Anushka Shahane D15A 55

Exp 3 Advance Devops

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

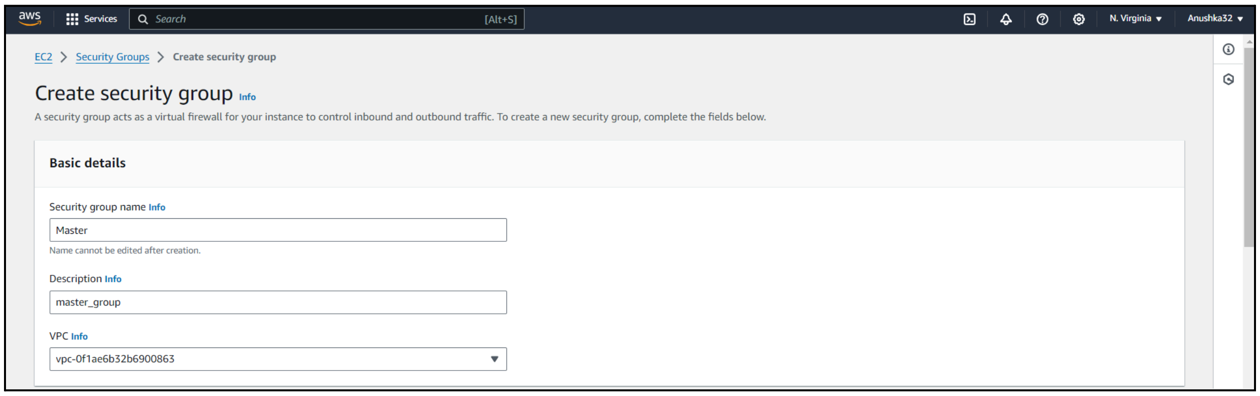
Theory:
Container-based microservices architectures have revolutionized how development and operations
teams test and deploy modern software. Containers allow companies to scale and deploy applications
more efficiently, but they also introduce new challenges, adding complexity by creating a whole new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

Anushka Shahane D15A 55

This ensured that the Kubernetes cluster had the necessary resources to function smoothly

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

master



Create security group Info

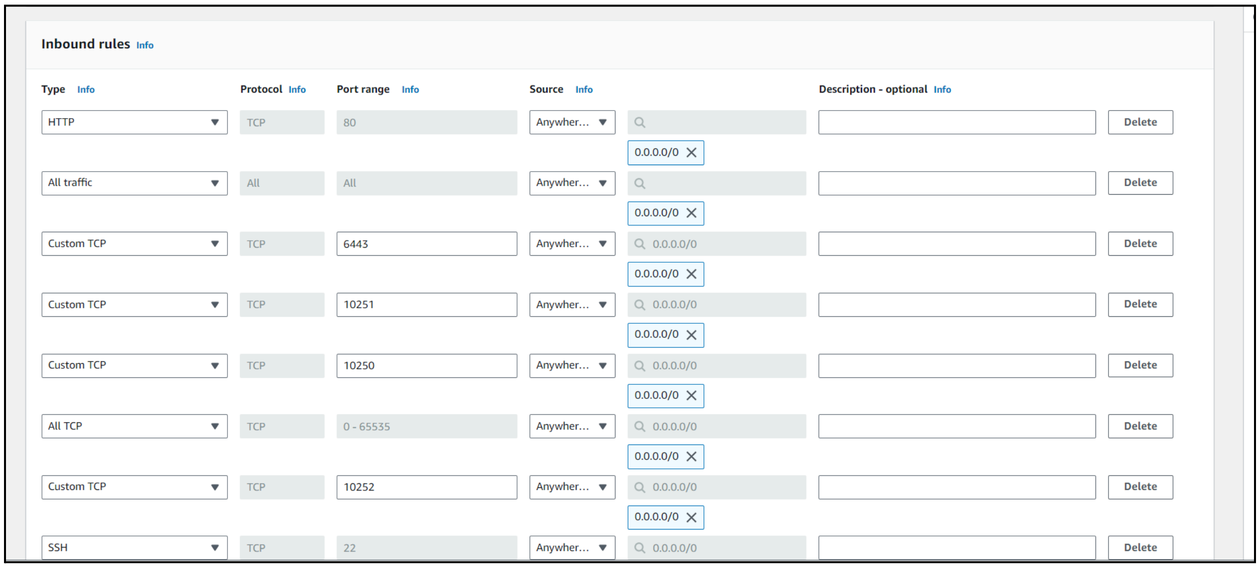
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info: Master
Name cannot be edited after creation.

Description Info: master_group

VPC Info: vpc-0f1ae6b32b6900863

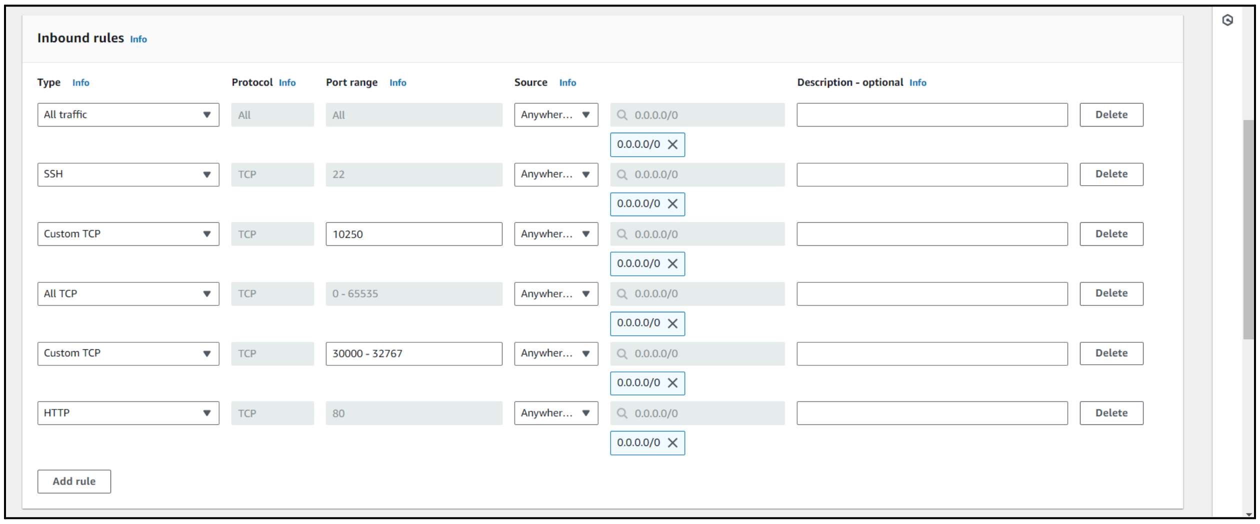


Inbound rules Info

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

Anushka Shahane D15A 55

Node

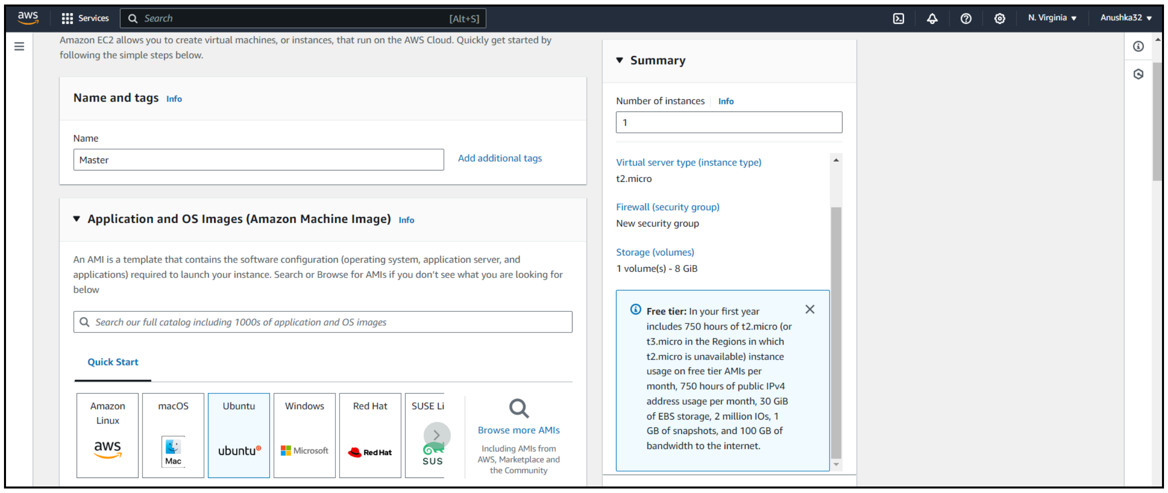


Inbound rules Info

Type	Protocol	Port range	Source		Description - optional
All traffic	All	All	Anywhere...	0.0.0.0/0	
SSH	TCP	22	Anywhere...	0.0.0.0/0	
Custom TCP	TCP	10250	Anywhere...	0.0.0.0/0	
All TCP	TCP	0 - 65535	Anywhere...	0.0.0.0/0	
Custom TCP	TCP	30000 - 32767	Anywhere...	0.0.0.0/0	
HTTP	TCP	80	Anywhere...	0.0.0.0/0	

Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances.
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 3 Different keys or 1 common
key also.
Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the
instance after the experiment because it is not available in the free tier.
Also Select Security groups from existing.
Master:

Anushka Shahane D15A 55



Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

Master

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

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

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Summary

Number of instances Info

1

Virtual server type (instance type)

t2.micro

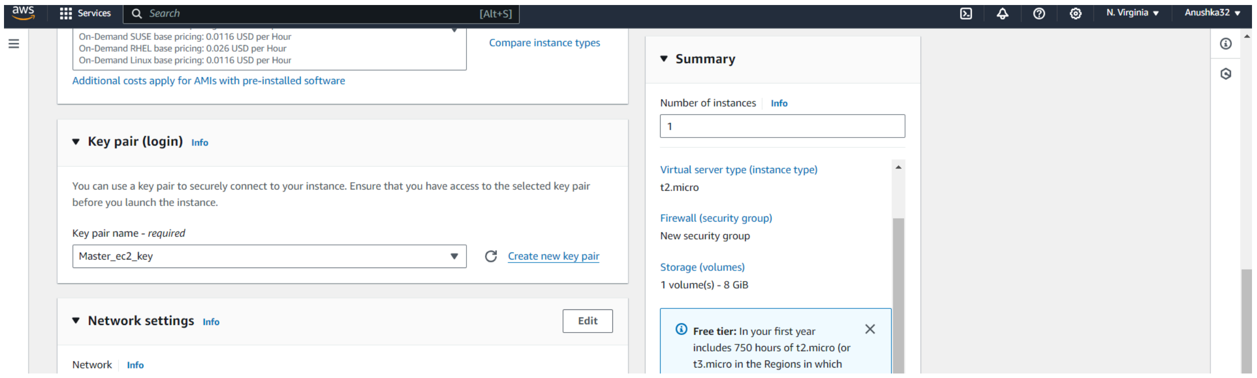
Firewall (security group)

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 IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.



On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

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

Summary

Number of instances Info

1

Virtual server type (instance type)

t2.micro

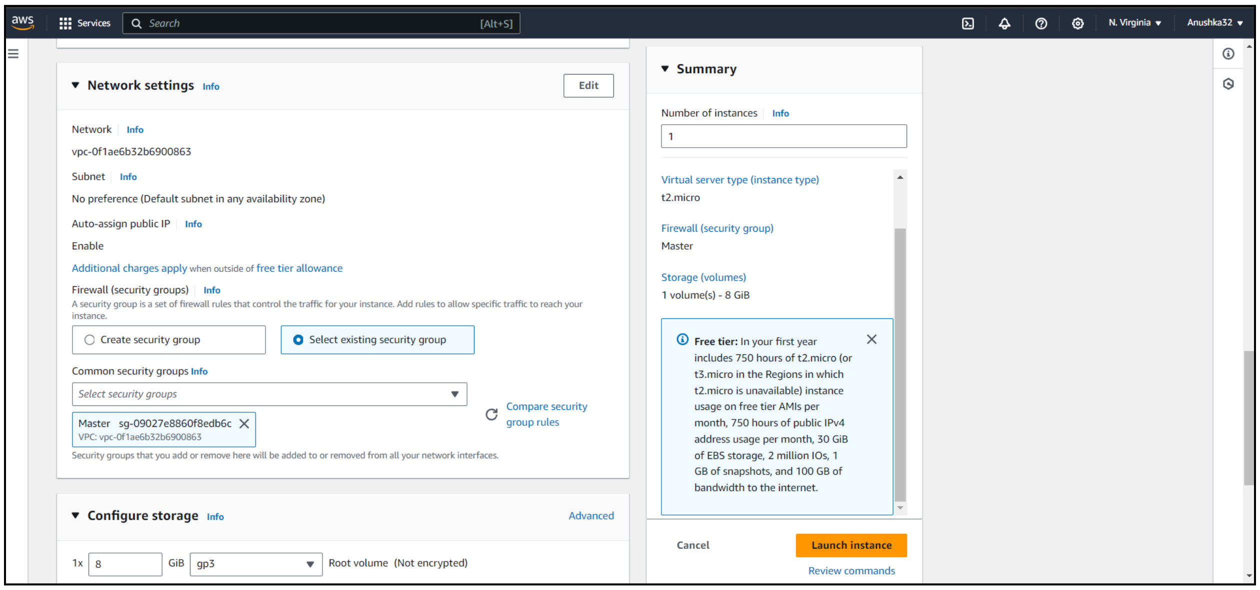
Firewall (security group)

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



Network settings Info

Edit

Network Info

vpc-0f1ae6b32b6900863

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-09027e8860f8edb6c

VPC: vpc-0f1ae6b32b6900863

Compare security group rules

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

Configure storage Info

Advanced

1x 8 GiB gp3 Root volume (Not encrypted)

Summary

Number of instances Info

1

Virtual server type (instance type)

t2.micro

Firewall (security group)

Master

Storage (volumes)

1 volume(s) - 8 GiB

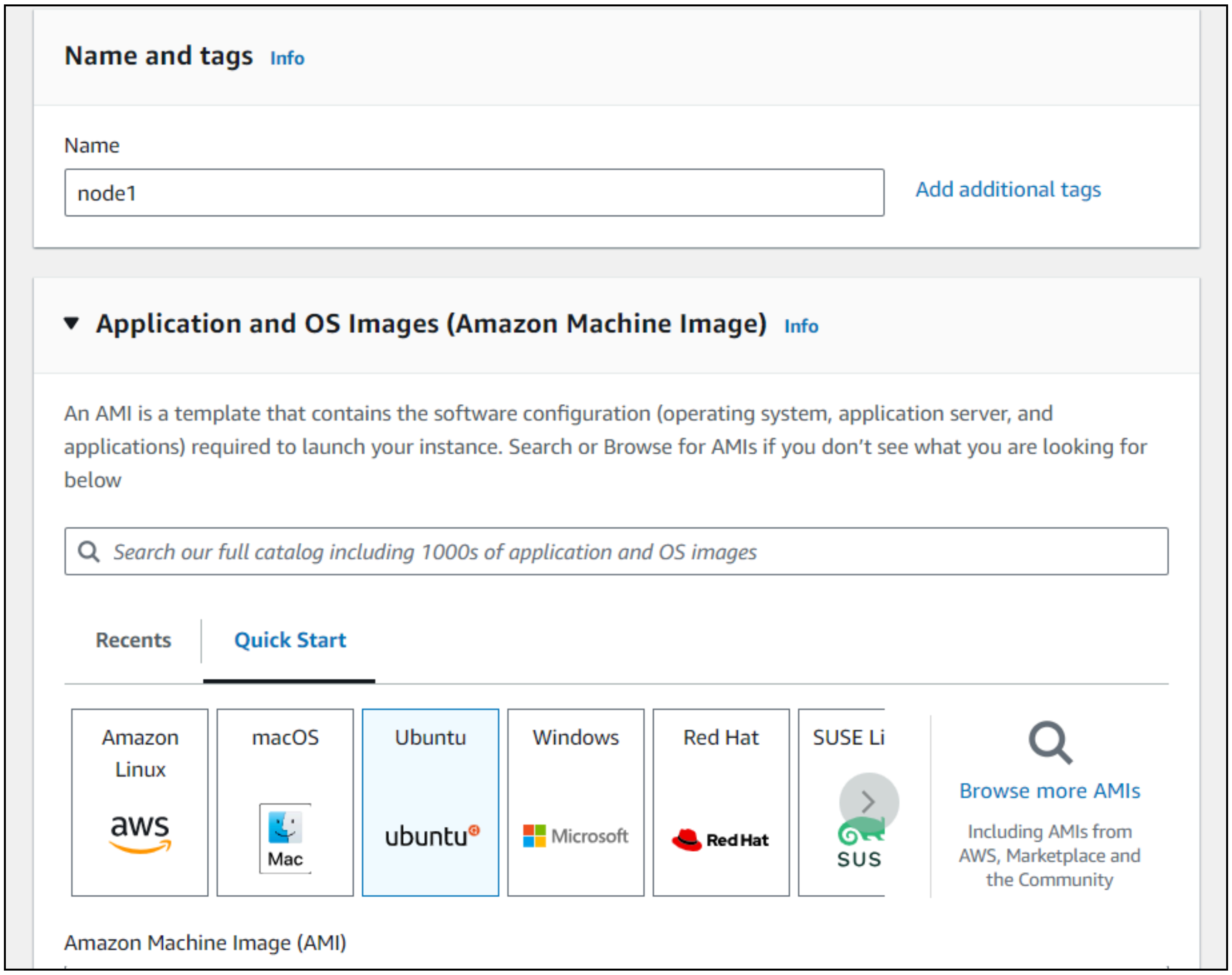
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel

Launch instance

Review commands

Anushka Shahane D15A 55



Name and tags Info

Name

node1

Add additional tags

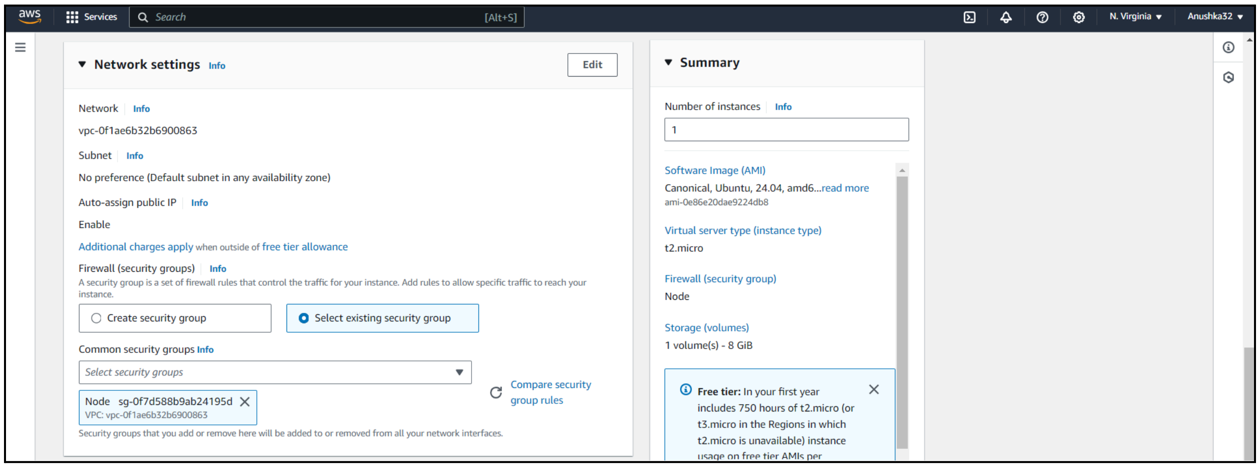
Application and OS Images (Amazon Machine Image) Info

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 Machine Image (AMI)



Network settings Info

Network Info

vpc-0f1ae6b32b6900863

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

Node sg-0f7d588b9ab24195d
VPC: vpc-0f1ae6b32b6900863

Compare security group rules

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

Summary

Number of instances Info

1

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8

Virtual server type (instance type)
t2.micro

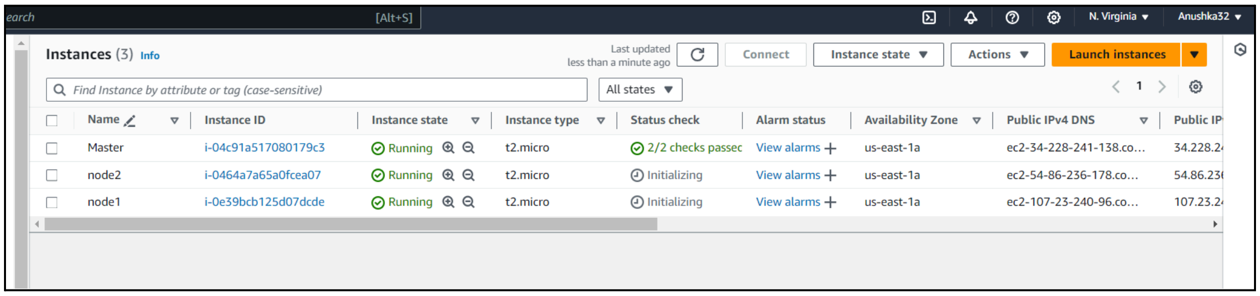
Firewall (security group)
Node

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

Anushka Shahane D15A 55

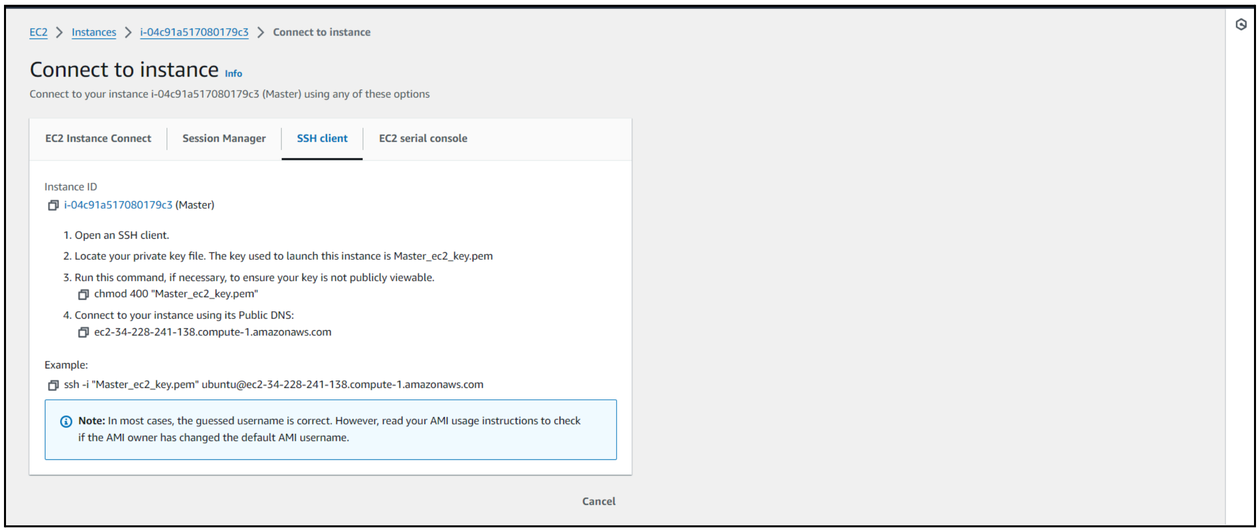
Do Same for 2 Nodes and use security groups of Node for that.
Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH
Client.



Instances (3) Info

Last updated less than a minute ago

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
	Master	i-04c91a517080179c3	Running	t2.micro	2/2 checks passec	View alarms	us-east-1a	ec2-34-228-241-138.co...	34.228.2
	node2	i-0464a7a65a0fcea07	Running	t2.micro	Initializing	View alarms	us-east-1a	ec2-54-86-236-178.co...	54.86.23
	node1	i-0e39bcb125d07dcde	Running	t2.micro	Initializing	View alarms	us-east-1a	ec2-107-23-240-96.co...	107.23.2



EC2 > Instances > i-04c91a517080179c3 > Connect to instance

Connect to instance Info

Connect to your instance i-04c91a517080179c3 (Master) using any of these options

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

Instance ID
i-04c91a517080179c3 (Master)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Master_ec2_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
```
chmod 400 "Master_ec2_key.pem"
```
4. Connect to your instance using its Public DNS:
```
ec2-34-228-241-138.compute-1.amazonaws.com
```

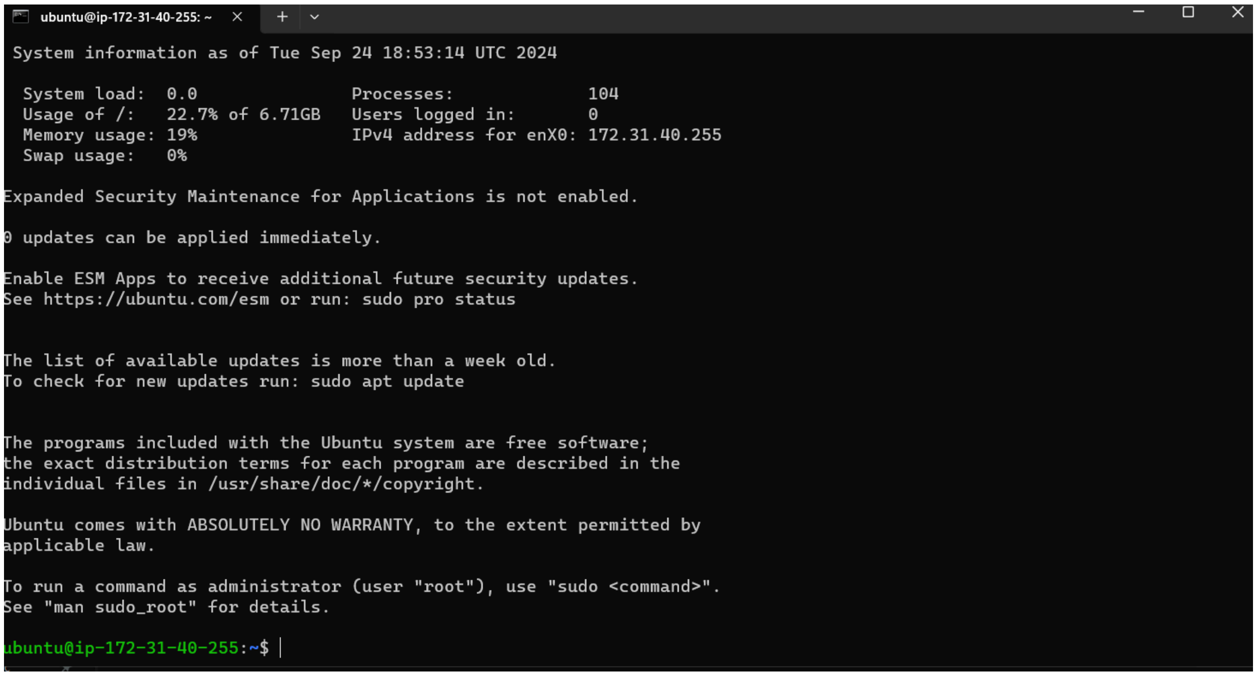
Example:
```
ssh -i "Master_ec2_key.pem" ubuntu@ec2-34-228-241-138.compute-1.amazonaws.com
```

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.

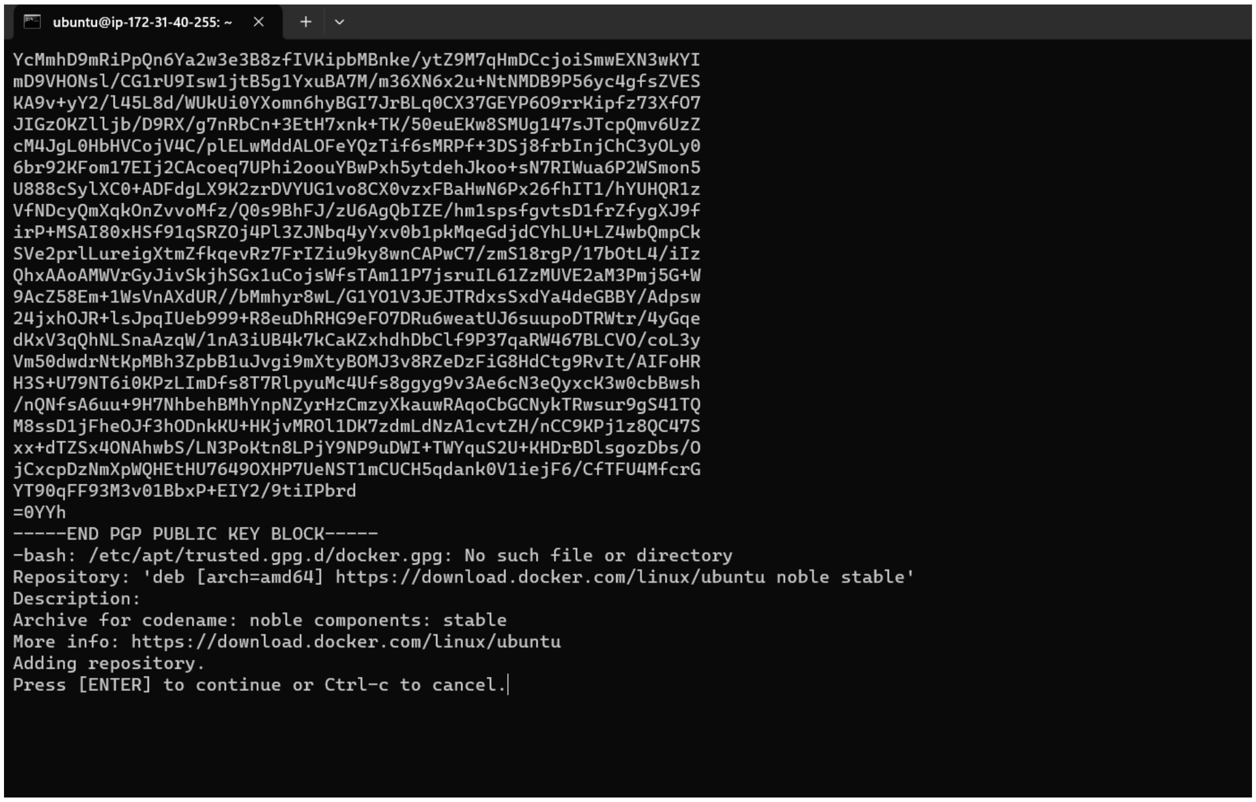
Cancel

Step 3: 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 (starting with ssh -i .....) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)
Master:

Anushka Shahane D15A 55

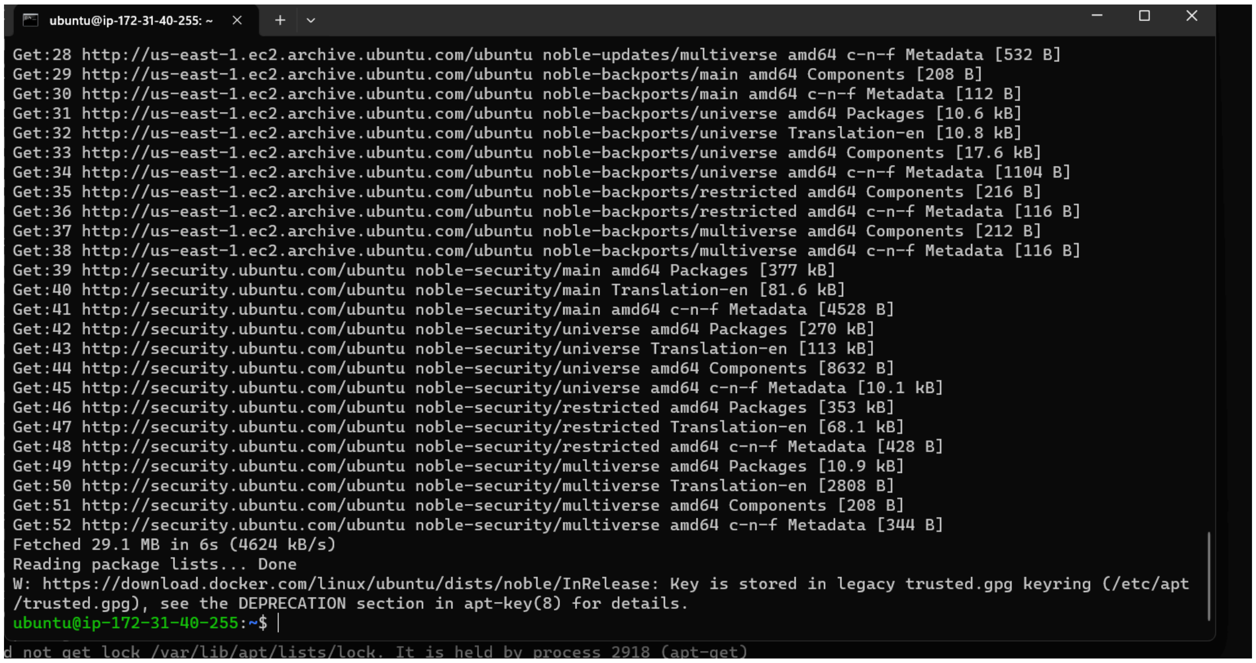


```
System information as of Tue Sep 24 18:53:14 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 19%               IPv4 address for enX0: 172.31.40.255
  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-172-31-40-255:~$
```



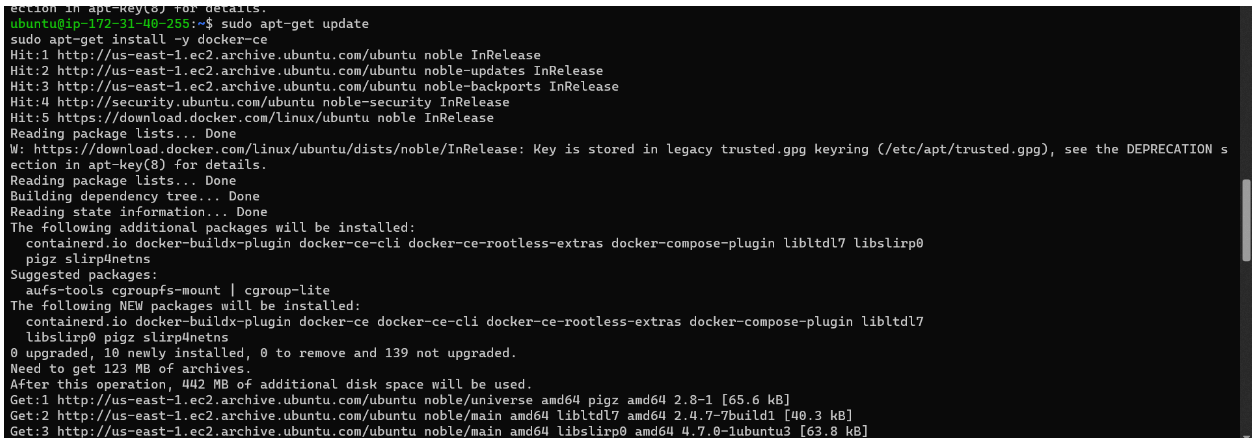
```
YcMmhD9mRiPpQn6Ya2w3e3B8zfIVKipbMBnke/ytZ9M7qHmDCcjoiSmwEXN3wKYI
mD9VHONsl/CG1rU9Isw1jtB5g1YxuBA7M/m36XN6x2u+NtNMDB9P56yc4gfsZVES
KA9v+yY2/l45L8d/WUkUi0YXomn6hyBGI7JrBLq0CX37GEYP6O9rrKipfz73XfO7
JIGzOKZlljb/D9RX/g7nRbCn+3EtH7xnk+TK/50euEKw8SMUg147sJTcpQmv6UzZ
cM4JgL0HbHVCojV4C/plELwMddALOFeYQzTif6sMRPf+3DSj8frbInjChC3yOLy0
6br92KFom17EIj2CAcoeq7UPhi2oouYBwPxh5ytdehJkoo+sN7RIWua6P2WSmon5
U888cSylXC0+ADFdgLX9K2zrDVYUG1vo8CX0vzxFBaHwN6Px26fhIT1/hYUHQR1z
VfNDcyQmXqkOnZvvoMfz/Q0s9BhFJ/zU6AgQbIZE/hm1spsfgvtsD1frZfygXJ9f
irP+MSAI80xHSf91qSRZOj4Pl3ZJNbq4yYxv0b1pkMqeGdjdCYhLU+LZ4wbQmpCk
SVe2prlLureigXtmZfkqevRz7FrIZiu9ky8wnCAPwC7/zmS18rgP/17bOtL4/iIz
QhxAAoAMWVrGyJivSkjhSGx1uCojsWfsTAm11P7jsruIL61ZzMUVE2aM3Pmj5G+W
9AcZ58Em+1WsVnAXdUR//bMmhyr8wL/G1YO1V3JEJTRdxsSxdYa4deGBBY/Adpsw
24jxhOJR+lsJpqIUeb999+R8euDhRHG9eFO7DRu6weatUJ6suupoDTRWtr/4yGqe
dKxV3qQhNLSnaAzqW/1nA3iUB4k7kCaKZxhdhDbClf9P37qaRW467BLCVO/coL3y
Vm50dwdrNtKpMBh3ZpbB1uJvgi9mXtyBOMJ3v8RZeDzFiG8HdCtg9RvIt/AIFoHR
H3S+U79NT6i0KPzLImDfs8T7RlpyuMc4Ufs8ggyg9v3Ae6cN3eQyxcK3w0cbBwsh
/nQNfsA6uu+9H7NhbehBMhYnpNZyrHzCmzyXkauwRAqoCbGCNykTRwsur9gS41TQ
M8ssD1jFheOJf3hODnkKU+HKjvMROl1DK7zdmLdNzA1cvtZH/nCC9KPj1z8QC47S
xx+dTZSx4ONAhwbS/LN3PoKtn8LPjY9NP9uDWI+TWYquS2U+KHDrBDlsgozDbs/O
jCxcpDzNmXpWQHEtHU7649OXHP7UeNST1mCUCH5qdank0V1iejF6/CfTFU4MfcrG
YT90qFF93M3v01BbxP+EIY2/9tiIPbrd
=0YYh
-----END PGP PUBLIC KEY BLOCK-----
-bash: /etc/apt/trusted.gpg.d/docker.gpg: No such file or directory
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

Anushka Shahane D15A 55



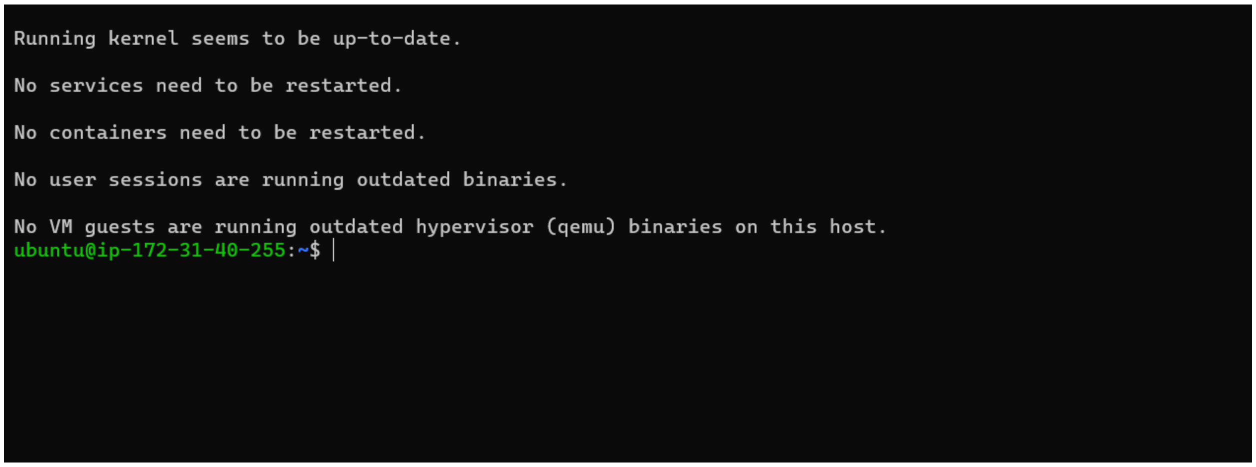
```
ubuntu@ip-172-31-40-255: ~
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [532 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main 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 [377 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [81.6 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4528 B]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [270 kB]
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 6s (4624 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-172-31-40-255:~$ |
d not get lock /var/lib/apt/lists/lock. It is held by process 2918 (apt-get)
```

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



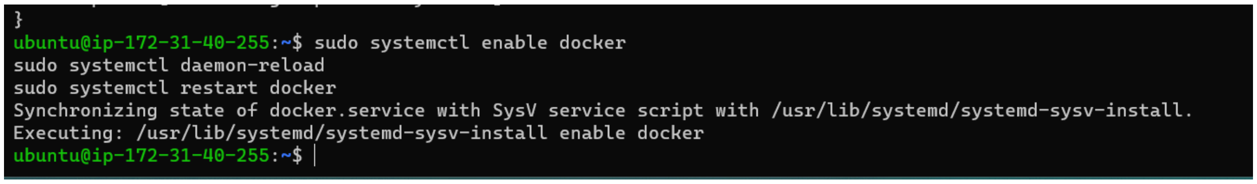
```
ection in apt-key(8) for details.
ubuntu@ip-172-31-40-255:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION s
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:
  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-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 139 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]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
```

Anushka Shahane D15A 55



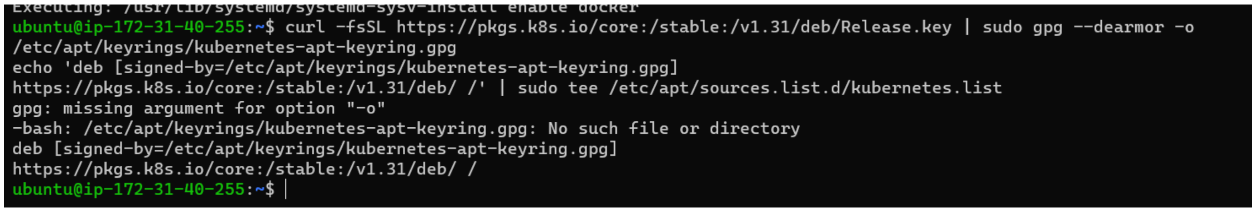
```
Running kernel seems to be up-to-date.

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-172-31-40-255:~$ |
```

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



```
}
ubuntu@ip-172-31-40-255:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-255:~$ |
```

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"

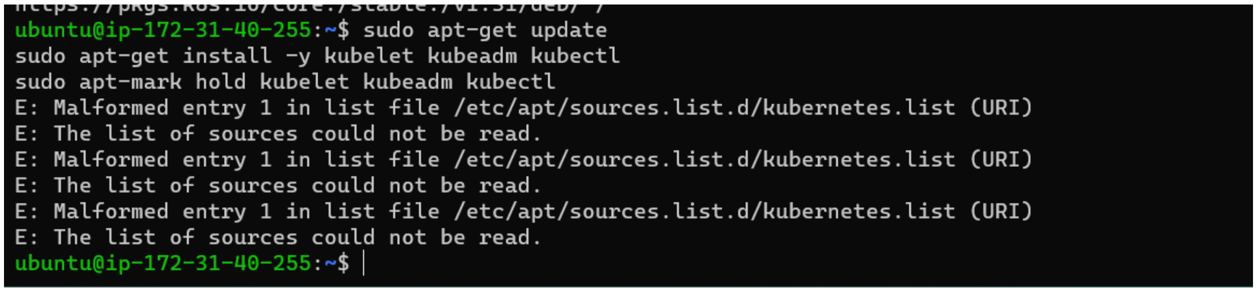


```
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-255:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | 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-172-31-40-255:~$ |
```

Anushka Shahane D15A 55

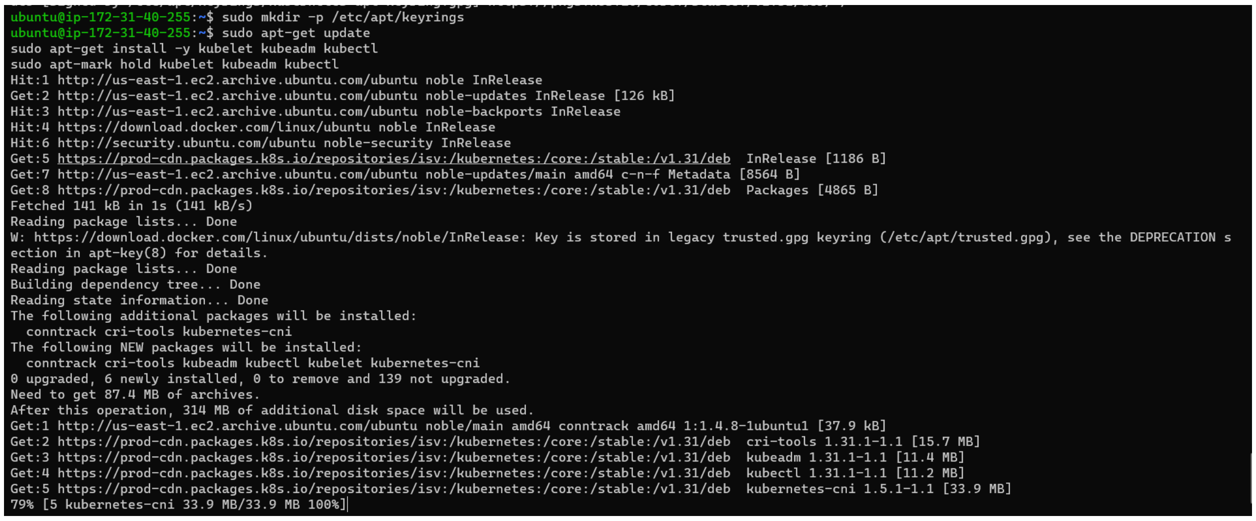
```
sudo apt-get update
sudo apt-get install -y docker-ce
```

Error

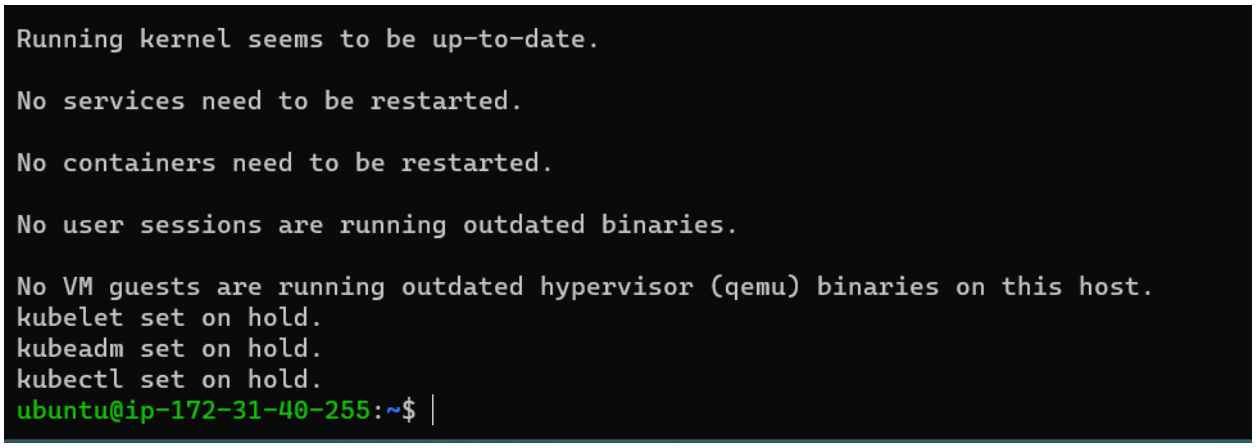


```
ubuntu@ip-172-31-40-255:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
ubuntu@ip-172-31-40-255:~$
```

To solve



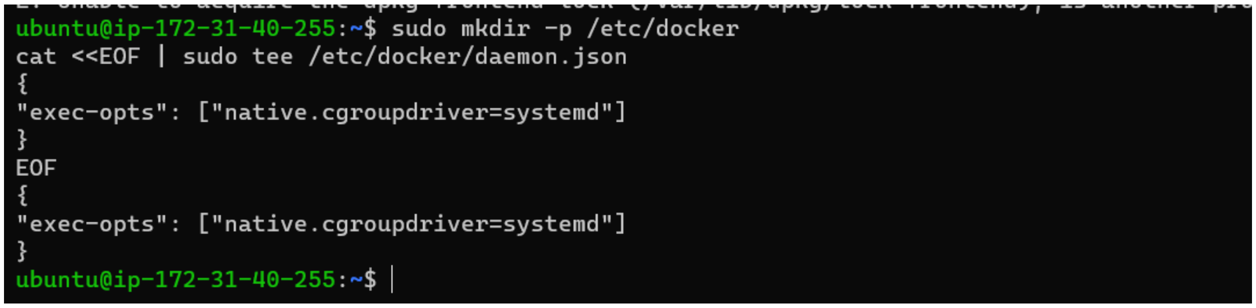
```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-40-255:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
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:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [8564 B]
Get:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 141 kB in 1s (141 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 s
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.
After this operation, 314 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:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
79% [5 kubernetes-cni 33.9 MB/33.9 MB 100%]
```



```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-40-255:~$
```

Anushka Shahane D15A 55

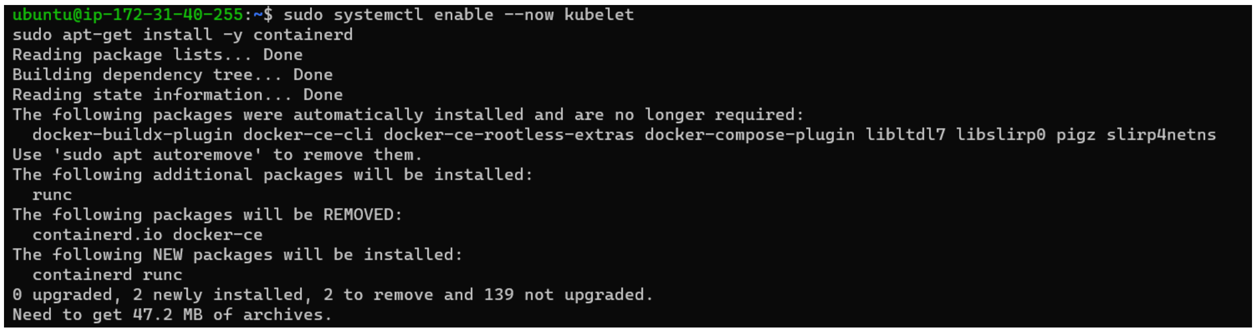
```
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```



```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-255:~$
```

Step 5: Run the below command to install Kubernets.

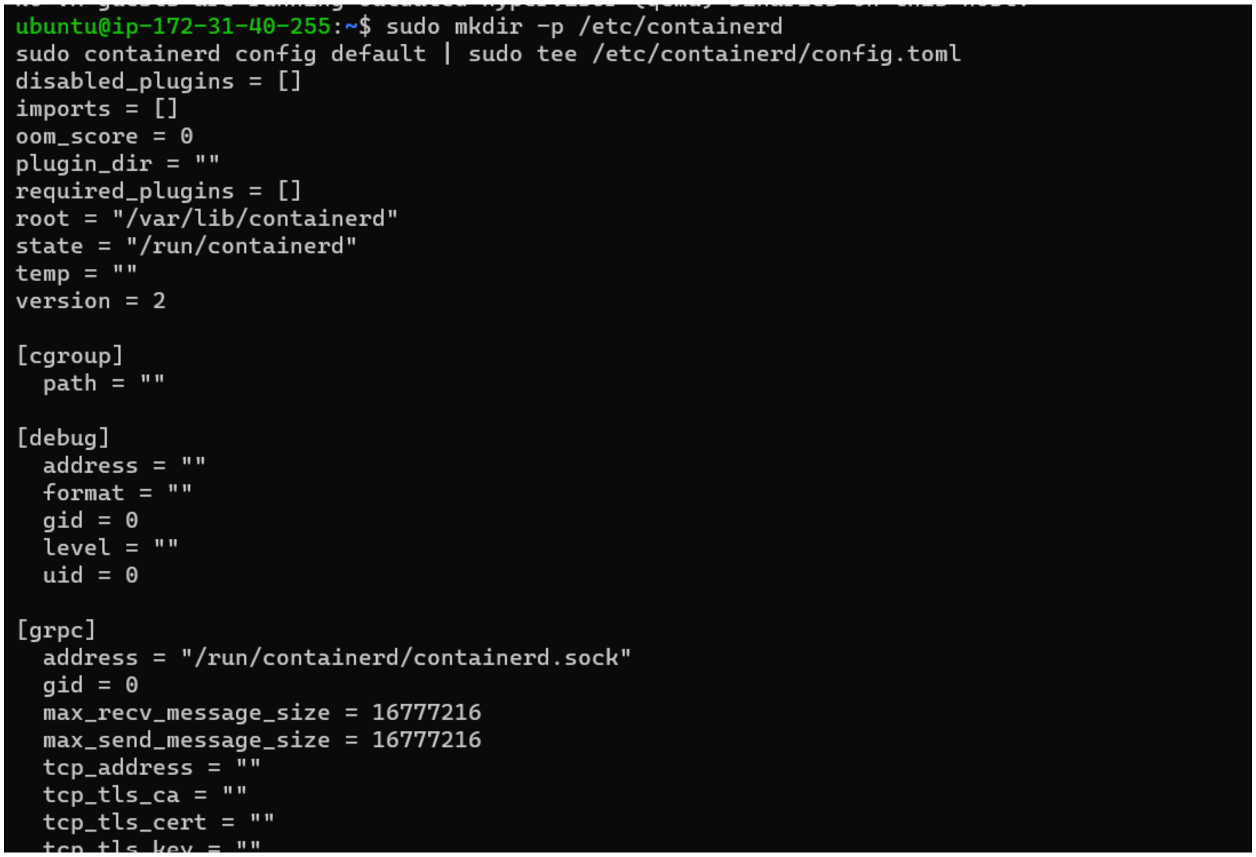
```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```



```
ubuntu@ip-172-31-40-255:~$ sudo systemctl enable --now kubelet
sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 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 139 not upgraded.
Need to get 47.2 MB of archives.
```

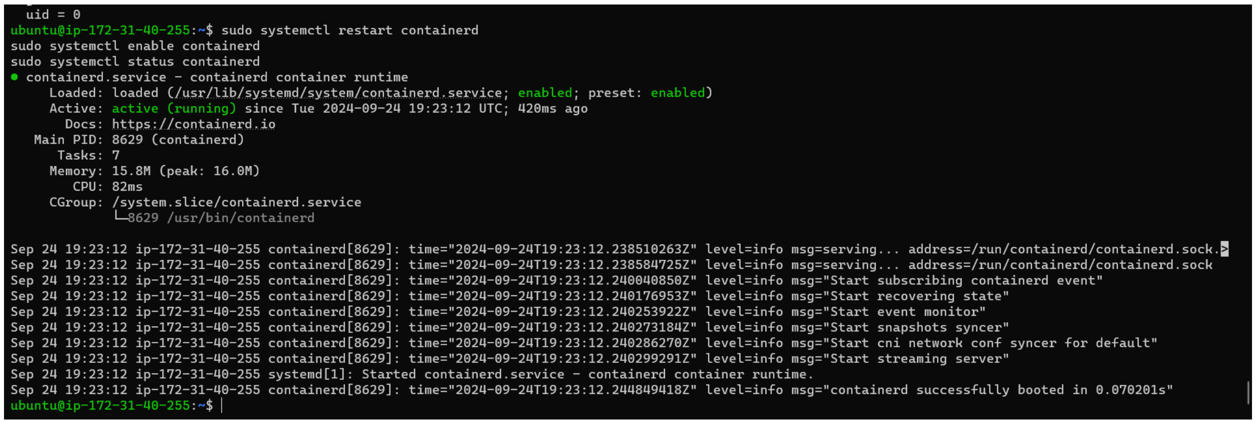
```
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

Anushka Shahane D15A 55



```
ubuntu@ip-172-31-40-255:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
```

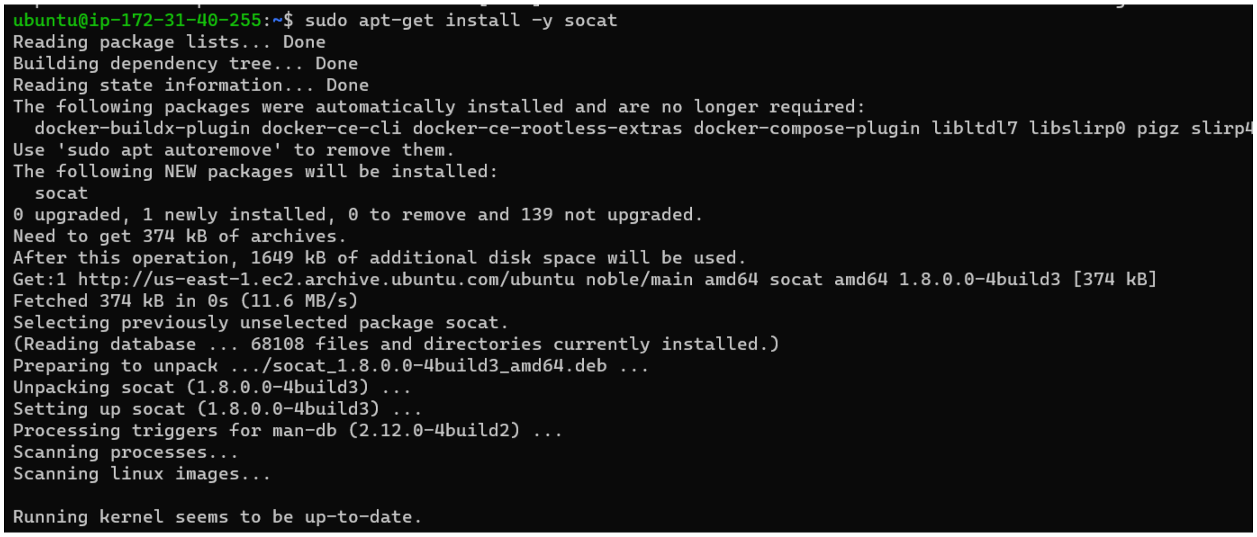
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd



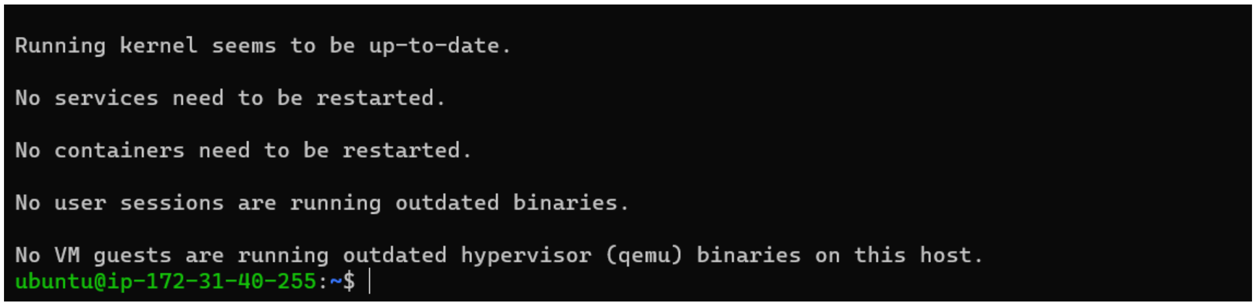
```
  uid = 0
ubuntu@ip-172-31-40-255:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-09-24 19:23:12 UTC; 420ms ago
       Docs: https://containerd.io
   Main PID: 8629 (containerd)
      Tasks: 7
     Memory: 15.8M (peak: 16.0M)
        CPU: 82ms
     CGroup: /system.slice/containerd.service
             └─8629 /usr/bin/containerd

Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.238510263Z" level=info msg=serving... address=/run/containerd/containerd.sock.>
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.238584725Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240040850Z" level=info msg="Start subscribing containerd event"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240176953Z" level=info msg="Start recovering state"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240253922Z" level=info msg="Start event monitor"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240273184Z" level=info msg="Start snapshots syncer"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240286270Z" level=info msg="Start cni network conf syncer for default"
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.240299291Z" level=info msg="Start streaming server"
Sep 24 19:23:12 ip-172-31-40-255 systemd[1]: Started containerd.service - containerd container runtime.
Sep 24 19:23:12 ip-172-31-40-255 containerd[8629]: time="2024-09-24T19:23:12.244849418Z" level=info msg="containerd successfully booted in 0.070201s"
ubuntu@ip-172-31-40-255:~$
```

sudo apt-get install -y socat

Anushka Shahane D15A 55

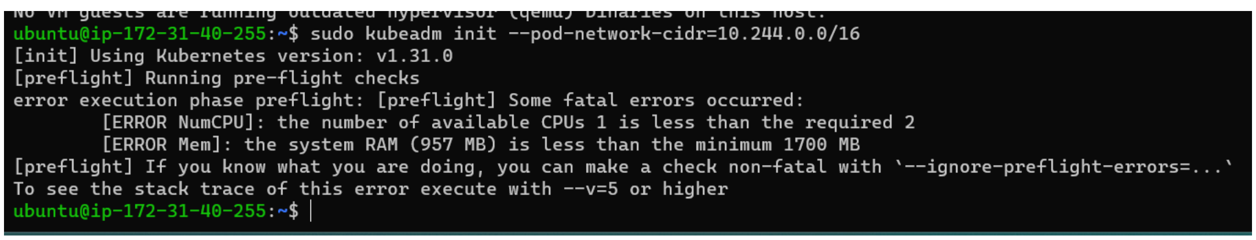


```
ubuntu@ip-172-31-40-255:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4
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 amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (11.6 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.
```



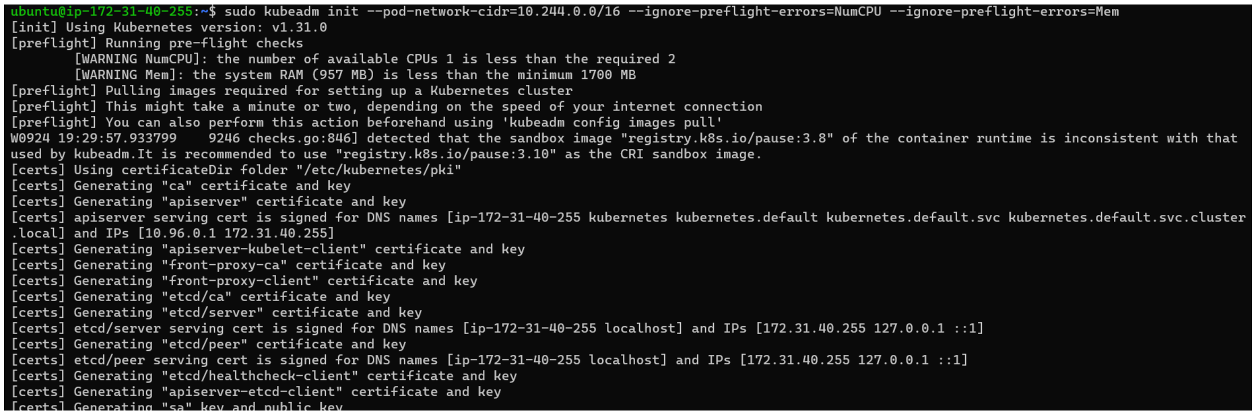
```
Running kernel seems to be up-to-date.

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-172-31-40-255:~$ 
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

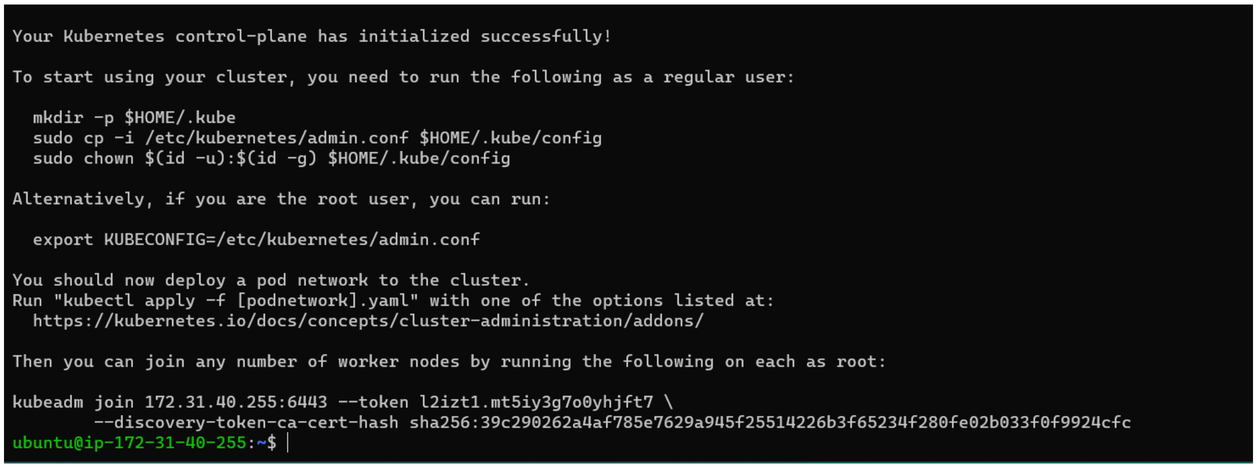


```
ubuntu@ip-172-31-40-255:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR NumCPU]: the number of available CPUs 1 is less than the required 2
        [ERROR Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-40-255:~$ 
```

Anushka Shahane D15A 55

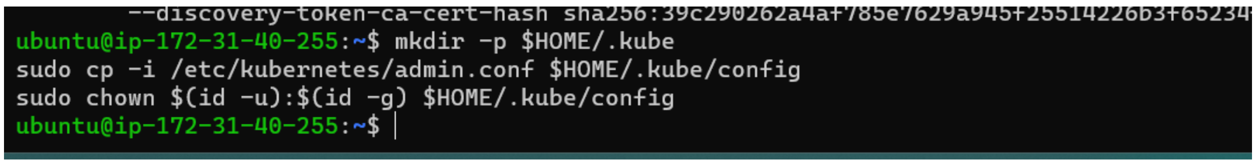


```
ubuntu@ip-172-31-40-255:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
[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'
W0924 19:29:57.933799    9246 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that
used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-40-255 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.40.255]
[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-172-31-40-255 localhost] and IPs [172.31.40.255 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-40-255 localhost] and IPs [172.31.40.255 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```



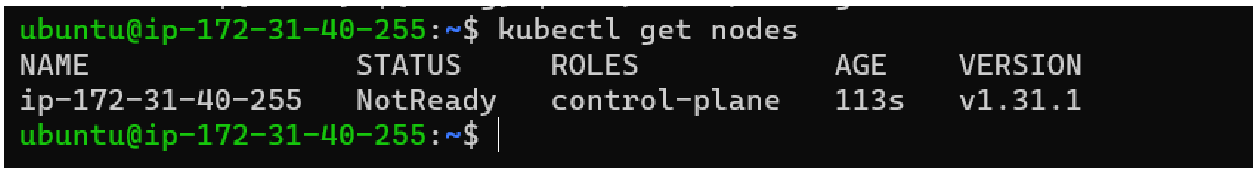
```
Your Kubernetes control-plane has initialized successfully!

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 172.31.40.255:6443 --token l2izt1.mt5iy3g7o0yhjft7 \
        --discovery-token-ca-cert-hash sha256:39c290262a4af785e7629a945f25514226b3f65234f280fe02b033f0f9924cfc
ubuntu@ip-172-31-40-255:~$
```

Run this command on master and also copy and save the Join command from above.
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config



```
        --discovery-token-ca-cert-hash sha256:39c290262a4af785e7629a945f25514226b3f65234
ubuntu@ip-172-31-40-255:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-40-255:~$
```

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

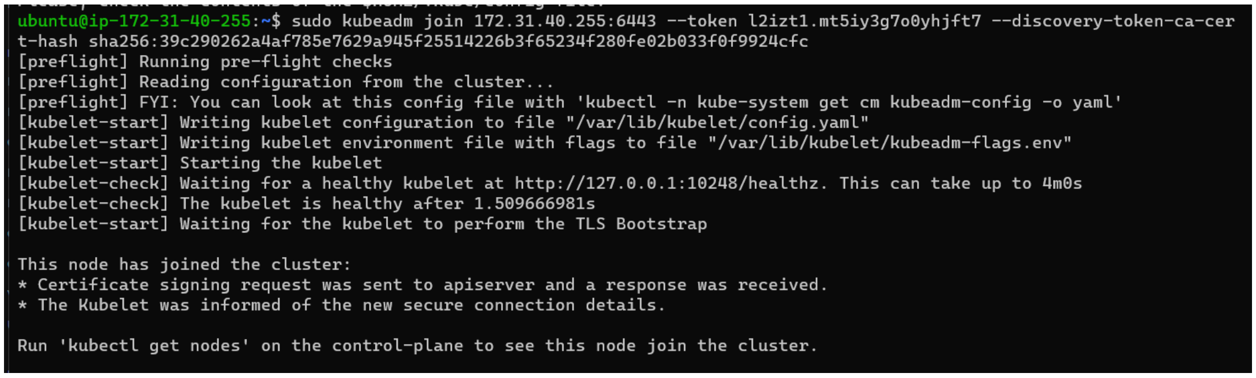


```
ubuntu@ip-172-31-40-255:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE    VERSION
ip-172-31-40-255   NotReady   control-plane   113s   v1.31.1
ubuntu@ip-172-31-40-255:~$
```

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

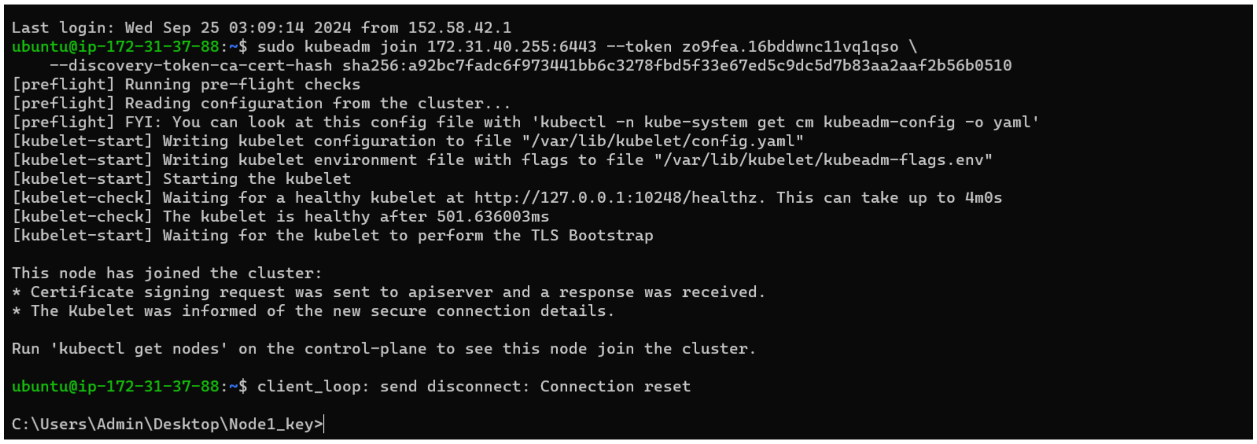
Anushka Shahane D15A 55

sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
--discovery-token-ca-cert-hash
sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6
Node 1:



```
ubuntu@ip-172-31-40-255:~$ sudo kubeadm join 172.31.40.255:6443 --token l2izt1.mt5iy3g7o0yhjft7 --discovery-token-ca-cer
t-hash sha256:39c290262a4af785e7629a945f25514226b3f65234f280fe02b033f0f9924cfc
[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 1.509666981s
[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.
```

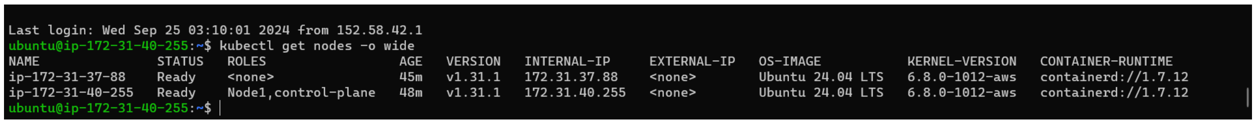
Step 9: Now Run the command kubectl get nodes to see the nodes after executing Join command on nodes.



```
Last login: Wed Sep 25 03:09:14 2024 from 152.58.42.1
ubuntu@ip-172-31-37-88:~$ sudo kubeadm join 172.31.40.255:6443 --token zo9fea.16bddwnc11vq1qso \
    --discovery-token-ca-cert-hash sha256:a92bc7fadc6f973441bb6c3278fbd5f33e67ed5c9dc5d7b83aa2aaf2b56b0510
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.636003ms
[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-172-31-37-88:~$ client_loop: send disconnect: Connection reset

C:\Users\Admin\Desktop\Node1_key>
```

Anushka Shahane D15A 55

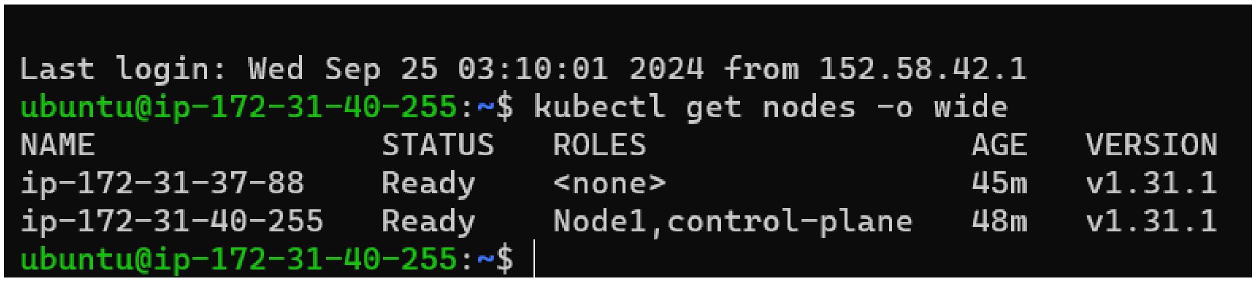
Now Run command kubectl get nodes -o wide we can see Status is ready.



```
Last login: Wed Sep 25 03:10:01 2024 from 152.58.42.1
ubuntu@ip-172-31-40-255:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES                  AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
ip-172-31-37-88    Ready    <none>                 45m   v1.31.1   172.31.37.88    <none>        Ubuntu 24.04 LTS     6.8.0-1012-aws     containerd://1.7.12
ip-172-31-40-255   Ready    Node1,control-plane    48m   v1.31.1   172.31.40.255   <none>        Ubuntu 24.04 LTS     6.8.0-1012-aws     containerd://1.7.12
ubuntu@ip-172-31-40-255:~$ 
```

Step 11: Run command kubectl get nodes -o wide . And Hence we can see we have Successfully
connected Node 1 and Node 2 to the Master.

Or run kubectl get nodes



```
Last login: Wed Sep 25 03:10:01 2024 from 152.58.42.1
ubuntu@ip-172-31-40-255:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES                  AGE   VERSION
ip-172-31-37-88    Ready    <none>                 45m   v1.31.1
ip-172-31-40-255   Ready    Node1,control-plane    48m   v1.31.1
ubuntu@ip-172-31-40-255:~$ 
```