

**EXPERIMENT NO: 3**

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

**STEPS :****1. Create Security Groups with the following inbound rules**

EC2 > Security Groups > Create security group

### Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name [Info](#)  
Master  
Name cannot be edited after creation.

Description [Info](#)  
Allows SSH access to developers

VPC [Info](#)  
vpc-06a8924e856c7a04d

**Inbound rules [Info](#)**

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
HTTP	TCP	80	Anywhere-I... 0.0.0.0/0 X		Delete
All traffic	All	All	Anywhere-I... 0.0.0.0/0 X		Delete
Custom TCP	TCP	6443	Anywhere-I... 0.0.0.0/0 X		Delete

**Inbound rules [Info](#)**

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
HTTP	TCP	80	Anywhere-I... 0.0.0.0/0 X		Delete
All traffic	All	All	Anywhere-I... 0.0.0.0/0 X		Delete
Custom TCP	TCP	6443	Anywhere-I... 0.0.0.0/0 X		Delete
Custom TCP	TCP	10251	Anywhere-I... 0.0.0.0/0 X		Delete
Custom TCP	TCP	10250	Anywhere-I... 0.0.0.0/0 X		Delete
All TCP	TCP	0 - 65535	Anywhere-I... 0.0.0.0/0 X		Delete
Custom TCP	TCP	10252	Anywhere-I... 0.0.0.0/0 X		Delete
SSH	TCP	22	Anywhere-I... 0.0.0.0/0 X		Delete

Add rule

EC2 > Security Groups > Create security group

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)

node

Name cannot be edited after creation.

Description [Info](#)

exp3

VPC [Info](#)

vpc-06a8924e856c7a04d

Inbound rules [Info](#)

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
HTTP	TCP	80	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
All traffic	All	All	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
Custom TCP	TCP	30000 - 32767	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>

Inbound rules [Info](#)

Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>
HTTP	TCP	80	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
All traffic	All	All	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
Custom TCP	TCP	30000 - 32767	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
Custom TCP	TCP	10250	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
SSH	TCP	22	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>
All TCP	TCP	0 - 65535	Anywhere-I... <div>Q0.0.0.0/0 X</div>	<div></div> <div>Delete</div>

Add rule

## 2.Create Instances:

We initiated the creation of three virtual machines or instances, naming them Master, node-1, and node-2. These instances will act as the nodes in our Kubernetes cluster.

EC2 > Instances > Launch an instance

### 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

master [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 Linux

Browse more AMIs  
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

#### ▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)  
Canonical, Ubuntu, 24.04, amd64...[read more](#)  
ami-0e86e20dae9224db8

Virtual server type (instance type)  
t2.medium

Firewall (security group)  
Master

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

Cancel [Launch instance](#) [Review commands](#)

Before you launch the instance:

Key pair name - required

key2309 [Create new key pair](#)

#### ▼ Network settings [Info](#)

[Edit](#)

Network [Info](#)  
vpc-06a8924e856c7a04d

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

Master sg-04d40f6a8d82c8941 X  
VPC: vpc-06a8924e856c7a04d [Compare security group rules](#)

Security groups that you add or remove here will be added to or removed from all your network interfaces.

#### ▼ Configure storage [Info](#)

[Advanced](#)

1x 8 GiB gp3 Root volume (Not encrypted)

#### ▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)  
Canonical, Ubuntu, 24.04, amd64...[read more](#)  
ami-0e86e20dae9224db8

Virtual server type (instance type)  
t2.medium

Firewall (security group)  
Master

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

Cancel [Launch instance](#) [Review commands](#)

EC2 > Instances > Launch an instance

Launch an instance

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

Name

node

Add additional tags

Application and OS Images (Amazon Machine Image)

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 Linux

Browse more AMIs

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

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

Free tier eligible

Summary

Number of instances

2

When launching more than 1 instance, consider EC2 Auto Scaling

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...read more

ami-0e86e20dae9224db8

Virtual server type (instance type)

t2.medium

Firewall (security group)

node

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

Review commands

Additional costs apply for AMIs with pre-installed software

Key pair (login)

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

key2309

Create new key pair

Network settings

Network

vpc-06a8924e856c7a04d

Subnet

No preference (Default subnet in any availability zone)

Auto-assign public IP

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

Create security group

Select existing security group

Common security groups

Select security groups

node sg-056d20b63b3793572

VPC: vpc-06a8924e856c7a04d

Compare security group rules

Summary

Number of instances

2

When launching more than 1 instance, consider EC2 Auto Scaling

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...read more

ami-0e86e20dae9224db8

Virtual server type (instance type)

t2.medium

Firewall (security group)

node

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

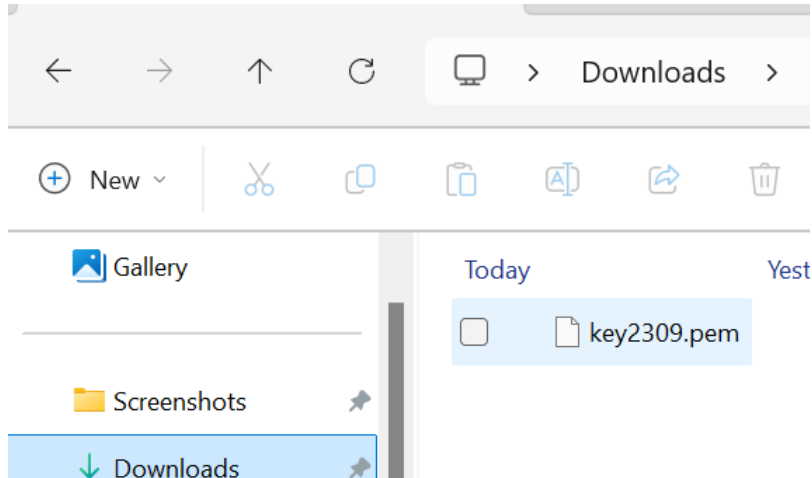
Launch instance

Review commands

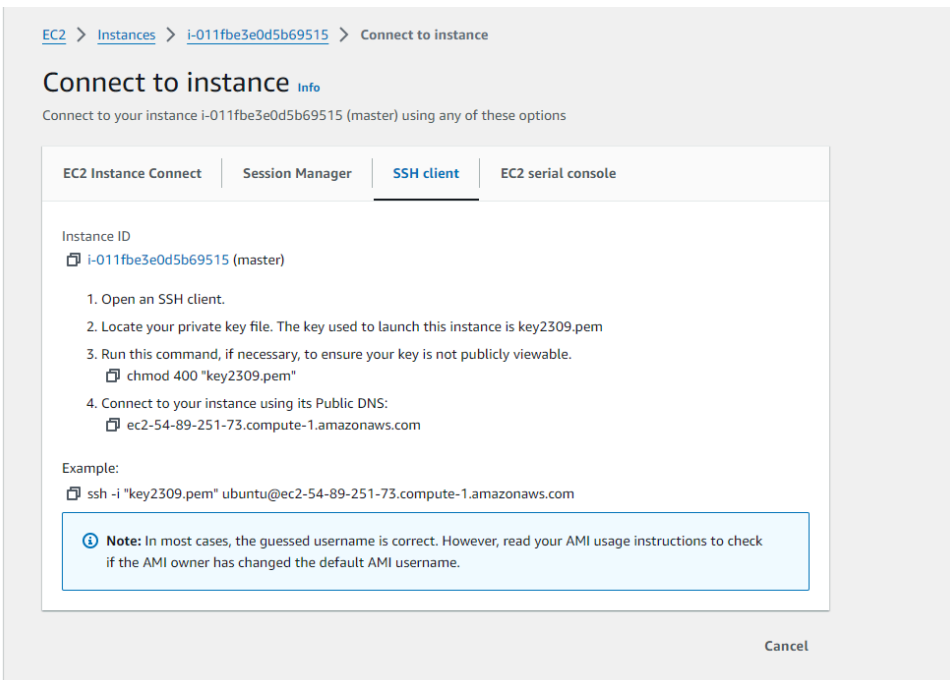
These the instances we have created successfully.

<input type="checkbox"/>	node-1	i-019d881a62651c90d	<span>Running</span>	t2.medium	Initializing	<a href="#">View alarms</a>	us-east-1b	ec2-3-86-95-118.comp...	3.86.95.118
<input type="checkbox"/>	node-2	i-05333ee8d27d26a64	<span>Running</span>	t2.medium	Initializing	<a href="#">View alarms</a>	us-east-1b	ec2-35-174-170-105.co...	35.174.170.105
<input type="checkbox"/>	master	i-011f8e3e0d5b69515	<span>Running</span>	t2.medium	Initializing	<a href="#">View alarms</a>	us-east-1b	ec2-54-89-251-73.com...	54.89.251.73

It is important to have the download file of your key.



Copy the command below in the example part and paste it in the cmd.



## 2.Install Docker:

```
PS C:\Users\91900> ssh -i "key2309.pem" ubuntu@ec2-35-174-170-105.compute-1.amazonaws.com
The authenticity of host 'ec2-35-174-170-105.compute-1.amazonaws.com (35.174.170.105)' can't be established.
ED25519 key fingerprint is SHA256:777d6VJLUANXWdvC2Rqcqg6Jscu79S7iwHCjSgnMwM0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-174-170-105.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 Mon Sep 23 15:19:47 UTC 2024
```

```
System load:  0.0          Processes:      112
Usage of /:   22.8% of 6.71GB Users logged in: 0
Memory usage: 5%          IPv4 address for enx0: 172.31.86.235
Swap usage:   0%
```

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-172-31-95-11:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOWQHxMAlaFhcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtgQmljZ/bJvXqnmivXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYf0S8QcixNzHUYNb6emjLANyEVLZzeqo7XKl7
UrwV5inawTSzWnvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
```

```
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 5s (6401 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg)
, see the DEPRECATION section in apt-key(8) for details.
```

sudo apt-get update

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-95-11:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy
, see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
```

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.

**No user sessions are running outdated binaries.**

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

**3.Start Docker:****sudo mkdir -p /etc/docker****cat <<EOF | sudo tee /etc/docker/daemon.json**

```
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}  
EOF
```

```
ubuntu@ip-172-31-95-11:~$ sudo mkdir -p /etc/docker  
cat <<EOF | sudo tee /etc/docker/daemon.json  
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}  
EOF  
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}
```

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

```
ubuntu@ip-172-31-95-11:~$ 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
```

**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-172-31-95-11:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyring  
s/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/ap  
t/sources.list.d/kubernetes.list  
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```



**Install Kubernetes components:**

```
sudo apt-get update
```

```
sudo apt-get install -y kubelet kubeadm kubectl
```

```
sudo apt-mark hold kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-95-11:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 6051 B in 1s (11.5 kB/s)
```

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

```
sudo systemctl enable --now kubelet
```

```
sudo apt-get install -y containerd
```

```
ubuntu@ip-172-31-95-11:~$ 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-plugin libltdl7 libsllp0 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:
```

```
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-172-31-95-11:~$ 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
```

```
[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
```

```
ubuntu@ip-172-31-95-11:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-09-23 15:35:29 UTC; 229ms ago
     Docs: https://containerd.io
   Main PID: 4614 (containerd)
    Tasks: 7
   Memory: 13.1M (peak: 14.4M)
      CPU: 64ms
   CGroup: /system.slice/containerd.service
           └─4614 /usr/bin/containerd

Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.093975568Z" level=info msg=serving... address=/run/contap
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094023950Z" level=info msg=serving... address=/run/contap
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094053439Z" level=info msg="Start subscribing containerd
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094080493Z" level=info msg="Start recovering state"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094120880Z" level=info msg="Start event monitor"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094128980Z" level=info msg="Start snapshots syncer"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094137306Z" level=info msg="Start cni network conf syncer"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094142936Z" level=info msg="Start streaming server"
Sep 23 15:35:29 ip-172-31-95-11 containerd[4614]: time="2024-09-23T15:35:29.094178857Z" level=info msg="containerd successfully boot
Sep 23 15:35:29 ip-172-31-95-11 systemd[1]: Started containerd.service - containerd container runtime.
Lines 1-21/21 (END)
```

```
ubuntu@ip-172-31-95-11:~$ 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:
```

```
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.

**6.Initialize kubeadm on Master Node:**

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

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

On the Master node, initialize the Kubernetes cluster using kubeadm. This process sets up the Kubernetes control plane and generates commands for joining worker nodes:

Copy the commands displayed in the output of the initialization process to configure permissions and obtain the join token. This includes a join command link needed for worker nodes to connect to the master.

```
ubuntu@ip-172-31-95-11:~$ 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'
W0923 15:48:50.266640 5352 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
```

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 172.31.95.11:6443 --token 16q9ib.pv09d9wixd10d50s \
--discovery-token-ca-cert-hash sha256:7ec275faa3af2297d3e809dea0a29175b2d2af6db63d2673456039bdba306fc6
ubuntu@ip-172-31-95-11:~$
```

```
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-172-31-95-11:~$ 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-172-31-95-11:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-95-11	NotReady	control-plane	2m50s	v1.31.1

## 7.Join Worker Nodes:

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
```

```
ubuntu@ip-172-31-86-235:~$ sudo kubeadm join 172.31.95.11:6443 --token 16q9ib.pv09d9wixd10d50s \
--discovery-token-ca-cert-hash sha256:7ec275faa3af2297d3e809dea0a29175b2d2af6db63d2673456039bdba306fc6
[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 501.238074ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap
```

This node has joined the cluster:

- \* Certificate signing request was sent to apiservert 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-172-31-95-11:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-86-235    NotReady <none>   70s   v1.31.1
ip-172-31-91-211    NotReady <none>   61s   v1.31.1
ip-172-31-95-11     NotReady control-plane 5m25s v1.31.1
ubuntu@ip-172-31-95-11:~$ 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/bgppconfigurations.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
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE           KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-86-235    Ready     <none>   3m44s v1.31.1   172.31.86.235 <none>         Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-91-211    Ready     <none>   3m35s v1.31.1   172.31.91.211 <none>         Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-95-11     Ready     control-plane 7m59s v1.31.1   172.31.95.11  <none>         Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-91-211 kubernetes.io/role=Node1
node/ip-172-31-91-211 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-95-11 kubernetes.io/role=worker
node/ip-172-31-95-11 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-86-235 kubernetes.io/role=Node2
error: 'kubernetes.io/role' already has a value (Node1), and --overwrite is false
```

```
ubuntu@ip-172-31-95-11:~$ kubectl label node ip-172-31-86-235 kubernetes.io/role=Node2 --overwrite
node/ip-172-31-86-235 labeled
```

```
ubuntu@ip-172-31-95-11:~$ kubectl get nodes -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION
ip-172-31-86-235	Ready	Node2	9m29s	v1.31.1	172.31.86.235	<none>	Ubuntu 24.04 LTS	6.8.0-1012-aws
ip-172-31-91-211	Ready	Node1	9m20s	v1.31.1	172.31.91.211	<none>	Ubuntu 24.04 LTS	6.8.0-1012-aws
ip-172-31-95-11	Ready	control-plane,worker	13m	v1.31.1	172.31.95.11	<none>	Ubuntu 24.04 LTS	6.8.0-1012-aws

## CONCLUSION :

**Docker installation:** After installing docker on all instances, sometimes docker services may fail to restart.

**Network Configuration Issue:** Connectivity issue between master and worker modes might be caused by the firewall blocking that required ports.

**CrashLoopBackOff:** There are errors indicating that the containers for Kubernetes components are restarting repeatedly but failing to start properly.