

Experiment No : 3

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

PREREQUISITES:

Create 2 Security Groups for Master Node and Worker Nodes and add the following inbound rules in those Groups.

Master Node Security Group:

Basic details

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

Description [Info](#)
Security group for master node

VPC info
vpc-04fadfb026daa5d28

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
HTTP	TCP	80	Anywhere... ▾	0.0.0.0/0 Delete
All traffic	All	All	Anywhere... ▾	0.0.0.0/0 Delete
Custom TCP	TCP	6443	Anywhere... ▾	0.0.0.0/0 Delete
Custom TCP	TCP	10251	Anywhere... ▾	0.0.0.0/0 Delete
Custom TCP	TCP	10250	Anywhere... ▾	0.0.0.0/0 Delete
All TCP	TCP	0 - 65535	Anywhere... ▾	0.0.0.0/0 Delete
Custom TCP	TCP	10252	Anywhere... ▾	0.0.0.0/0 Delete
SSH	TCP	22	Anywhere... ▾	0.0.0.0/0 Delete

Security group (sg-02237795f21a51527 | master_security_node) was created successfully
[Details](#)

EC2 > Security Groups > sg-02237795f21a51527 - master_security_node

sg-02237795f21a51527 - master_security_node

Actions ▾

Details			
Security group name master_security_node	Security group ID sg-02237795f21a51527	Description security group for master node	VPC ID vpc-07187e57bd9cb1f9
Owner 073011525842	Inbound rules count 8 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Tags](#)

Inbound rules (8)

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-0513cc2b25f03b64b	IPv4	Custom TCP	TCP	10250	0.0.0.0/0
<input type="checkbox"/>	-	sgr-050c02db76b670e...	IPv4	Custom TCP	TCP	10251	0.0.0.0/0

Worker Node Security Group :

aws Services Search [Alt+S] N. Virginia v oclabs/user3382213=PRAJAPATI_SHIVAM_ROHITKUMAR @ 3805-5794-4473 ▾

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](#)

Name cannot be edited after creation.

Description [Info](#)

VPC [Info](#)

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
<input type="checkbox"/> CloudShell	<input type="checkbox"/> Feedback			

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Inbound rules Info					
Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
All traffic	All	All	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete
SSH	TCP	22	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete
Custom TCP	TCP	10250	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete
All TCP	TCP	0 - 65535	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete
Custom TCP	TCP	30000 - 32767	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete
HTTP	TCP	80	Anywhere... ▾	<input type="text" value="0.0.0.0/0"/> 0.0.0.0/0 X	Delete

[Add rule](#)

Added Required Inbound rules for worker nodes

⌚ Security group (sg-0b1e0d81a11a7ee55 | worker_node_security) was created successfully
▶ Details

EC2 > [Security Groups](#) > sg-0b1e0d81a11a7ee55 - worker_node_security Actions ▾

Details			
Security group name worker_node_security	Security group ID sg-0b1e0d81a11a7ee55	Description security group for worker node	VPC ID vpc-07187e57bdb9cb1f9
Owner 073011525842	Inbound rules count 6 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Tags](#)

Inbound rules (6)

Inbound rules (6)								
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	
<input type="checkbox"/>	-	sgr-0e1d255da246e6b...	IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0	Edit inbound rules

Step 1: Access your AWS Academy or personal account and create **three new EC2 instances**. For the AMI, choose **Ubuntu** and set the **instance type to t2.medium**. Generate an RSA key in .pem format and move the downloaded key to a new folder for eg: **Newfolder**; you can either use three separate keys or one shared key for all instances ,in my case I have made three different keys .

It's essential to select t2.medium, as it includes at least 2 CPUs, which are required for this setup.. Additionally, make sure to select security groups from the ones already

available i.e master node and worker node will select their own security groups respectively

Master Node:

The screenshot shows the AWS EC2 'Launch an instance' wizard. It consists of two main sections: 'Name and tags' and 'Application and OS Images (Amazon Machine Image)'.

Name and tags

Name: 61_exp_3_Master

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

Recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE

Quick Start: Browse more AMIs (Including AMIs from AWS, Marketplace and the Community)

Selected AMI: Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

Details: 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>).

Architecture: 64-bit (x86)

AMI ID: ami-0e86e20dae9224db8

Username: ubuntu

Verified provider

AMI as Ubuntu

This screenshot shows the first step of the AWS CloudFormation Create New Stack wizard. It's titled "Set instance type and key pair".

- 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
- Key pair (login):** lab3_41_kube
- Network settings:** (Edit)
- Summary:**
 - Number of instances: 1
 - Software Image (AMI): Canonical, Ubuntu, 24.04, amd64... (with a "read more" link)
 - Virtual server type (instance type): t2.medium
 - Firewall (security group): Master-kube
 - Storage (volumes): 1 volume(s) - 8 GiB
- Launch instance** button (orange)
- Review commands**

Generate Instance type and create key pair login

This screenshot is identical to the one above, showing the "Set instance type and key pair" step of the CloudFormation wizard.

- 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
- Key pair (login):** exp3
- Launch instance** button (orange)

Worker Node 1:

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' step, the 'Name' field contains '61_exp_3_worker'. There is a link to 'Add additional tags'.

Give a name to your instance .

The screenshot shows the 'Application and OS Images (Amazon Machine Image)' section. It lists various AMIs including Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE. A search bar at the top says 'Search our full catalog including 1000s of application and OS images'. Below the search bar, there are tabs for 'Recents' and 'Quick Start', with 'Quick Start' selected. A large 'Ubuntu' button is highlighted. To the right, there is a search icon and a link to 'Browse more AMIs'. A detailed view of the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI is shown, including its AMI ID (ami-0e86e20dae9224db8), architecture (64-bit (x86)), and other details like ENA enabled and root device type. The 'Description' section provides a brief overview of the Ubuntu Server 24.04 LTS AMI.

AMI as Ubuntu

▼ 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

exp3

Create new key pair

Created the Key pair login

▼ Network settings [Info](#) Edit

Network [Info](#)
vpc-07187e57bdb9cb1f9

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

worker_node_security sg-0b1e0d81a11a7ee55 X Compare security group rules

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

Select required Security Group

Instances are :

<input type="checkbox"/>	61_exp_3_Mas...	i-05d1cf82d5660367e	Running	t2.medium	
<input type="checkbox"/>	61_exp_3_wor...	i-04343d785c0fca4d4	Running	t2.medium	

Step 2: After creating the instances click on Connect for all the instances one by one and navigate to the SSH Client section .

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	Worker1_41	i-0817bcdpcf6a277c1	Running	t2.medium
<input type="checkbox"/>	Master_41	i-04d83d2955a09bc03	Running	t2.medium
<input type="checkbox"/>	Worker2_41	i-0fe66de19378bcaa1	Running	t2.medium

Master Node:

EC2 > Instances > i-04d83d2955a09bc03 > Connect to instance

Connect to instance Info

Connect to your instance i-04d83d2955a09bc03 (Master_41) using any of these options

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

Instance ID
i-04d83d2955a09bc03 (Master_41)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is lab3_41_kube.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "lab3_41_kube.pem"
4. Connect to your instance using its Public DNS:
ec2-54-158-48-115.compute-1.amazonaws.com

Example:
ssh -i "lab3_41_kube.pem" ubuntu@ec2-54-158-48-115.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.

Step 3: Now open the folder in the terminal 3 times by right clicking on it for Master, Node1 & Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i) from the ssh client section for instance in the terminal as shown above.

This will basically make our terminal to do remote login on our ec2 instance via SSH

Mini_Project	08-03-2024 11:15	File folder
Newfolder	29-09-2024 06:19	File folder
React_Scrimba	21-09-2024 21:16	File folder

MasterNode:

EC2 > Instances > i-04d83d2955a09bc03 > Connect to instance

Connect to instance Info

Connect to your instance i-04d83d2955a09bc03 (Master_41) using any of these options

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

Instance ID
i-04d83d2955a09bc03 (Master_41)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is lab3_41_kube.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "lab3_41_kube.pem"
4. Connect to your instance using its Public DNS:
ec2-54-158-48-115.compute-1.amazonaws.com

Example:
ssh -i "lab3_41_kube.pem" ubuntu@ec2-54-158-48-115.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.

Successful Connection:

```
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Sun Sep 29 00:57:09 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: 172.31.84.76
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sun Sep 29 00:55:44 2024 from 103.87.29.122
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-84-76:~$ |
```

Worker Node 1:

EC2 > Instances > i-0817bcdcbf6a277c1 > Connect to instance

Connect to instance Info

Connect to your instance i-0817bcdcbf6a277c1 (Worker1_41) using any of these options

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

Instance ID
i-0817bcdcbf6a277c1 (Worker1_41)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is lab3_41_W1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "lab3_41_W1.pem"
4. Connect to your instance using its Public DNS:
ec2-3-82-160-230.compute-1.amazonaws.com

Example:
ssh -i "lab3_41_W1.pem" ubuntu@ec2-3-82-160-230.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.

Successful Connection:

```
ubuntu@ip-172-31-93-130: ~
```

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 Sun Sep 29 00:58:10 UTC 2024

System load: 0.02	Processes: 116
Usage of /: 22.9% of 6.71GB	Users logged in: 0
Memory usage: 5%	IPv4 address for enX0: 172.31.93.130
Swap usage: 0%	

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sun Sep 29 00:56:27 2024 from 103.87.29.122
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

```
ubuntu@ip-172-31-93-130:~$ |
```

Worker Node 2:

EC2 > Instances > i-0fe66de19378bcaa1 > Connect to instance

Connect to instance Info

Connect to your instance i-0fe66de19378bcaa1 (Worker2_41) using any of these options

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

Instance ID
i-0fe66de19378bcaa1 (Worker2_41)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is lab3_41_W2.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "lab3_41_W2.pem"
4. Connect to your instance using its Public DNS:
ec2-3-93-79-152.compute-1.amazonaws.com

Example:
ssh -i "lab3_41_W2.pem" ubuntu@ec2-3-93-79-152.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.

Successful Connection:

```
ubuntu@ip-172-31-93-235: ~ + - ×
System information as of Sun Sep 29 00:59:06 UTC 2024
System load: 0.0 Processes: 113
Usage of /: 22.8% of 6.71GB Users logged in: 0
Memory usage: 6% IPv4 address for enX0: 172.31.93.235
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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

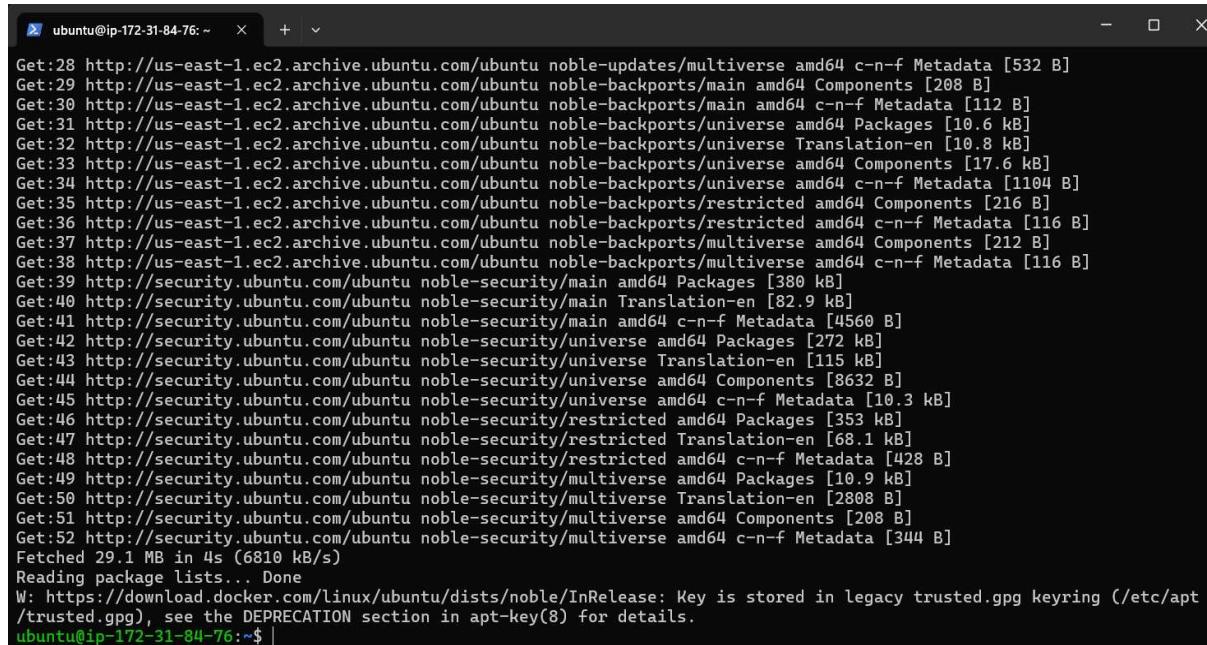
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

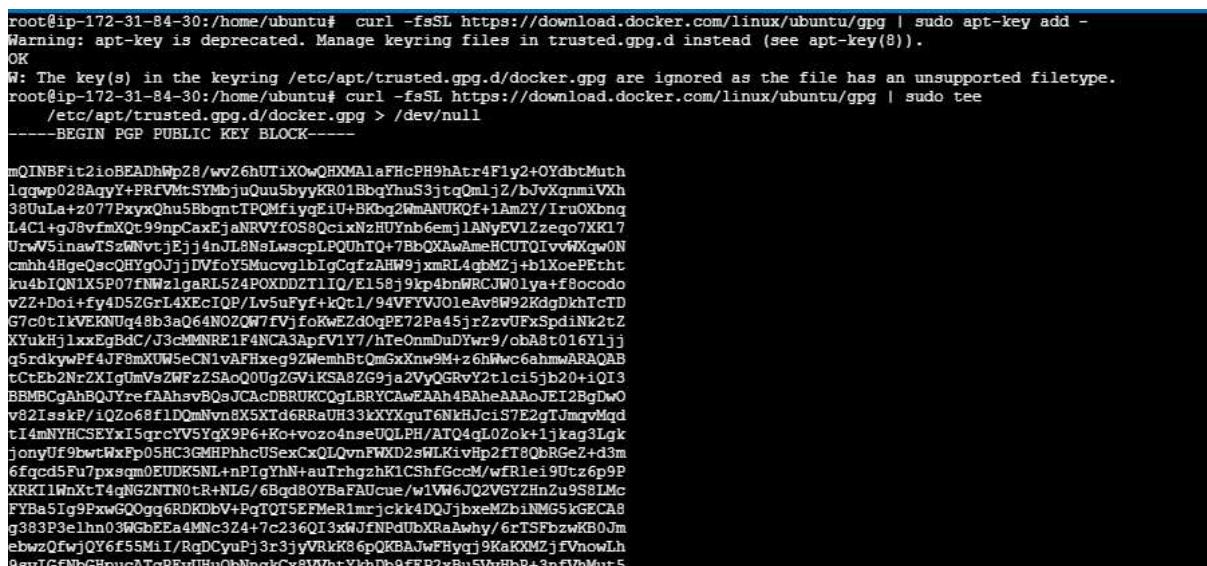
ubuntu@ip-172-31-93-235:~$ |
```

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

a) `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-172-31-84-76: ~ + v
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [10.6 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [10.8 kB]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.6 kB]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:39 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [380 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.9 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4560 B]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [272 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 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 [10.3 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 (6810 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.
ubuntu@ip-172-31-84-76:~$ |
```



```
root@ip-172-31-84-30:/home/ubuntu# 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
W: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
root@ip-172-31-84-30:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
    /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINBFit2ioBEADhWpZ8/wvZ6hUTiXoWQHXMaIaFHcPH9hAtr4F1y2+0YdbtMuth
lgwp02&AqyX+PfRfVfMtSYMbjuOuuSbvyKR01BbqYhuS3jtqQmljZ/bJvXgnmiVXn
38JuLa+z077PxynQhu5BbqntTPQmfiyqEiU+BKbq2WmANUKof+lAmZY/IruOXbng
L4C1+jJ8fmXqT99npCaxXjaNRVYf0S8QcxNzHUYNb6emj1ANyEVl2zeqo7XK17
UrW5inawTSzWnvvtjEj14nJL8NsLwscplPQUhT0+7bbOXAwAmeHCUTQIVvWXqWON
cmhh4HgeQscQHYgOjjDVfoY5Mucvg1bIgCqfzAHW9jxmrL14qbMZj+b1KoePEtht
ku4bIQN1X5P07fNWz1gaRL524POXDDZT1IQ/E158j9kp4bnWRCuW01ya+f8ocodo
vZ2+Doi+fy4D5ZGrL4XEcICP/Lv5uFyt+k0t1/94VFYYVJ0leAvsR92KdgLhTcTD
G7eoCt1KVKNuq48b3aQ64N02QW7fVjfoKwZd0qPE72Pa45j1zzvUFxSpdiNk2tZ
XYukHj1xxEgBdC/J3cmNRE1f4NC43ApfV1Y7/hTeOnmbuDYwr9/obA8t016Y1jj
q5rdkywPf4JF8mXUW5eCN1vAFHxeq9ZWembBtQmGxXnw9M+26hWwc6ahmmARQAAB
tCtEb2NrZXlgUmVs2WFzzSaQoQUGzGViKSA8ZG9ja2VvYQGRvY2t1ci5jb20+iQ13
BBMBCgAhQJYrefaAhsvB9sJCACdBRUKCQgLBRYCAwEEAh4BAheAAoAEJ12BgDwO
v82IsskP/iO2c68f1D0mNvn8X5Xtd6RRaUH33kXYXquT6NkhJc1s7E2gIJmqyMqd
tIAmNHcSEYx15qrcYV5YqX9P6+Ko+vzo4nseUQLPH/ATQ4gL02ok+1jkag3Lgk
jonyUf9bwtpWxfp05HC3GMPhhcUSexCxQLQvnFWX2dswWLKivHpf2fT80bRGeZ+d3m
6fcqd5Fu7pxsgmDEUDK5NL+nPIgYH+N+auTrhgzhK1CSnfGccM/wfRleiu9utz69P
XRK11WnXtT4qNGZNT0tR+NLG/6Bqd8OYBaFAUcue/w1W6JQ2VGY2Hn2u9S8lMc
FYba51g9fxwGOog6RDKDbV+PqTQT5EFMerk1mrjckk4DQJbjxeMzb1MG5kGECA8
g383P3elhn03RGEfEa4MNc324+7c236Q13xWJFNPDubXRaAwh/y/6rTSFbzkwKB0Jm
ebwzQfwjQY6f55MiI/RqDCyuPj3r3jyVRkX86pQKBAJwFHyqj9KaKXMZj1VhovLh
9av1GfNbGhnucaT0REvUHnObNngkCx8VvhrlkhDh9fEP2xRu5VvBbR+3nfvhMu5
```

b) sudo apt-get update

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

```
root@ip-172-31-88-203:/home/ubuntu# 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 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: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg).
details.
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 143 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
```

```
root@ip-172-31-84-30:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg).
details.
root@ip-172-31-84-30:/home/ubuntu# sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
```

i-04343d785c0fca4d4 (61_exp_3_worker)

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

```
root@ip-172-31-88-203:/home/ubuntu# mkdir -p /etc/docker  
root@ip-172-31-88-203:/home/ubuntu# cat <<EOF | sudo tee /etc/docker/daemon.json  
> {  
  >   "exec-opts": ["native.cgroupdriver=systemd"]  
  > }  
> EOF  
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}
```

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

```
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/sysv.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl daemon-reload  
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl restart docker  
root@ip-172-31-88-203:/home/ubuntu# docker -v  
Docker version 27.3.1, build ce12230
```

Step 5: Run the below command to **install Kubernetes** on all the three terminals one by one

a) **sudo rm -f /etc/apt/sources.list.d/kubernetes.list**

```
root@ip-172-31-88-203:/home/ubuntu# rm -f /etc/apt/sources.list.d/kubernetes.list
```

b) **sudo apt-get update**

```
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s)
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
details.
```

i-05d1cf82d5660367e (61_exp_3_Master)

c) **sudo apt-get install -y apt-transport-https ca-certificates curl gpg**

```
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
gpg is already the newest version (2.4.4-2ubuntu17).
gpg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  curl libcurl3t64-gnutls libcurl4t64
3 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 904 kB of archives.
After this operation, 38.9 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14bu
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.5.0-2ubuntu10.4
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl4t64 amd64 8.5.0-2ubu
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libcurl3t64-gnutls amd64 8.5
Fetched 904 kB in 0s (16.1 MB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 68007 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...

```

d) **sudo mkdir -p -m 755 /etc/apt/keyrings**

```
root@ip-172-31-88-203:/home/ubuntu# mkdir -p -m 755 /etc/apt/keyrings
```

f) **curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg**

```
root@ip-172-31-88-203:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

g) 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

```
root@ip-172-31-88-203:/homeecho '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/
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:v1.31/deb InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:v1.31/deb Packages [4865 B]
Fetched 6051 B in 1s (9980 B/s)
```

i-05d1cf82d5660367e (61_exp_3_Master)

PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

f) sudo apt-get update

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

```
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get update  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease  
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease  
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.  
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.  
Fetched 6051 B in 1s (9980 B/s)  
Reading package lists... Done  
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the k  
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in le  
details.  
root@ip-172-31-88-203:/home/ubuntu# sudo apt-get install -y kubelet kubeadm kubectl  
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.  
kubelet set on hold.  
kubeadm set on hold.  
kubectl set on hold.
```

```
root@ip-172-31-88-203:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl  
kubelet set on hold.  
kubeadm set on hold.  
kubectl set on hold.
```

g) sudo systemctl enable --now kubelet

```
root@ip-172-31-88-203:/home/ubuntu# systemctl enable --now kubelet  
root@ip-172-31-88-203:/home/ubuntu# 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 libl  
Use 'sudo apt autoremove' to remove them.
```

```
i-05d1cf82d5660367e (61_exp_3_Master)
```

```
PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203
```

h) sudo apt-get install -y containerd

```
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.
```

i) **sudo mkdir -p /etc/containerd**

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

```
root@ip-172-31-88-203:/home/ubuntu# sudo mkdir -p /etc/containerd
root@ip-172-31-88-203:/home/ubuntu# 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
```

j) sudo systemctl restart containerd
 sudo systemctl enable containerd
 sudo systemctl status containerd

```
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl restart containerd
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl enable containerd
root@ip-172-31-88-203:/home/ubuntu# sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-10-01 05:55:35 UTC; 18s ago
       Docs: https://containerd.io
   Main PID: 5525 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 13.8M)
        CPU: 126ms
      CGroup: /system.slice/containerd.service
              └─5525 /usr/bin/containerd

Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.342731956Z" level=err
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.3429271822" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.3429763992" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.3429988392" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.343024047Z" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.343064226Z" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.343095363Z" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.343104271Z" level=info
Oct 01 05:55:35 ip-172-31-88-203 containerd[5525]: time="2024-10-01T05:55:35.343109924Z" level=info
```

k) sudo apt-get install -y socat

```
ubuntu@ip-172-31-84-76:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (15.2 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68112 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.
```

```
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.
```

Step 6: Set up the Kubernetes cluster by running the following command exclusively on the master node:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16. This command initializes the Kubernetes control plane and specifies the pod network range to be used within the cluster i.e kubeadm will initialize the kubernetes cluster and cidr is Classless Inter Domain Range which decides the range of network range of the pods

```
root@ip-172-31-88-203:/home/ubuntu# 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 conn  
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'  
W1001 05:56:40.054582      5726 checks.go:846] detected that the sandbox image "registry.k8s.io/k8s.gcr.io/pause:3.10" is not available. 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-172-31-88-203 kubernetes kubernetes.172.31.88.203]  
[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
```

i-05d1cf82d5660367e (61_exp_3_Master)

Public IPs: 3.83.154.116 Private IPs: 172.31.88.203

Run this command on master:

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

These commands create a .kube directory in your home folder, copy the Kubernetes admin configuration file into it, and change the ownership of that configuration file to the current user to ensure proper access.

```
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 172.31.88.203:6443 --token ef5xtx.olbe4aouwj2i4fdf \
--discovery-token-ca-cert-hash sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76
root@ip-172-31-88-203:/home/ubuntu# mkdir -p $HOME/.kube
root@ip-172-31-88-203:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@ip-172-31-88-203:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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

command on nodes. kubectl is a command-line tool used to interact with and manage Kubernetes clusters. It allows users to deploy applications, inspect and manage cluster resources, and view logs, among other tasks, by sending commands to the Kubernetes API server.

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes
NAME           STATUS    ROLES      AGE   VERSION
ip-172-31-88-203   NotReady control-plane   57s   v1.31.1
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master. For this from the kubeadm init command output for the master node copy and paste this on both the node

```
[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 172.31.84.76:6443 --token 6zxtbp.go2qmwge82ih7w9t \
    --discovery-token-ca-cert-hash sha256:6b96e72028e4e3b98fc453d2b2f41f3c3ee9a013155c2dc9d2dda313bd2bc88
ubuntu@ip-172-31-84-76:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-84-76:~$ kubectl get nodes
NAME           STATUS    ROLES      AGE     VERSION
ip-172-31-84-76   NotReady   control-plane   6m42s   v1.31.1
ubuntu@ip-172-31-84-76:~$ |
```

Copy and paste: **kubeadm join 172.31.88.203:6443 --token ef5xtx.okbe4aouwj2i4fdf**

|

--discovery-token-ca-cert-hash

sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76

Worker Node 1:

```
root@ip-172-31-84-30:/home/ubuntu# kubeadm join 172.31.88.203:6443 --token ef5xtx.okbe4aouwj2i4fdf \
--discovery-token-ca-cert-hash sha256:71074273c6192db966a142598ef7d65305691636883a1cf66233df241d4c7c76
[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 1.001611728s
[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.
root@ip-172-31-84-30:/home/ubuntu# |||
```

i-04343d785c0fca4d4 (61_exp_3_worker)

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

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERFACES
ip-172-31-84-30   Ready   Worker_Node1_61   2m43s   v1.31.1   172.31.84.30
ip-172-31-88-203   Ready   control-plane   13m     v1.31.1   172.31.88.203
```

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> . This command applies the Calico network plugin configuration, which enables networking capabilities in the Kubernetes cluster .It enhances the effective communication between pods in network

```
root@ip-172-31-88-203:/home/ubuntu# 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
```

i-05d1cf82d5660367e (61_exp_3_Master)

PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

Now perform **sudo systemctl status kubelet**

```
root@ip-172-31-88-203:/home/ubuntu# 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 Tue 2024-10-01 05:56:58 UTC; 11min ago
      Docs: https://kubernetes.io/docs/
   Main PID: 6396 (kubelet)
     Tasks: 10 (limit: 4676)
    Memory: 32.5M (peak: 33.0M)
```

i-05d1cf82d5660367e (61_exp_3_Master)

PublicIPs: 3.83.154.116 PrivateIPs: 172.31.88.203

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

```
ubuntu@ip-172-31-84-76:~$ kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE        KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-84-76   Ready    control-plane   39m   v1.31.1  172.31.84.76   <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
ip-172-31-93-130  Ready    <none>      12m   v1.31.1  172.31.93.130  <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
ip-172-31-93-235  Ready    <none>      8m45s  v1.31.1  172.31.93.235  <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
ubuntu@ip-172-31-84-76:~$ |
```

Now to Rename run this command

Rename to Worker_Node1_61: kubectl label node ip-172-31-84-30
kubernetes.io/role=Worker_Node_61

Make Sure to give Correct corresponding IP while renaming which can be seen from previous output also.

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE        KERNEL-VERSION   CONTAINER-RUNTIME
CONTAINER-RUNTIME
ip-172-31-84-30   Ready    Worker_Node1_61      3m47s  v1.31.1  172.31.84.30   <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
ip-172-31-88-203  Ready    Master_Node_61,control-plane  14m   v1.31.1  172.31.88.203  <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
root@ip-172-31-88-203:/home/ubuntu# |
```

```
i-05d1cf82d5660367e (61_exp_3_Master)
Public IPs: 3.83.154.116 Private IPs: 172.31.88.203
```

Step 11: Run command **kubectl get nodes -o wide** . And Hence we can see we have Successfully connected both the Worker Nodes to the Master Node.

```
root@ip-172-31-88-203:/home/ubuntu# kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE        KERNEL-VERSION   CONTAINER-RUNTIME
CONTAINER-RUNTIME
ip-172-31-84-30   Ready    Worker_Node1_61      3m47s  v1.31.1  172.31.84.30   <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
ip-172-31-88-203  Ready    Master_Node_61,control-plane  14m   v1.31.1  172.31.88.203  <none>       Ubuntu 24.04 LTS  6.8.0-1012-aws  containerd://1.7.12
root@ip-172-31-88-203:/home/ubuntu# |
```

```
i-05d1cf82d5660367e (61_exp_3_Master)
Public IPs: 3.83.154.116 Private IPs: 172.31.88.203
```

CONCLUSION:

In this experiment, we established a Kubernetes cluster on AWS EC2 instances, consisting of one master node and two worker nodes. After installing Docker and the necessary Kubernetes tools (kubelet, kubeadm, kubectl, and containerd) on all nodes, we initialized the master and added the worker nodes to the cluster. Initially, the nodes showed a NotReady status, which we resolved by installing the Calico network plugin. We also tagged the nodes with their respective roles (control-plane and worker). Ultimately, all nodes transitioned to the Ready state, demonstrating our successful setup and management of the Kubernetes cluster.