NAME: Bhushan Malpani **DIV: D15C** **Roll No:33**

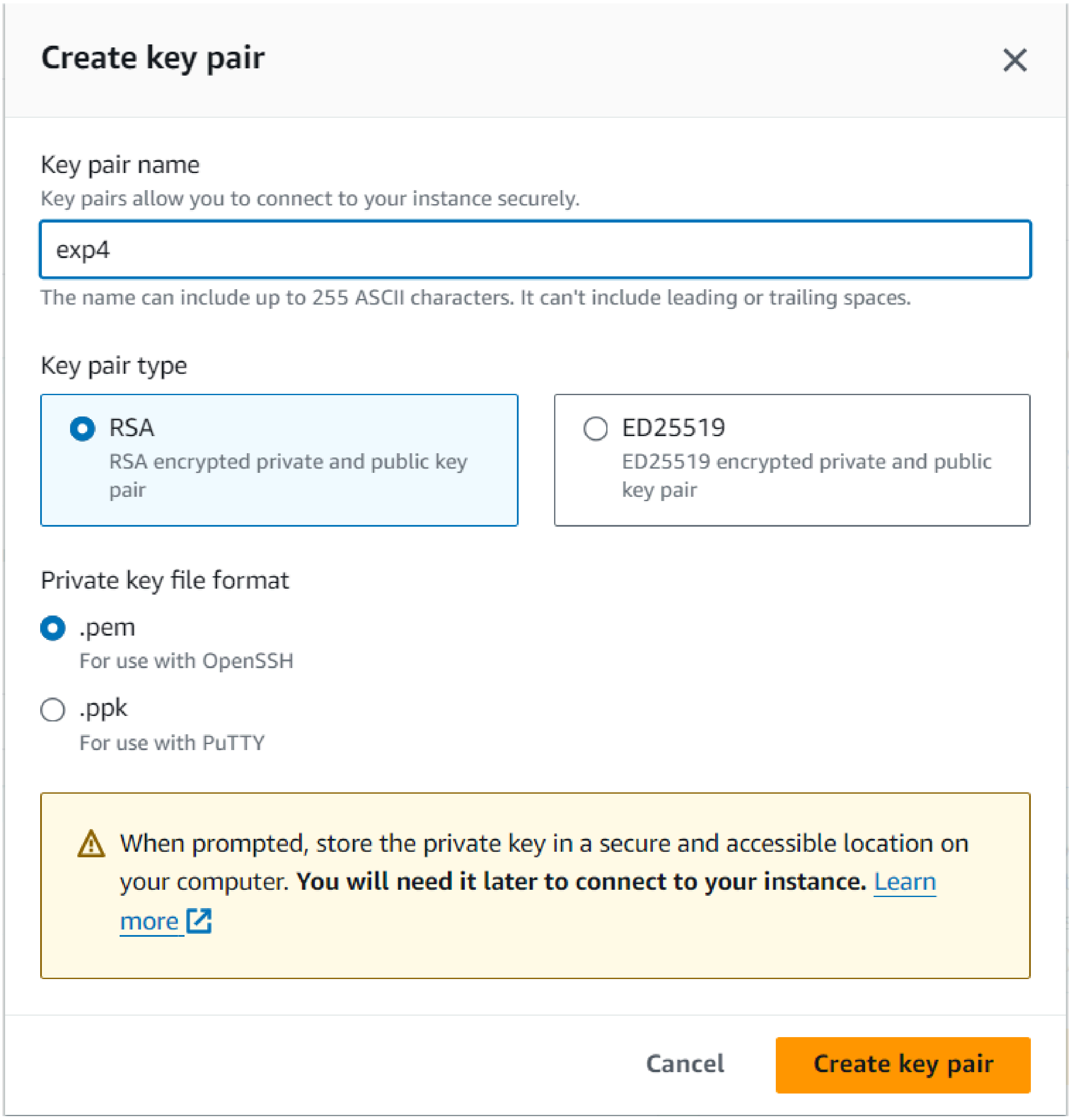
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Theory

kubectl is the command-line tool used to interact with Kubernetes clusters. It serves as the primary interface for managing and orchestrating containers in a Kubernetes environment. By sending commands to the Kubernetes API server, kubectl allows you to control clusters, manage workloads, and inspect resource states.

To begin using Kubernetes, installing `kubectl` is essential. The installation process varies based on the operating system (Linux, Windows, or macOS). After installing, kubectl connects to the Kubernetes cluster using the kubeconfig file, which stores details like cluster name, server address, and access credentials. With this connection established, you can use kubectl to perform a variety of operations, such as creating, updating, scaling, and deleting applications.

Step 1:Create an EC2 instance use ubuntu application and select t2 .medium category in instance type create a new key rsa type save it in local machine in an folder:



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

exp4

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

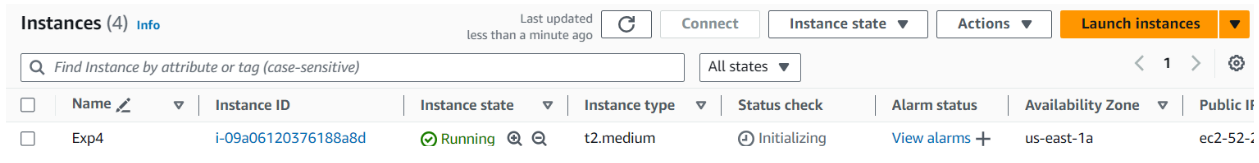
Private key file format

[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Step 2:Click create to create the instance:

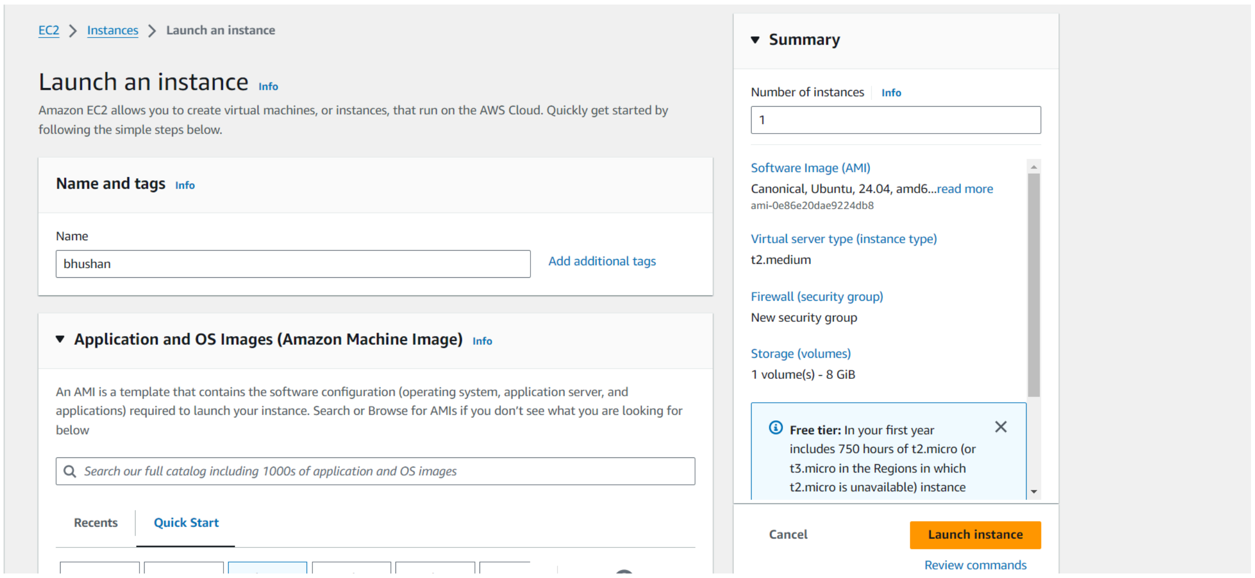


Instances (4) Info

Last updated less than a minute ago

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
	Exp4	i-09a06120376188a8d	Running	t2.medium	Initializing	View alarms	us-east-1a	ec2-52-

Step 3:
Navigate to ssh client copy the key:



EC2 > Instances > Launch an instance

Launch an instance Info

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

Name and tags Info

Name: bhushan

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

Summary

Number of instances Info: 1

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

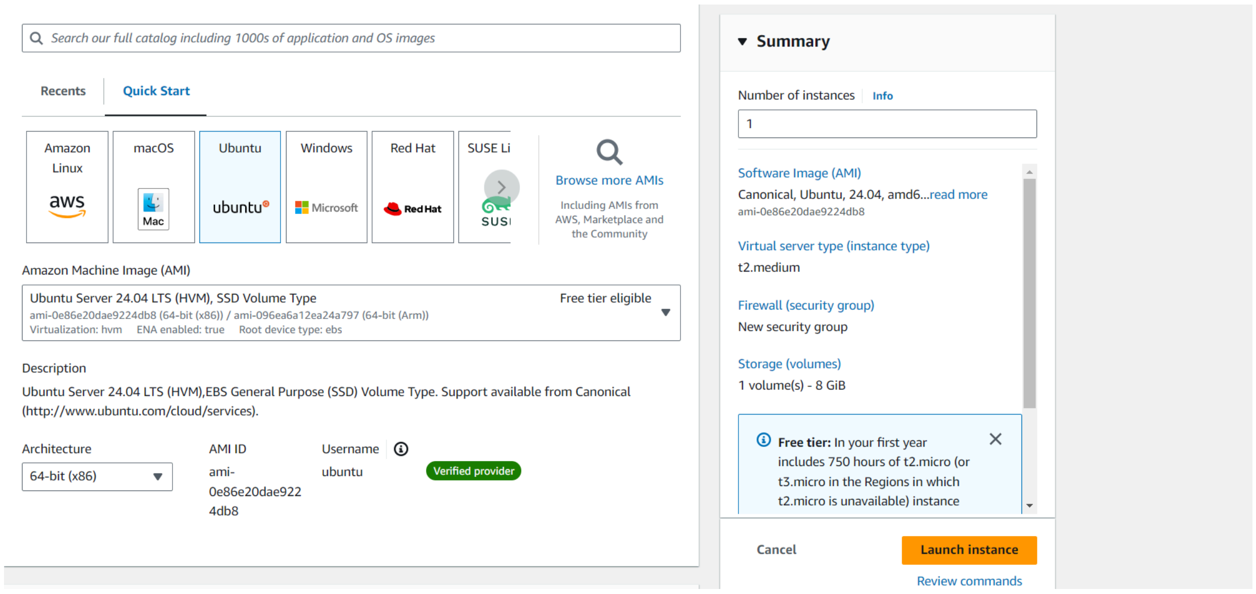
Virtual server type (instance type): t2.medium

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel | Launch instance | Review commands



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

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li | Browse more AMIs (Including AMIs from AWS, Marketplace and the Community)

Amazon Machine Image (AMI)

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

Description

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

Architecture	AMI ID	Username	
64-bit (x86)	ami-0e86e20dae9224db8	ubuntu	Verified provider

Summary

Number of instances Info: 1

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

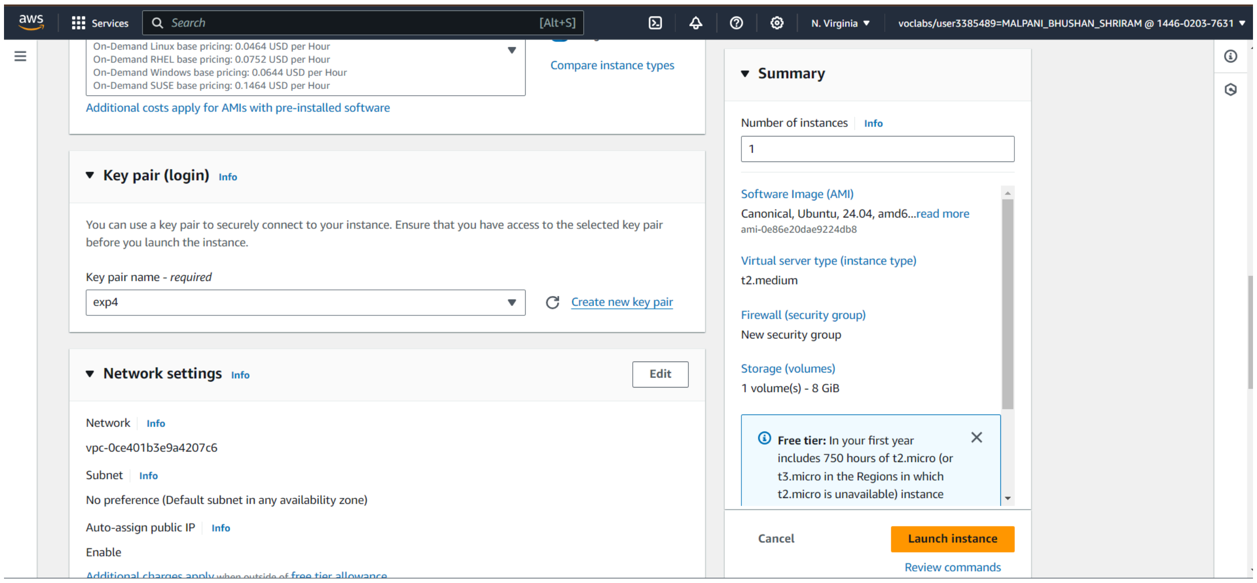
Virtual server type (instance type): t2.medium

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel | Launch instance | Review commands


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3385489=MALPANI_BHUSHAN_SHRIRAM @ 1446-0203-7631

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

Additional costs apply for AMIs with pre-installed software

Compare instance types

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*

exp4

Create new key pair

Network settings Info

Edit

Network Info

vpc-0ce401b3e9a4207c6

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Summary

Number of instances Info

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd6...read more

ami-0e86e20dae9224db8

Virtual server type (instance type)

t2.medium

Firewall (security group)

New security group

Storage (volumes)

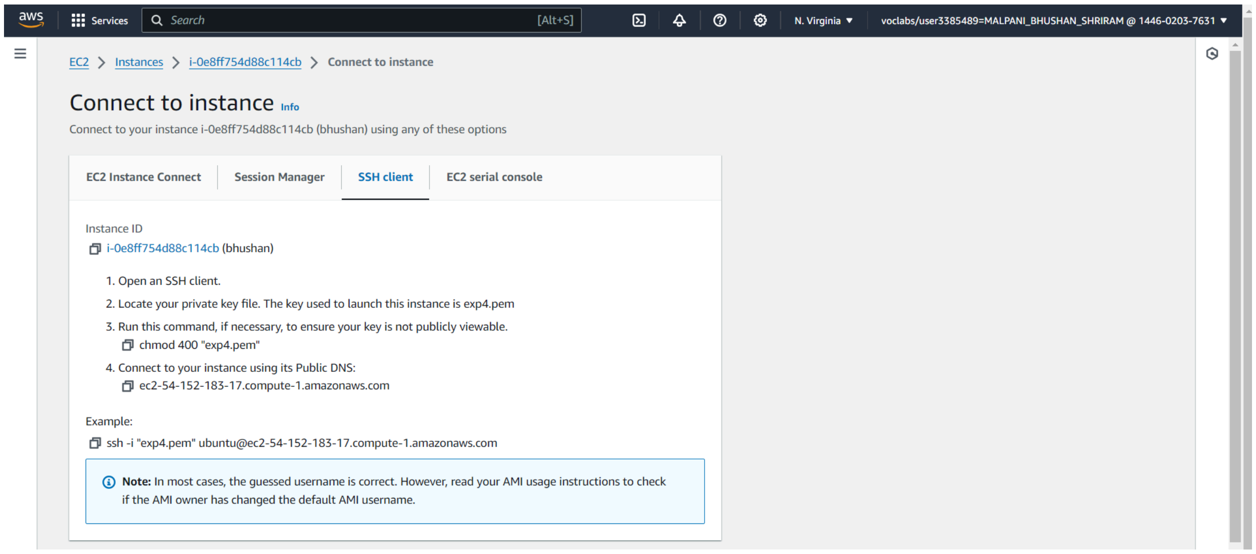
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance

Cancel

Launch instance

Review commands


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3385489=MALPANI_BHUSHAN_SHRIRAM @ 1446-0203-7631

EC2 > Instances > i-0e8ff754d88c114cb > Connect to instance

Connect to instance Info

Connect to your instance i-0e8ff754d88c114cb (bhushan) using any of these options

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

Instance ID

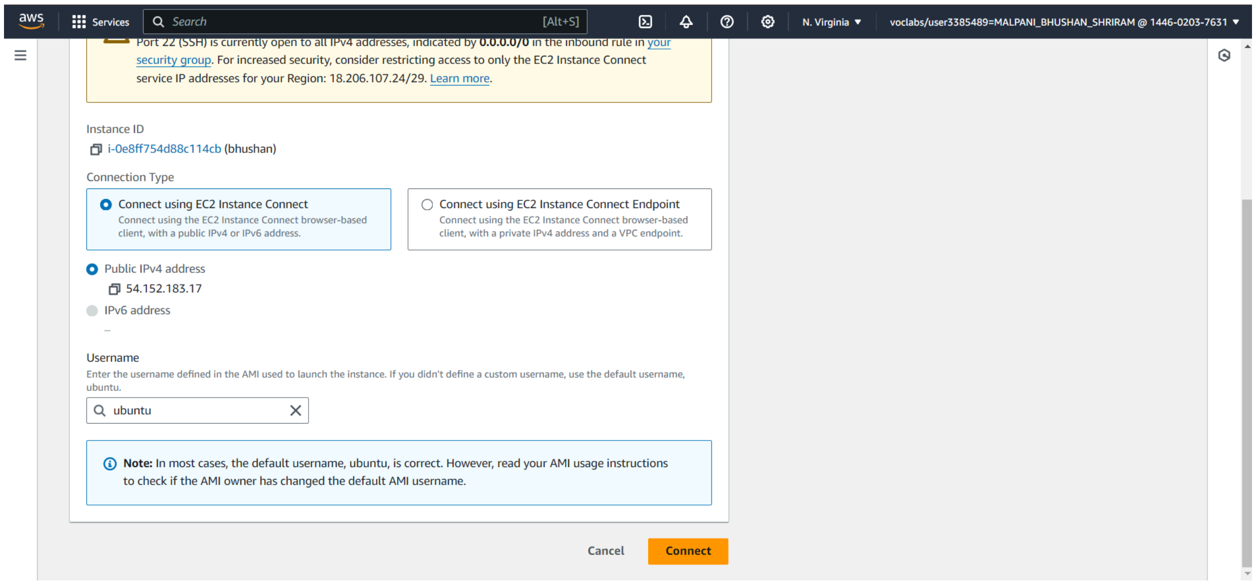
i-0e8ff754d88c114cb (bhushan)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp4.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   ```
   chmod 400 "exp4.pem"
   ```
4. Connect to your instance using its Public DNS:
   ```
   ec2-54-152-183-17.compute-1.amazonaws.com
   ```

Example:

```
ssh -i "exp4.pem" ubuntu@ec2-54-152-183-17.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.


aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user3385489=MALPANI_BHUSHAN_SHRIRAM @ 1446-0203-7631

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by 0.0.0.0/0 in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. Learn more.

Instance ID
i-0e8ff754d88c114cb (bhushan)

Connection Type
Connect using EC2 Instance Connect — Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
Connect using EC2 Instance Connect Endpoint — Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

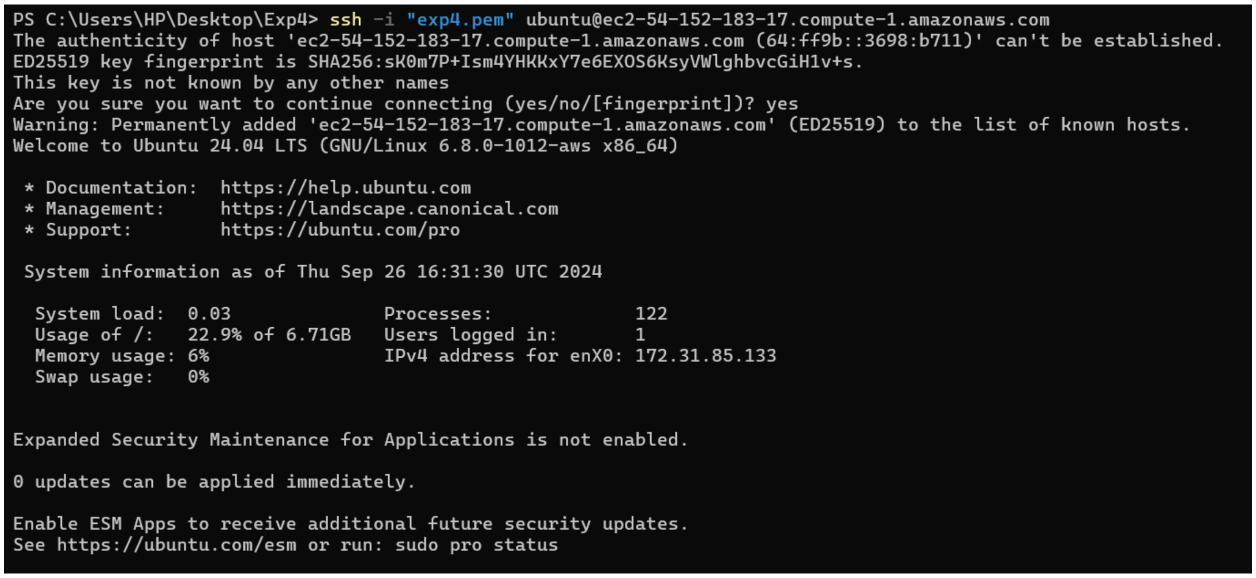
Public IPv4 address
54.152.183.17
IPv6 address
–

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.
ubuntu

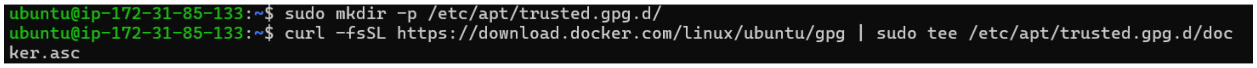
Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

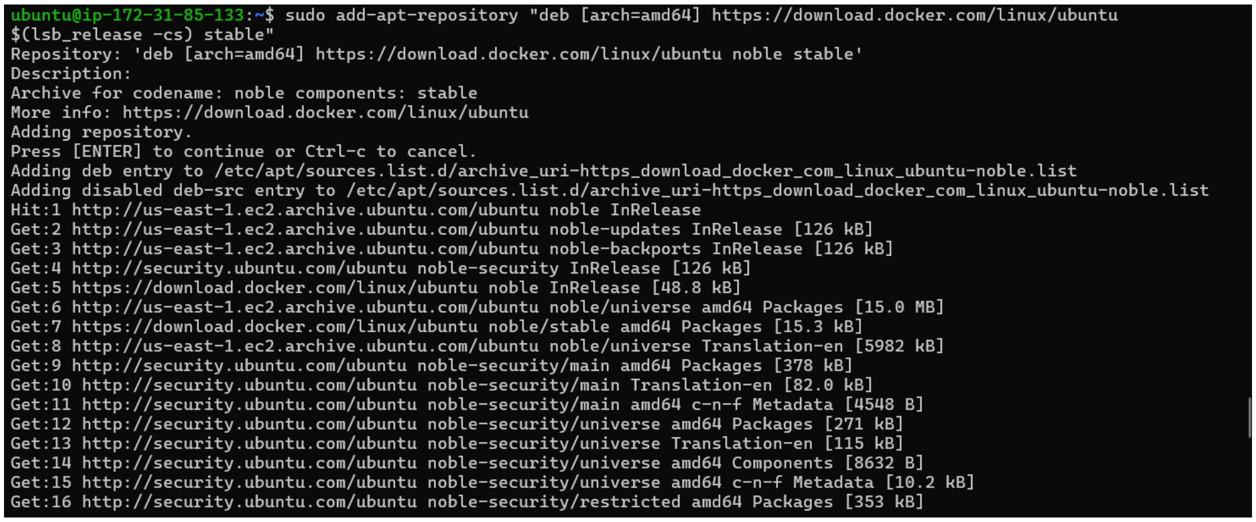
Cancel Connect

Step 4:navigate to the folder open the terminal and paste the ssh command:
ssh -i "sahilexp4key.pem" ubuntu@ec2-52-201-236-39.compute-1.amazonaws.com


```
PS C:\Users\HP\Desktop\Exp4> ssh -i "exp4.pem" ubuntu@ec2-54-152-183-17.compute-1.amazonaws.com
The authenticity of host 'ec2-54-152-183-17.compute-1.amazonaws.com (64:ff9b::3698:b711)' can't be established.
ED25519 key fingerprint is SHA256:sK0m7P+Ism4YHKKxY7e6EXOS6KsyVWlghbvcGiH1v+s.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-152-183-17.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Thu Sep 26 16:31:30 UTC 2024

  System load:  0.03              Processes:             122
  Usage of /:   22.9% of 6.71GB   Users logged in:       1
  Memory usage: 6%                IPv4 address for enX0: 172.31.85.133
  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
```

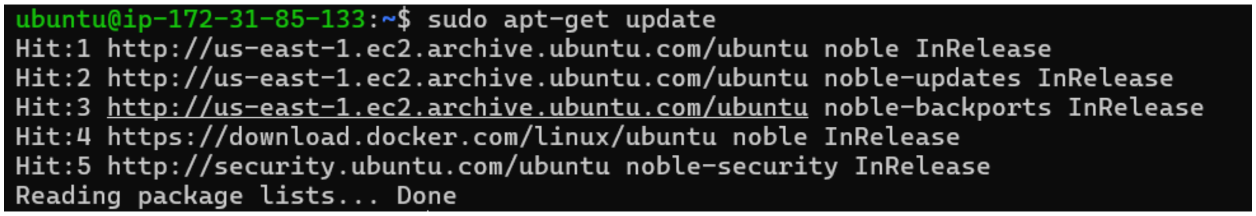
Step 5:Install docker
Use the commands given below to install docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"


```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/apt/trusted.gpg.d/
ubuntu@ip-172-31-85-133:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/doc
ker.asc
```



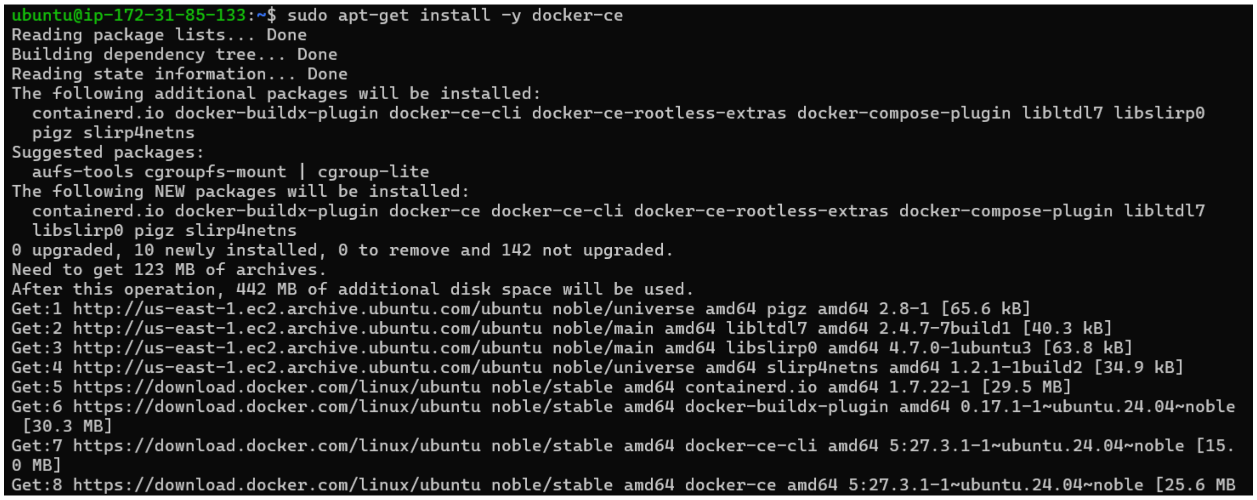
```
ubuntu@ip-172-31-85-133:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
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]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [378 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [82.0 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4548 B]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [271 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [115 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:15 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.2 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
```

Use:
sudo apt-get update



```
ubuntu@ip-172-31-85-133:~$ 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
```

Use:
sudo apt-get install -y docker-ce



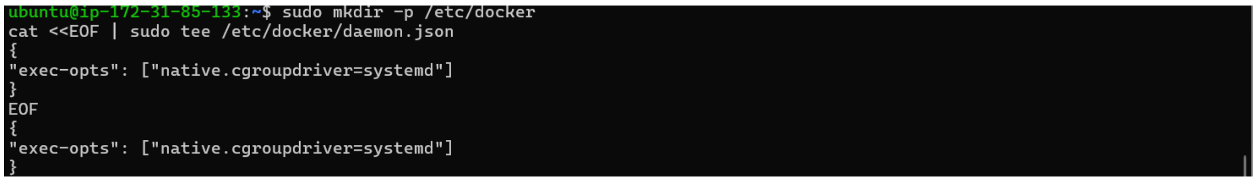
```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin 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-cli docker-ce-rootless-extras docker-compose-plugin libltdl7
  libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 142 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]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.24.04~noble
 [30.3 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.24.04~noble [15.
0 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.24.04~noble [25.6 MB
```

Now the docker is installed;

Now lets enable the docker:

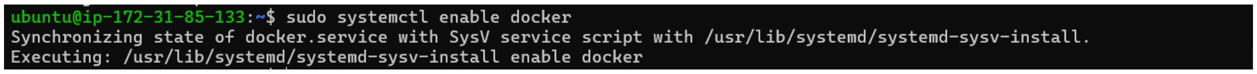
sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json {

"exec-opts": ["native.cgroupdriver=systemd"] } EOF



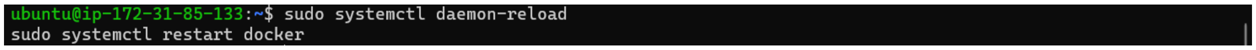
```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

sudo systemctl enable docker



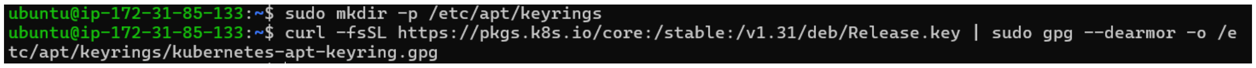
```
ubuntu@ip-172-31-85-133:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

sudo systemctl daemon-reload



```
ubuntu@ip-172-31-85-133:~$ sudo systemctl daemon-reload
sudo systemctl restart docker
```

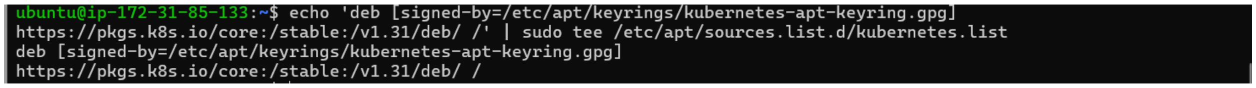
sudo systemctl restart docker



```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/apt/keyrings
ubuntu@ip-172-31-85-133:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /e
tc/apt/keyrings/kubernetes-apt-keyring.gpg
```

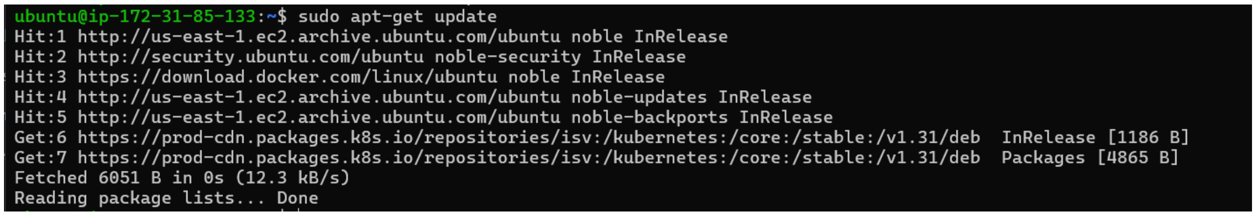
Step 6:Now lets install kubernetes;
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

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



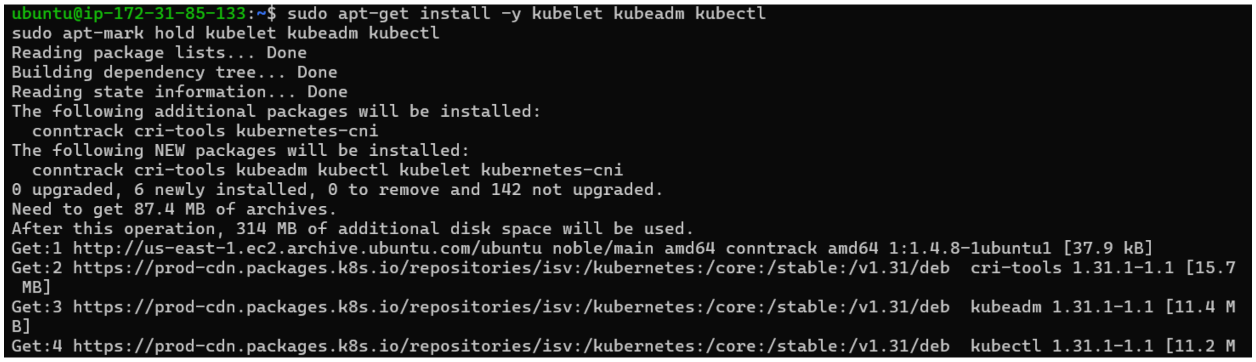
```
ubuntu@ip-172-31-85-133:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

sudo apt-get update



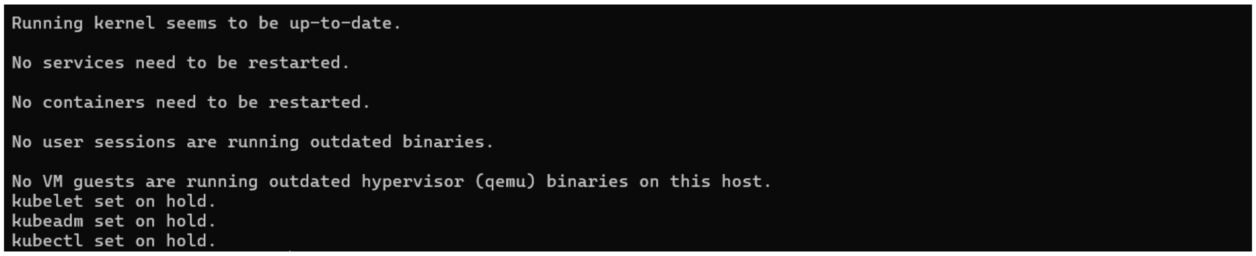
```
ubuntu@ip-172-31-85-133:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 https://download.docker.com/linux/ubuntu noble InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports 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 0s (12.3 kB/s)
Reading package lists... Done
```

sudo apt-get install -y kubelet kubeadm kubectl



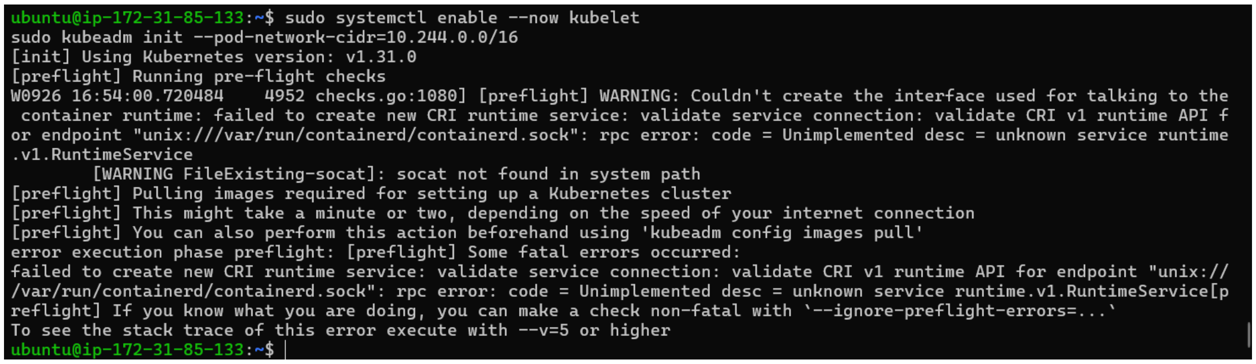
```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
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:/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 M
B]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 M
```

sudo apt-mark hold kubelet kubeadm kubectl



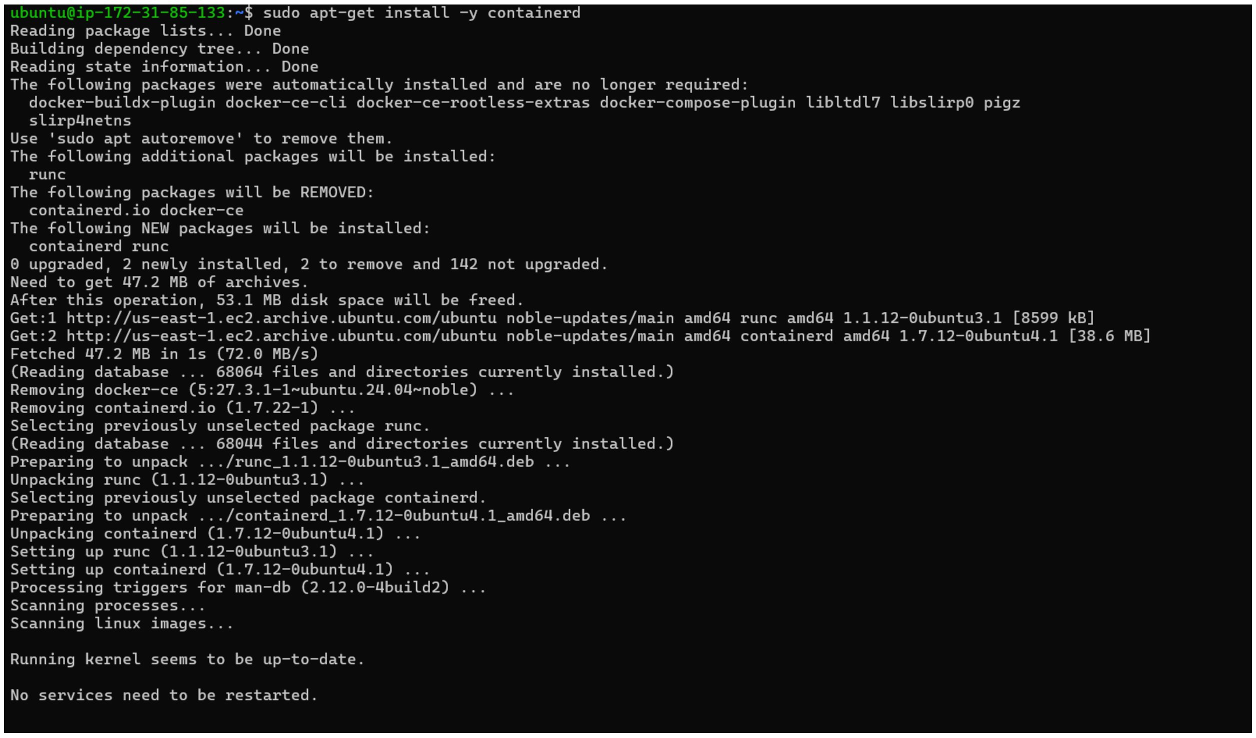
```
Running kernel seems to be up-to-date.

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.
```

sudo systemctl enable --now kubelet
//Skip:sudo kubeadm init --pod-network-cidr=10.244.0.0/16

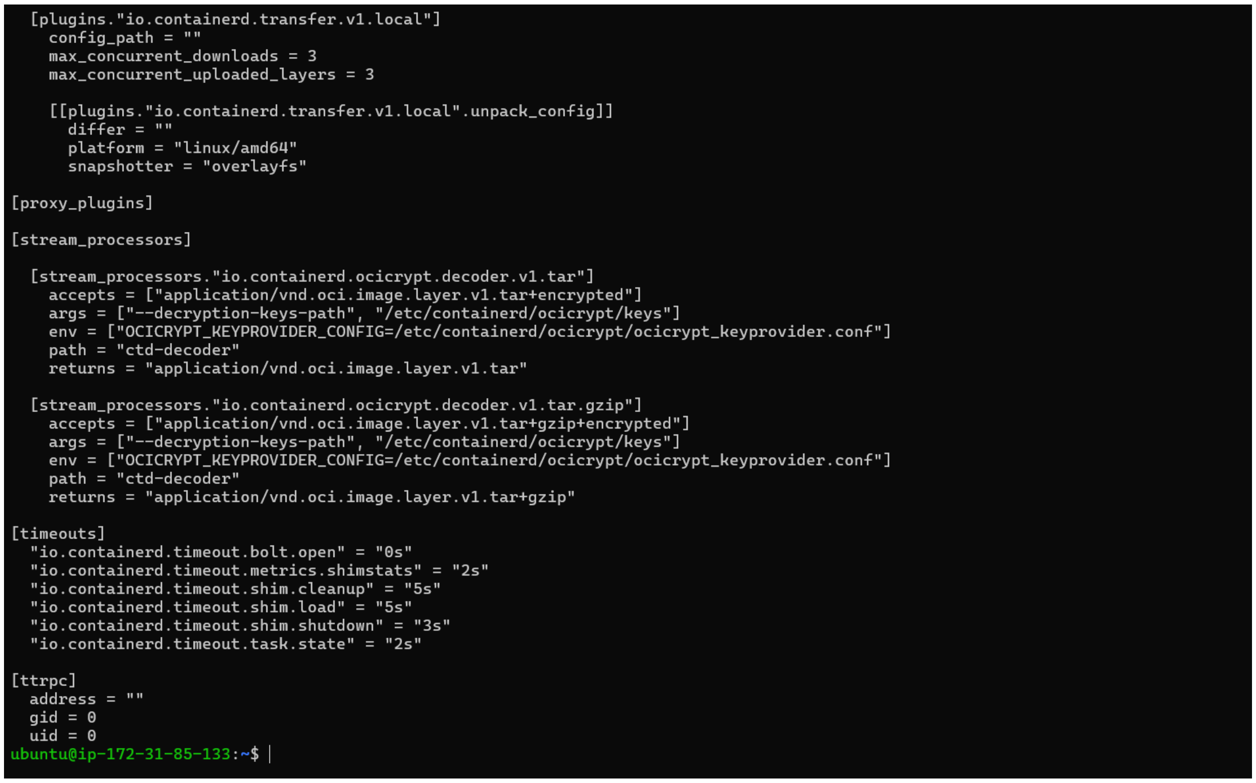


```
ubuntu@ip-172-31-85-133:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0926 16:54:00.720484    4952 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the
 container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API f
or endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime
.v1.RuntimeService
        [WARNING FileExisting-socat]: socat not found in system path
[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'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix://
/var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[p
reflight] 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-172-31-85-133:~$
```

sudo apt-get install -y containerd

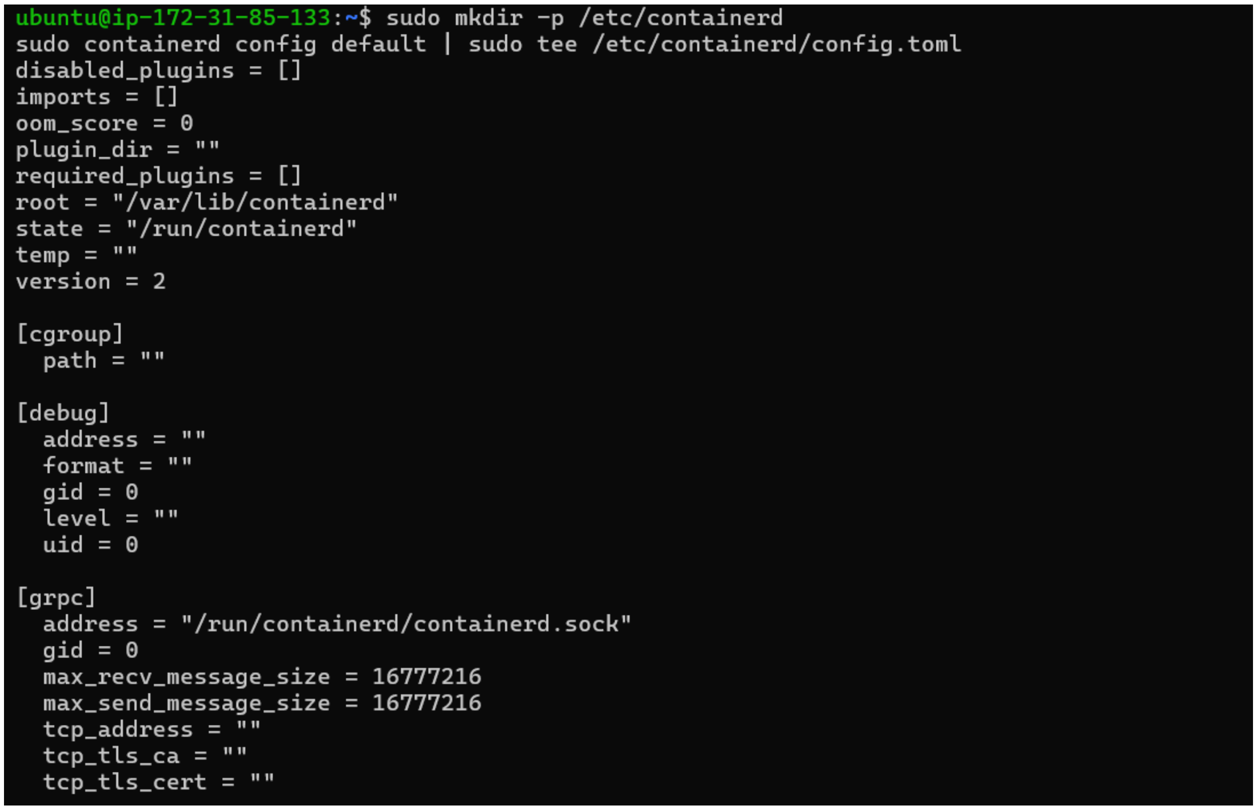


```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras 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 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (72.0 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68044 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
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.
```



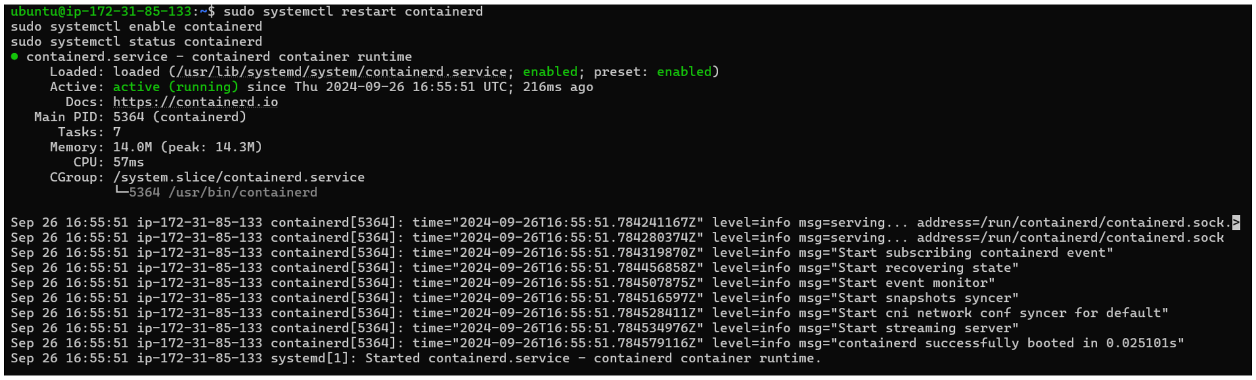
```
  [plugins."io.containerd.transfer.v1.local"]
    config_path = ""
    max_concurrent_downloads = 3
    max_concurrent_uploaded_layers = 3

    [[plugins."io.containerd.transfer.v1.local".unpack_config]]
      differ = ""
      platform = "linux/amd64"
      snapshotter = "overlayfs"

[proxy_plugins]

[stream_processors]

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar"

  [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
    accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
    args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
    env = ["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider.conf"]
    path = "ctd-decoder"
    returns = "application/vnd.oci.image.layer.v1.tar+gzip"

[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
ubuntu@ip-172-31-85-133:~$ |
```

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



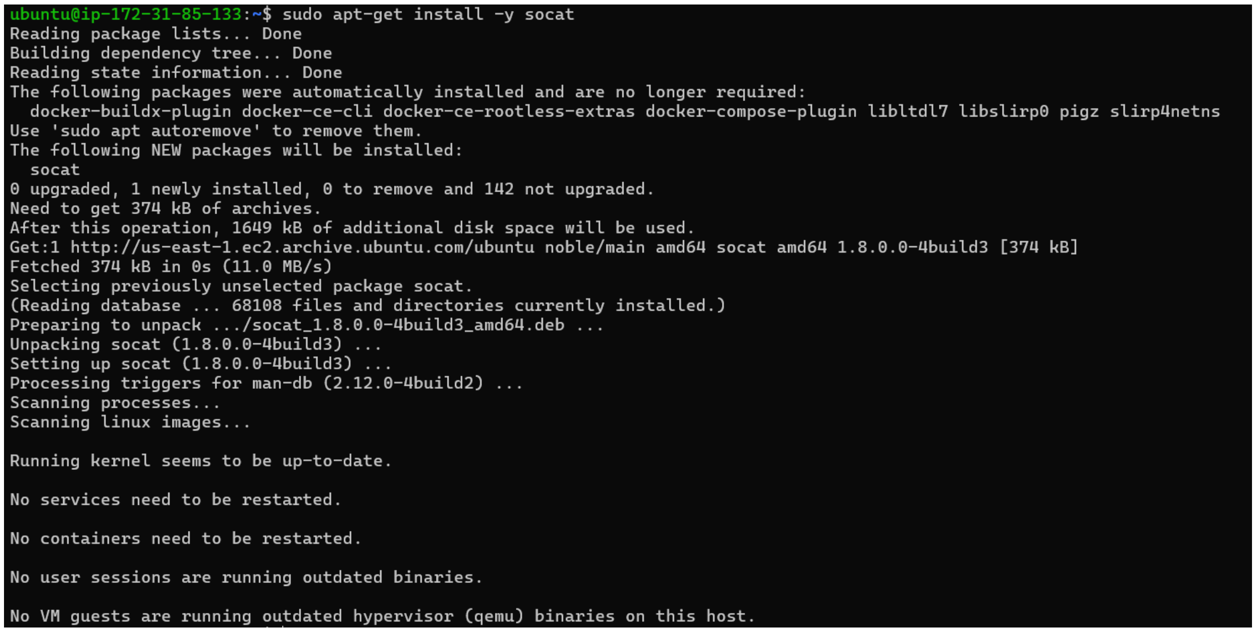
```
ubuntu@ip-172-31-85-133:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[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 = ""
```

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



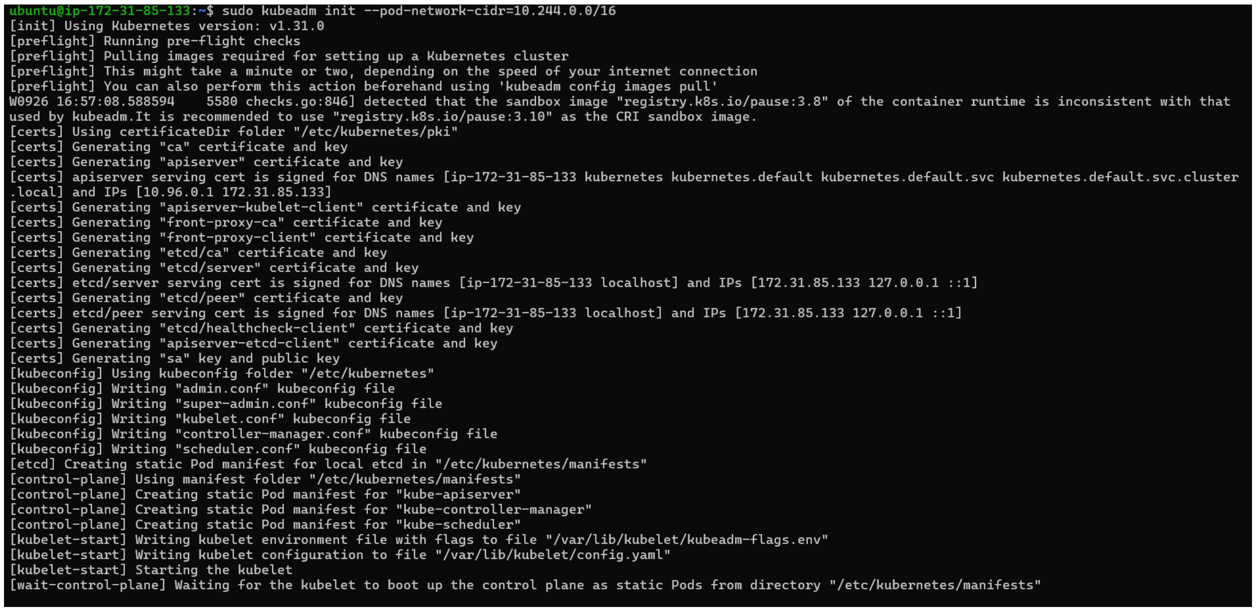
```
ubuntu@ip-172-31-85-133:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-09-26 16:55:51 UTC; 216ms ago
       Docs: https://containerd.io
   Main PID: 5364 (containerd)
      Tasks: 7
     Memory: 14.0M (peak: 14.3M)
        CPU: 57ms
     CGroup: /system.slice/containerd.service
             └─5364 /usr/bin/containerd

Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784241167Z" level=info msg=serving... address=/run/containerd/containerd.sock.>
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784280374Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784319870Z" level=info msg="Start subscribing containerd event"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784456858Z" level=info msg="Start recovering state"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784507875Z" level=info msg="Start event monitor"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784516597Z" level=info msg="Start snapshots syncer"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784528411Z" level=info msg="Start cni network conf syncer for default"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784534976Z" level=info msg="Start streaming server"
Sep 26 16:55:51 ip-172-31-85-133 containerd[5364]: time="2024-09-26T16:55:51.784579116Z" level=info msg="containerd successfully booted in 0.025101s"
Sep 26 16:55:51 ip-172-31-85-133 systemd[1]: Started containerd.service - containerd container runtime.
```

sudo apt-get install -y socat

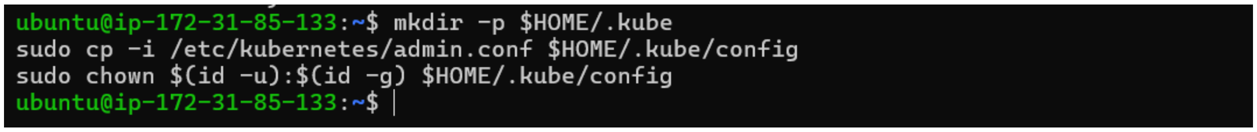


```
ubuntu@ip-172-31-85-133:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras 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 amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (11.0 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Step 7: Intitialize the kubernete:
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

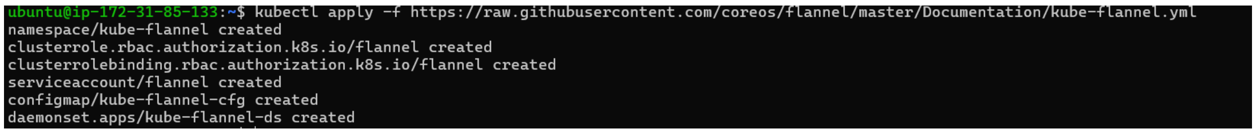


```
ubuntu@ip-172-31-85-133:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0926 16:57:08.588594    5580 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-172-31-85-133 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster
.local] and IPs [10.96.0.1 172.31.85.133]
[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-172-31-85-133 localhost] and IPs [172.31.85.133 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-85-133 localhost] and IPs [172.31.85.133 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
```



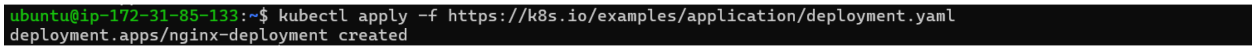
```
ubuntu@ip-172-31-85-133:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-85-133:~$
```

kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



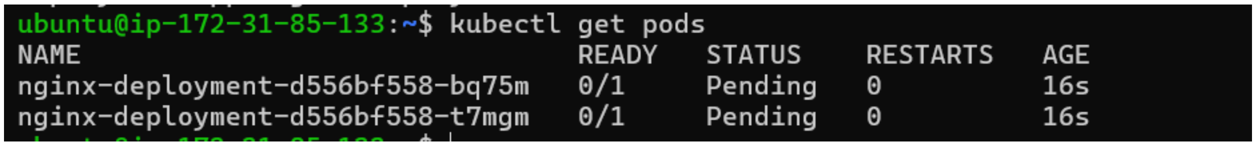
```
ubuntu@ip-172-31-85-133:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Step 8:Now we can deploy our nginx server on this cluster using following steps:
kubectl apply -f https://k8s.io/examples/application/deployment.yaml



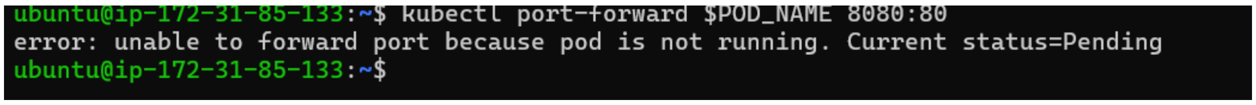
```
ubuntu@ip-172-31-85-133:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods



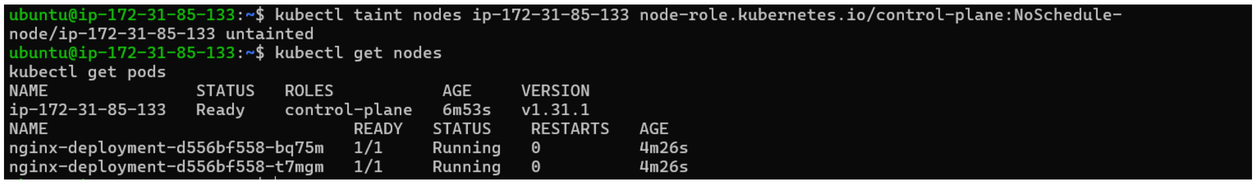
```
ubuntu@ip-172-31-85-133:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-bq75m    0/1     Pending   0          16s
nginx-deployment-d556bf558-t7mgm    0/1     Pending   0          16s
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80



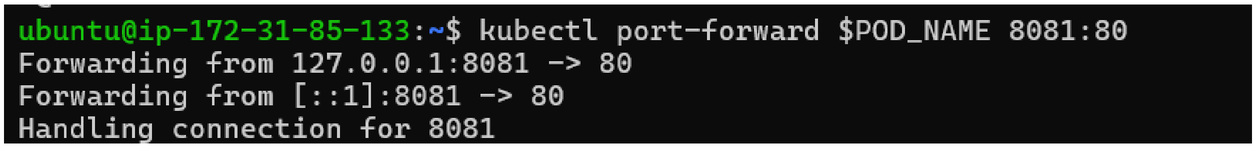
```
ubuntu@ip-172-31-85-133:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-85-133:~$
```

kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted kubectl get nodes



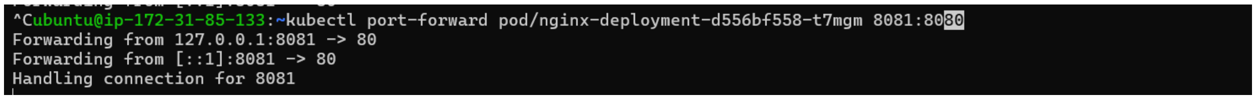
```
ubuntu@ip-172-31-85-133:~$ kubectl taint nodes ip-172-31-85-133 node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-85-133 untainted
ubuntu@ip-172-31-85-133:~$ kubectl get nodes
kubectl get pods
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-85-133   Ready    control-plane   6m53s   v1.31.1
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-bq75m   1/1     Running   0          4m26s
nginx-deployment-d556bf558-t7mgm   1/1     Running   0          4m26s
```

kubectl get pods



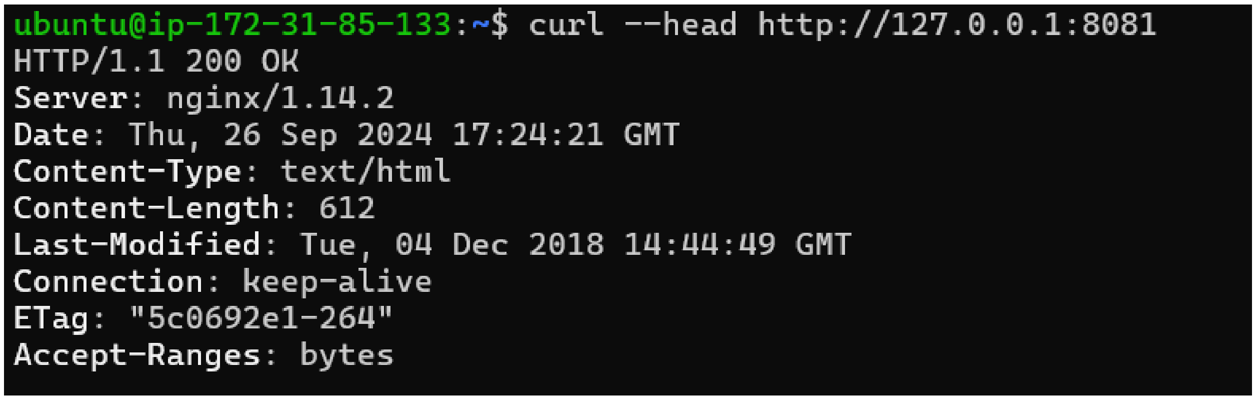
```
ubuntu@ip-172-31-85-133:~$ kubectl port-forward $POD_NAME 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80



```
^Cubuntu@ip-172-31-85-133:~kubectl port-forward pod/nginx-deployment-d556bf558-t7mgm 8081:8080
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
```

Step 9 :check deployment:
Open new terminal in folder,
Paste the ssh key,
Type
curl --head http://127.0.0.8081



```
ubuntu@ip-172-31-85-133:~$ curl --head http://127.0.0.1:8081
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 26 Sep 2024 17:24:21 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
```

Status Code 200 tells us that we have successfully deployed our nginx server on ec2 instance
Now we have successfully deployed our nginx server on our ec2 instance.

Conclusion

In this experiment, we installed kubectl, the command-line tool for managing Kubernetes clusters, using the appropriate package manager for our system. After setting up the kubeconfig file to connect with the Kubernetes cluster, we verified the connection and the cluster status using kubectl get nodes. We then deployed our first Kubernetes application by writing the necessary configuration in files, ensuring the application was defined correctly. Following the deployment, we inspected the resources and application status with kubectl get pods to confirm the application was running successfully in the cluster. This experiment demonstrated the basic interaction with Kubernetes using kubectl for managing resources and monitoring deployments.