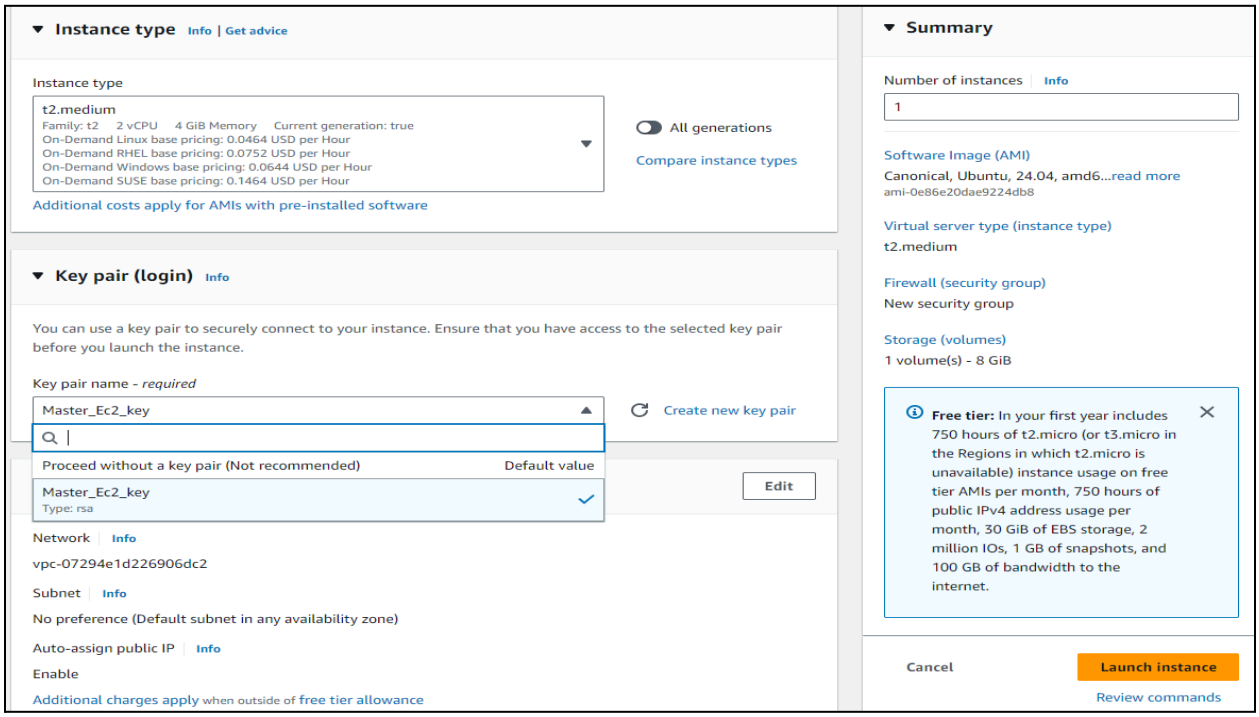
Node:

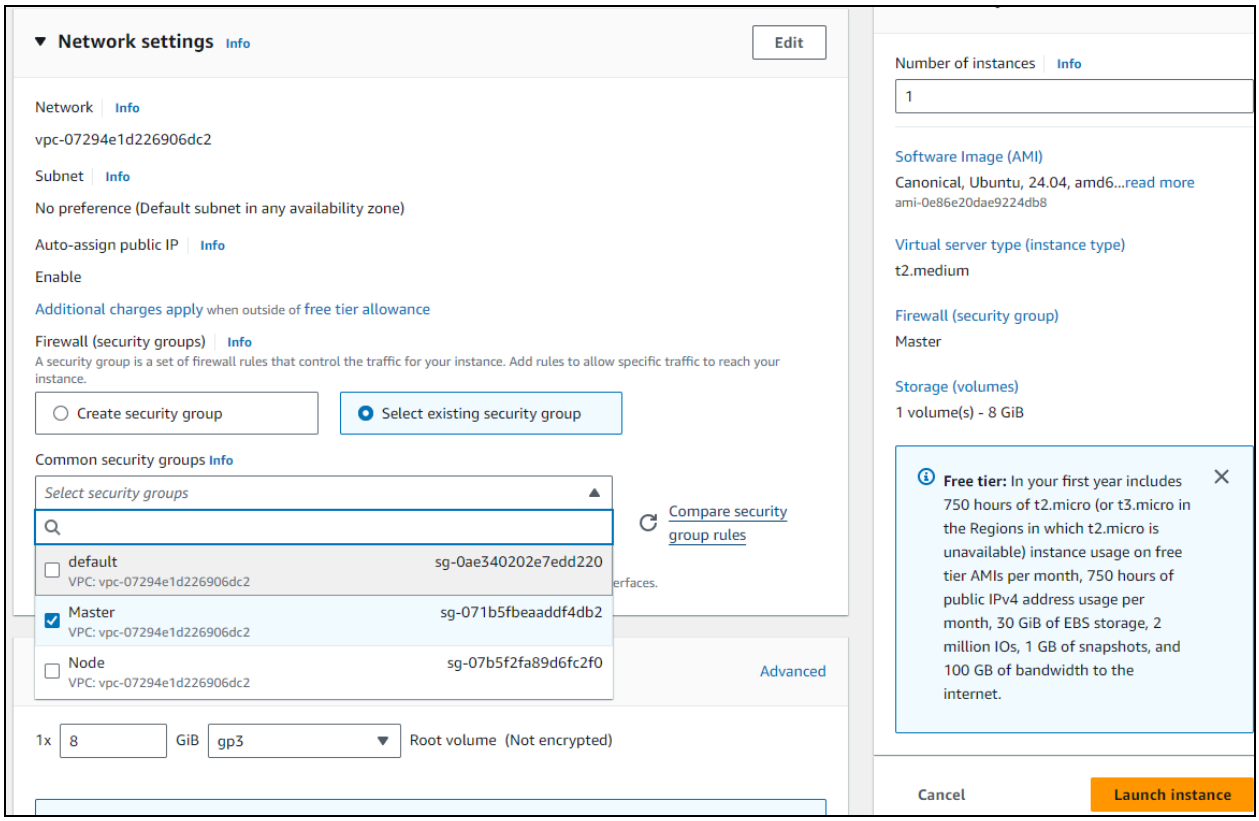


Step 2: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances(1 for Master and 2 for Node).Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 2 Different keys, 1 for Master and 1 for Node. Also Select Security Groups from the existing.

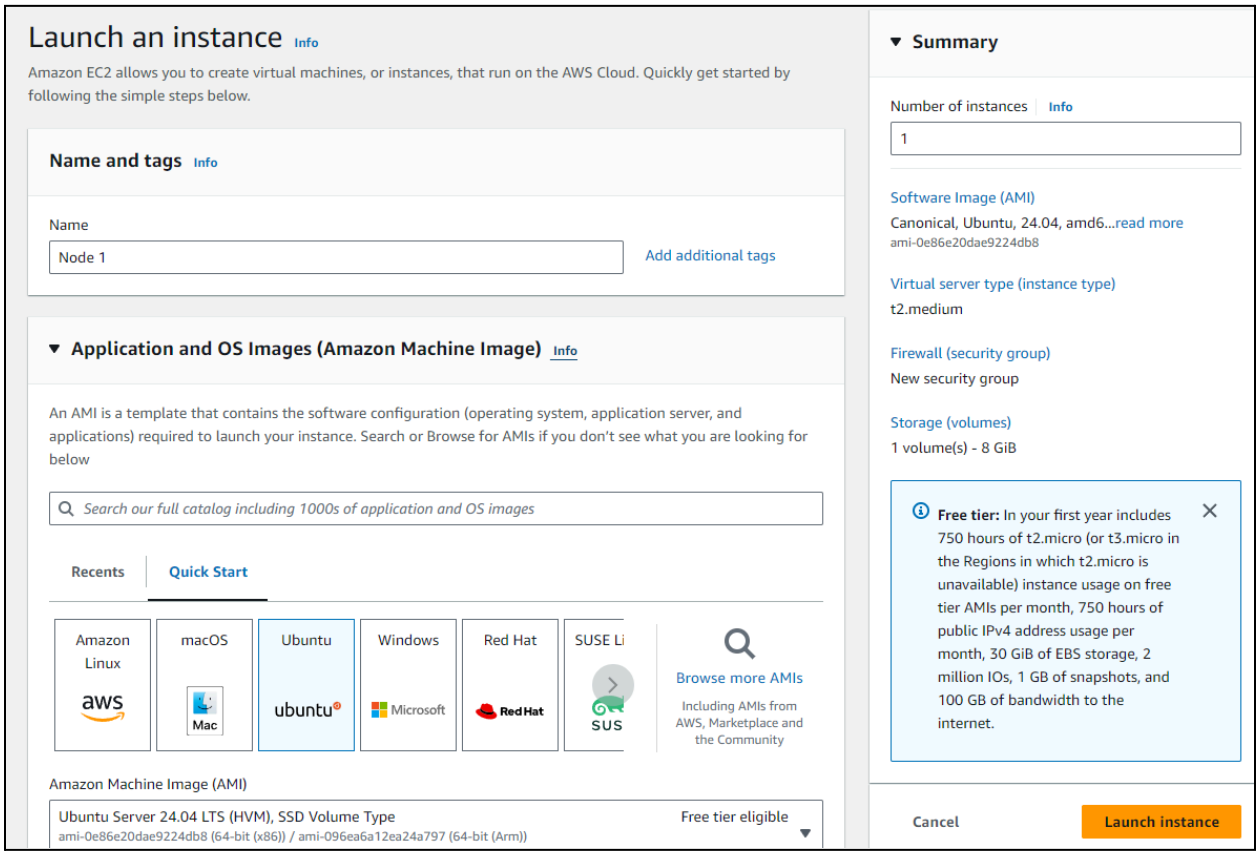
Master:

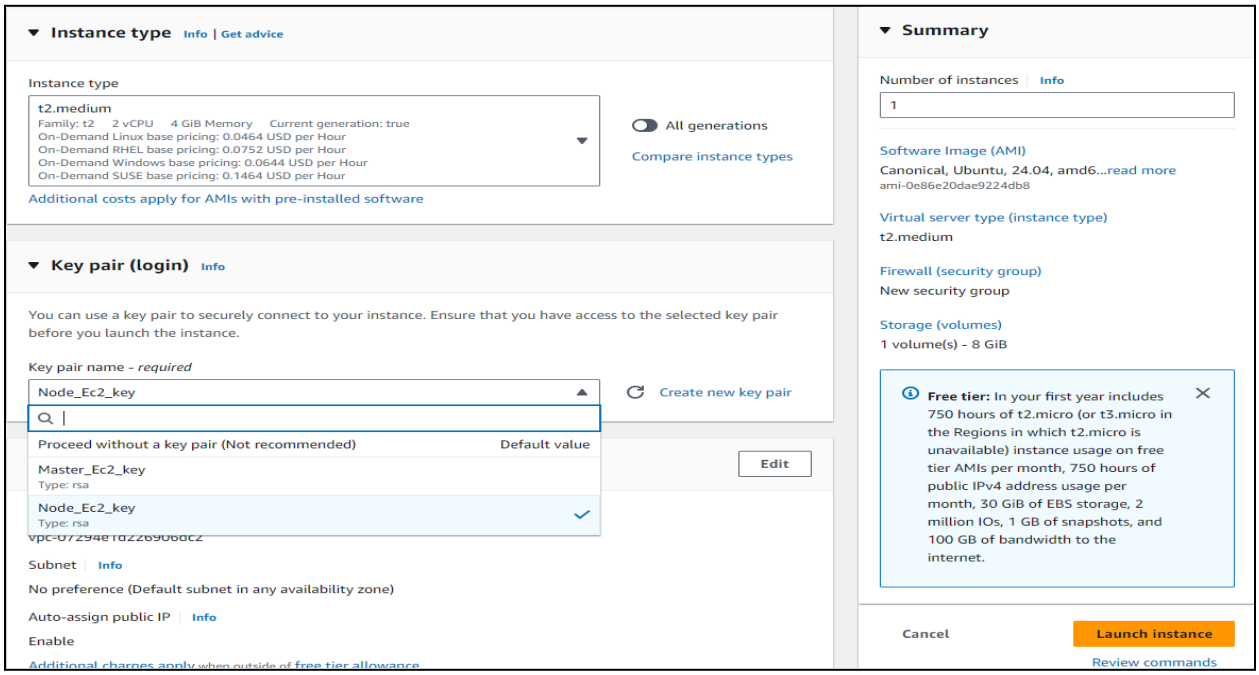


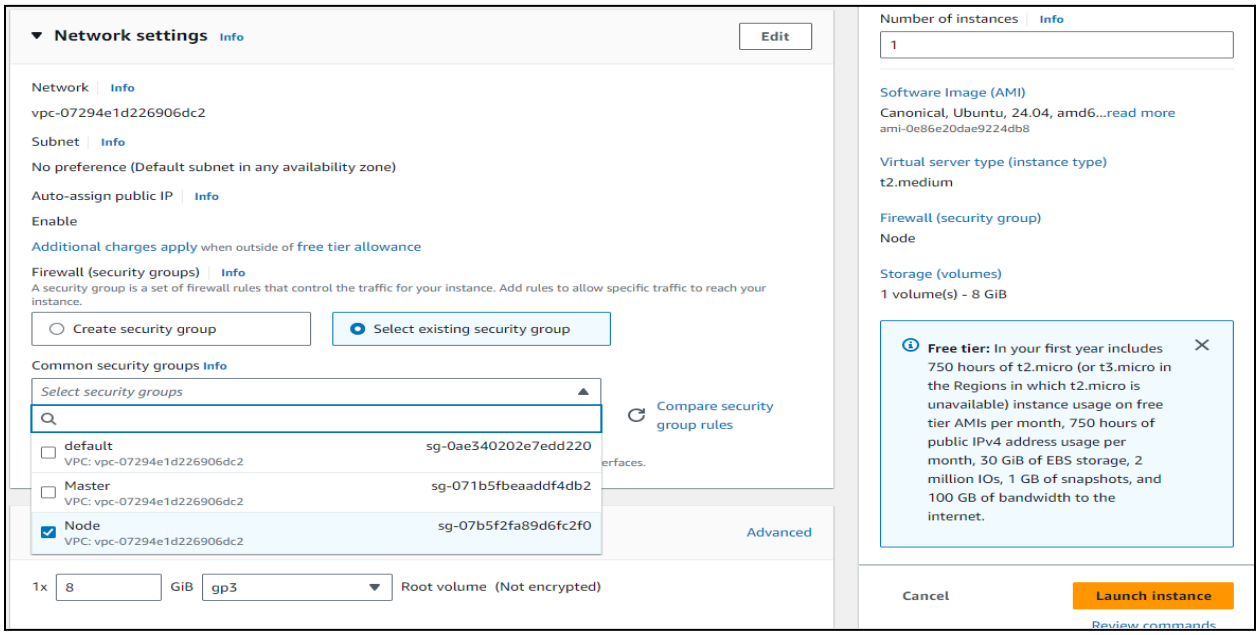


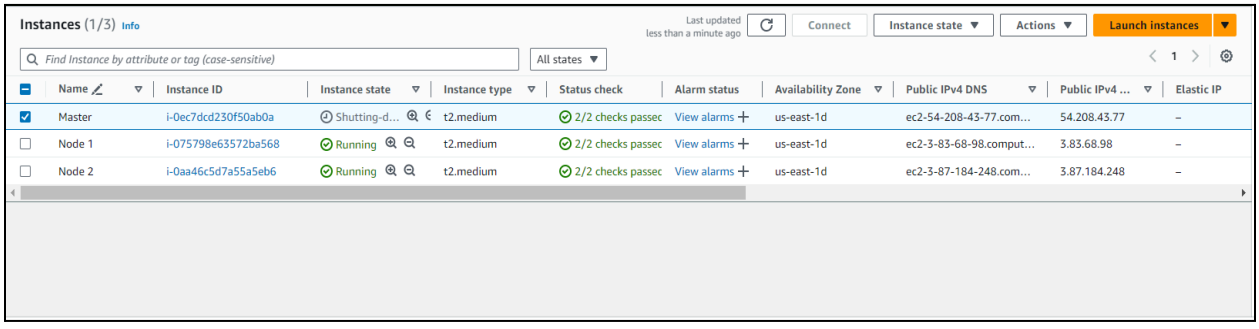

Network settings Info
Edit
Network Info
vpc-07294e1d226906dc2
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
default VPC: vpc-07294e1d226906dc2 sg-0ae340202e7edd220
Master VPC: vpc-07294e1d226906dc2 sg-071b5fbeaaddf4db2
Node VPC: vpc-07294e1d226906dc2 sg-07b5f2fa89d6fc2f0
Compare security group rules
Advanced
1x 8 GiB gp3 Root volume (Not encrypted)
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.medium
Firewall (security group)
Master
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 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance

Node:


Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
Node 1
Add additional tags
Application and OS Images (Amazon Machine Image) Info
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
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Free tier eligible
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.medium
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 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance

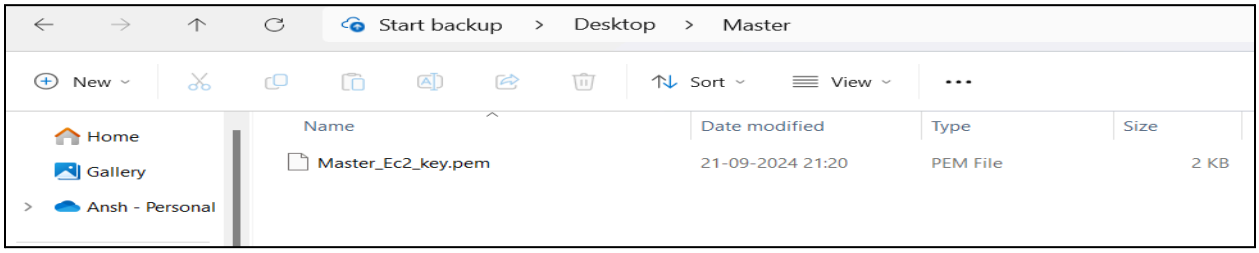

Instance type Info | Get advice
Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
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
Node_Ec2_key
Create new key pair
Proceed without a key pair (Not recommended) Default value
Master_Ec2_key
Type: rsa
Node_Ec2_key
Type: rsa
Edit
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
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.medium
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 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance
Review commands


Network settings Info
Edit
Network Info
vpc-07294e1d226906dc2
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
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
default
VPC: vpc-07294e1d226906dc2
sg-0ae340202e7edd220
Master
VPC: vpc-07294e1d226906dc2
sg-071b5fbeaaddf4db2
Node
VPC: vpc-07294e1d226906dc2
sg-07b5f2fa89d6fc2f0
Compare security group rules
Advanced
1x 8 GiB gp3 Root volume (Not encrypted)
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8
Virtual server type (instance type)
t2.medium
Firewall (security group)
Node
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 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance
Review commands

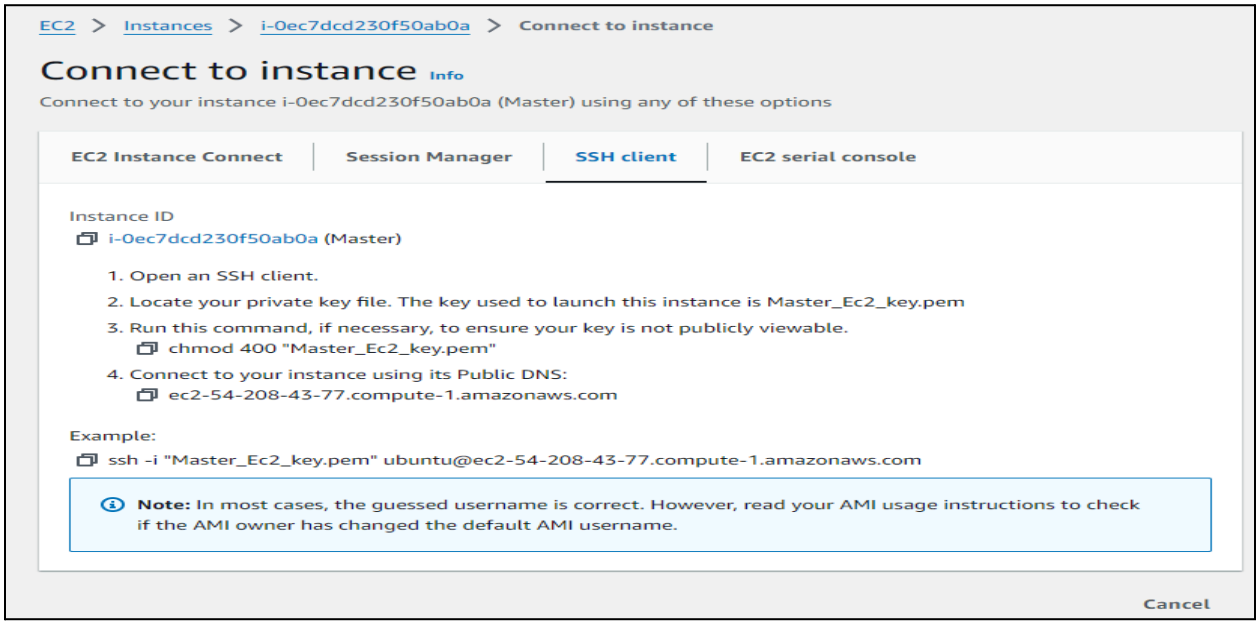

Instances (1/3) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elastic IP
Master i-0ec7dcd230f50ab0a Shutting-d... t2.medium 2/2 checks passec View alarms us-east-1d ec2-54-208-43-77.com... 54.208.43.77 –
Node 1 i-075798e63572ba568 Running t2.medium 2/2 checks passec View alarms us-east-1d ec2-3-83-68-98.comput... 3.83.68.98 –
Node 2 i-0aa46c5d7a55a5eb6 Running t2.medium 2/2 checks passec View alarms us-east-1d ec2-3-87-184-248.com... 3.87.184.248 –

Step 3: Connect the instance and navigate to SSH client and copy the example command. 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 from ssh client (starting with ssh -i .....) in the terminal.

Downloaded Key:

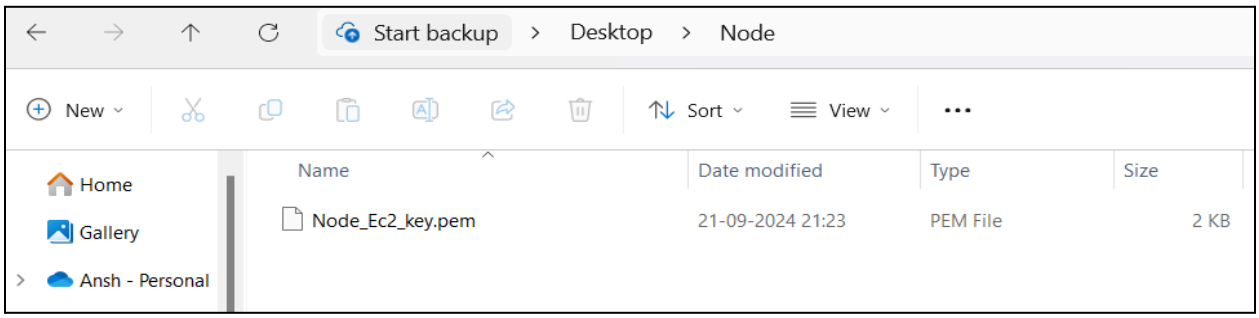


Master:

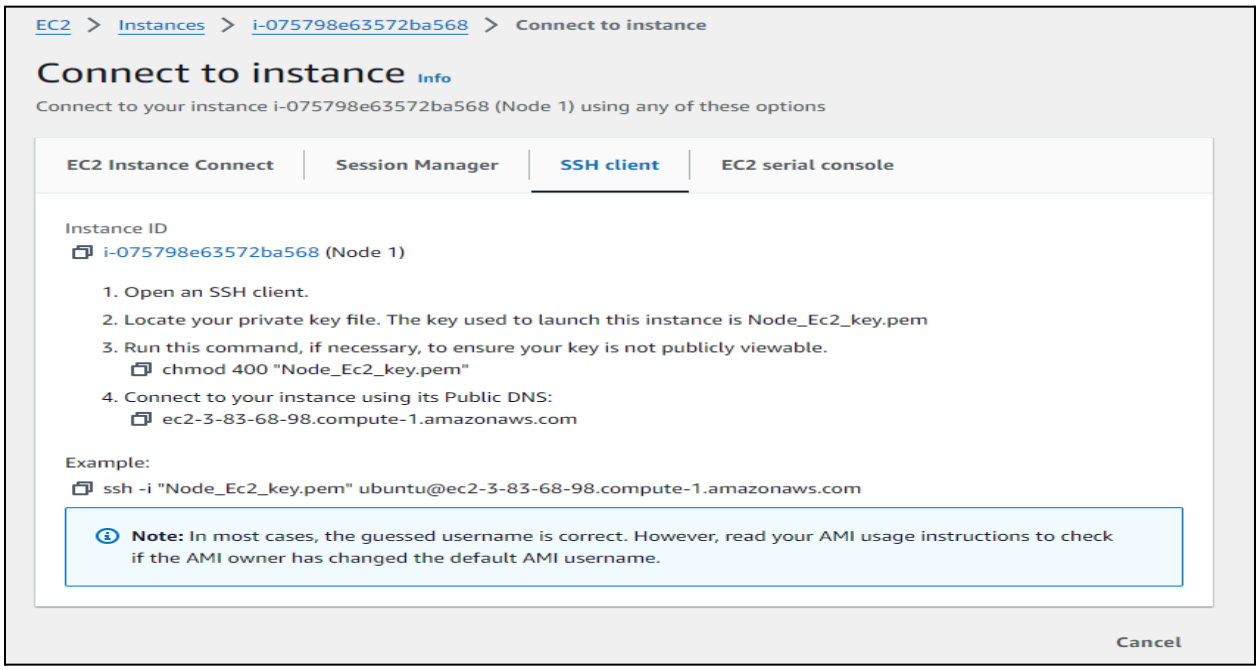



```
PS C:\Users\Ansh\Desktop\Master> ssh -i "Master_Ec2_key.pem" ubuntu@ec2-54-208-43-77.compute-1.amazonaws.com
The authenticity of host 'ec2-54-208-43-77.compute-1.amazonaws.com (54.208.43.77)' can't be established.
ED25519 key fingerprint is SHA256:GEXNsxYEo5wgzQPomUHsm+3oE1vRL4EAEjhm1lggTpU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-208-43-77.compute-1.amazonaws.com' (ED25519) 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 Sat Sep 21 16:47:07 UTC 2024

  System load:  0.0               Processes:             115
  Usage of /:   23.1% of 6.71GB   Users logged in:       0
  Memory usage: 7%                IPv4 address for enX0: 172.31.95.244
  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

Last login: Sat Sep 21 16:28:45 2024 from 18.206.107.28
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-95-244:~$
```

Downloaded Key:

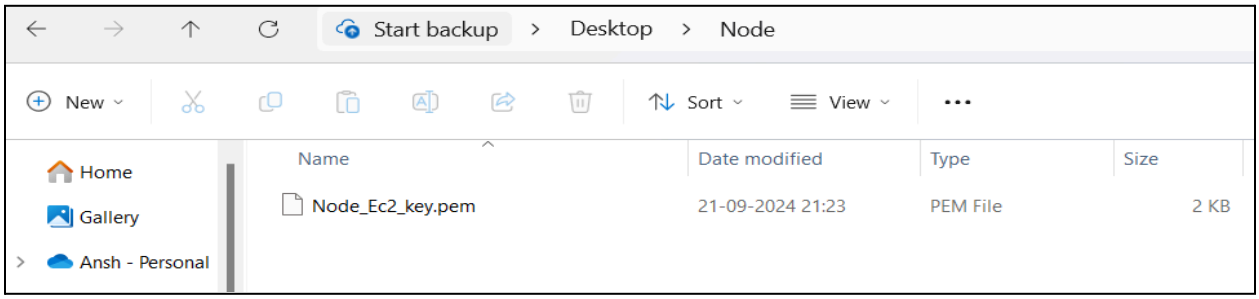


Node 1:

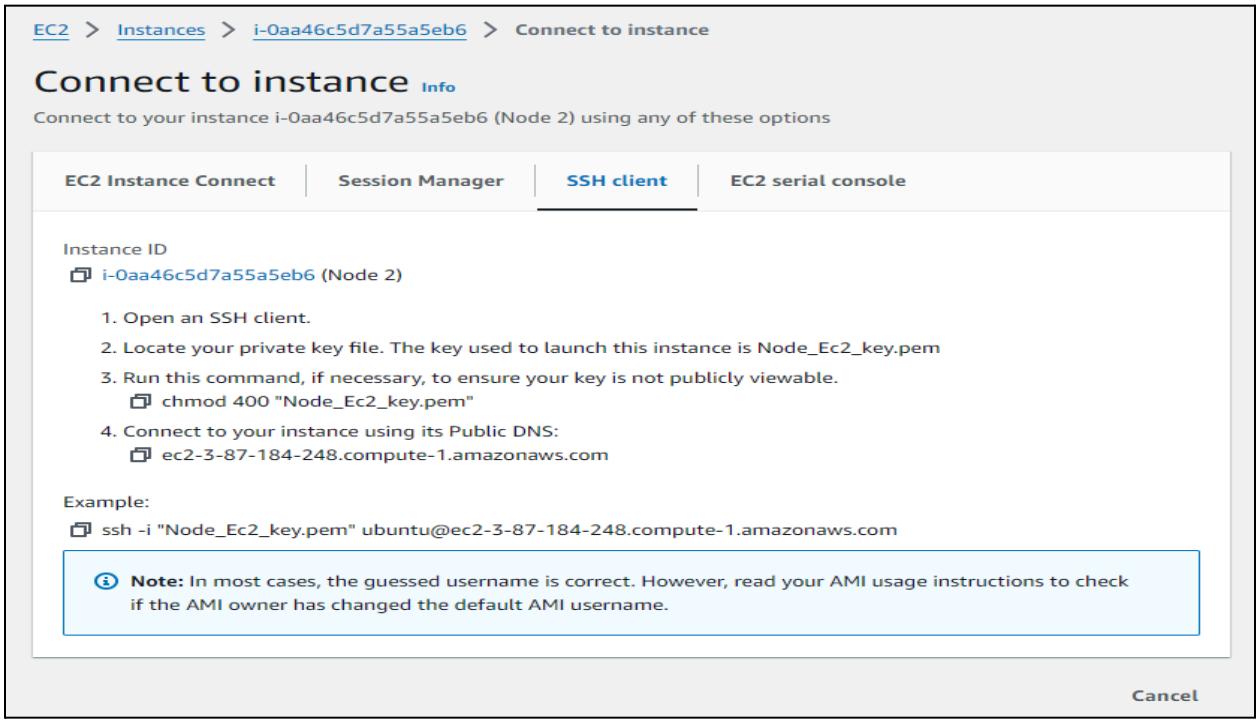



```
PS C:\Users\Ansh\Desktop\Node> ssh -i "Node_Ec2_key.pem" ubuntu@ec2-3-83-68-98.compute-1.amazonaws.com
The authenticity of host 'ec2-3-83-68-98.compute-1.amazonaws.com (3.83.68.98)' can't be established.
ED25519 key fingerprint is SHA256:21QMbe+vHlvpbqWK7g3/dY14ckA4LLH0mijbam4WIvQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-83-68-98.compute-1.amazonaws.com' (ED25519) 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 Sat Sep 21 17:31:03 UTC 2024

  System load:  0.0               Processes:             117
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 8%                IPv4 address for enX0: 172.31.84.209
  Swap usage:   0%
```

Downloaded Key:



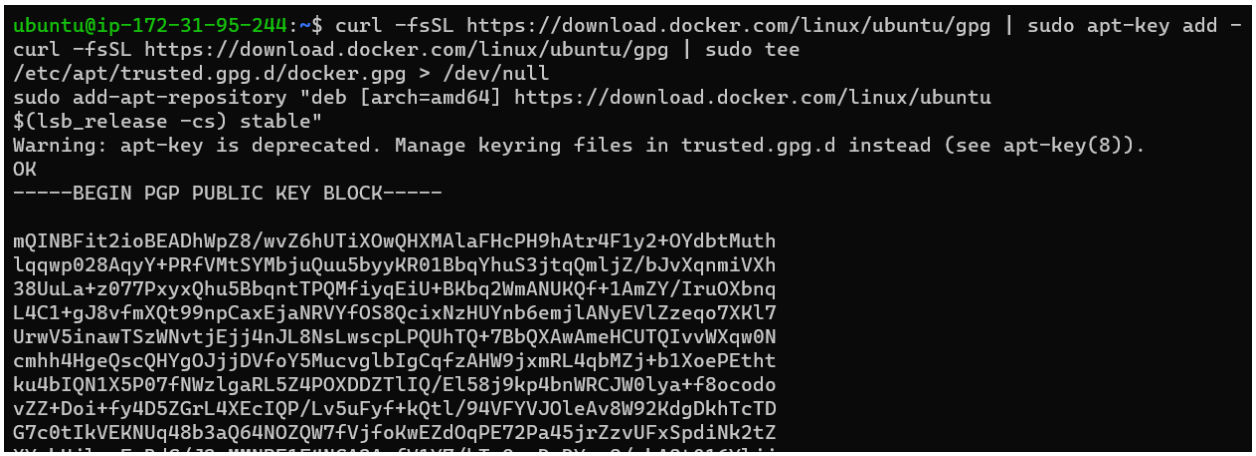
Node 2:

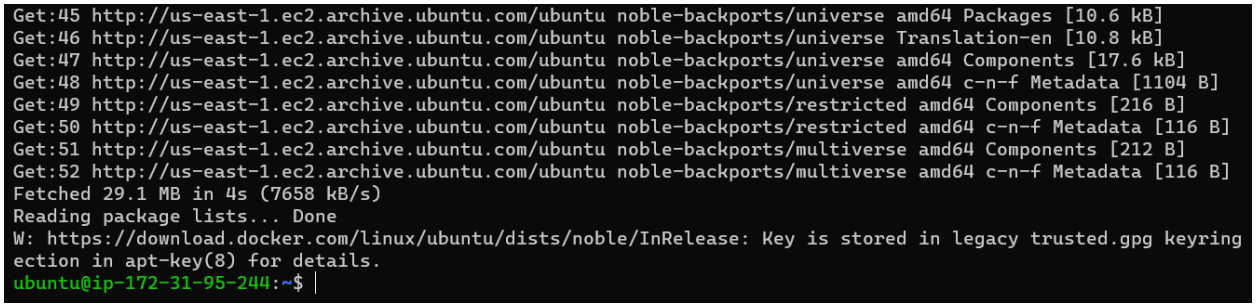



```
PS C:\Users\Ansh\Desktop\Node> ssh -i "Node_Ec2_key.pem" ubuntu@ec2-3-87-184-248.compute-1.amazonaws.com
The authenticity of host 'ec2-3-87-184-248.compute-1.amazonaws.com (3.87.184.248)' can't be established.
ED25519 key fingerprint is SHA256:cwvHL/f3y1isWTr/hU72bPMq6+a03thKQtCkQfII2gg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-87-184-248.compute-1.amazonaws.com' (ED25519) 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 Sat Sep 21 17:32:30 UTC 2024

  System load:  0.08              Processes:             113
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.87.15
  Swap usage:   0%
```

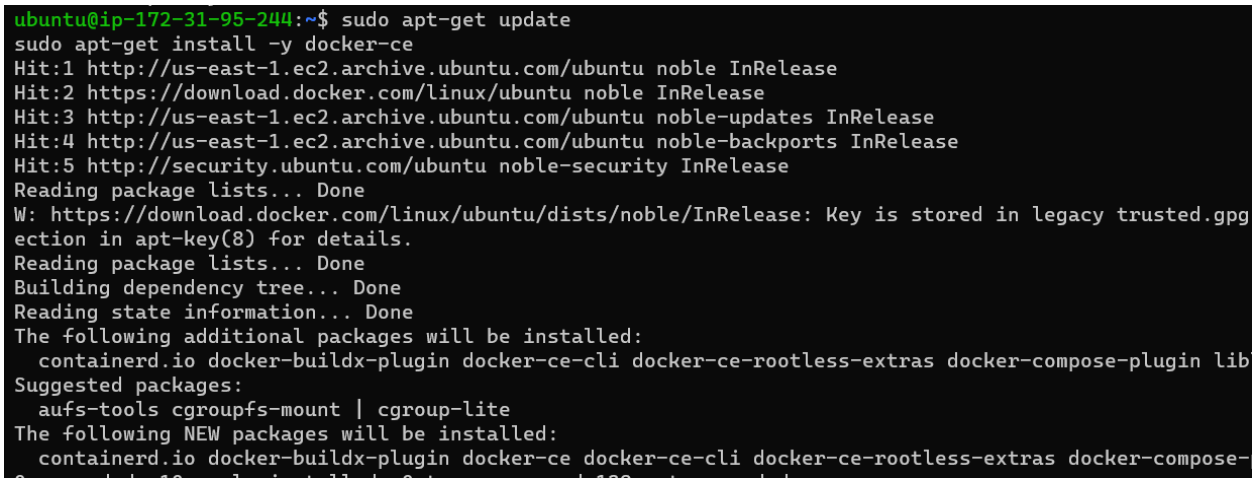
Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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



```
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-172-31-95-244:~$
```

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

```
ubuntu@ip-172-31-95-244:~$ 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-172-31-95-244:~$
```

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

```
ubuntu@ip-172-31-95-244:~$ 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-172-31-95-244:~$
```

Step 5: Run the below command to install Kubernetes.

- 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-172-31-95-244:~$ 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
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-95-244:~$
```

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

error:

```
ubuntu@ip-172-31-95-244:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
ubuntu@ip-172-31-95-244:~$
```

To solve the error:
Added **sudo mkdir -p /etc/apt/keyrings** in the previous command.

```
ubuntu@ip-172-31-95-244:~$ 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 InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble 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 (10.1 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/
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 139 not upgraded.
Need to get 87.4 MB of archives.
```

```
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-172-31-95-244:~$
```

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

```
ubuntu@ip-172-31-95-244:~$ 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 required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-pl
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 ru
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 co
Fetched 47.2 MB in 1s (76.6 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
```

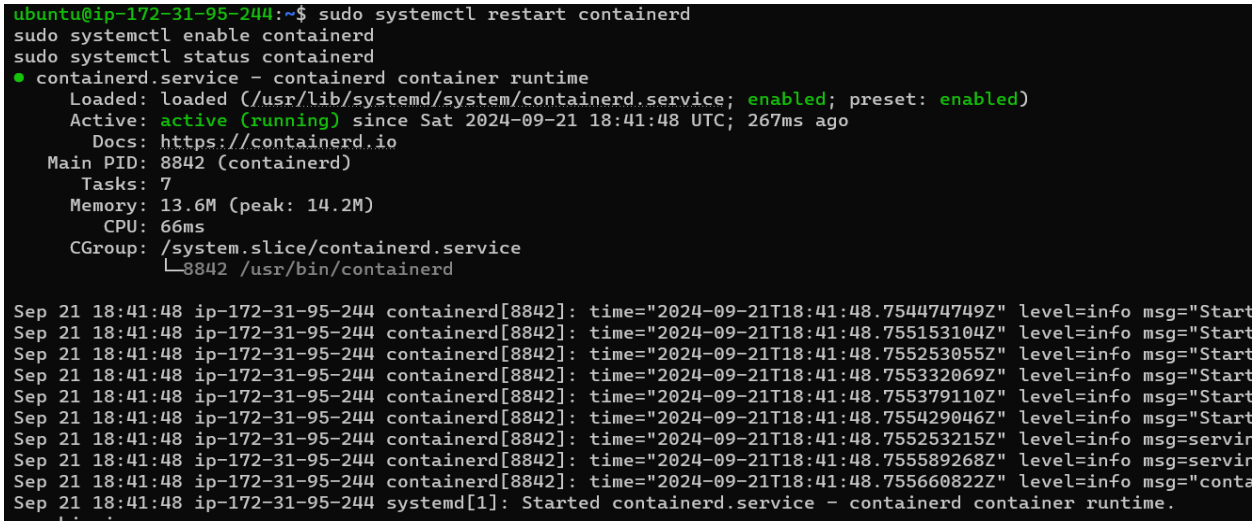
```
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

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.
ubuntu@ip-172-31-95-244:~$
```

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

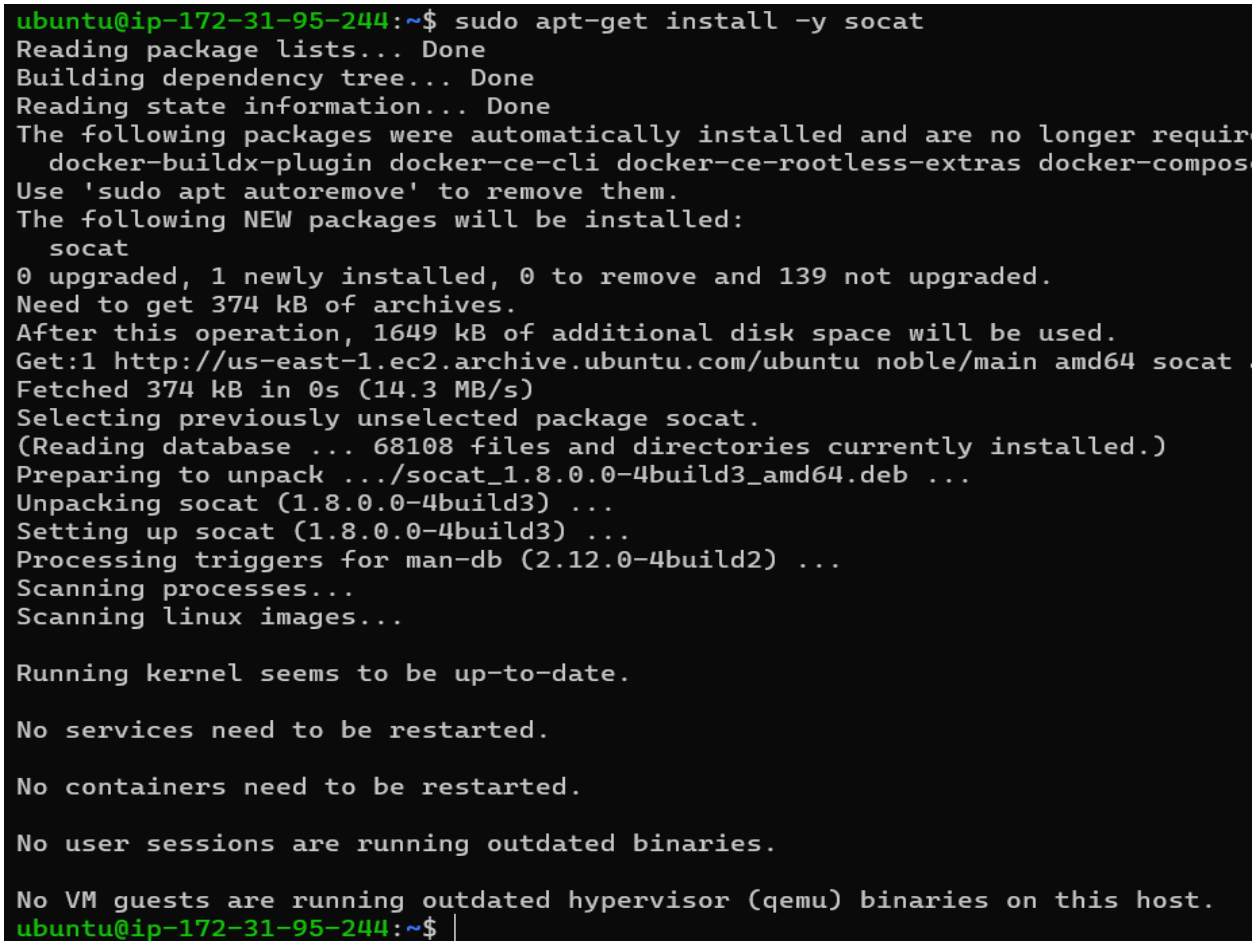
```
ubuntu@ip-172-31-95-244:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[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
```

```
  [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
    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-172-31-95-244:~$
```

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



- sudo apt-get install -y socat



Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.

- sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-95-244:~$ 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'
W0921 18:47:47.470667    9264 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container ru
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-172-31-95-244 kubernetes kubernetes.default kubernetes.default.sv
.local] and IPs [10.96.0.1 172.31.95.244]
[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-172-31-95-244 localhost] and IPs [172.31.95.244 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-95-244 localhost] and IPs [172.31.95.244 127.0.0.1 ::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
```

```
[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 172.31.95.244:6443 --token kzfth2.ug3970lp3qeeieb4 \
        --discovery-token-ca-cert-hash sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa88ad75c7ea83f15c529f4ca
ubuntu@ip-172-31-95-244:~$ mkdir -p $HOME/.kube
```

Copy the kudeadm join any number of worker nodes command to use it later for joining Node 1 and Node 2 with master

- 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-172-31-95-244:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-95-244:~$
```

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

```
ubuntu@ip-172-31-95-244:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-95-244   NotReady   control-plane   2m52s   v1.31.1
ubuntu@ip-172-31-95-244:~$
```

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

- sudo kubeadm join 172.31.95.244:6443 --token kzfth2.ug3970lp3qeeieb4\
  --discovery-token-ca-cert-hash
  sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa88ad75c7ea83f15c529f4ca

Node 1:

```
ubuntu@ip-172-31-84-209:~$ sudo kubeadm join 172.31.95.244:6443 --token kzfth2.ug3970lp3qeeieb4 \
        --discovery-token-ca-cert-hash sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa88ad75c7ea83f15c529f4ca
[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 1.001303324s
[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-172-31-84-209:~$
```

Node 2:

```
ubuntu@ip-172-31-87-15:~$ sudo kubeadm join 172.31.95.244:6443 --token kzfth2.ug3970lp3qeeieb4 \
        --discovery-token-ca-cert-hash sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa88ad75c7ea83f15c529f4ca
[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 1.501340478s
[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-172-31-87-15:~$
```

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

```
ubuntu@ip-172-31-95-244:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-84-209   NotReady   <none>          113s    v1.31.1
ip-172-31-87-15    NotReady   <none>          65s     v1.31.1
ip-172-31-95-244   NotReady   control-plane   8m30s   v1.31.1
ubuntu@ip-172-31-95-244:~$
```

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-172-31-95-244:~$ 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-172-31-95-244:~$
```

- sudo systemctl status kubelet

```
ubuntu@ip-172-31-95-244:~$ 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 Sat 2024-09-21 18:48:07 UTC; 9min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 9932 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.4M (peak: 32.9M)
        CPU: 9.201s
     CGroup: /system.slice/kubelet.service
             └─9932 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/k

Sep 21 18:57:49 ip-172-31-95-244 kubelet[9932]: I0921 18:57:49.455496    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:49 ip-172-31-95-244 kubelet[9932]: I0921 18:57:49.457800    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:49 ip-172-31-95-244 kubelet[9932]: I0921 18:57:49.468016    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:49 ip-172-31-95-244 kubelet[9932]: I0921 18:57:49.478177    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:52 ip-172-31-95-244 kubelet[9932]: I0921 18:57:52.466936    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:52 ip-172-31-95-244 kubelet[9932]: E0921 18:57:52.467071    9932 pod_workers.go:1301] "Error syncing pod,
Sep 21 18:57:58 ip-172-31-95-244 kubelet[9932]: I0921 18:57:58.981488    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:58 ip-172-31-95-244 kubelet[9932]: E0921 18:57:58.981598    9932 pod_workers.go:1301] "Error syncing pod,
Sep 21 18:57:59 ip-172-31-95-244 kubelet[9932]: I0921 18:57:59.349434    9932 scope.go:117] "RemoveContainer" containe
Sep 21 18:57:59 ip-172-31-95-244 kubelet[9932]: E0921 18:57:59.349579    9932 pod_workers.go:1301] "Error syncing pod,
lines 1-23/23 (END)
```

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

```
ubuntu@ip-172-31-95-244:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES          AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-209  Ready    <none>         4m24s  v1.31.1   172.31.84.209   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-87-15   Ready    <none>         3m36s  v1.31.1   172.31.87.15    <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-95-244  Ready    control-plane  11m    v1.31.1   172.31.95.244   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-95-244:~$
```

The Roles are not yet assigned to the Nodes

```
ubuntu@ip-172-31-95-244:~$ kubectl get nodes
NAME              STATUS   ROLES          AGE    VERSION
ip-172-31-84-209  Ready    <none>         5m27s  v1.31.1
ip-172-31-87-15   Ready    <none>         4m39s  v1.31.1
ip-172-31-95-244  Ready    control-plane  12m    v1.31.1
ubuntu@ip-172-31-95-244:~$
```

- **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-172-31-95-244:~$ kubectl label node ip-172-31-84-209 kubernetes.io/role=Node1
node/ip-172-31-84-209 labeled
ubuntu@ip-172-31-95-244:~$ kubectl label node ip-172-31-95-15 kubernetes.io/role=Node2
Error from server (NotFound): nodes "ip-172-31-95-15" not found
ubuntu@ip-172-31-95-244:~$ kubectl label node ip-172-31-87-15 kubernetes.io/role=Node2
node/ip-172-31-87-15 labeled
ubuntu@ip-172-31-95-244:~$
```

- Run **kubectl get nodes** to check if roles are assigned now to the nodes

```
ubuntu@ip-172-31-95-244:~$ kubectl get nodes
NAME              STATUS   ROLES          AGE    VERSION
ip-172-31-84-209  Ready    Node1          8m57s  v1.31.1
ip-172-31-87-15   Ready    Node2          8m9s   v1.31.1
ip-172-31-95-244  Ready    control-plane  15m    v1.31.1
ubuntu@ip-172-31-95-244:~$
```