ADVANCE DEVOPS EXP-3

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

Theory:
Container-based microservices architectures have revolutionized how development and operations teams test and deploy modern software. Containers allow companies to scale and deploy applications more efficiently, but they also introduce new challenges, adding complexity by creating a whole new infrastructure ecosystem.

Today, both large and small software companies are deploying thousands of container instances daily. Managing this level of complexity at scale requires advanced tools. Enter Kubernetes.
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Kubernetes has quickly become the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), supported by major players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes simplifies the deployment and operation of applications in a microservice architecture by providing an abstraction layer over a group of hosts. This allows development teams to deploy their applications while Kubernetes takes care of key tasks, including:

- ● Managing resource consumption by applications or teams
- ● Distributing application load evenly across the infrastructure
- ● Automatically load balancing requests across multiple instances of an application
- ● Monitoring resource usage to prevent applications from exceeding resource limits and
- automatically restarting them if needed
- ● Moving application instances between hosts when resources are low or if a host fails
- ● Automatically utilizing additional resources when new hosts are added to the cluster
- ● Facilitating canary deployments and rollbacks with ease
- Necessary Requirements:
- ● EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as
- Kubernetes demands sufficient resources for effective functioning.
- ● Minimum Requirements:

○ Instance Type: t2.medium
○ CPUs: 2
○ Memory: Adequate for container orchestration.
This ensured that the Kubernetes cluster had the necessary resources to function smoothly

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

Master:

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-088cc3ff8808aa44d	Custom TCP	TCP	6443	Custom	0.0.0.0/0		Delete
sgr-069da2a3c819ccba2	Custom TCP	TCP	10250	Custom	0.0.0.0/0		Delete
sgr-0e6cbc7a4c1270b60	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-088467a6ddfc73fe9	Custom TCP	TCP	10251	Custom	0.0.0.0/0		Delete
sgr-0dced61d56e719f9f	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-07048153ce3523dc9	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-02ce8b0567f4c3351	All TCP	TCP	0 - 65535	Custom	0.0.0.0/0		Delete
sgr-09b161b78fc97e86e	Custom TCP	TCP	10252	Custom	0.0.0.0/0		Delete

Add rule

Node:

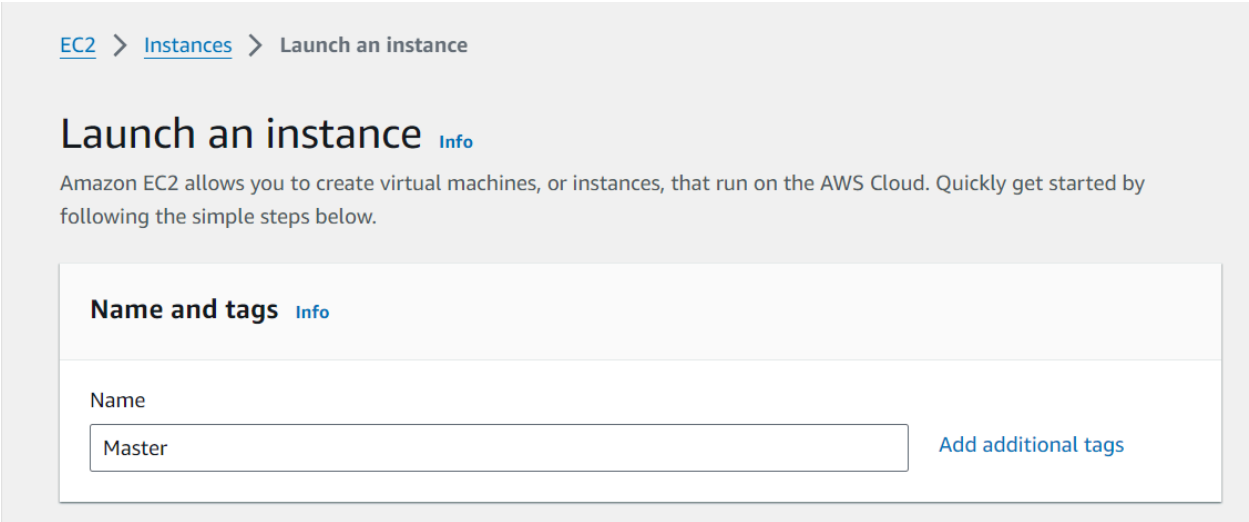
Inbound rules Info

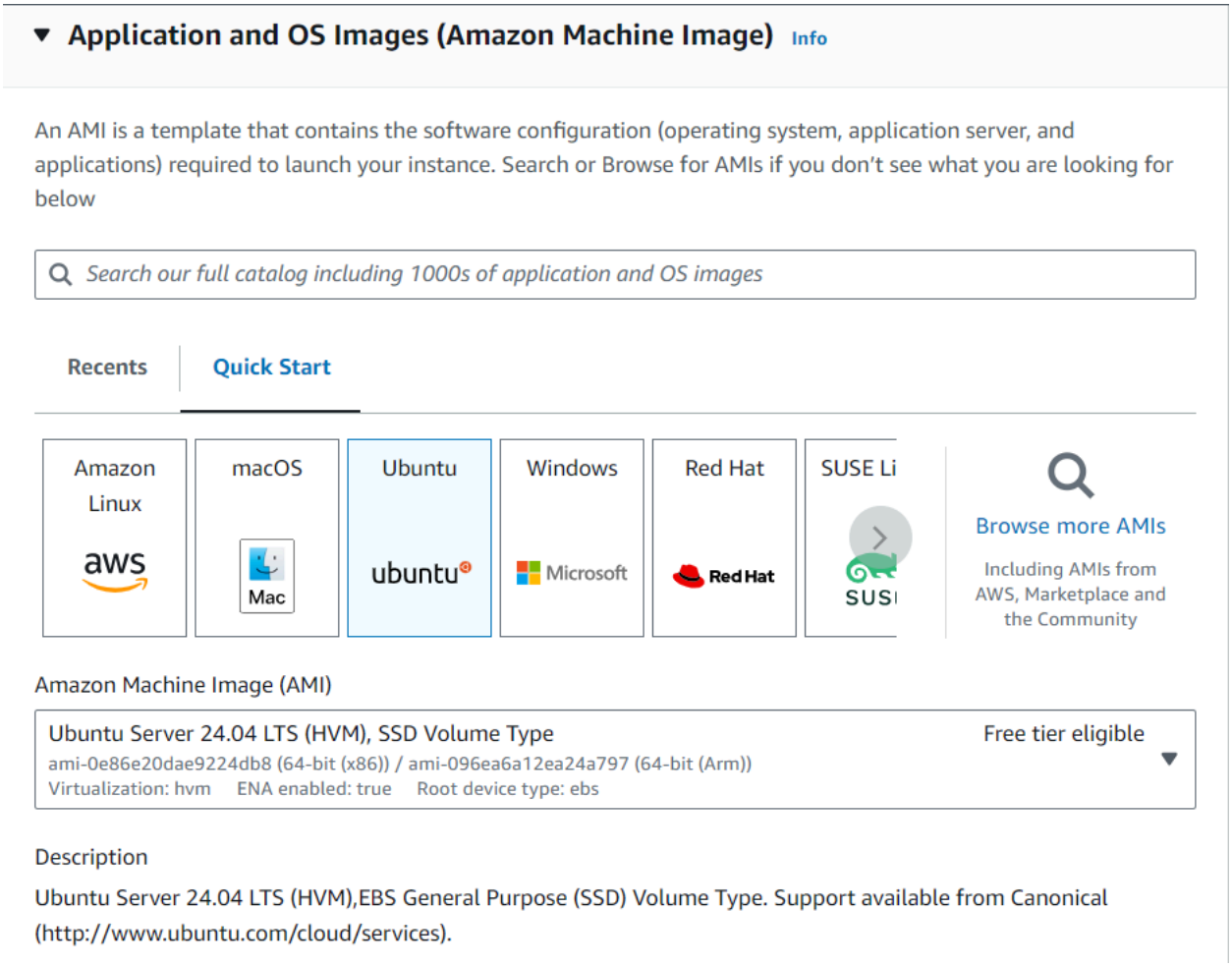
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0d6072a8c79e947c8	All TCP	TCP	0 - 65535	Custom	0.0.0.0/0		Delete
sgr-0dcfcab7177656606	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-09438da8cb6119119	Custom TCP	TCP	30000 - 32	Custom	0.0.0.0/0		Delete
sgr-0e9faedc577341fd6	Custom TCP	TCP	10250	Custom	0.0.0.0/0		Delete
sgr-0fe9772bbc777ebf0	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-0c2a28feaf2c8d86f	SSH	TCP	22	Custom	0.0.0.0/0		Delete

Add rule

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:





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*

Master_ec2_key

Create new key pair

Network settings Info

Edit

Network Info

vpc-024c98e3c11533db9

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-0db43ee2a0858c50c
VPC: vpc-024c98e3c11533db9

Compare security group rules

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

Node:

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

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 — Free tier eligible

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

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

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

SHRAVANI RASAM D15A 46

Network settings Info

Edit

Network Info

vpc-024c98e3c11533db9

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

Nodes sg-019d7373dd3c972e8 ✕
VPC: vpc-024c98e3c11533db9

Compare security group rules

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

Instances (5) Info

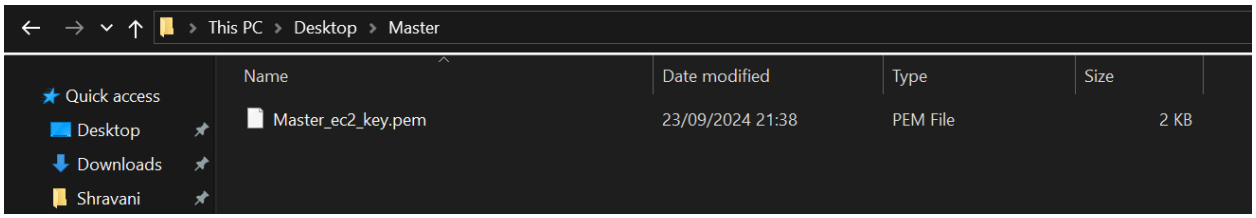
Last updated less than a minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states | 1

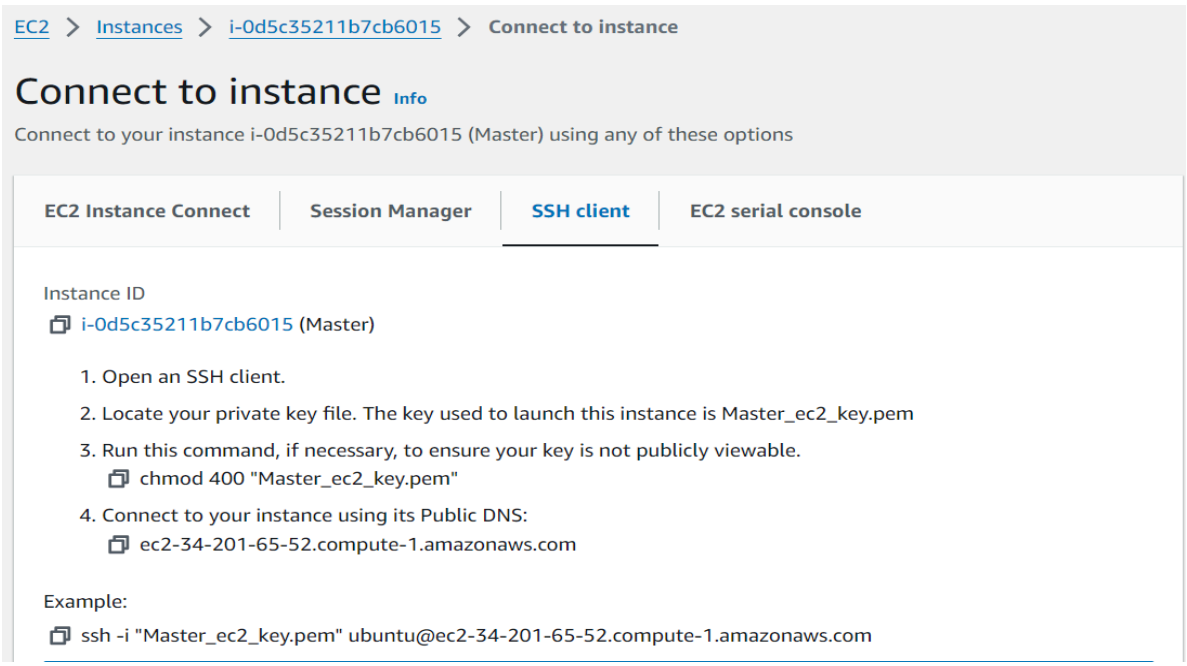
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
shravani-webs...	i-00f3310aafe64a5b9	Stopped	t2.micro	–	View alarms	us-east-1e	–
ShravaniR021...	i-0ece95f31565de7ee	Running	t2.small	2/2 checks passec	View alarms	us-east-1e	ec2-18-205-118-127.c
Node 2	i-08e8b706c4c048ea8	Running	t2.medium	2/2 checks passec	View alarms	us-east-1d	ec2-3-92-229-59.comp
Node 1	i-0cad8aaad24835d3c	Running	t2.medium	2/2 checks passec	View alarms	us-east-1d	ec2-18-208-184-75.co
Master	i-0d5c35211b7cb6015	Running	t2.medium	2/2 checks passec	View alarms	us-east-1d	ec2-34-201-65-52.com

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.

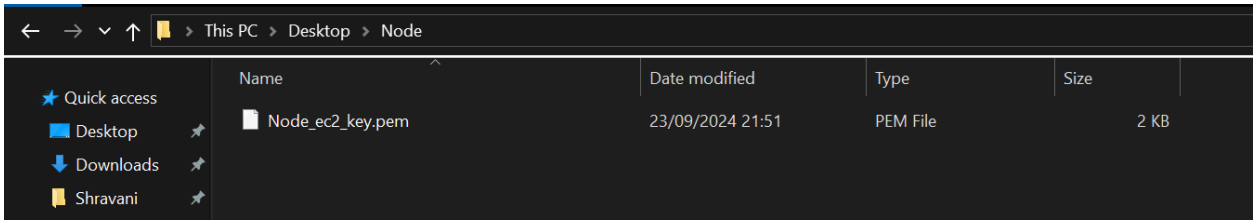
Downloaded Key:



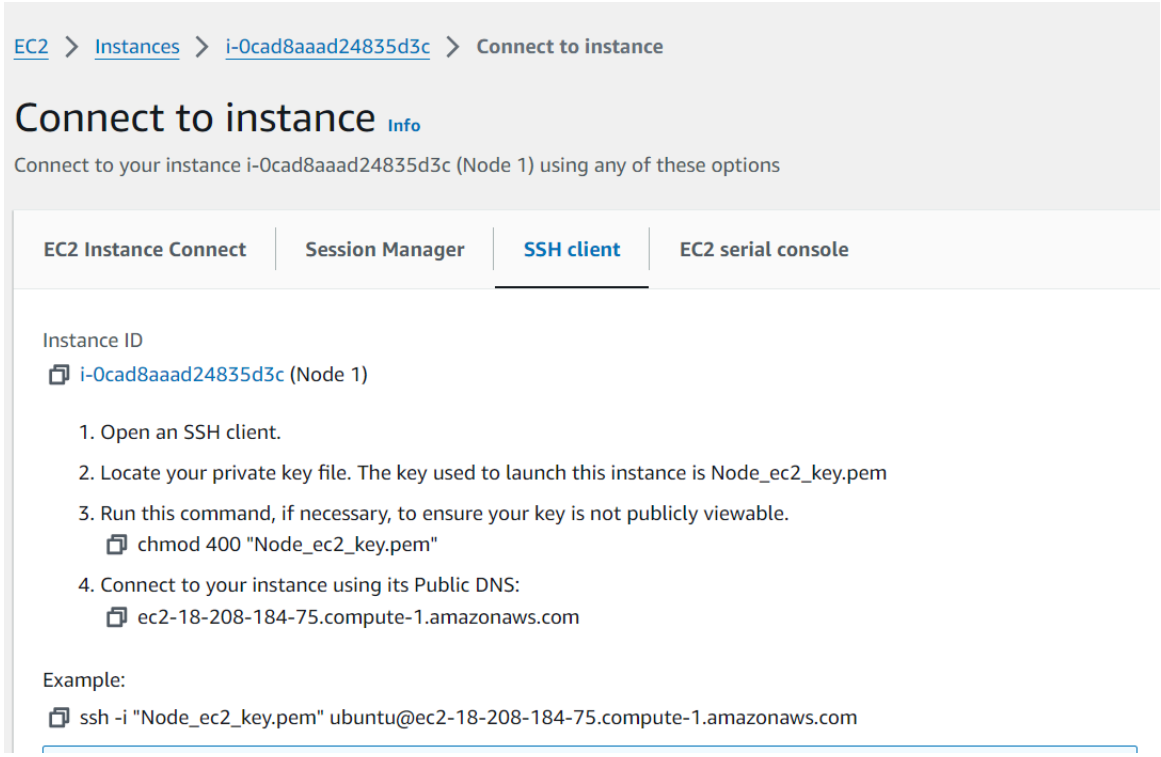
Master:




```
C:\Users\Shravani\Desktop\Master>ssh -i "Master_ec2_key.pem" ubuntu@ec2-34-201-65-52.compute-1.amazonaws.com
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 16:43:43 UTC 2024

  System load:  0.0               Processes:             116
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.84.221
  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
```



Node 1:




```
C:\Users\Shravani\Desktop\Node>ssh -i "Node_ec2_key.pem" ubuntu@ec2-18-208-184-75.compute-1.amazonaws.com
The authenticity of host 'ec2-18-208-184-75.compute-1.amazonaws.com (18.208.184.75)' can't be established.
ECDSA key fingerprint is SHA256:Mt3R8xcNRQpug+O8YjlPo+4OyaB1xn/43dC9MQA87+A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-208-184-75.compute-1.amazonaws.com,18.208.184.75' (ECDSA) 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 16:48:32 UTC 2024

  System load:  0.08               Processes:             113
  Usage of /:   22.7% of 6.71GB    Users logged in:       0
  Memory usage: 5%                 IPv4 address for enX0: 172.31.95.119
  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
```

Node 2:

EC2 > Instances > i-08e8b706c4c048ea8 > Connect to instance

Connect to instance Info

Connect to your instance i-08e8b706c4c048ea8 (Node 2) using any of these options

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

Instance ID

i-08e8b706c4c048ea8 (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:
   ec2-3-92-229-59.compute-1.amazonaws.com

Example:

ssh -i "Node_ec2_key.pem" ubuntu@ec2-3-92-229-59.compute-1.amazonaws.com

```
C:\Users\Shravani\Desktop\Node>ssh -i "Node_ec2_key.pem" ubuntu@ec2-3-92-229-59.compute-1.amazonaws.com
The authenticity of host 'ec2-3-92-229-59.compute-1.amazonaws.com (64:ff9b::35c:e53b)' can't be established.
ECDSA key fingerprint is SHA256:jKzi3rD9OgtRdm2AJ5oS4Ndayn4cxMLRUQGQVXEMsck.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-92-229-59.compute-1.amazonaws.com,64:ff9b::35c:e53b' (ECDSA) 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 16:50:15 UTC 2024

  System load:  0.0               Processes:             114
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.80.164
  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
```

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

● curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

●curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null

● sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee/etc/apt/
sudo: tee/etc/apt/trusted.gpg.d/docker.gpg: command not found
ubuntu@ip-172-31-84-221:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
gpg.d/docker.gpg > /dev/null-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtEb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
```

```
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [113 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.1 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
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 4s (7159 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
key(8) for details.
ubuntu@ip-172-31-84-221:~$
```

● sudo apt-get update

● sudo apt-get install -y docker-c

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
key(8) for details.
ubuntu@ip-172-31-84-221:~$
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo mkdir -p /etc/docker
driver=systemd"]
}
EOFcat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFubuntu@ip-172-31-84-221:~$ sudo mkdir -p /etc/docker
 tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFcat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOFubuntu@ip-172-31-84-221:~$
```

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

```
EOFubuntu@ip-172-31-84-221:~$ sudo systemctl enable docker
ctl daemon-reload
sudo systemctl restart dockersudo systemctl daemon-reload
```

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-172-31-84-221:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
ngs/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | gpg: missing argument for option "-o"
sudo tee /etc/apt/sources.list.d/kubernetes.list


ubuntu@ip-172-31-84-221:~$ /etc/apt/keyrings/kubernetes-apt-keyring.gpg
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
ubuntu@ip-172-31-84-221:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 136 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 335 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  cri-tools 1.28.0-1.1 [19.6 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubernetes-cni 1.2.0-2.1 [27.6 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubelet 1.28.14-2.1 [19.6 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubectl 1.28.14-2.1 [10.4 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  kubeadm 1.28.14-2.1 [10.1 MB]
Fetched 87.4 MB in 1s (77.5 MB/s)
```

```
ubuntu@ip-172-31-84-221:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-84-221:~$
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo apt-get update
Warning: The unit file, source configuration file or drop-ins of apt-news.service changed on disk. Run 'systemctl daemon-reload' to reload u
nits.
Warning: The unit file, source configuration file or drop-ins of esm-cache.service changed on disk. Run 'systemctl daemon-reload' to reload
units.
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
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Err:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
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 t
he DEPRECATION section in apt-key(8) for details.
W: GPG error: https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease: The following signatures co
uldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
E: The repository 'https://pkgs.k8s.io/core:/stable:/v1.31/deb  InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release 404 Not Found [IP: 64.233.180.139 443]

- sudo rm /etc/apt/sources.list.d/kubernetes.list
- sudo nano /etc/apt/sources.list.d/kubernetes.list
- deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /

```
ubuntu@ip-172-31-84-221:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-84-221:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 136 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (90.1 MB/s)
```

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

```
ubuntu@ip-172-31-84-221:~$ 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
```

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

```
ubuntu@ip-172-31-84-221:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-84-221:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-84-221:~$ 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 20:47:25 UTC; 14s ago
       Docs: https://containerd.io
   Main PID: 19202 (containerd)
      Tasks: 7
     Memory: 13.0M (peak: 13.8M)
        CPU: 113ms
     CGroup: /system.slice/containerd.service
             └─19202 /usr/bin/containerd

Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572213616Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572255061Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572281095Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572298184Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572313100Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572322058Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572328397Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572313683Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 containerd[19202]: time="2024-09-23T20:47:25.572786584Z" level=info
Sep 23 20:47:25 ip-172-31-84-221 systemd[1]: Started containerd.service - containerd container runtim
lines 1-21/21 (END)...skipping...
```

● sudo apt-get install -y socat

```
ubuntu@ip-172-31-84-221:~$ 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 lib
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 136 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3
Fetched 374 kB in 0s (13.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-84-221:~$
```

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-172-31-84-221:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0923 20:56:13.230794   19947 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.14
[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 in beforehand using 'kubeadm config images pull'
W0923 20:56:20.561492   19947 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container ru
used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" 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-172-31-84-221 kubernetes kubernetes.default kubernetes.default.sv
.local] and IPs [10.96.0.1 172.31.84.221]
[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-172-31-84-221 localhost] and IPs [172.31.84.221 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-84-221 localhost] and IPs [172.31.84.221 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get l
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a N
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the c
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 \
        --discovery-token-ca-cert-hash sha256:ffdb051e04077afecd5ea7a5702131537f9aa5c3dd13785ed4442327fb39f9cf
ubuntu@ip-172-31-84-221:~$
```

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

sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 \--discovery-token-ca-cert -hash sha256:ffdb051e04077afecd5ea7a5702131537f9aa5c3dd13785ed4442327fb39f9cf

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-172-31-84-221:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-84-221:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
cp: overwrite '/home/ubuntu/.kube/config'? y
ubuntu@ip-172-31-84-221:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-221:~$
```

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

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-84-221    NotReady   control-plane   8m27s   v1.28.14
ubuntu@ip-172-31-84-221:~$
```

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-172-31-95-119:~$ sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 --discovery-token-ca-cert-hash sha256:f
a5c3dd13785ed4442327fb39f9cf --ignore-preflight-errors=FileContent--proc-sys-net-bridge-bridge-nf-call-iptables
[preflight] Running pre-flight checks
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[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-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-172-31-95-119:~$
```

Node 2:

```
ubuntu@ip-172-31-80-164:~$ sudo kubeadm join 172.31.84.221:6443 --token yjt10w.maqlf98vcw88kw96 --discove
a5c3dd13785ed4442327fb39f9cf --ignore-preflight-errors=FileContent--proc-sys-net-bridge-bridge-nf-call-ip
[preflight] Running pre-flight checks
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-n
[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 y
[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-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.

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-80-164    NotReady   <none>          16s     v1.28.14
ip-172-31-84-221    NotReady   control-plane   30m     v1.28.14
ip-172-31-95-119    NotReady   <none>          6m43s   v1.28.14
ubuntu@ip-172-31-84-221:~$
```

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-172-31-84-221:~$ 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-172-31-84-221:~$
```

● sudo systemctl status kubelet

```
ubuntu@ip-172-31-84-221:~$ 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 Mon 2024-09-23 20:56:33 UTC; 32min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 20621 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 38.0M (peak: 38.5M)
        CPU: 25.017s
     CGroup: /system.slice/kubelet.service
             └─20621 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kube

Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:  > pod="kube-system/calico-kube-controller
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.385530   20621 remote_runti
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:         rpc error: code = Unknown desc = f
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:         : unknown
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]:  > podSandboxID="0ac51787037fdb883ecf57aad
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.385606   20621 kuberuntime_
Sep 23 21:29:20 ip-172-31-84-221 kubelet[20621]: E0923 21:29:20.505019   20621 kubelet.go:1
Sep 23 21:29:21 ip-172-31-84-221 kubelet[20621]: I0923 21:29:21.388923   20621 pod_startup_
Sep 23 21:29:26 ip-172-31-84-221 kubelet[20621]: I0923 21:29:26.828223   20621 scope.go:117
Sep 23 21:29:26 ip-172-31-84-221 kubelet[20621]: E0923 21:29:26.828431   20621 pod_workers.
lines 1-23/23 (END)
```

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

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES          AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-80-164  Ready    <none>         3m29s   v1.28.14  172.31.80.164   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-84-221  Ready    control-plane  33m     v1.28.14  172.31.84.221   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ip-172-31-95-119  Ready    <none>         9m56s   v1.28.14  172.31.95.119   <none>        Ubuntu 24.04 LTS  6.8.0-1012-aws   containerd://1.7.12
ubuntu@ip-172-31-84-221:~$
```

The Roles are not yet assigned to the Nodes

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-80-164    Ready    <none>          4m14s   v1.28.14
ip-172-31-84-221    Ready    control-plane   34m     v1.28.14
ip-172-31-95-119    Ready    <none>          10m     v1.28.14
ubuntu@ip-172-31-84-221:~$
```

- 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-172-31-84-221:~$ kubectl label node ip-172-31-80-164 kubernetes.io/role=Node1
node/ip-172-31-80-164 labeled
ubuntu@ip-172-31-84-221:~$ kubectl label node ip-172-31-95-119 kubernetes.io/role=Node2
node/ip-172-31-95-119 labeled
ubuntu@ip-172-31-84-221:~$
```

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

```
ubuntu@ip-172-31-84-221:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE    VERSION
ip-172-31-80-164    Ready    Node1           8m2s   v1.28.14
ip-172-31-84-221    Ready    control-plane   38m    v1.28.14
ip-172-31-95-119    Ready    Node2           14m    v1.28.14
ubuntu@ip-172-31-84-221:~$
```

Conclusion: In this experiment, we successfully set up a Kubernetes cluster with one master and two worker nodes on AWS EC2 instances. After installing Docker, Kubernetes tools (kubelet, kubeadm, kubectl), and containerd on all nodes, the master node was initialized and the worker nodes were joined to the cluster. Initially, the nodes were in the NotReady state, which was resolved by installing the Calico network plugin. We also labeled the nodes with appropriate roles (control-plane and worker). The cluster became fully functional with all nodes in the Ready state, demonstrating the successful configuration and orchestration of Kubernetes.

SHRAVANI RASAM D15A 46