Name: Pratik Patil

Division: D15C

Academic Year:2024-2025

Roll No: 40

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Theory:

Kubernetes, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the **Cloud Native Computing Foundation (CNCF)**, with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

Necessary Requirements:

- **EC2 Instance:** The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- **Minimum Requirements:**
  - **Instance Type:** t2.medium
  - **CPUs:** 2
  - **Memory:** Adequate for container orchestration.

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

Note:
AWS Personal Account is preferred but we can also perform it on AWS Academy(adding some ignores in the command if any error occurs in below as the below experiment is performed on Personal Account
.).
If You are using AWS Academy Account Errors you will face in kubeadm init command so you have to add some ignores with this command.

Name: Pratik Patil

Academic Year:2024-2025

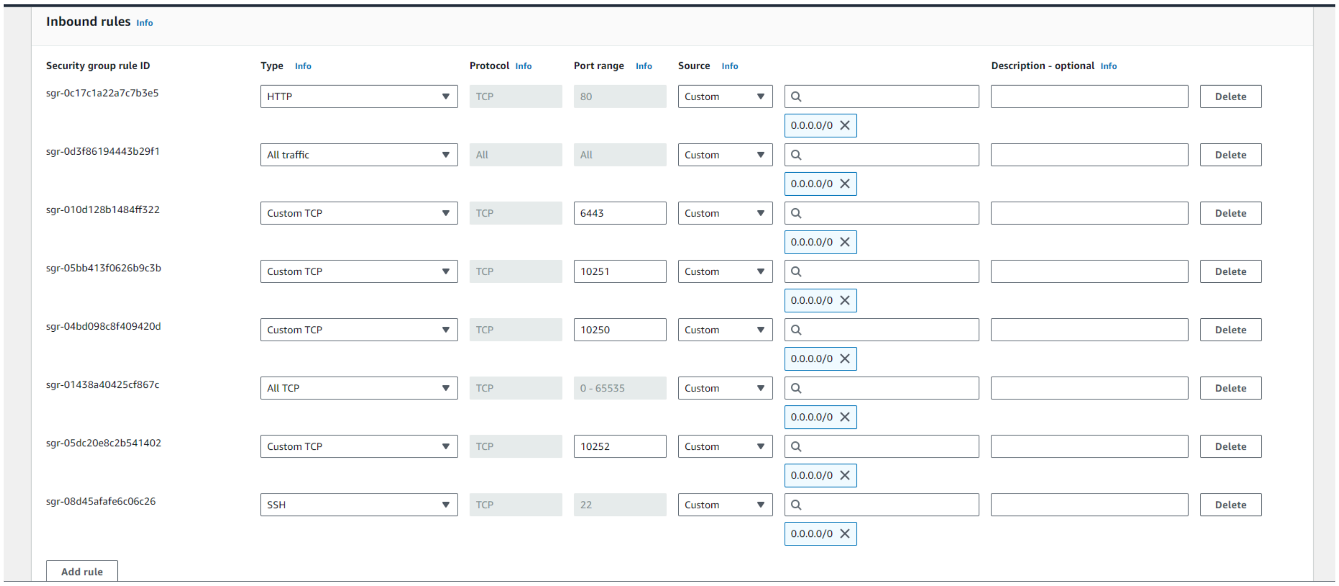
Division: D15C

Roll No: 40

Step 1: Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.

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.

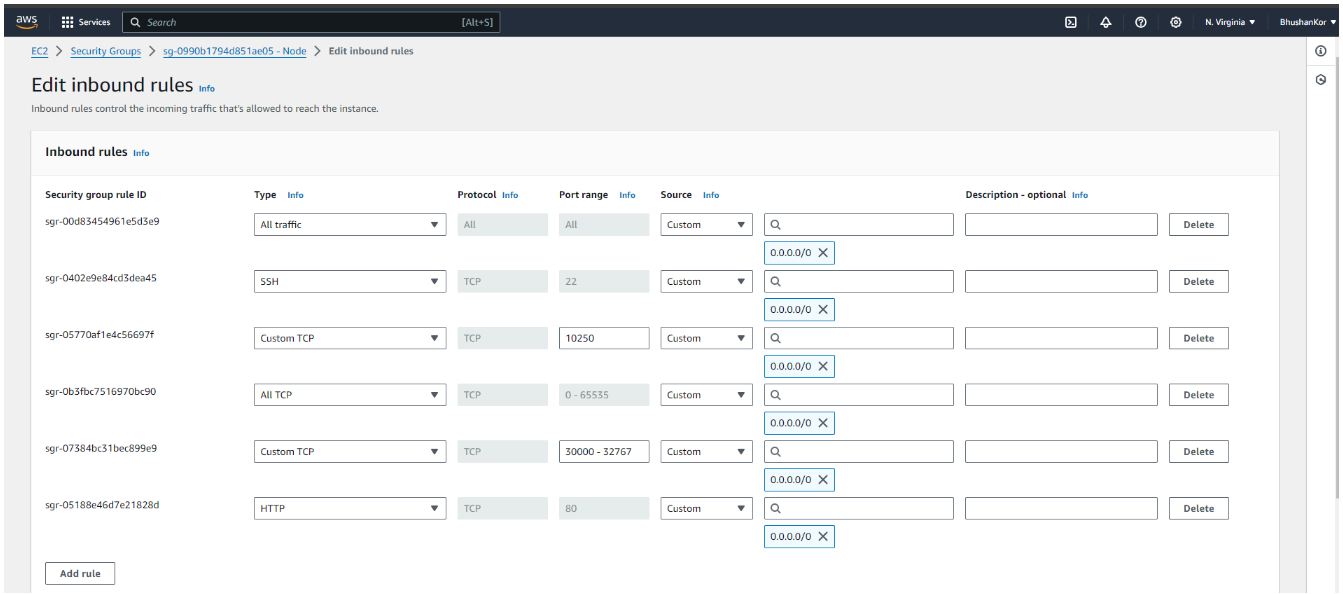
Master:



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0c17c1a22a7c7b3e5	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-0d3f86194443b29f1	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-010d128b1484ff322	Custom TCP	TCP	6443	Custom	0.0.0.0/0		Delete
sgr-05bb413f0626b9c3b	Custom TCP	TCP	10251	Custom	0.0.0.0/0		Delete
sgr-04bd098c8f409420d	Custom TCP	TCP	10250	Custom	0.0.0.0/0		Delete
sgr-01438a40425cf867c	All TCP	TCP	0 - 65535	Custom	0.0.0.0/0		Delete
sgr-05dc20e8c2b541402	Custom TCP	TCP	10252	Custom	0.0.0.0/0		Delete
sgr-08d45afafe6c06c26	SSH	TCP	22	Custom	0.0.0.0/0		Delete

Node :



EC2 > Security Groups > sg-0990b1794d851ae05 - Node > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-00d83454961e5d3e9	All traffic	All	All	Custom	0.0.0.0/0		Delete
sgr-0402e9e84cd3dea45	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-05770af1e4c56697f	Custom TCP	TCP	10250	Custom	0.0.0.0/0		Delete
sgr-0b3fbc7516970bc90	All TCP	TCP	0 - 65535	Custom	0.0.0.0/0		Delete
sgr-07384bc31bec899e9	Custom TCP	TCP	30000 - 32767	Custom	0.0.0.0/0		Delete
sgr-05188e46d7e21828d	HTTP	TCP	80	Custom	0.0.0.0/0		Delete

Name: Pratik Patil

Academic Year:2024-2025

Division: D15C

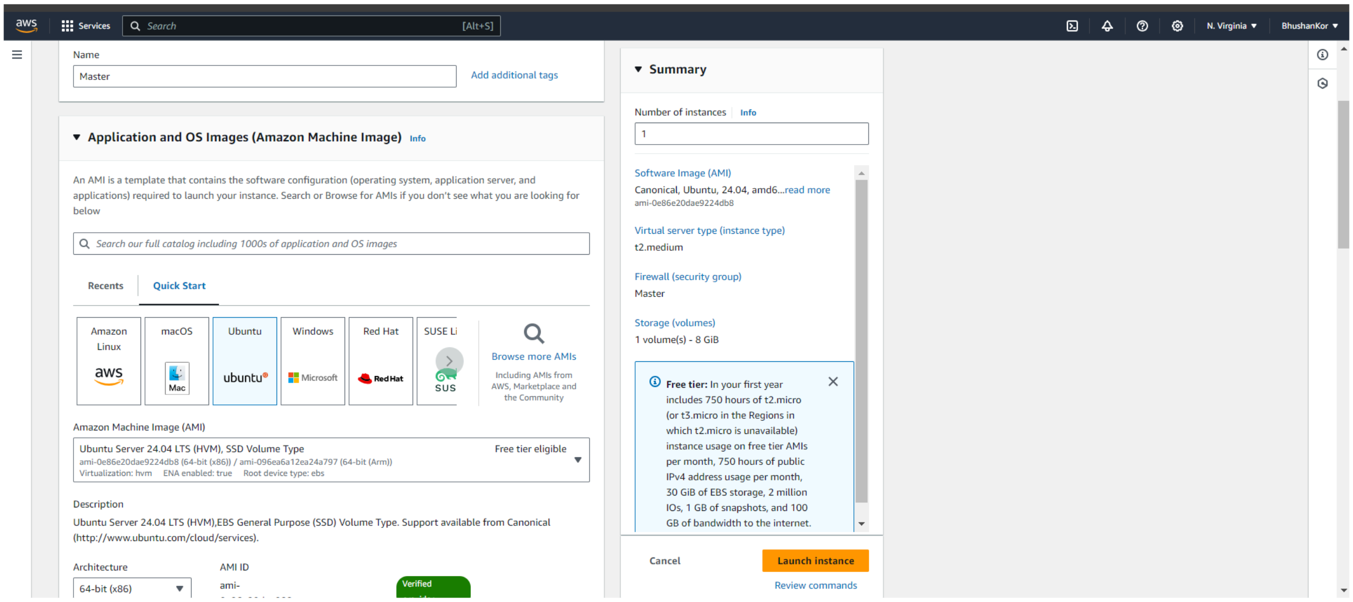
Roll No: 40

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:



Name: 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

Quick Start: Ubuntu

Amazon Machine Image (AMI): Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

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

Architecture: 64-bit (x86)

Summary

Number of instances: 1

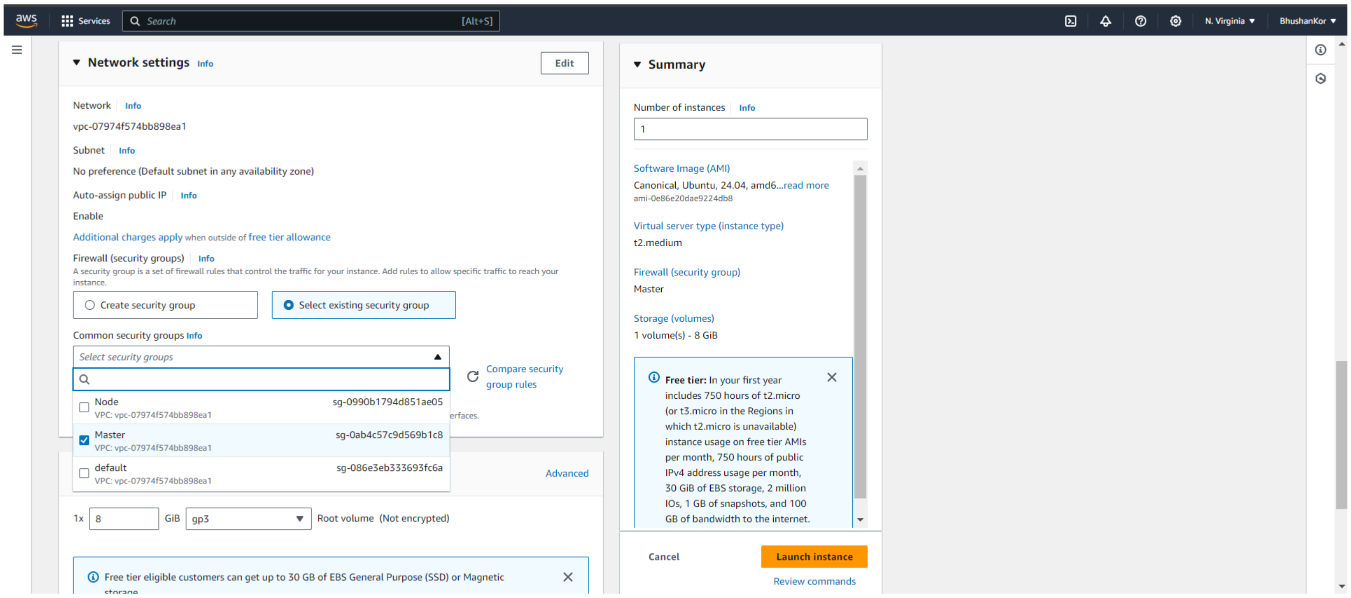
Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-0e86e20dae9224db8

Virtual server type (instance type): t2.medium

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.



Network settings Info

Network: vpc-07974f574bb898ea1

Subnet: No preference (Default subnet in any availability zone)

Auto-assign public IP: Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

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

	Name	VPC	Security group ID
□	Node	VPC: vpc-07974f574bb898ea1	sg-0990b1794d851ae05
☑	Master	VPC: vpc-07974f574bb898ea1	sg-0ab4c57c9d569b1c8
□	default	VPC: vpc-07974f574bb898ea1	sg-086e3eb333693fc6a

1x 8 GiB gp3 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Summary

Number of instances: 1

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

Virtual server type (instance type): t2.medium

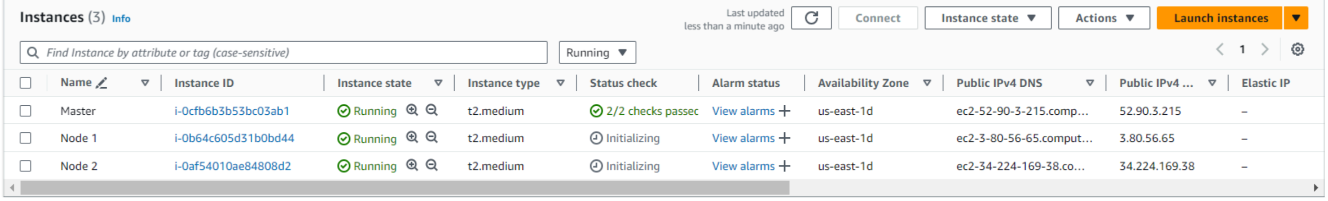
Firewall (security group): Master

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.

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.



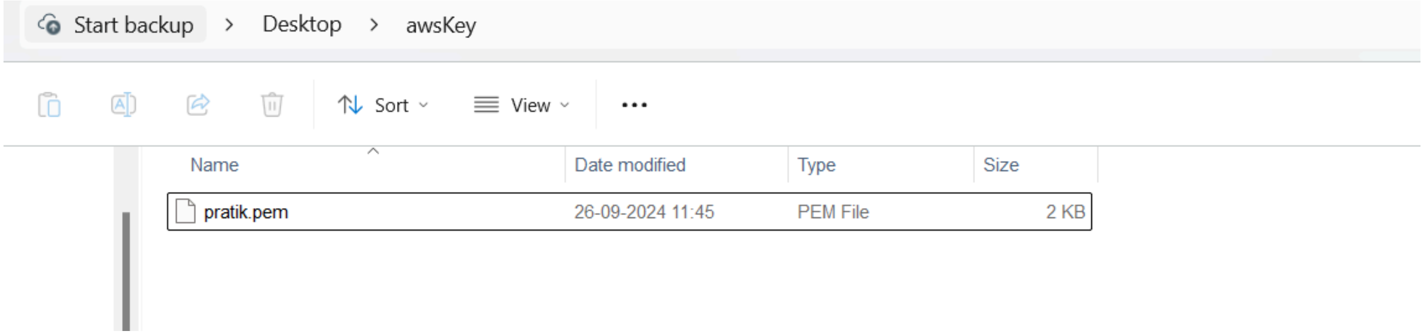
Instances (3)

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive) | Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Master	i-0cfb6b3b53bc03ab1	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1d	ec2-52-90-3-215.comp...	52.90.3.215	–
Node 1	i-0b64c605d31b0bd44	Running	t2.medium	Initializing	View alarms +	us-east-1d	ec2-3-80-56-65.comput...	3.80.56.65	–
Node 2	i-0af54010ae84808d2	Running	t2.medium	Initializing	View alarms +	us-east-1d	ec2-34-224-169-38.co...	34.224.169.38	–

(Downloaded Key

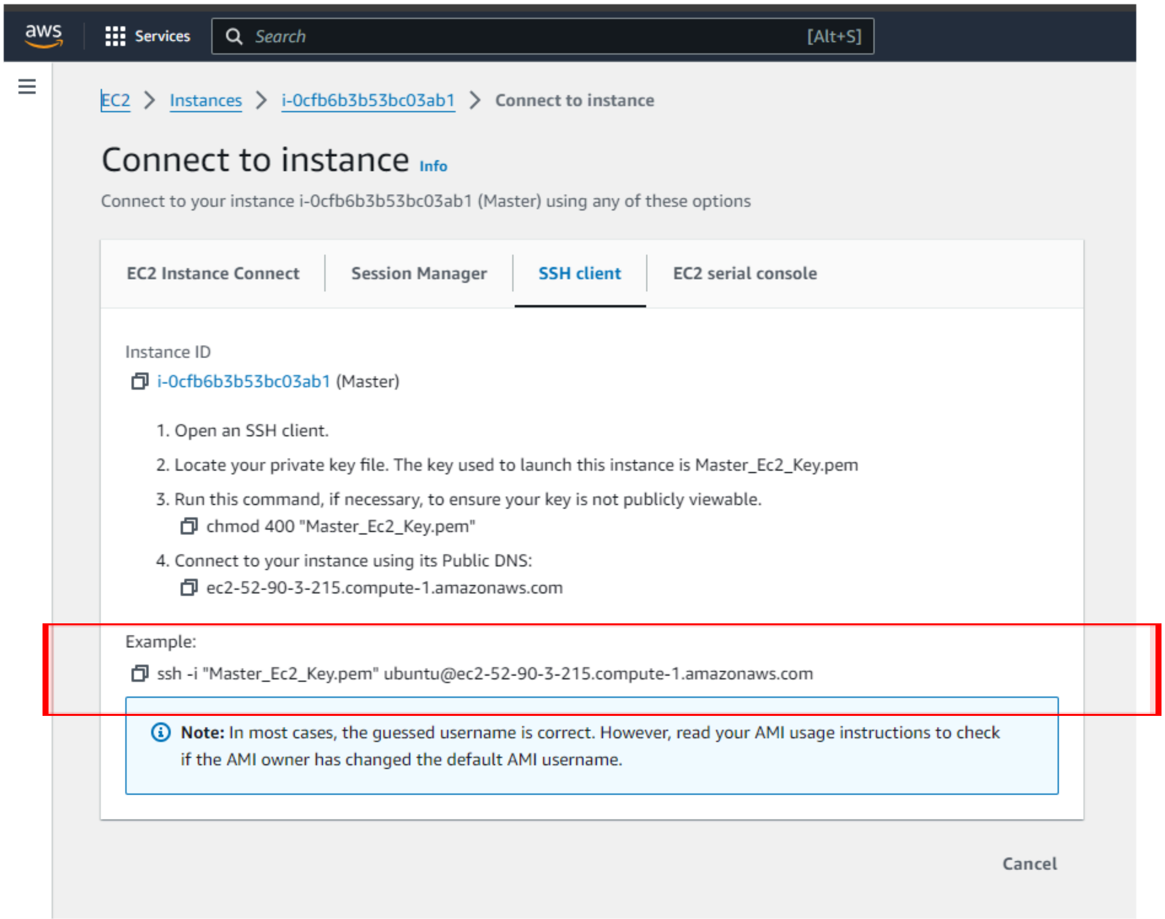


Start backup > Desktop > awsKey

Name	Date modified	Type	Size
pratik.pem	26-09-2024 11:45	PEM File	2 KB

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:



Connect to instance

Connect to your instance i-0cfb6b3b53bc03ab1 (Master) using any of these options

Instance ID

i-0cfb6b3b53bc03ab1 (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:

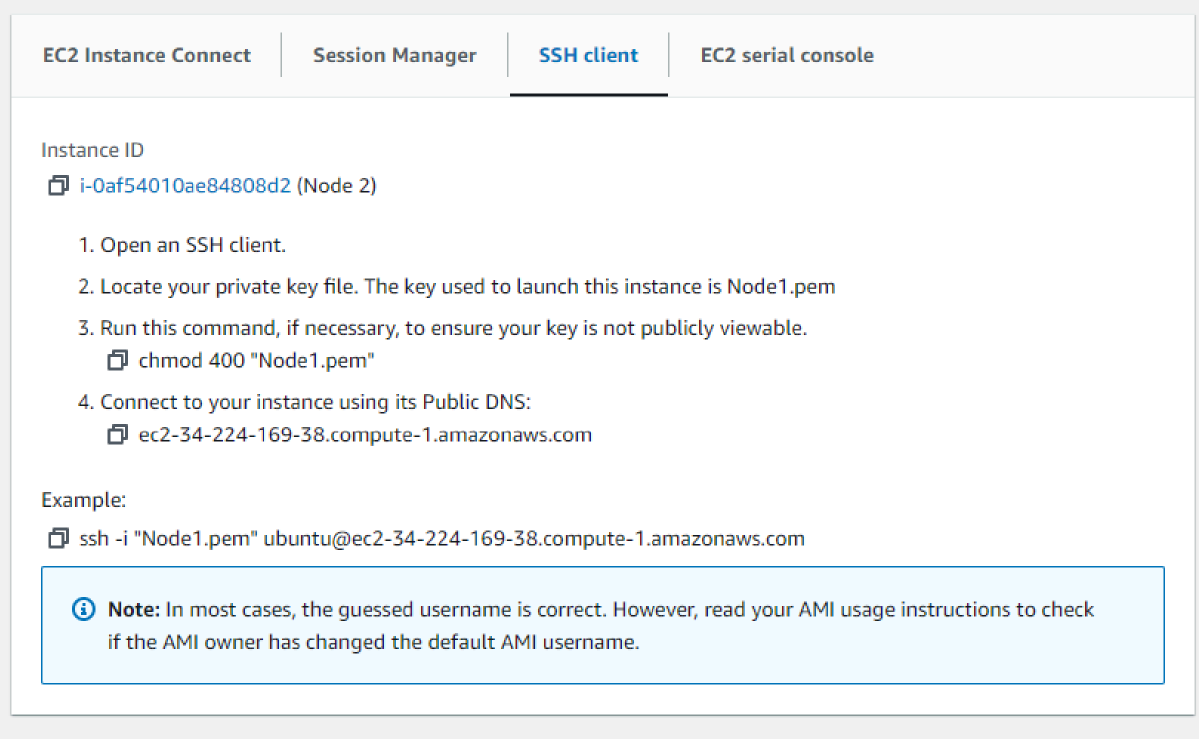
```
ec2-52-90-3-215.compute-1.amazonaws.com
```

Example:

```
ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-52-90-3-215.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.

Node 2:



Instance ID

i-0af54010ae84808d2 (Node 2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Node1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 "Node1.pem"
```

4. Connect to your instance using its Public DNS:

```
ec2-34-224-169-38.compute-1.amazonaws.com
```

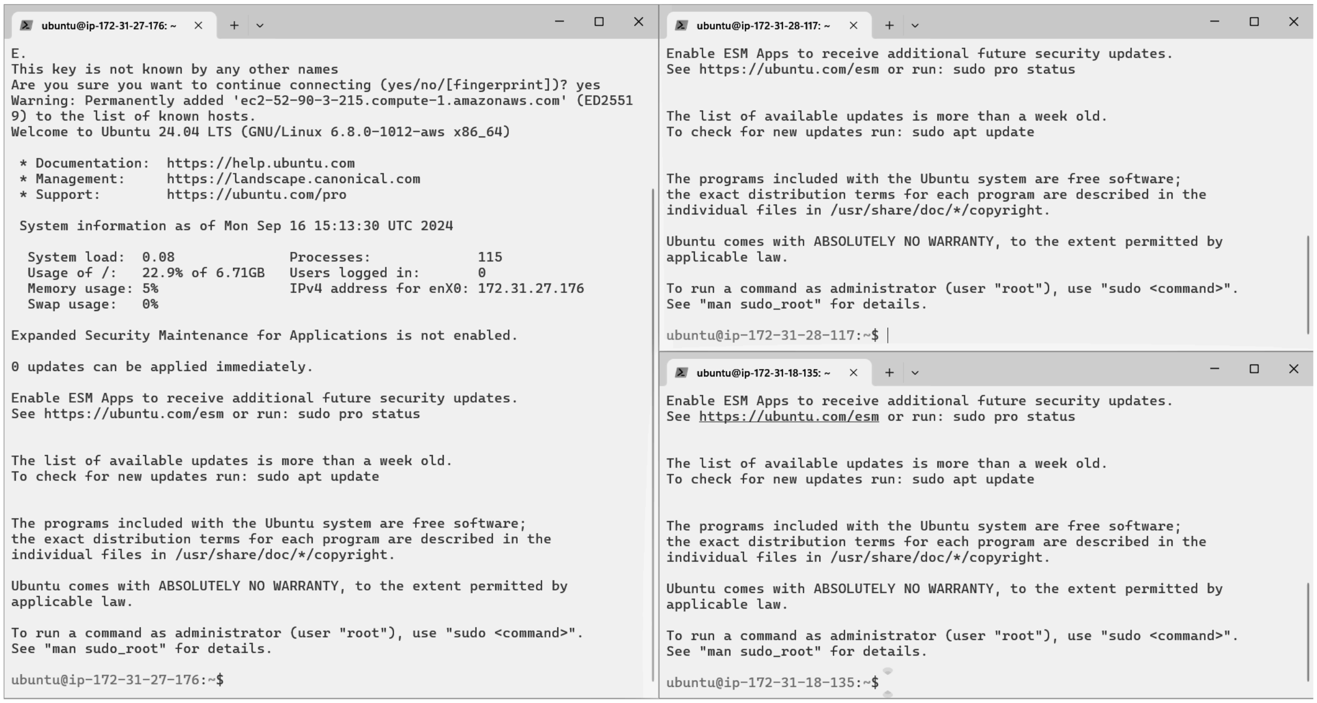
Example:

```
ssh -i "Node1.pem" ubuntu@ec2-34-224-169-38.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.

Here I have use 2 keys 1 for master and 1 for 2 node so I have to run open 3 terminals.In master key folder 1 terminal and 2 terminals in node 1 key folder.
If you use 1 Key only, you can open 3 terminal in one folder only.

Successful Connection:



```
E.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-90-3-215.compute-1.amazonaws.com' (ED2551
9) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 16 15:13:30 UTC 2024

  System load:  0.08               Processes:             115
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 5%                 IPv4 address for enX0: 172.31.27.176
  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-172-31-27-176:~$
```

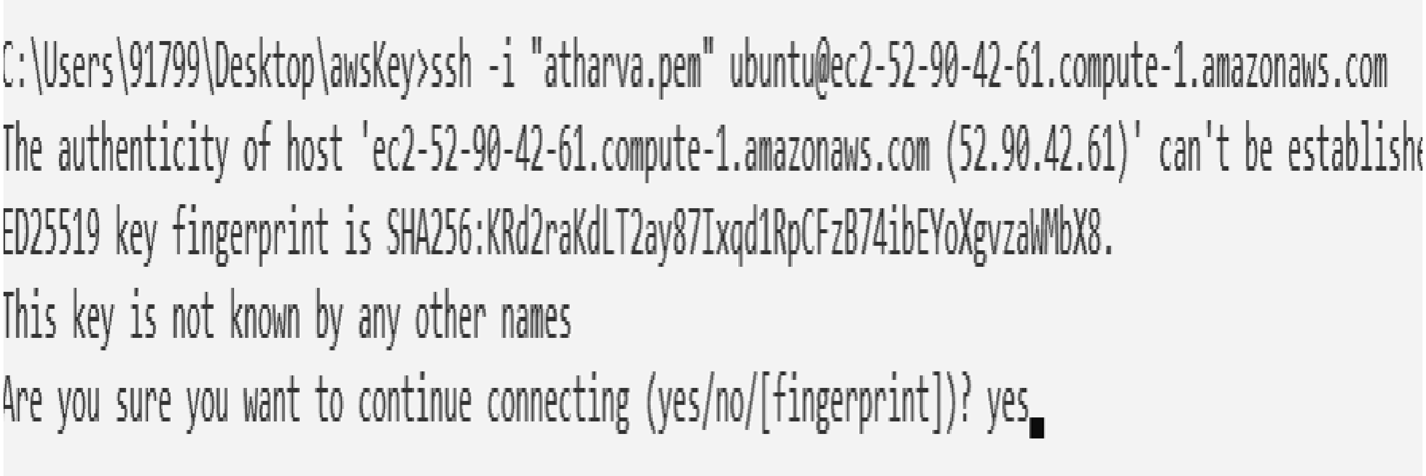
```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


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-172-31-28-117:~$ 
```

```
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-172-31-18-135:~$
```

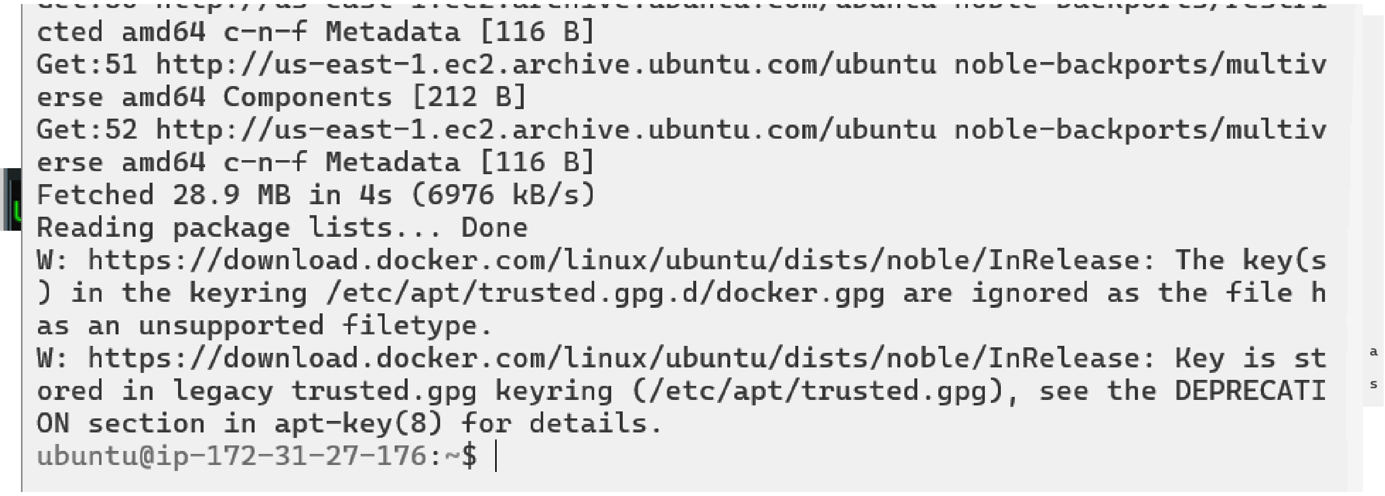
Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

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

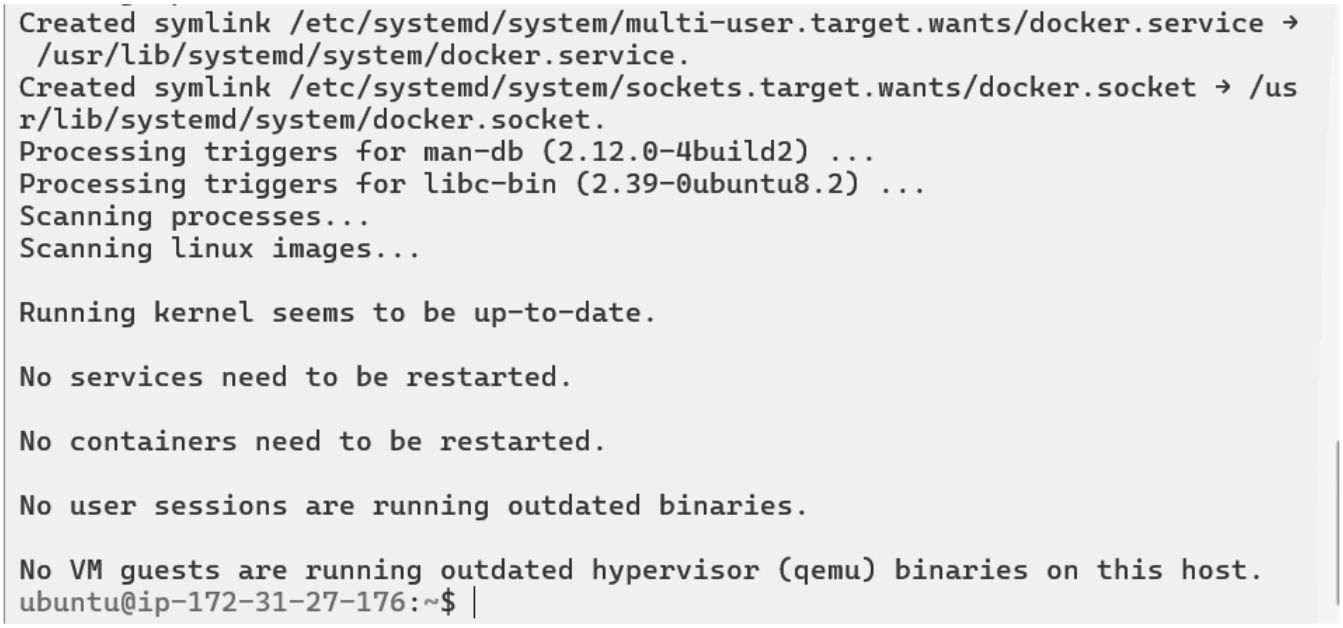
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



```
C:\Users\91799\Desktop\awsKey>ssh -i "atharva.pem" ubuntu@ec2-52-90-42-61.compute-1.amazonaws.com
The authenticity of host 'ec2-52-90-42-61.compute-1.amazonaws.com (52.90.42.61)' can't be establishe
ED25519 key fingerprint is SHA256:KRd2raKdLT2ay87Ixqd1RpCFzB74ibEYoXgvzaWMbX8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

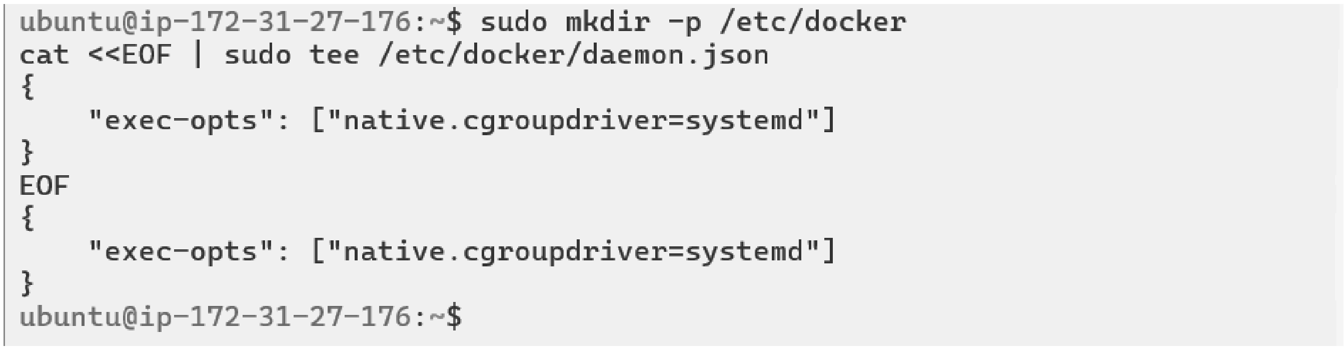


```
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restri
cted amd64 c-n-f Metadata [116 B]
Get:51 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 Components [212 B]
Get:52 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiv
erse amd64 c-n-f Metadata [116 B]
Fetched 28.9 MB in 4s (6976 kB/s)
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 as the file h
as an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is st
ored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATI
ON section in apt-key(8) for details.
ubuntu@ip-172-31-27-176:~$
```



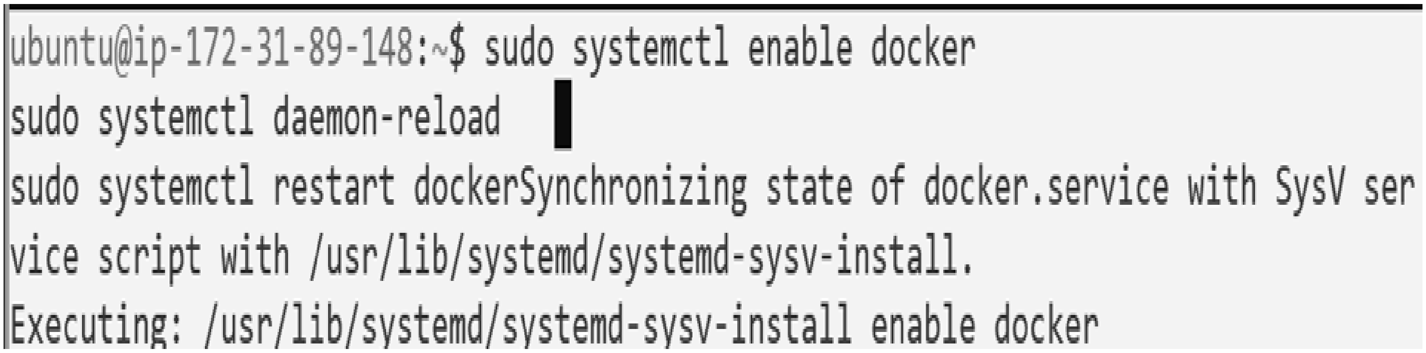
```
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service →
 /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /us
r/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-27-176:~$ 
```

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



```
ubuntu@ip-172-31-27-176:~$ 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-172-31-27-176:~$
```

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

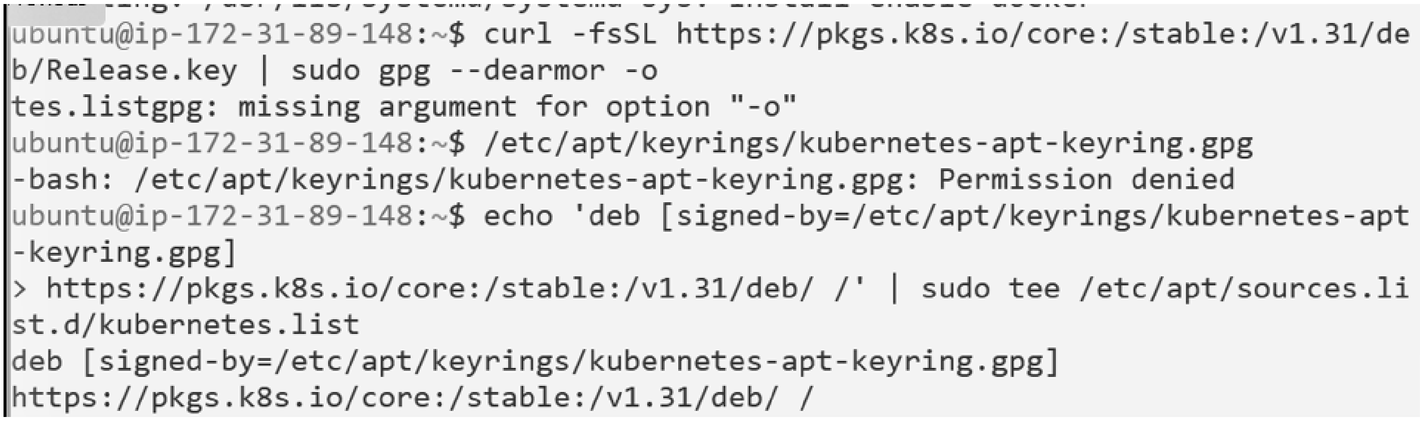


```
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart dockerSynchronizing state of docker.service with SysV ser
vice script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 5: Run the below command to install Kubernets.

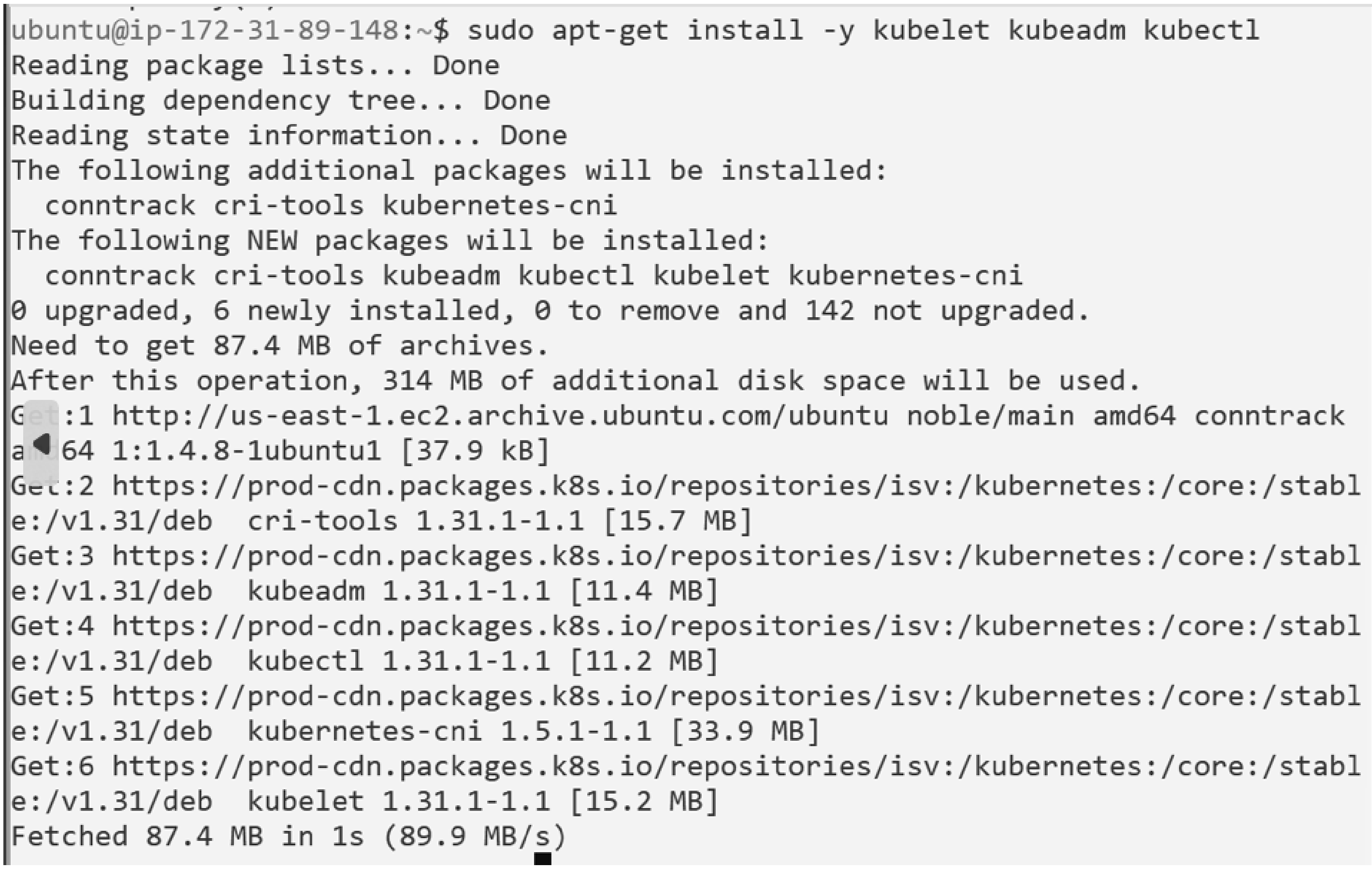
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

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

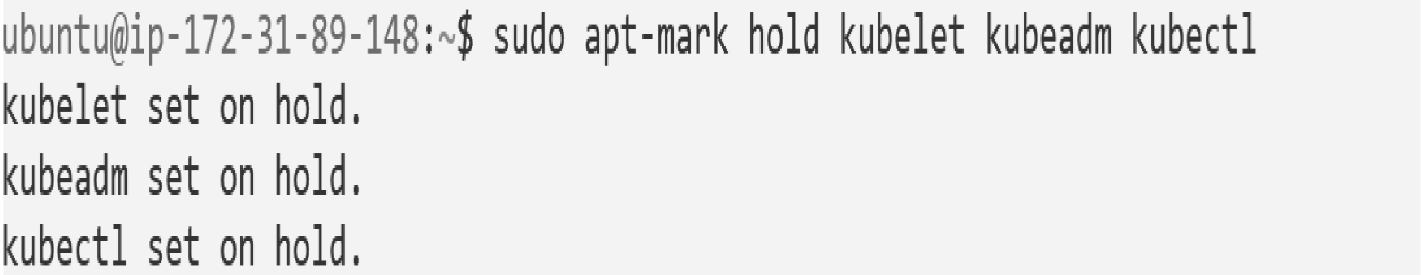


```
ubuntu@ip-172-31-89-148:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/de
b/Release.key | sudo gpg --dearmor -o
tes.listgpg: missing argument for option "-o"
ubuntu@ip-172-31-89-148:~$ /etc/apt/keyrings/kubernetes-apt-keyring.gpg
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: Permission denied
ubuntu@ip-172-31-89-148:~$ 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.li
st.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

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

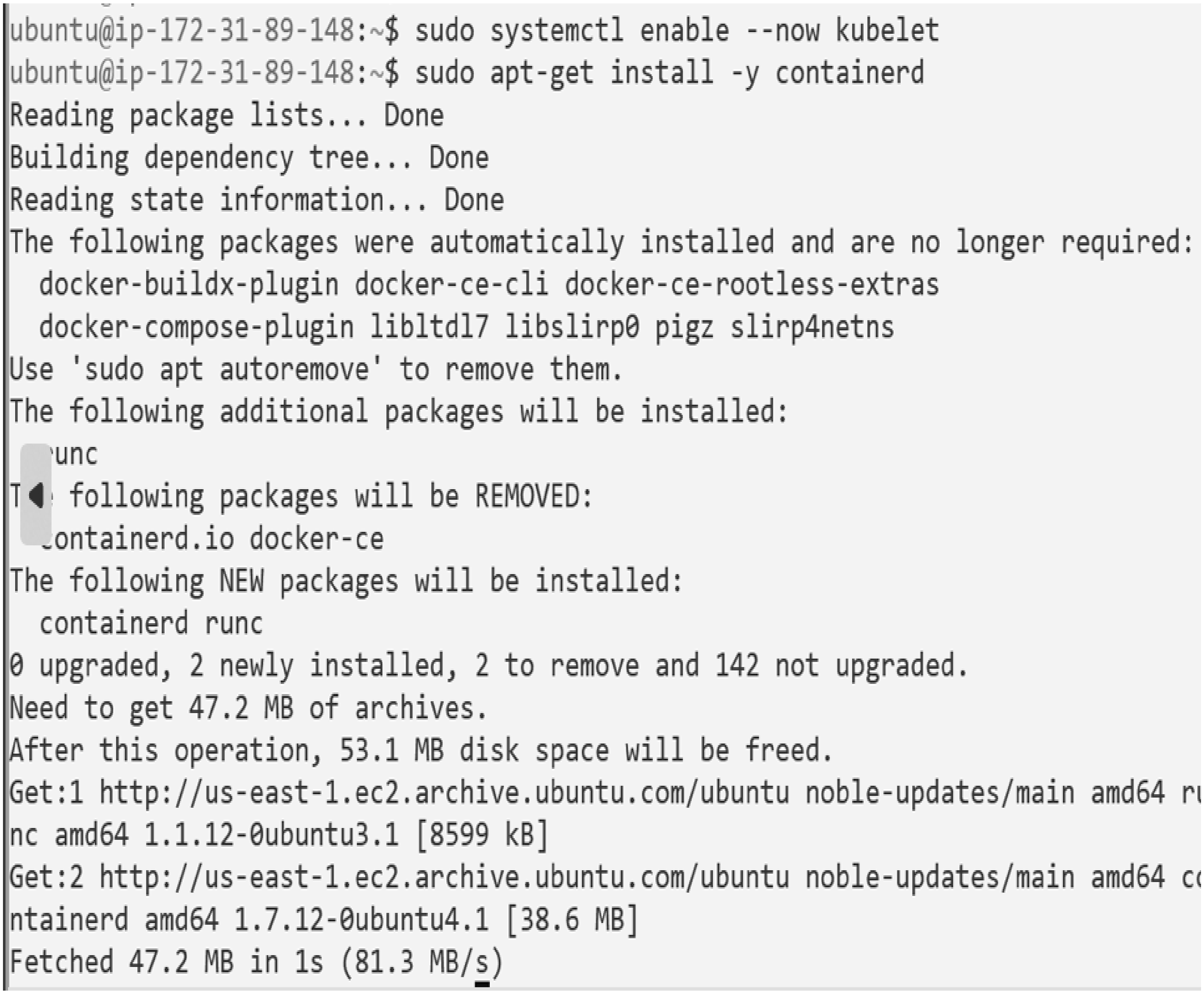


```
ubuntu@ip-172-31-89-148:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 142 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:/stabl
e:/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:/stabl
e:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stabl
e:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (89.9 MB/s)
```



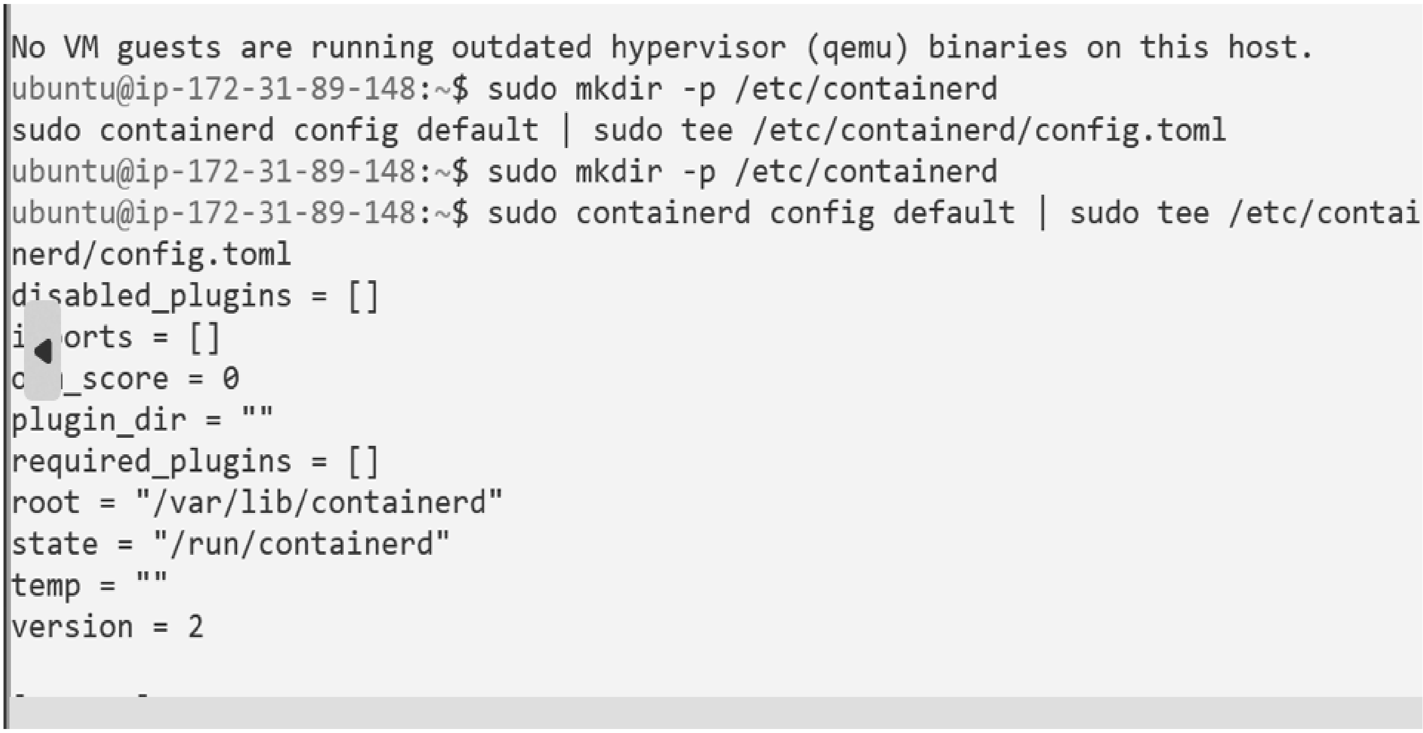
```
ubuntu@ip-172-31-89-148:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd



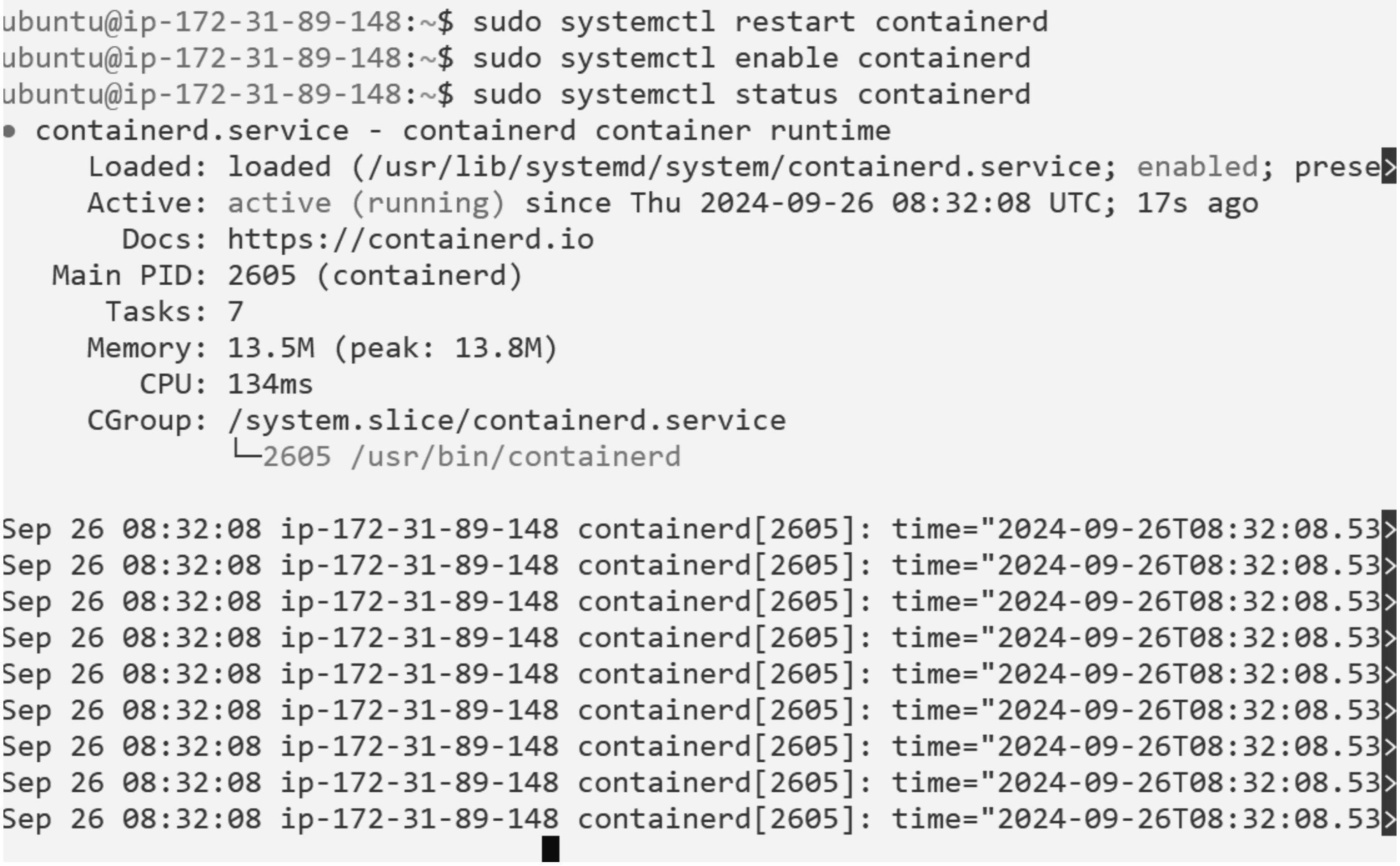
```
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-89-148:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  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 142 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 ru
nc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 co
ntainerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (81.3 MB/s)
```

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



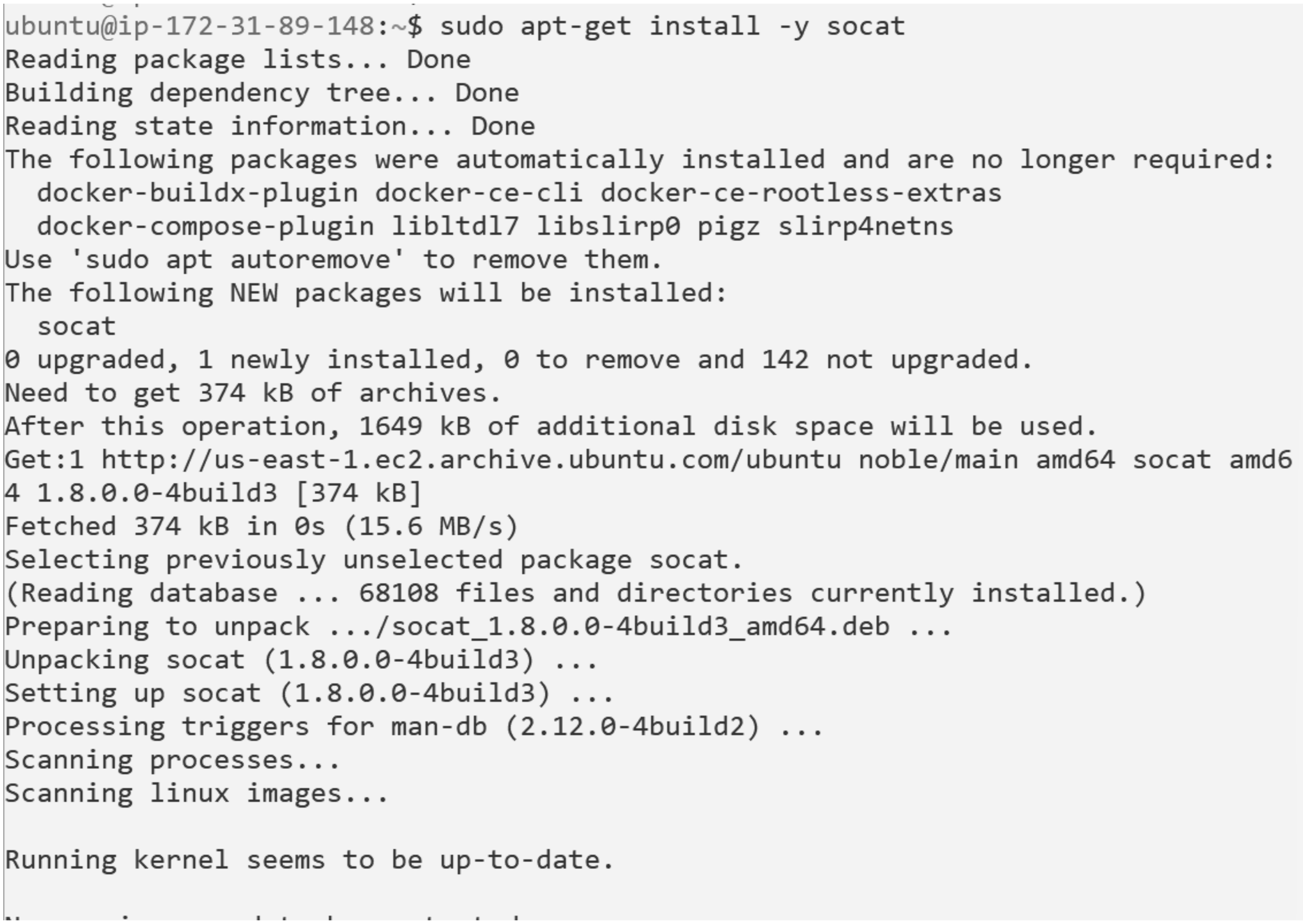
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-89-148:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
ubuntu@ip-172-31-89-148:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-89-148:~$ sudo containerd config default | sudo tee /etc/contai
nerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2
```

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



```
ubuntu@ip-172-31-89-148:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-89-148:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-89-148:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; prese>
     Active: active (running) since Thu 2024-09-26 08:32:08 UTC; 17s ago
       Docs: https://containerd.io
   Main PID: 2605 (containerd)
      Tasks: 7
     Memory: 13.5M (peak: 13.8M)
        CPU: 134ms
     CGroup: /system.slice/containerd.service
             └─2605 /usr/bin/containerd

Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
Sep 26 08:32:08 ip-172-31-89-148 containerd[2605]: time="2024-09-26T08:32:08.53>
```

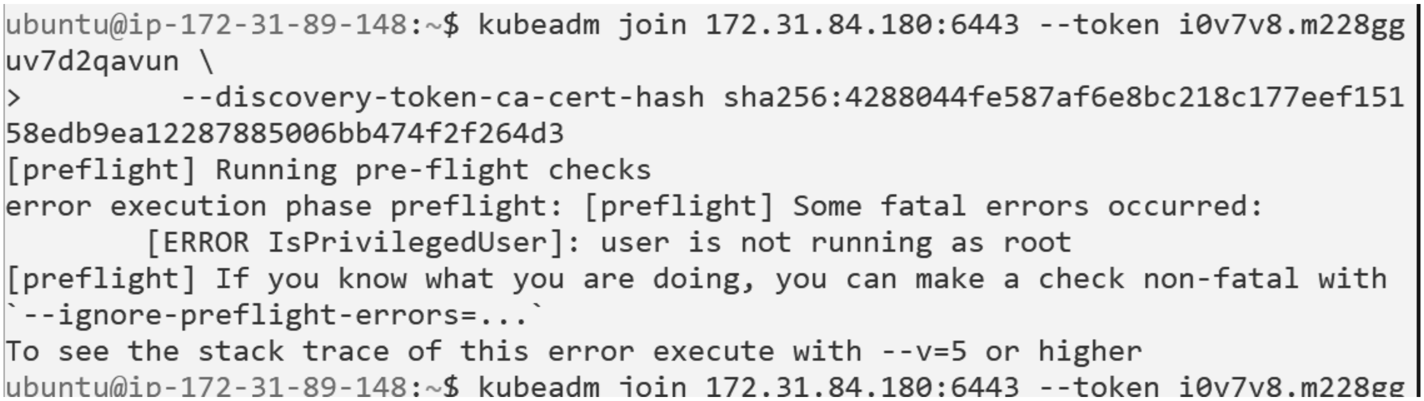
sudo apt-get install -y socat



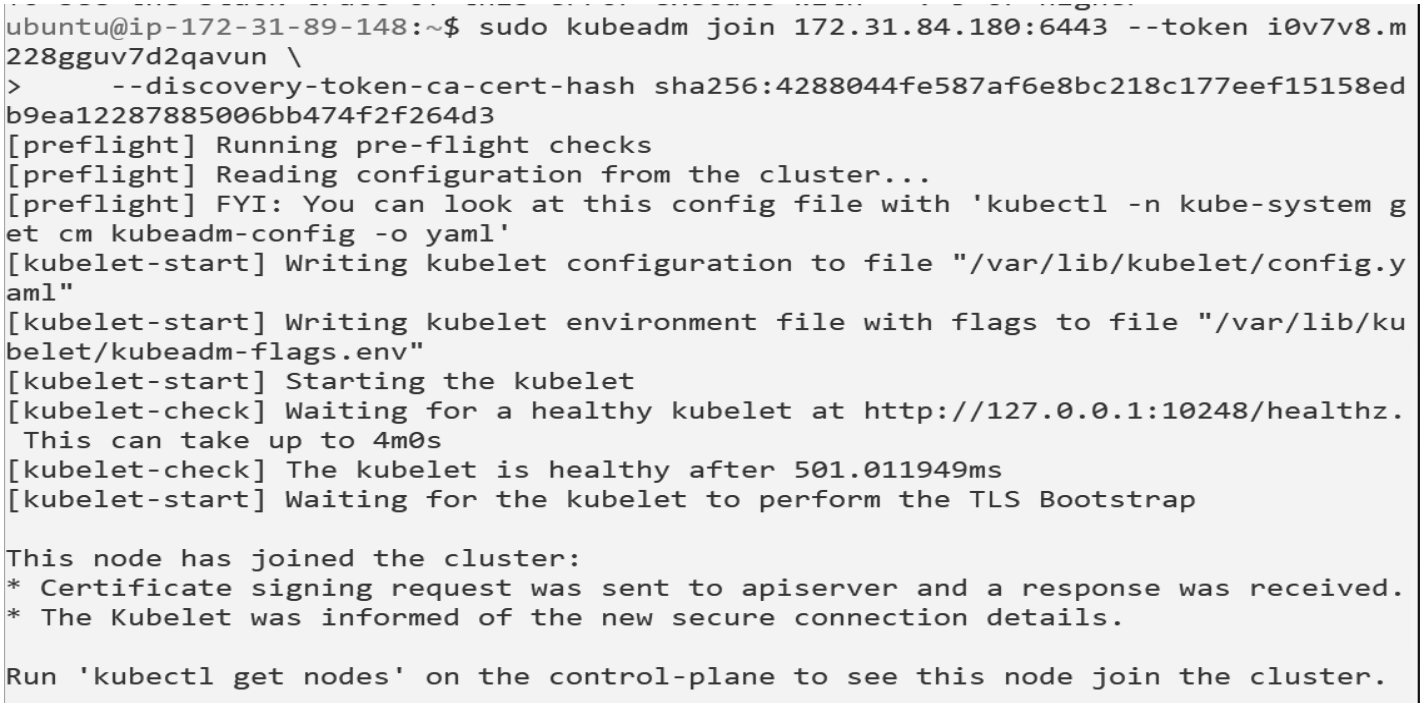
```
ubuntu@ip-172-31-89-148:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  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 142 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 amd6
4 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (15.6 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.
```

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-172-31-89-148:~$ kubeadm join 172.31.84.180:6443 --token i0v7v8.m228gg
uv7d2qavun \
>         --discovery-token-ca-cert-hash sha256:4288044fe587af6e8bc218c177eef151
58edb9ea12287885006bb474f2f264d3
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR IsPrivilegedUser]: user is not running as root
[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-172-31-89-148:~$ kubeadm join 172.31.84.180:6443 --token i0v7v8.m228gg
```

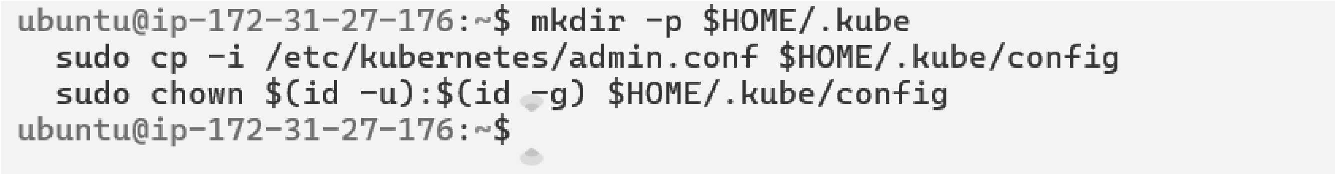


```
ubuntu@ip-172-31-89-148:~$ sudo kubeadm join 172.31.84.180:6443 --token i0v7v8.m
228gguv7d2qavun \
>     --discovery-token-ca-cert-hash sha256:4288044fe587af6e8bc218c177eef15158ed
b9ea12287885006bb474f2f264d3
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system g
et cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/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 501.011949ms
[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.
```

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



```
ubuntu@ip-172-31-27-176:~$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-27-176:~$
```

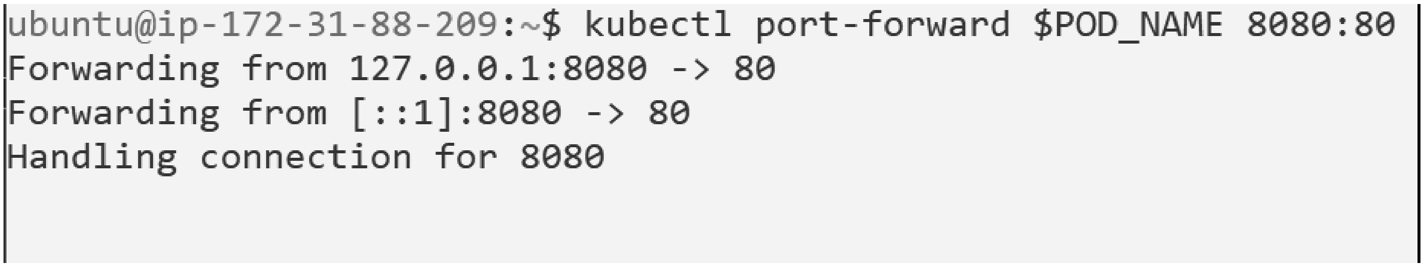
Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Step 7: Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

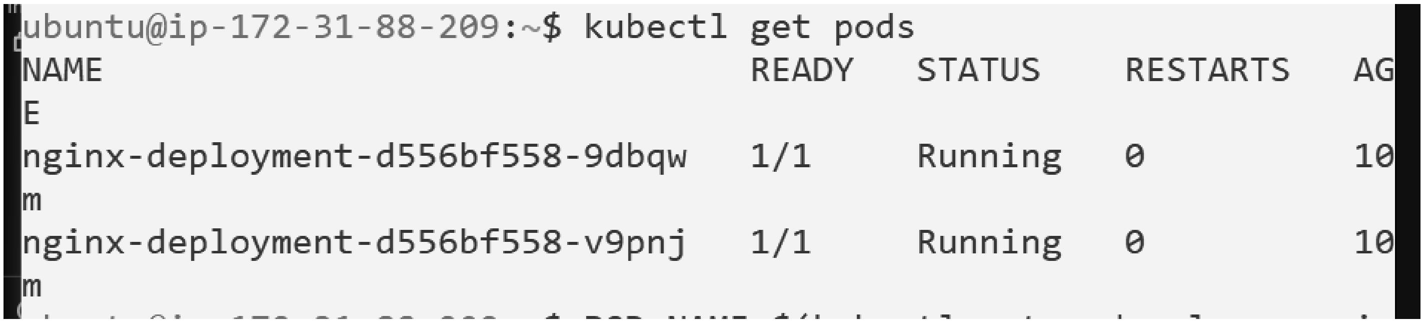
kubectl get pods

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
note : We have faced an error as pod status is pending so make it running run below commands then again run above 2 commands.
kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted

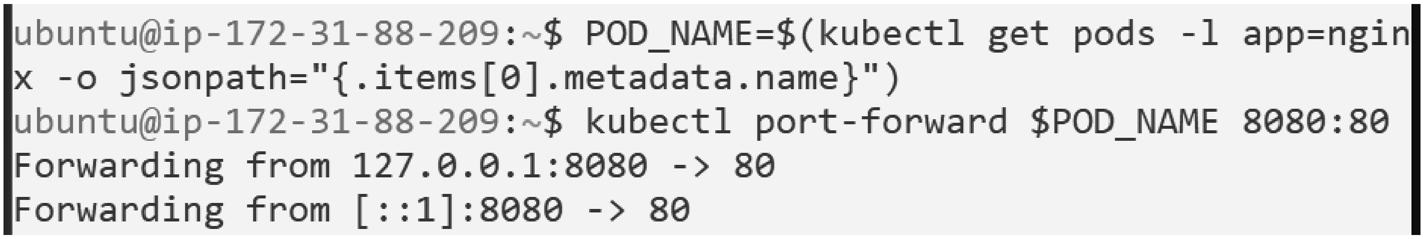
kubectl get nodes



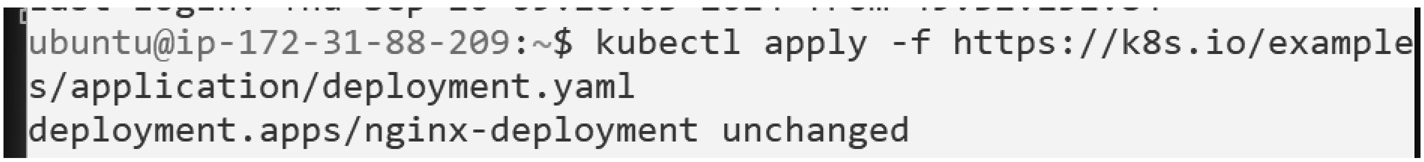
```
ubuntu@ip-172-31-88-209:~$ kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```



```
ubuntu@ip-172-31-88-209:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AG
E
nginx-deployment-d556bf558-9dbqw   1/1     Running   0          10
m
nginx-deployment-d556bf558-v9pnj   1/1     Running   0          10
m
```

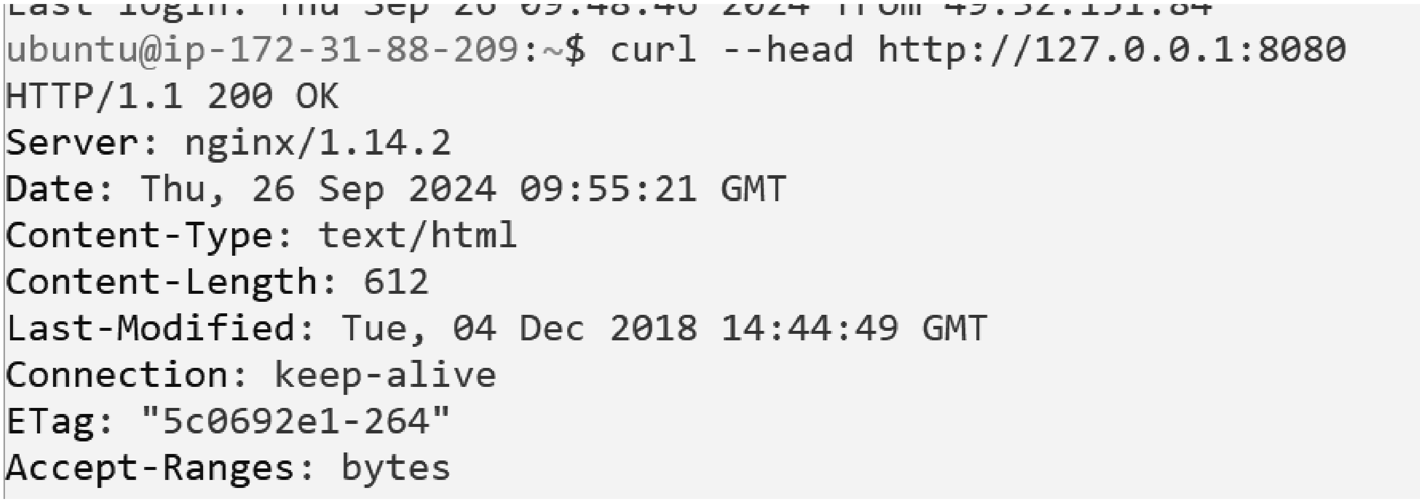


```
ubuntu@ip-172-31-88-209:~$ POD_NAME=$(kubectl get pods -l app=ngin
x -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-88-209:~$ kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```



```
ubuntu@ip-172-31-88-209:~$ kubectl apply -f https://k8s.io/example
s/application/deployment.yaml
deployment.apps/nginx-deployment unchanged
```

final Result :



```
Last login: Thu Sep 26 09:48:46 2024 from 49.32.151.84
ubuntu@ip-172-31-88-209:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 26 Sep 2024 09:55:21 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

Conclusion:

Installing and configuring `kubectl` on an EC2 instance allowed successful interaction with a Kubernetes cluster. Kubernetes streamlines the management of containerized applications by automating processes like deployment, scaling, and rollback. Deploying an application using Kubernetes demonstrates how it ensures high availability and fault tolerance by maintaining the desired state through Deployments and ReplicaSets.