

ADVANCE DEVOPS EXPERIMENT -3

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

Theory :

Kubernetes (often abbreviated as K8s) is a powerful open-source platform for managing containerized applications across multiple hosts. It provides an abstraction layer for deploying and scaling applications, ensuring high availability and fault tolerance. Kubernetes helps you to build cloud-native microservices-based apps. Kubernetes is built to be used anywhere, allowing you to run your applications across on-site deployments and public clouds; as well as hybrid deployments in between. So you can run your applications where you need them. The Kubernetes Cluster architecture consists of several key components divided into two major categories:

1. **Master Node Components**
2. **Worker Node Components**

1. Master Node Components

The **Master Node** is responsible for managing the entire Kubernetes cluster. It acts as the brain of the cluster, orchestrating container deployments, scaling, and communication between the nodes.

a. API Server

- Acts as the front-end for the Kubernetes control plane.
- All internal and external communication with the cluster is through the API server.
- It validates and processes RESTful requests.

b. Scheduler

- Responsible for distributing workload across the cluster.
- Determines which worker node will host a newly created pod based on resource availability, policy constraints, and other factors.

c. Controller Manager

- Ensures that the desired state of the cluster matches the actual state.
- Examples of controllers include the **Node Controller**, **Replication Controller**, **Endpoint Controller**, and **Service Controller**.

-

d. etcd

- A consistent and highly-available key-value store used as Kubernetes' backing store.
- Stores the entire configuration and state of the Kubernetes cluster.
- etcd must be backed up regularly as it's critical to the integrity of the cluster.

e. Cloud Controller Manager (optional)

- Integrates cloud-specific APIs into Kubernetes.
- Handles cloud-related services like load balancing, managing cloud storage, and node lifecycle events in cloud platforms like AWS, GCP, Azure, etc.

2. Worker Node Components

The **Worker Nodes** are where your applications run. Each worker node communicates with the master node and executes the commands given.

a. Kubelet

- The primary agent that runs on each worker node.
- Ensures that the containers described in a pod are running and healthy.
- Communicates with the API server and ensures that the desired state is met on the node.

b. Kube-proxy

- Manages the networking for the worker node.
- Ensures that traffic is correctly routed to and from the pods running on the worker node.
- Facilitates communication between services within the cluster.

c. Container Runtime

- Responsible for pulling container images and running the containers.
- Common runtimes include Docker, containerd, or CRI-O.

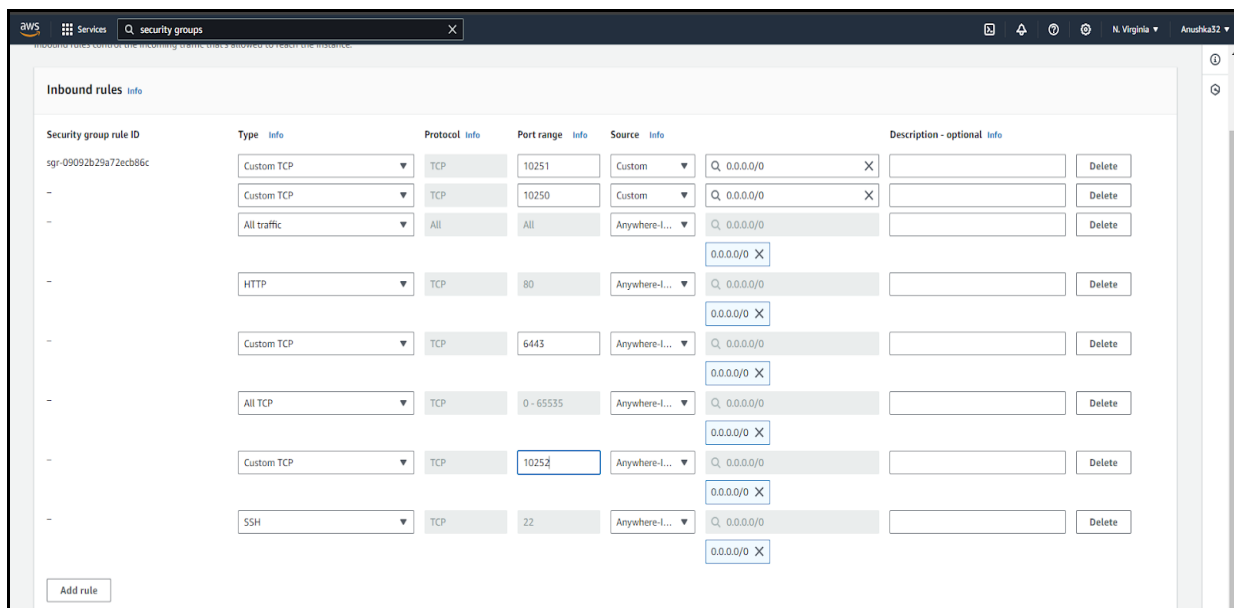
d. Pods

- The smallest deployable unit in Kubernetes.

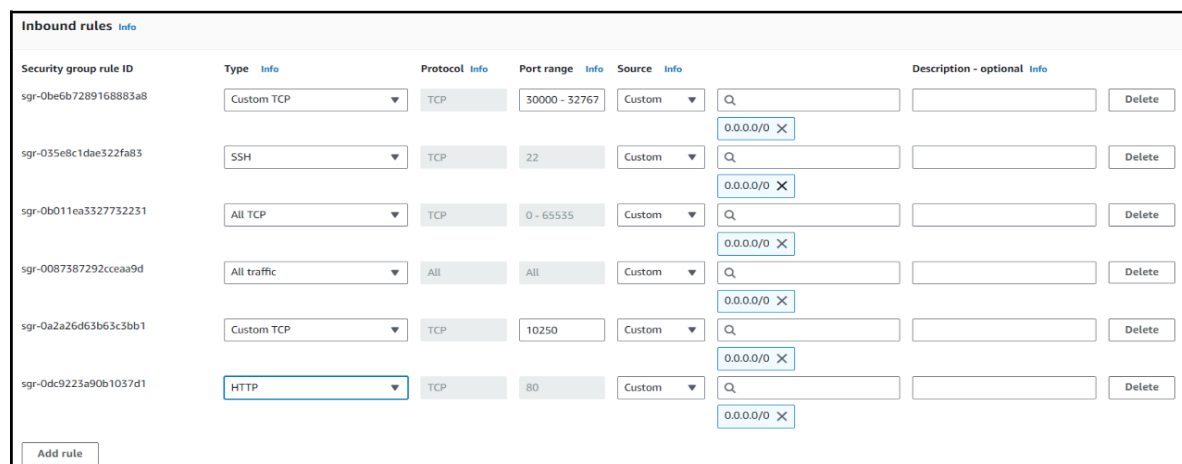
A pod encapsulates one or more containers and their shared storage, network, and configuration options.

Step 1: Create 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.

Master:

Name and tags [Info](#)

Name

[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

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)

Number of instances [Info](#)

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[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 I/Os

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

▼ 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

[Create new key pair](#)

▼ Network settings [Info](#)

[Edit](#)

Network [Info](#)

vpc-0fae6b32b6900863

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

We'll create a new security group called 'launch-wizard-1' with the following rules:

▼ Summary

Number of instances [Info](#)

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[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 I/Os

Node:

Name and tags [Info](#)

Name


[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

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)

▼ Summary

Number of instances [Info](#)

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64...[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 I/Os

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

Q |

Create new key pair

Proceed without a key pair (Not recommended) Default value

Master_Ec2_key
Type: rsa

Node_Ec2_key
Type: rsa

vpc-07294e1d226906dc2

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, amd64...[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](#)

1178

▼ 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

Q |

Compare security group rules

default
VPC: vpc-07294e1d226906dc2 sg-0ae340202e7edd220

Master
VPC: vpc-07294e1d226906dc2 sg-071b5fbaaddf4db2

Node
VPC: vpc-07294e1d226906dc2 sg-07b5f2fa89d6fc2f0

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)

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

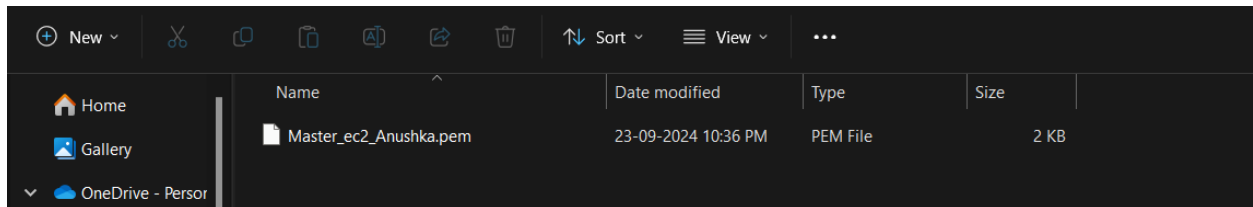
Launch instance

[Review commands](#)

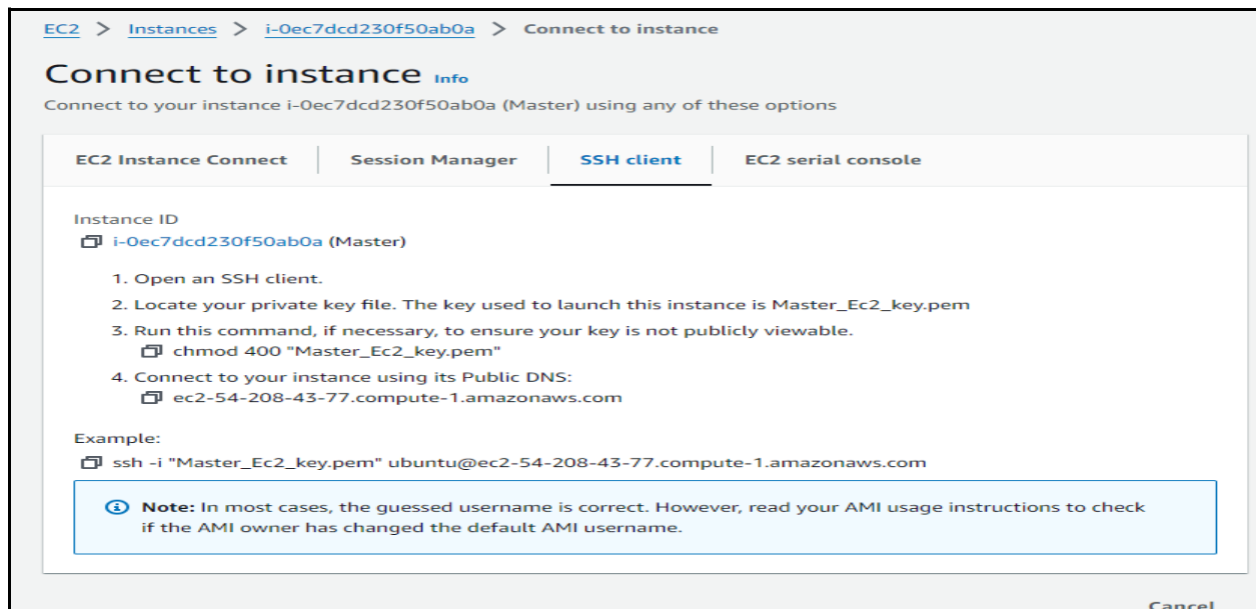
1178

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

Key:



Master:



```
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

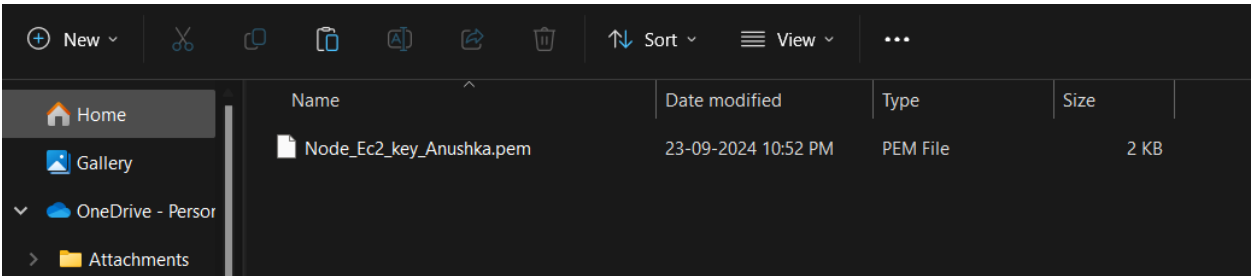
System load:  0.0          Processes:    115
Usage of /:   23.1% of 6.71GB Users logged in:  0
Memory usage: 7%          IPv4 address for enx0: 172.31.95.244
Swap usage:   0%

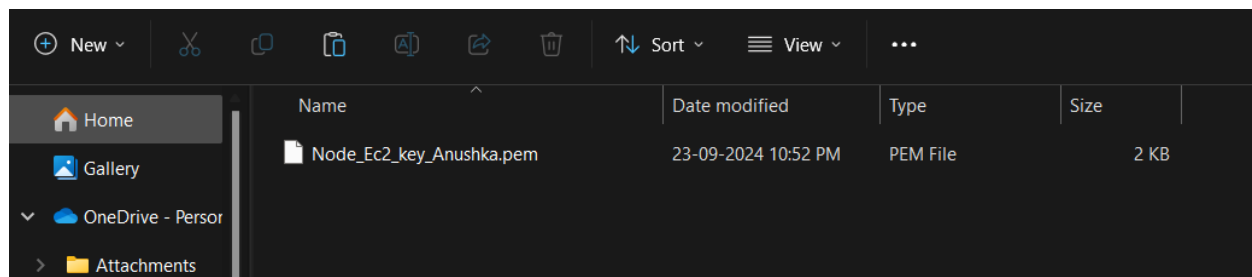
Expanded Security Maintenance for Applications is not enabled.

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
```

Key:





Node 2:

EC2 > Instances > i-0aa46c5d7a55a5eb6 > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0aa46c5d7a55a5eb6 (Node 2) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
i-0aa46c5d7a55a5eb6 (Node 2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Node_Ec2_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "Node_Ec2_key.pem"
4. Connect to your instance using its Public DNS:
ec2-3-87-184-248.compute-1.amazonaws.com

Example:
ssh -i "Node_Ec2_key.pem" ubuntu@ec2-3-87-184-248.compute-1.amazonaws.com

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

Cancel

```
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/f3ylisWTr/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, Node 1, and Node 2 the below commands to install and setup .

```
ubuntu@ip-172-31-95-244:~$ 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/wvZ6hUTiX0wQHXMAlaFhcPH9hAtr4Fly2+OYdbtMuth
lqqwp028AqyY+PRfVmtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmVXh
38UuLa+z077PyxQhu5BbqntTPQMfiyqEiU+8Kbq2WmANUKQf+1AmZY/Iru0Xbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYNb6emjLANyEVLZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscLPQUhTQ+7BbQXAwAmeHCUTQIvvWxqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzLgaRL5Z4POXDDZTLIQ/EL58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtL/94VFYVJ0leAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
-----
Get:45 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:46 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Translation-en [10.8 kB]
Get:47 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:48 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:49 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:51 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:52 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 29.1 MB in 4s (7658 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
action in apt-key(8) for details.
ubuntu@ip-172-31-95-244:~$
```

```
ubuntu@ip-172-31-95-244:~$ 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 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
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg
action in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin lib
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-
```

Step 5: Run the below command to install Kubernetes.

```
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

```
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 (C
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.
```

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.

```
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 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
```

```

Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.754474749Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755153104Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755253055Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755332069Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755379110Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755429046Z" level=info msg="Start
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755253215Z" level=info msg="servin
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755589268Z" level=info msg="servin
Sep 21 18:41:48 ip-172-31-95-244 containerd[8842]: time="2024-09-21T18:41:48.755660822Z" level=info msg="conta
Sep 21 18:41:48 ip-172-31-95-244 systemd[1]: Started containerd.service - containerd container runtime.

```

```

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
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
Fetched 374 kB in 0s (14.3 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.

```


Step 6: Initialize the Kubecluster.

```
[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 runtime 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 kubernet.es kubernet.es.default kubernet.es.default.svc.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
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!
```

Step 7: Now Run the command `kubectl get nodes` to see the nodes.

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-95-244	NotReady	control-plane	2m52s	v1.31.1

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

```
sudo kubeadm join 172.31.95.244:6443 --token kzft2.ug3970lp3qeeieb4\ --
discovery-token-ca-cert-hash
```

Node 1:

```
--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.
```

Node 2:

```
--discovery-token-ca-cert-hash sha256:dec27d33f1bfd1dca7a50caa2c05d4cad1d0a18aa8ad75c7ea83f15c529f4ca
[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.
```

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

Step 10: Since Status is 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>**

```
poddissruptionbudget.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

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

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

The Roles are not yet assigned to the Nodes

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`

Step 11 : After that Verify cluster connection on master node

```
ubuntu@ip-172-31-44-131:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-40-114    Ready     <none>    26s   v1.29.0
ip-172-31-44-131    Ready     control-plane 23m   v1.29.0
```