Advance DevOps Lab
Experiment 04

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

Roll No.	42
Name	NAIKWADI YASH SHIVDAS
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO2: To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
Grade:	

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 interface (CLI) tool that allows users to interact with a Kubernetes cluster. As a central component of Kubernetes, **kubectl** provides the functionality needed to manage applications, inspect cluster resources, and perform administrative tasks through simple commands executed in a terminal.

Importance of Kubectl in Kubernetes Management

kubectl is essential for effective Kubernetes management for several reasons:

1. User-Friendly Interface: **kubectl** offers a command-line interface that simplifies complex operations, making it accessible for developers and administrators.

2. Resource Management: Users can create, update, and delete Kubernetes resources such as pods, deployments, services, and namespaces with straightforward commands.

3. Deployment and Scaling: **kubectl** facilitates the deployment of containerized applications and allows users to easily scale them up or down based on current demands.

4. Monitoring and Troubleshooting: The tool enables users to monitor the health and status of applications running in the cluster. It provides commands to view logs, describe resources, and check the current state of pods and services, which aids in troubleshooting issues.

5. Configuration Management: **kubectl** supports YAML configuration files that define the desired state of applications and resources, allowing users to apply changes consistently and repeatedly across different environments.

Key Features of Kubectl

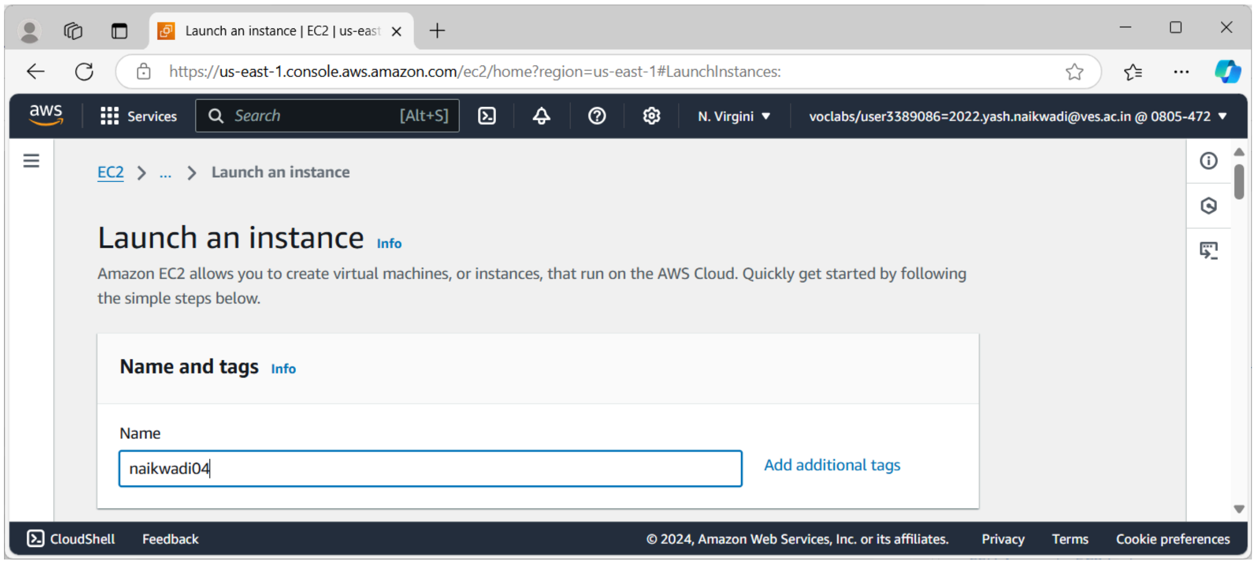
- Resource Discovery: **kubectl** can list all resources in a Kubernetes cluster, providing an overview of what is running and its current status.

- Detailed Resource Descriptions: The tool can display detailed information about specific resources, including configuration, current state, events, and resource utilization.

- Access to Container Logs: Users can view the logs generated by application containers, helping diagnose issues and understand application behavior.

- Namespace Management: **kubectl** allows for the management of namespaces, which help organize resources and provide isolation within a cluster.

Launch an EC2 Instance


Launch an instance | EC2 | us-east
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Services
Search [Alt+S]
N. Virgini
voclabs/user3389086=2022.yash.naikwadi@ves.ac.in @ 0805-472
EC2 > ... > Launch an instance
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Launch an instance Info

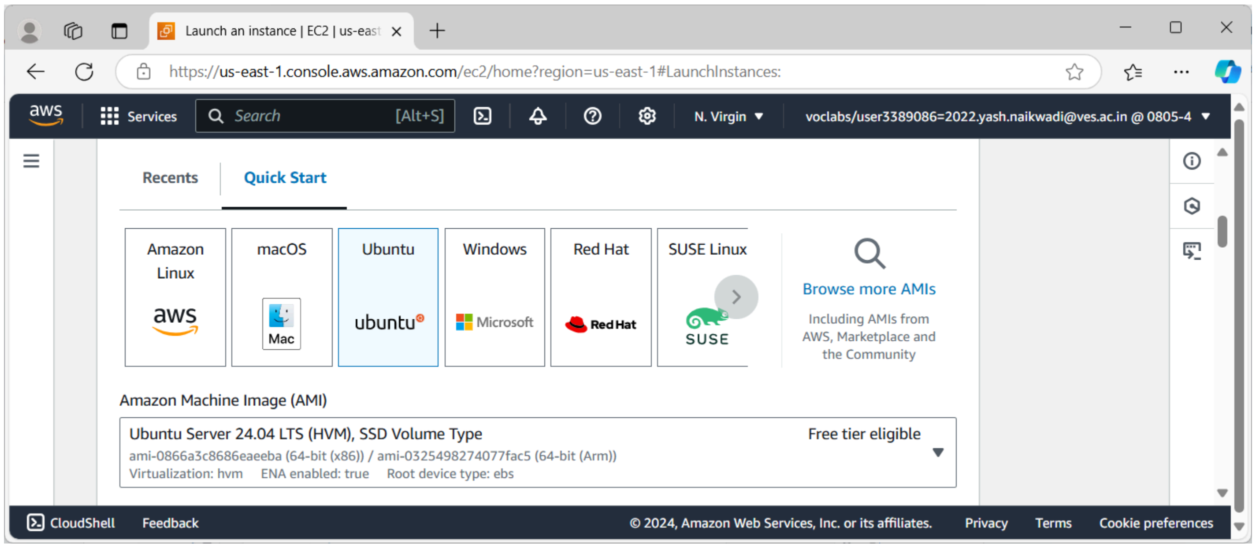
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: naikwadi04

Add additional tags

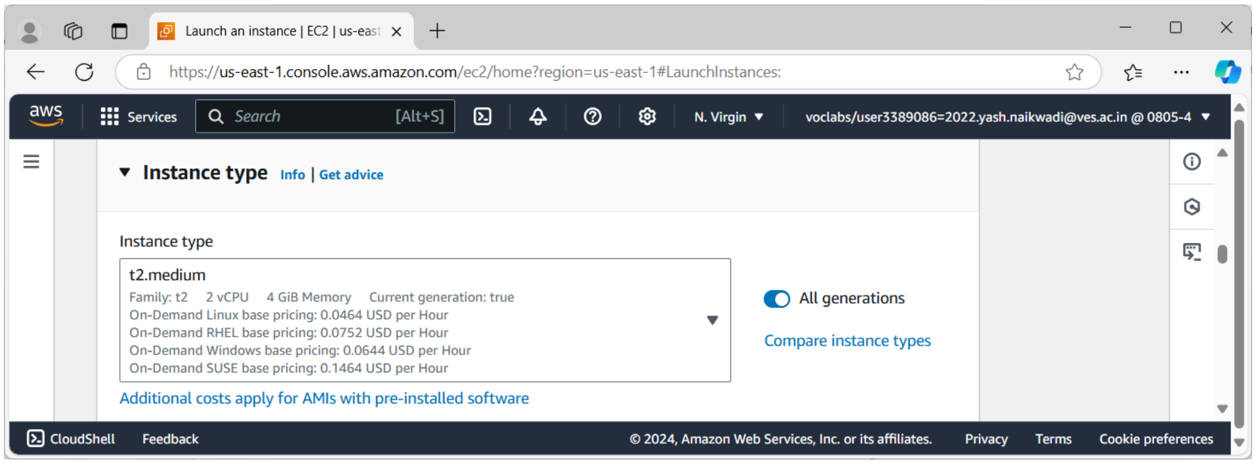
Choose Ubuntu Server 20.04 LTS (HVM), SSD Volume Type as your AMI.



Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Select t2.medium as the instance type (2 CPUs).



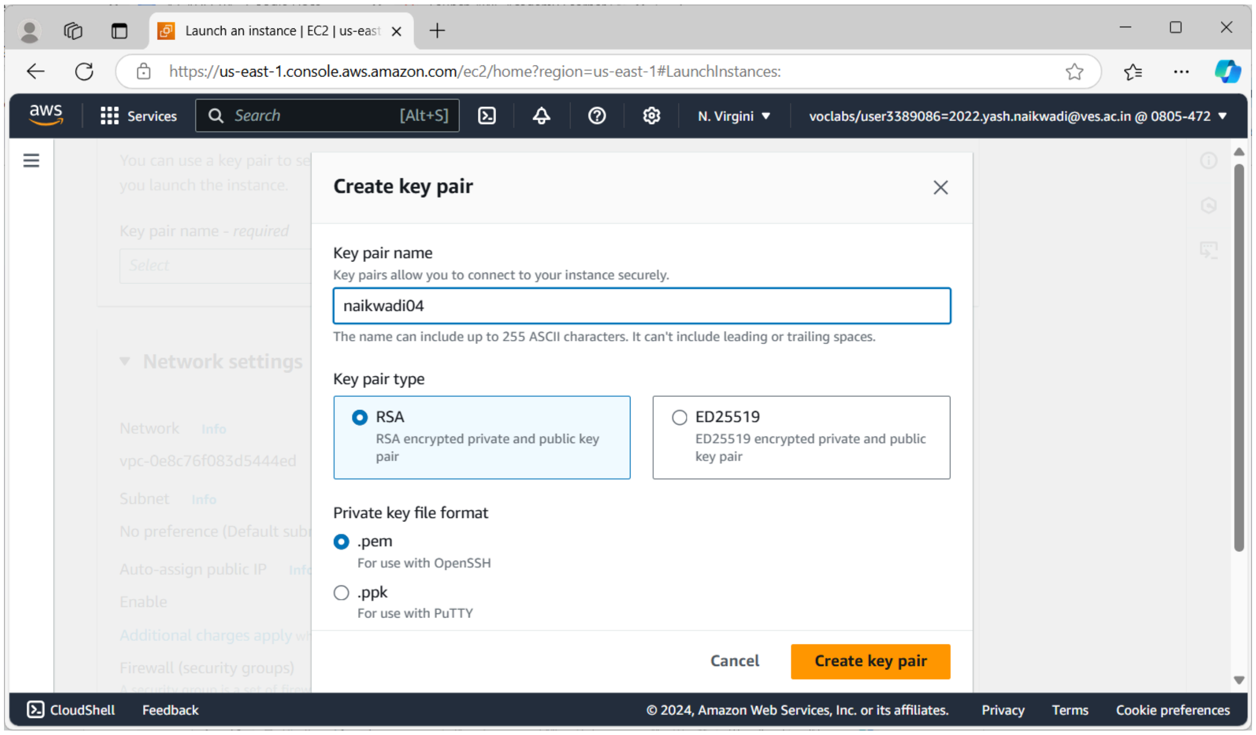
Instance type Info | Get advice

Instance type

t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour

Additional costs apply for AMIs with pre-installed software

Select Create a new key pair, name it (e.g., naikwadi04), and click Download Key Pair. This will download a .pem file to your computer.



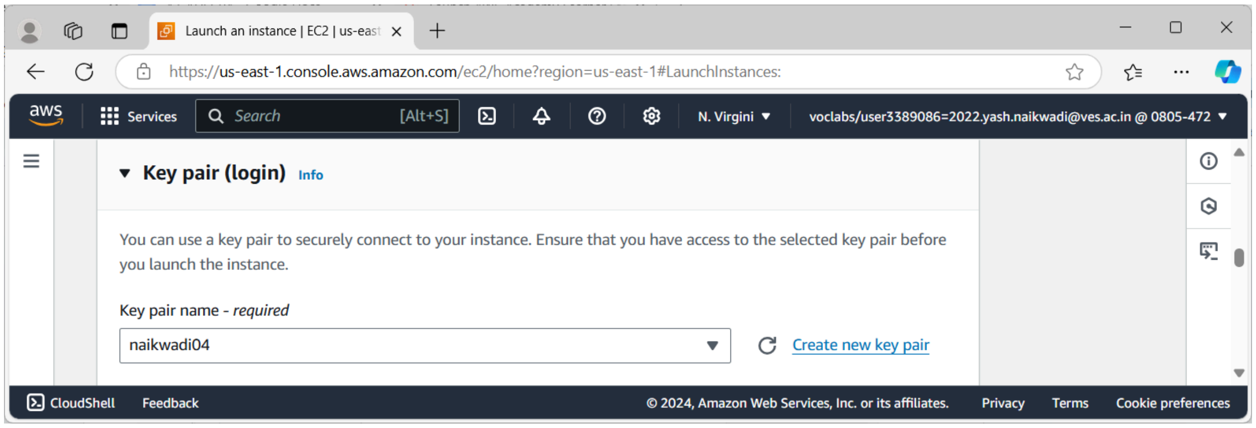
Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.
naikwadi04
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

Cancel | Create key pair

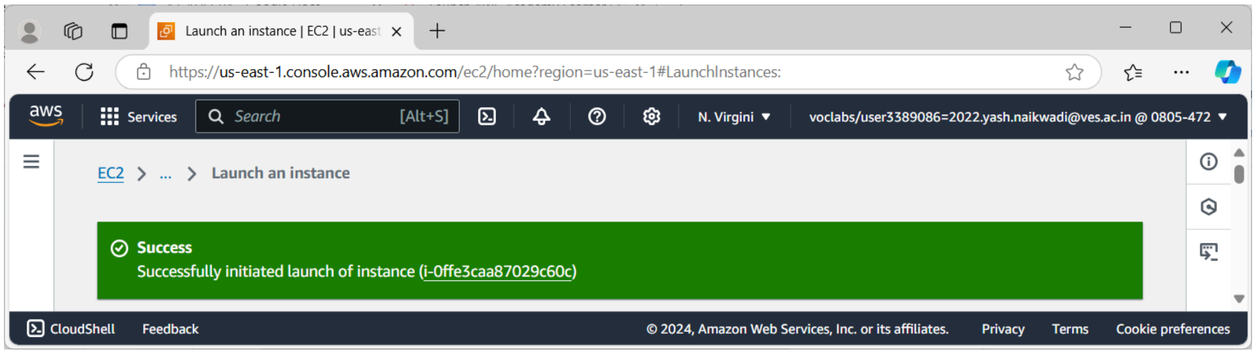


Key pair (login) Info

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.

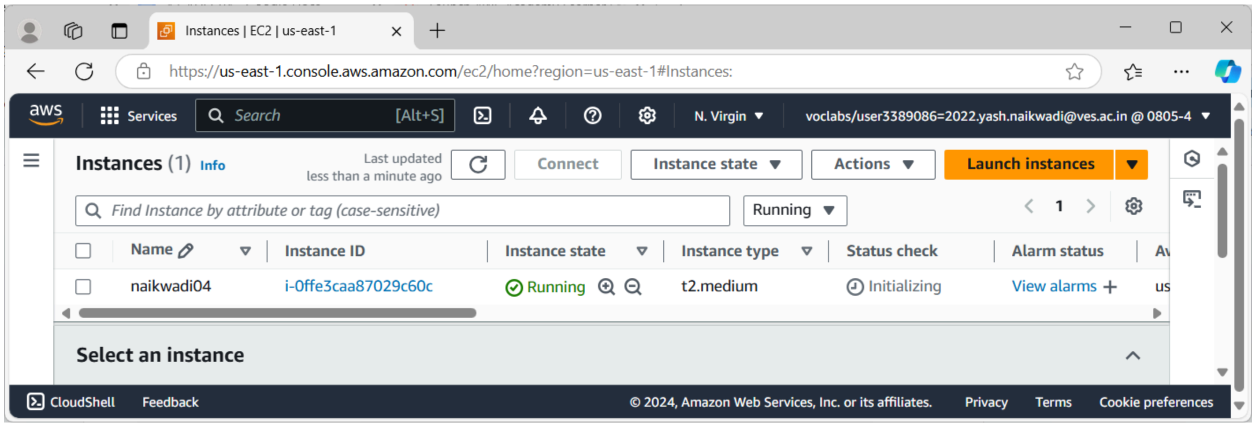
Key pair name - *required*
naikwadi04
Create new key pair

Launch the EC2 instance.



EC2 > ... > Launch an instance

Success
Successfully initiated launch of instance (i-0ffe3caa87029c60c)



Instances (1) Info

Last updated less than a minute ago

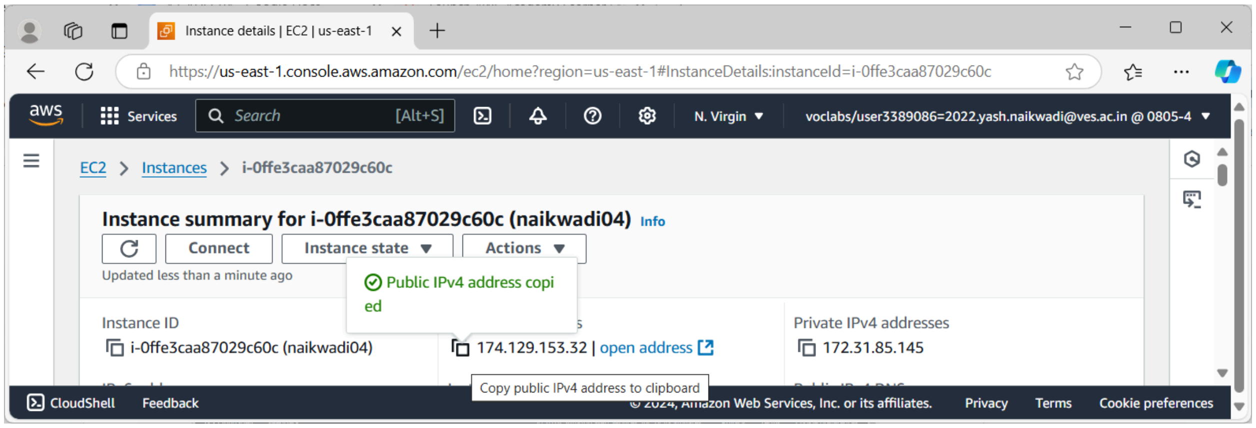
Find Instance by attribute or tag (case-sensitive) — Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Av
naikwadi04	i-0ffe3caa87029c60c	Running	t2.medium	Initializing	View alarms +	us

Select an instance

© 2024, Amazon Web Services, Inc. or its affiliates.

Click on the instance id of the newly created ec2 instance and copy the public url of it.



https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-0ffe3caa87029c60c

Instance summary for i-0ffe3caa87029c60c (naikwadi04) Info

Updated less than a minute ago

Public IPv4 address copied

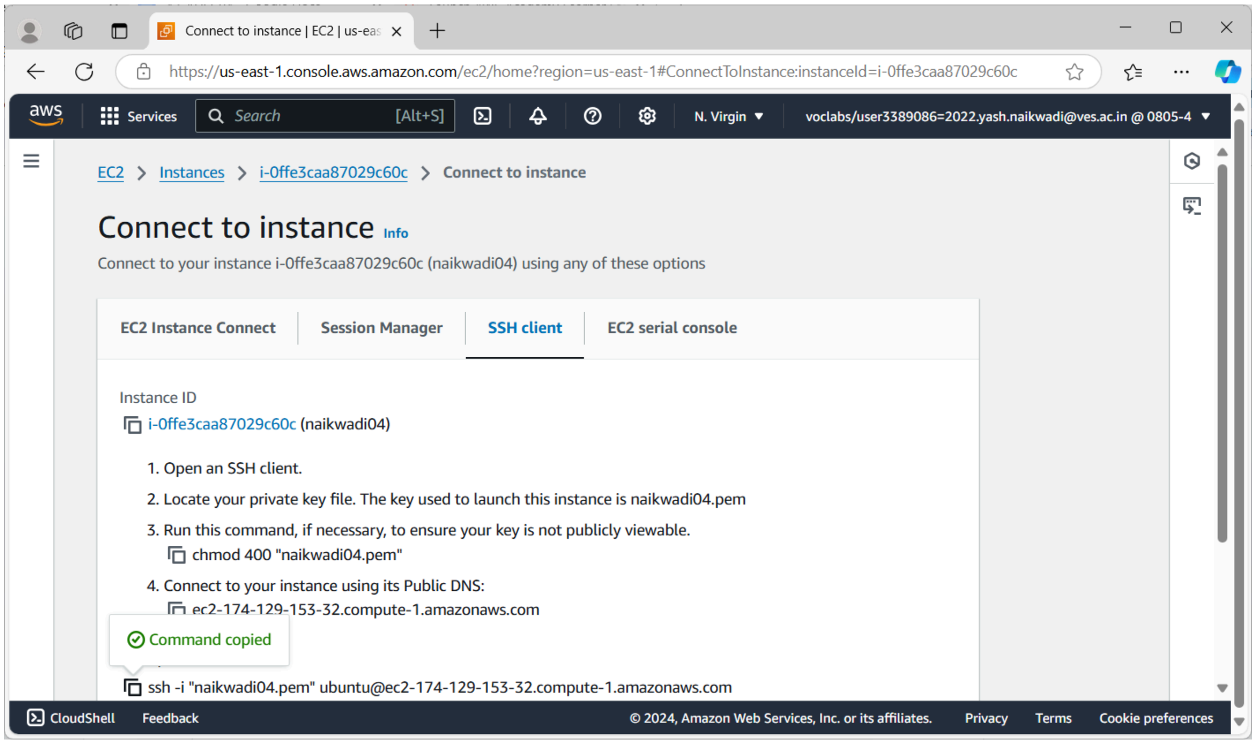
Instance ID: i-0ffe3caa87029c60c (naikwadi04)

174.129.153.32 | open address

Private IPv4 addresses: 172.31.85.145

Copy public IPv4 address to clipboard

Click on connect and copy the command as shown



Connect to instance Info

Connect to your instance i-0ffe3caa87029c60c (naikwadi04) using any of these options

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

Instance ID: i-0ffe3caa87029c60c (naikwadi04)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is naikwadi04.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 "naikwadi04.pem"
```

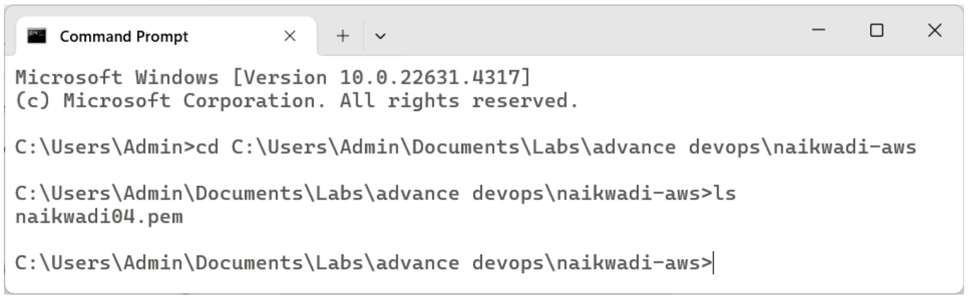
4. Connect to your instance using its Public DNS:

```
ec2-174-129-153-32.compute-1.amazonaws.com
```

Command copied

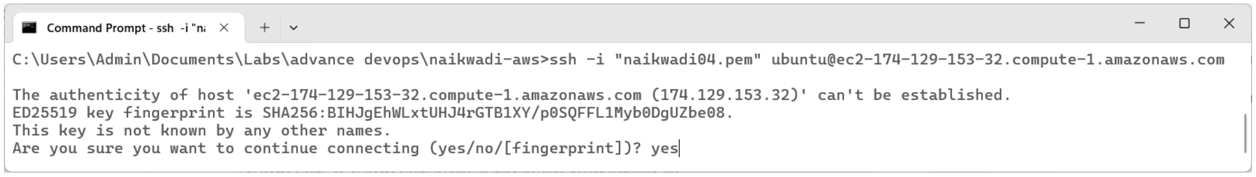
```
ssh -i "naikwadi04.pem" ubuntu@ec2-174-129-153-32.compute-1.amazonaws.com
```

If you are using Windows, you might need a terminal like **Git Bash** or **PuTTY**.
Use the cd command to navigate to the folder where your downloaded key is located.



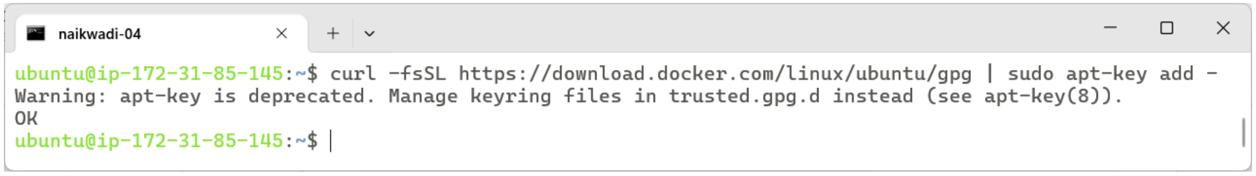
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws

C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws>ls
naikwadi04.pem

C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws>
```

Run the following command, replacing the placeholder with your actual EC2 public DNS:
ssh -i "naikwadi-exp04.pem" ubuntu@ec2-54-92-219-25.compute-1.amazonaws.com



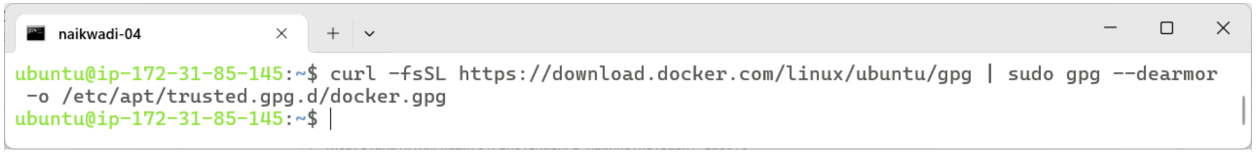
```
C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws>ssh -i "naikwadi04.pem" ubuntu@ec2-174-129-153-32.compute-1.amazonaws.com

The authenticity of host 'ec2-174-129-153-32.compute-1.amazonaws.com (174.129.153.32)' can't be established.
ED25519 key fingerprint is SHA256:BIHJgEhWLxtUHJ4rGTB1XY/p0SQFFL1Myb0DgUZbe08.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

To install Docker, Run the Following Commands:
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -



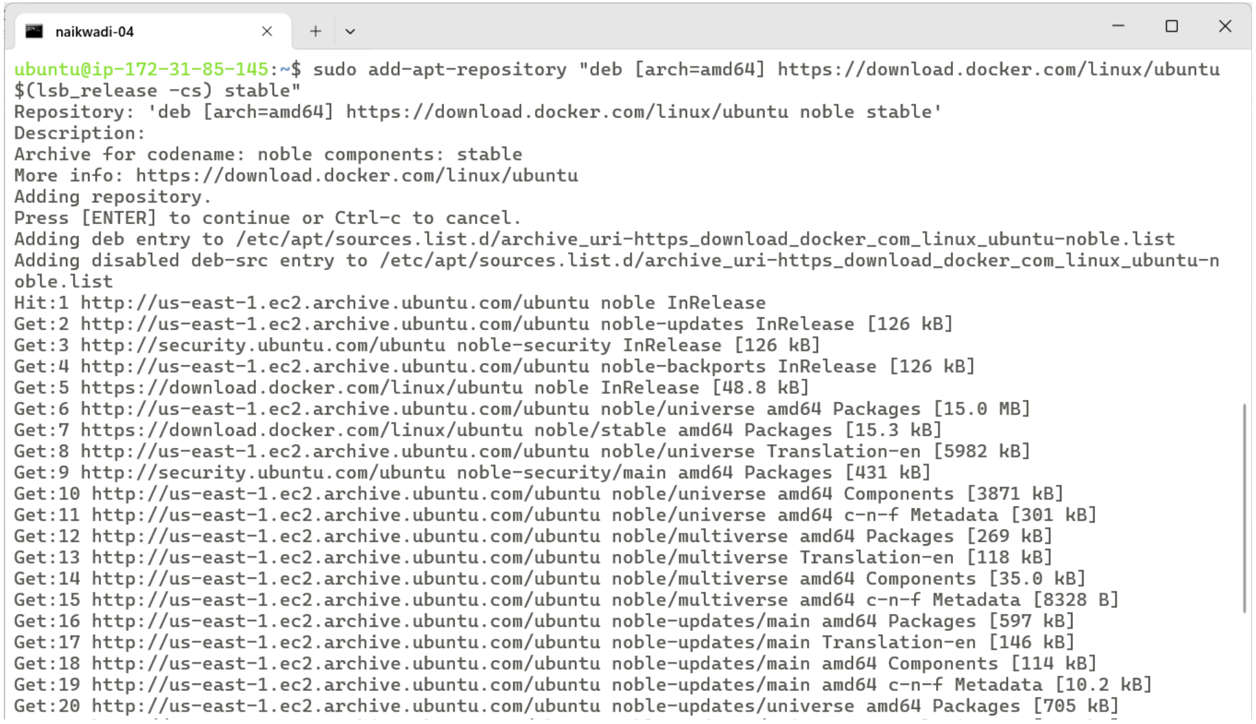
```
ubuntu@ip-172-31-85-145:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-85-145:~$ 
```

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg



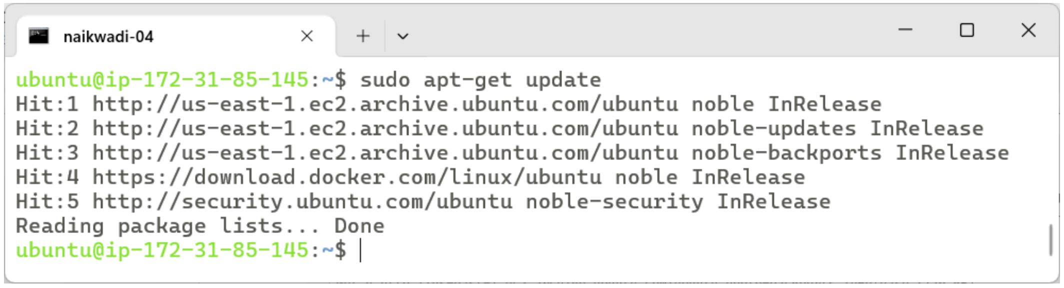
```
ubuntu@ip-172-31-85-145:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
 -o /etc/apt/trusted.gpg.d/docker.gpg
ubuntu@ip-172-31-85-145:~$ 
```

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



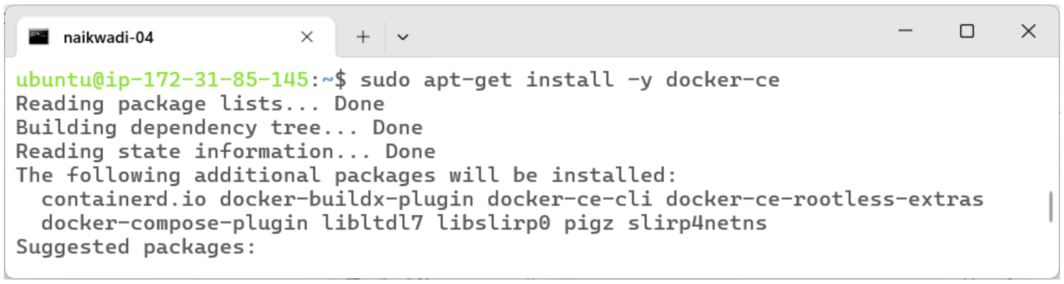
```
ubuntu@ip-172-31-85-145:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
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-n
oble.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://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports 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 [431 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [597 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [146 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [114 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [10.2 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [705 kB]
```

sudo apt-get update



```
ubuntu@ip-172-31-85-145:~$ 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
ubuntu@ip-172-31-85-145:~$
```

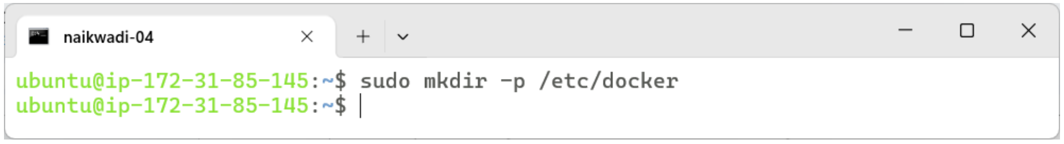
sudo apt-get install -y docker-ce



```
ubuntu@ip-172-31-85-145:~$ 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:
```

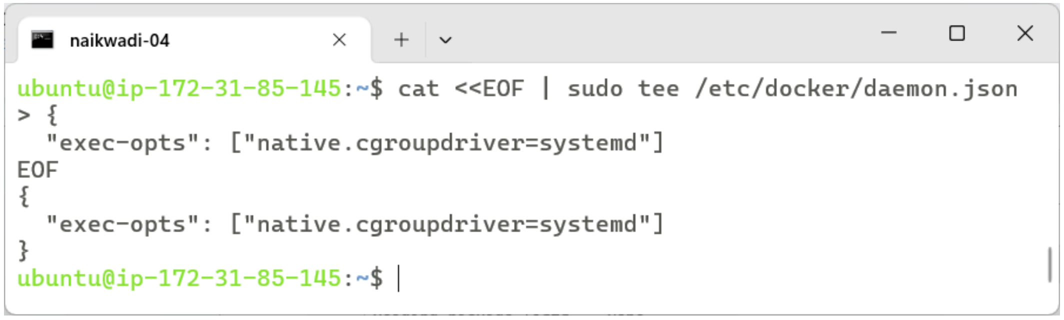
Configure Docker

sudo mkdir -p /etc/docker



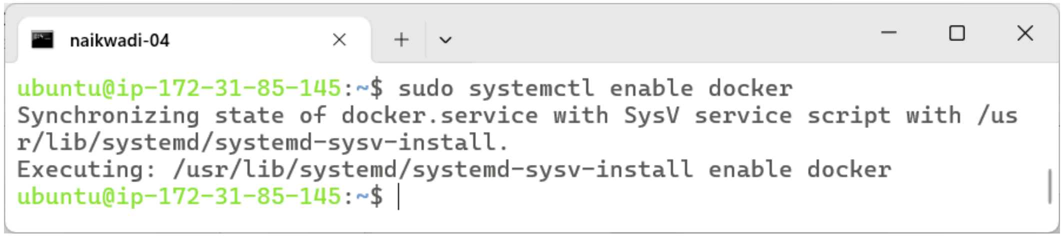
```
ubuntu@ip-172-31-85-145:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-85-145:~$
```

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



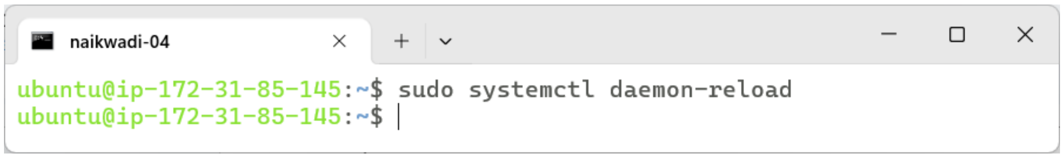
```
ubuntu@ip-172-31-85-145:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
> {
  "exec-opts": ["native.cgroupdriver=systemd"]
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-85-145:~$
```

sudo systemctl enable docker



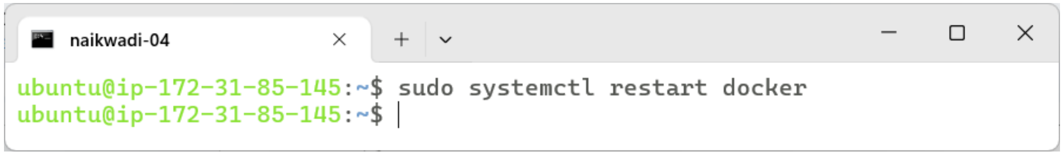
```
ubuntu@ip-172-31-85-145:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /us
r/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-85-145:~$
```

sudo systemctl daemon-reload



```
ubuntu@ip-172-31-85-145:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-85-145:~$
```

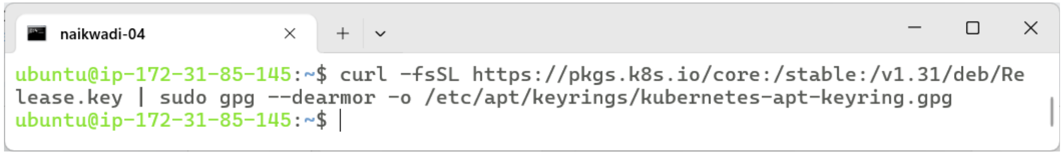
sudo systemctl restart docker



```
ubuntu@ip-172-31-85-145:~$ sudo systemctl restart docker
ubuntu@ip-172-31-85-145:~$ |
```

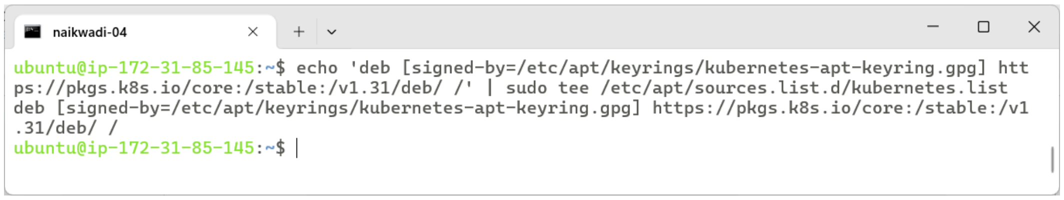
To Install Kubernetes, Add the Kubernetes Repository

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



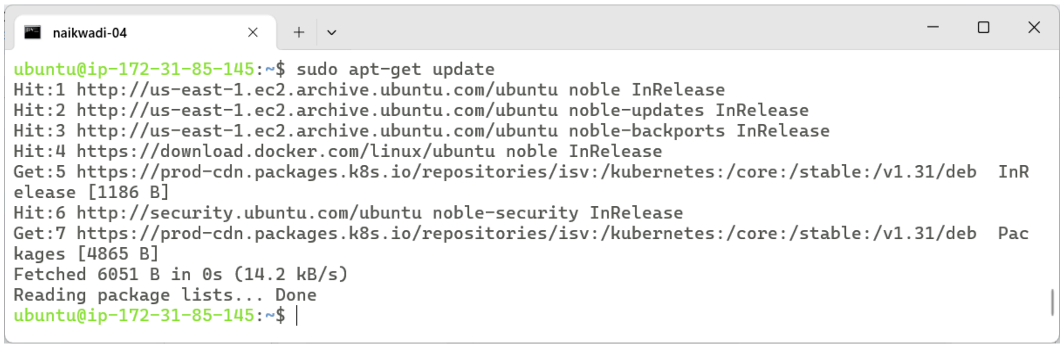
```
ubuntu@ip-172-31-85-145:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Re
lease.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-85-145:~$ |
```

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



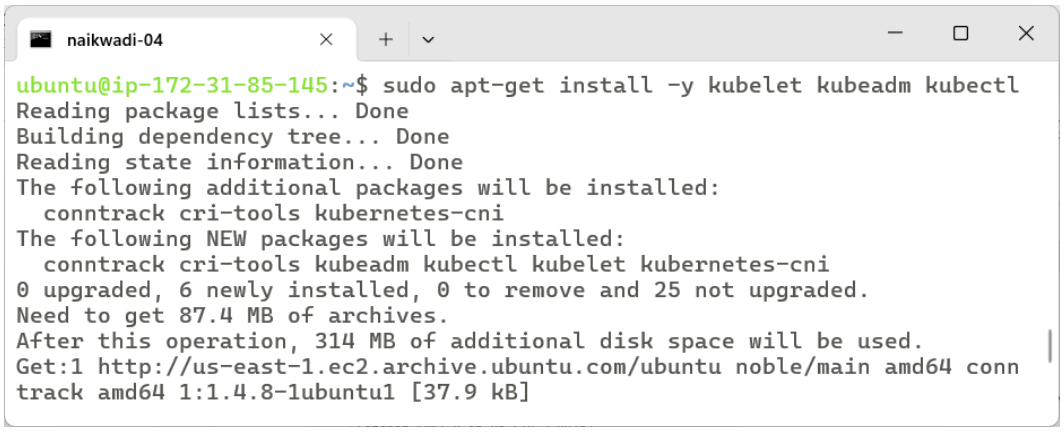
```
ubuntu@ip-172-31-85-145:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] htt
ps://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1
.31/deb/ /
ubuntu@ip-172-31-85-145:~$ |
```

sudo apt-get update



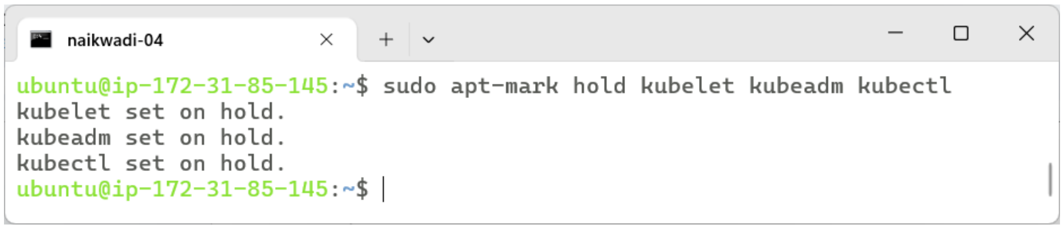
```
ubuntu@ip-172-31-85-145:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InR
elease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Pac
kages [4865 B]
Fetched 6051 B in 0s (14.2 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-85-145:~$ |
```

sudo apt-get install -y kubelet kubeadm kubectl



```
ubuntu@ip-172-31-85-145:~$ 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 25 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 conn
track amd64 1:1.4.8-1ubuntu1 [37.9 kB]
```

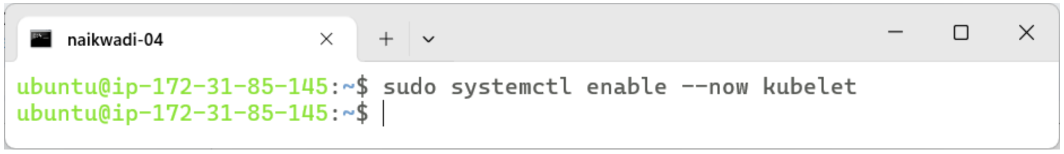
sudo apt-mark hold kubelet kubeadm kubectl



naikwadi-04

```
ubuntu@ip-172-31-85-145:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-85-145:~$
```

Enable and Start Kubelet:
sudo systemctl enable --now kubelet

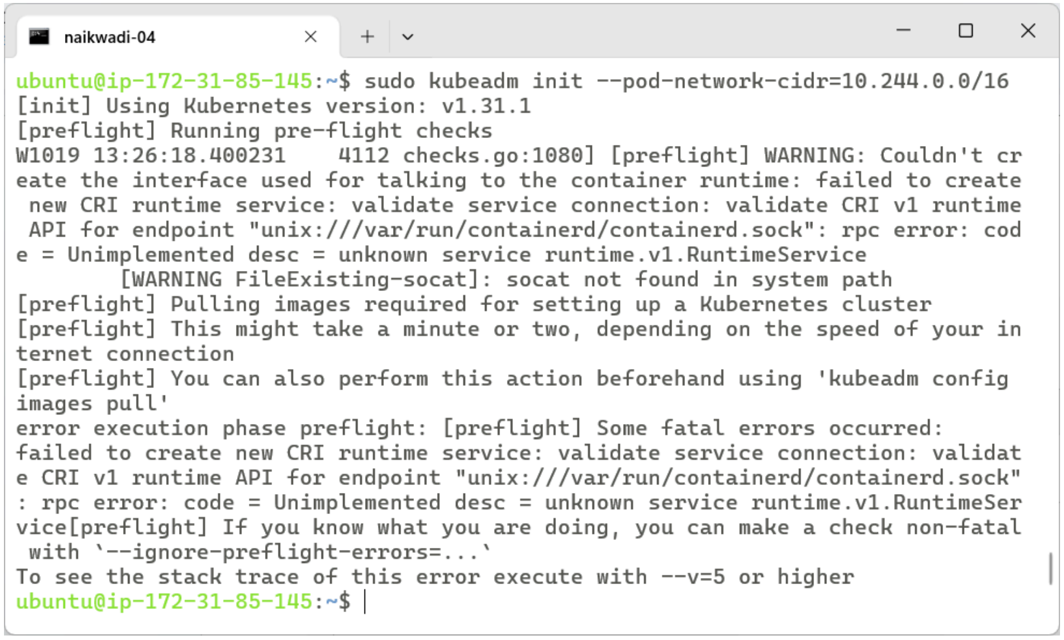


naikwadi-04

```
ubuntu@ip-172-31-85-145:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-85-145:~$
```

To Initialize the Kubernetes Cluster, Run the Command

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

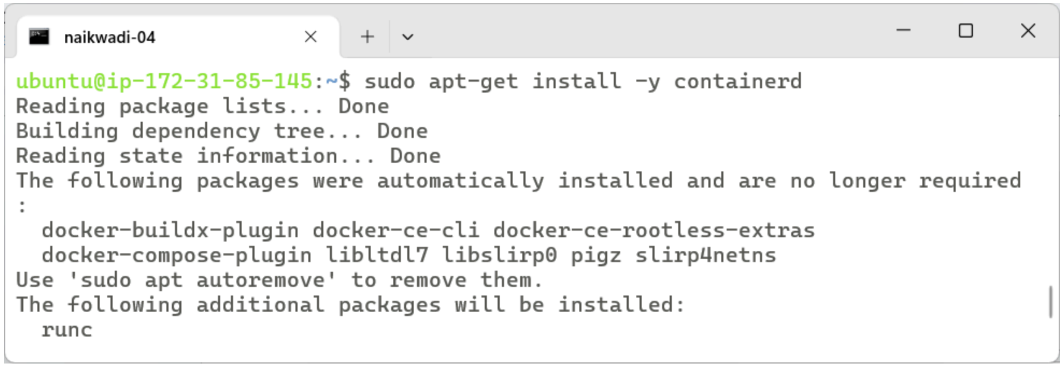


naikwadi-04

```
ubuntu@ip-172-31-85-145:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
W1019 13:26:18.400231    4112 checks.go:1080] [preflight] WARNING: Couldn't cr
eate the interface used for talking to the container runtime: 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: cod
e = 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 in
ternet 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: validat
e CRI v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock"
: rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeSer
vice[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-172-31-85-145:~$
```

If you encounter errors, run the following commands to fix containerd issues:

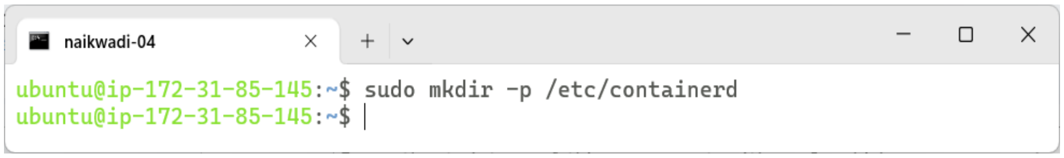
sudo apt-get install -y containerd



naikwadi-04

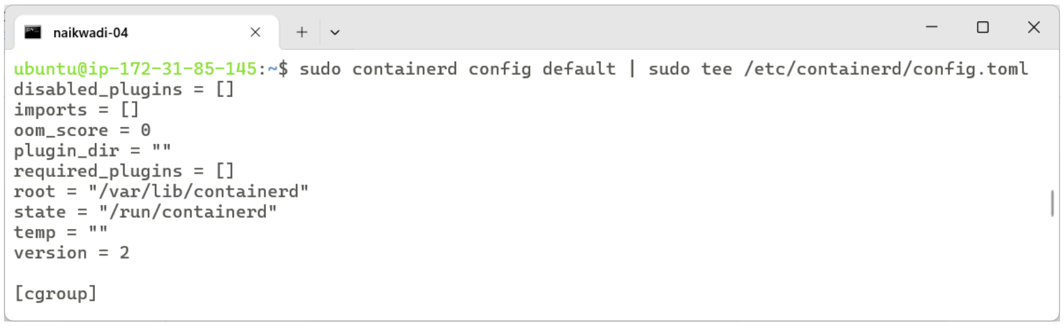
```
ubuntu@ip-172-31-85-145:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required
:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
```

sudo mkdir -p /etc/containerd



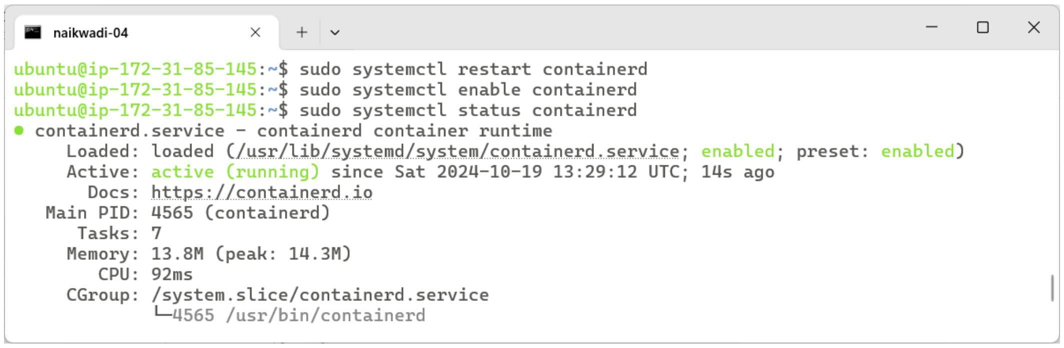
```
ubuntu@ip-172-31-85-145:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-85-145:~$ |
```

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



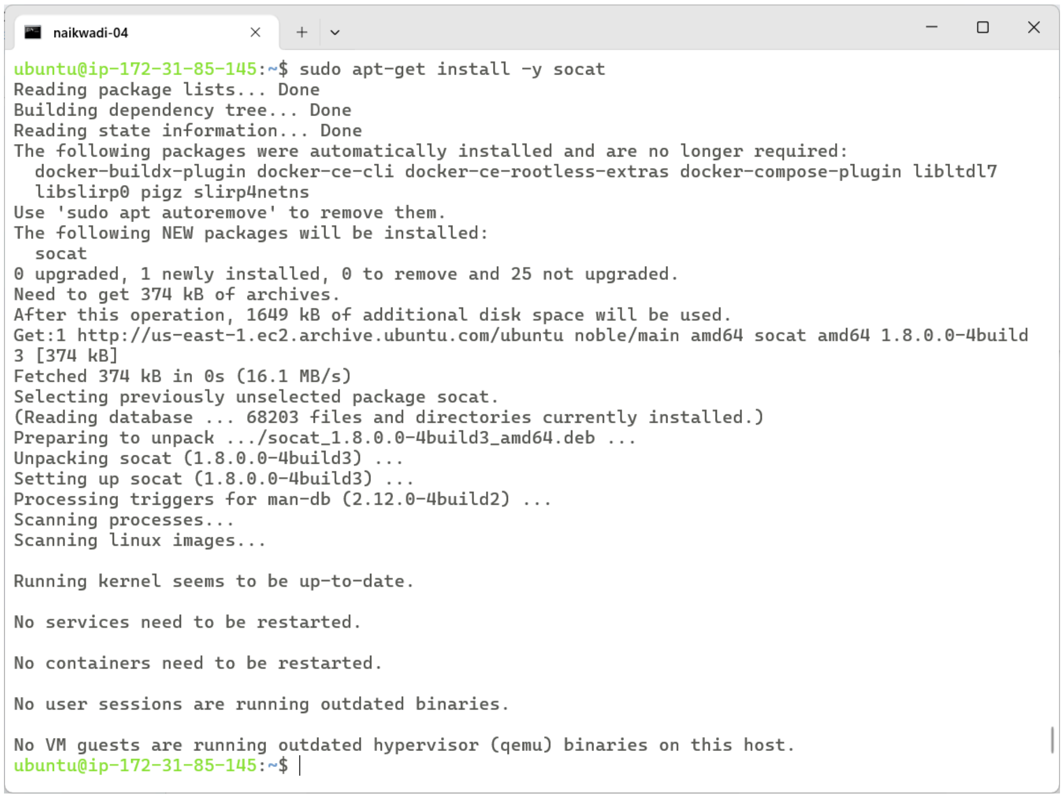
```
ubuntu@ip-172-31-85-145:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
```

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



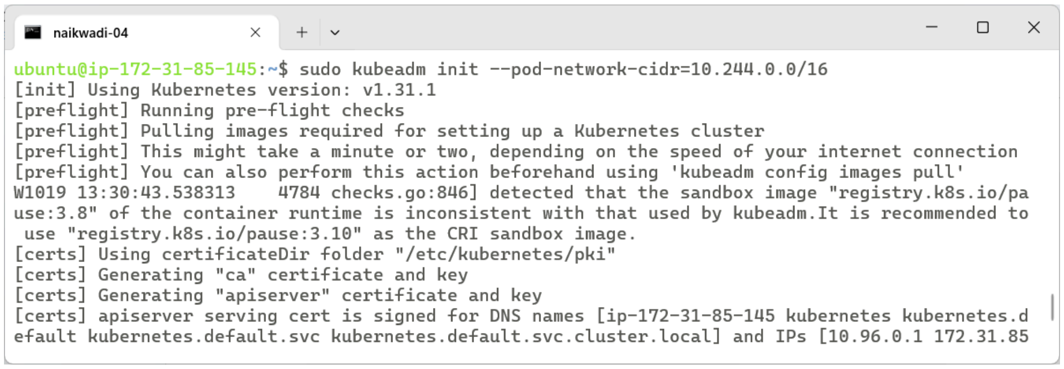
```
ubuntu@ip-172-31-85-145:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-85-145:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-85-145:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-10-19 13:29:12 UTC; 14s ago
       Docs: https://containerd.io
   Main PID: 4565 (containerd)
      Tasks: 7
     Memory: 13.8M (peak: 14.3M)
        CPU: 92ms
     CGroup: /system.slice/containerd.service
             └─4565 /usr/bin/containerd
```

sudo apt-get install -y socat



```
ubuntu@ip-172-31-85-145:~$ 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 25 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 1.8.0.0-4build
3 [374 kB]
Fetched 374 kB in 0s (16.1 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68203 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-85-145:~$ |
```

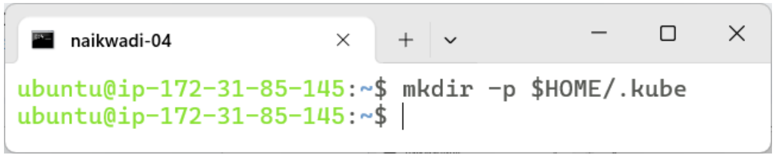
Re-run the Init Command:
sudo kubeadm init --pod-network-cidr=10.244.0.0/16



```
ubuntu@ip-172-31-85-145:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[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'
W1019 13:30:43.538313    4784 checks.go:846] detected that the sandbox image "registry.k8s.io/pa
use: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-172-31-85-145 kubernetes kubernetes.d
efault kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.85
```

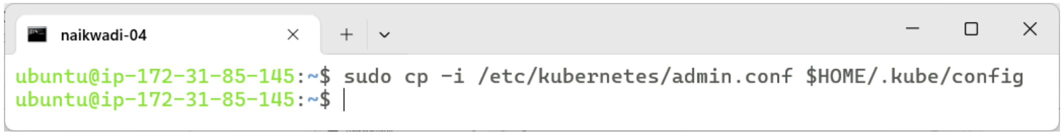
To Configure kubectl, **Set Up kubeconfig**

mkdir -p $HOME/.kube



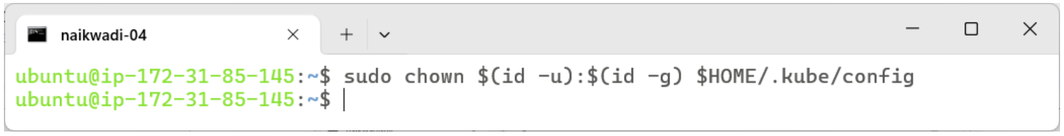
```
ubuntu@ip-172-31-85-145:~$ mkdir -p $HOME/.kube
ubuntu@ip-172-31-85-145:~$ |
```

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config



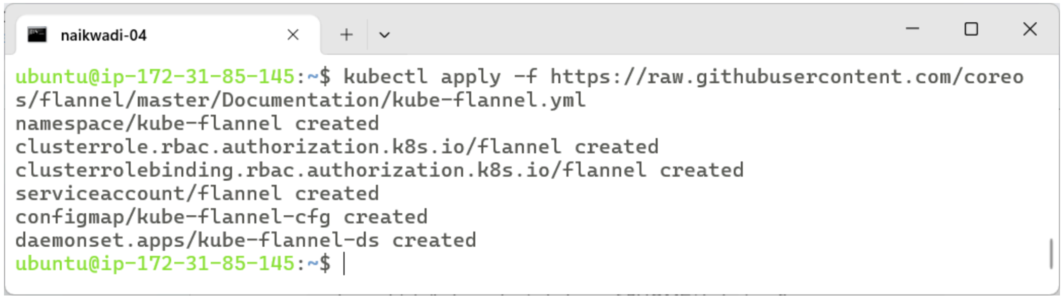
```
ubuntu@ip-172-31-85-145:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
ubuntu@ip-172-31-85-145:~$ |
```

sudo chown $(id -u):$(id -g) $HOME/.kube/config



```
ubuntu@ip-172-31-85-145:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@ip-172-31-85-145:~$ |
```

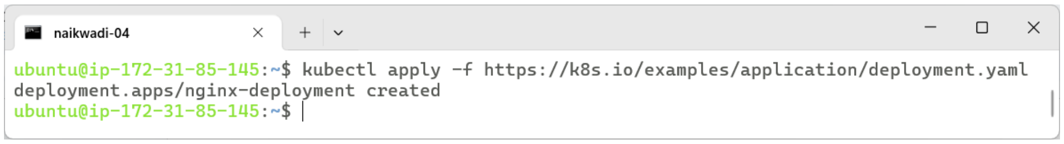
Install Flannel (a networking plugin):
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml



```
ubuntu@ip-172-31-85-145:~$ kubectl apply -f https://raw.githubusercontent.com/coreo
s/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
ubuntu@ip-172-31-85-145:~$ |
```

To Deploy Nginx Server, **Create a Deployment:**

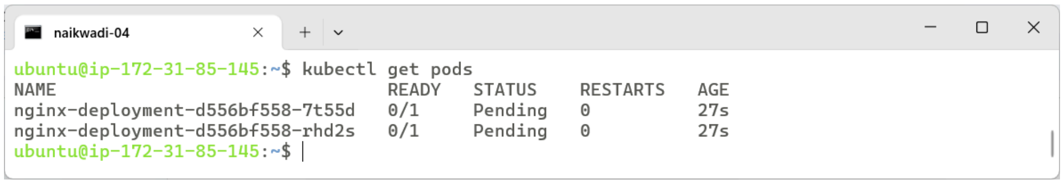
kubectl apply -f https://k8s.io/examples/application/deployment.yaml



```
ubuntu@ip-172-31-85-145:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-85-145:~$ |
```

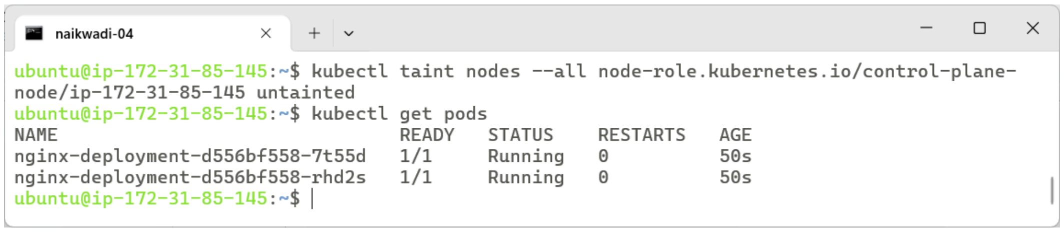
Check Pods:

kubectl get pods



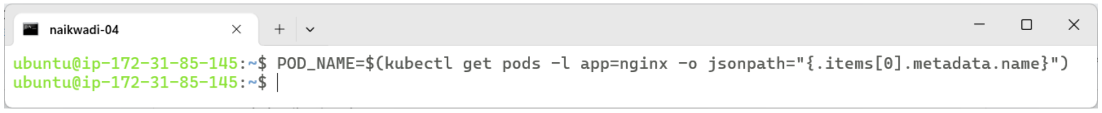
```
ubuntu@ip-172-31-85-145:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-7t55d  0/1     Pending   0          27s
nginx-deployment-d556bf558-rhd2s  0/1     Pending   0          27s
ubuntu@ip-172-31-85-145:~$
```

If the pod status is pending, you might need to remove the control-plane taint:
kubectl taint nodes --all node-role.kubernetes.io/control-plane-



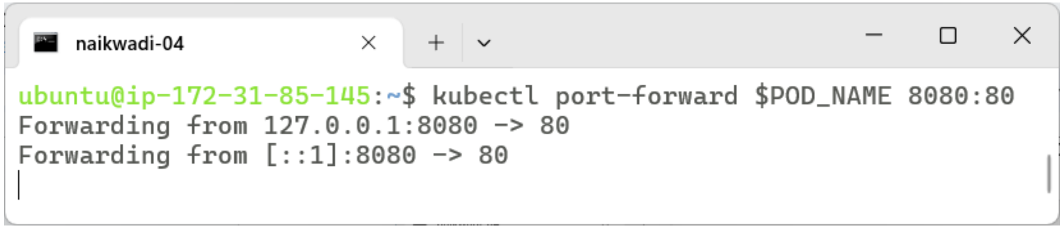
```
ubuntu@ip-172-31-85-145:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/ip-172-31-85-145 untainted
ubuntu@ip-172-31-85-145:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-7t55d  1/1     Running   0          50s
nginx-deployment-d556bf558-rhd2s  1/1     Running   0          50s
ubuntu@ip-172-31-85-145:~$
```

Port Forward to Access Nginx: Find the Pod name
POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")



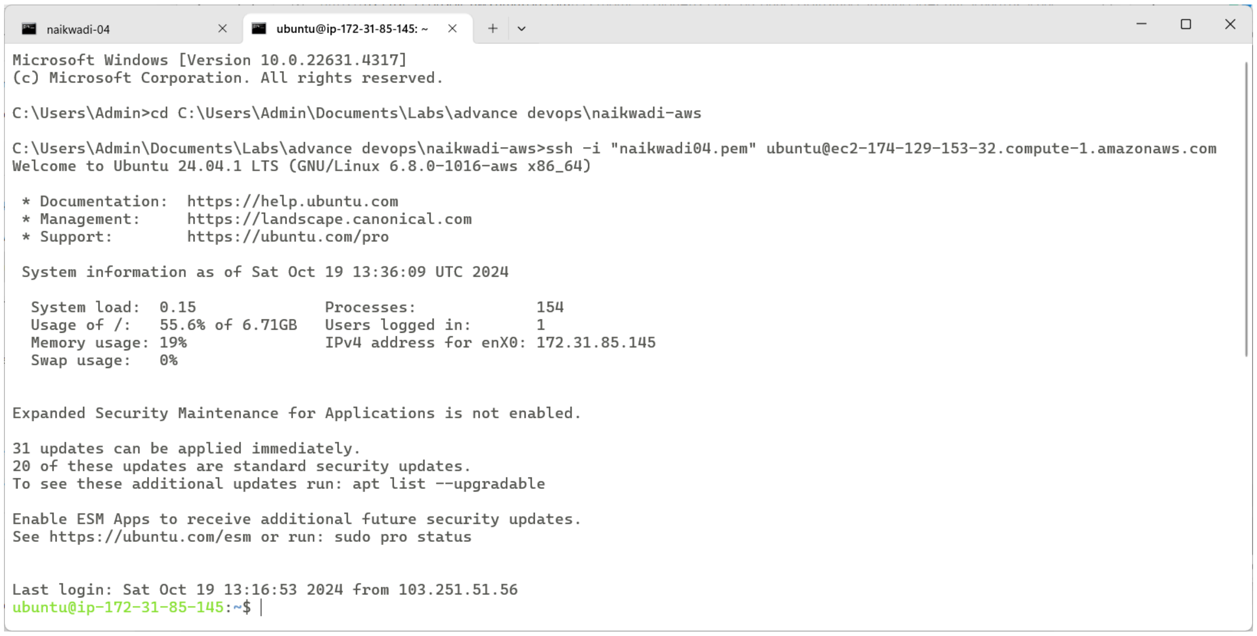
```
ubuntu@ip-172-31-85-145:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-85-145:~$
```

kubectl port-forward $POD_NAME 8080:80



```
ubuntu@ip-172-31-85-145:~$ kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

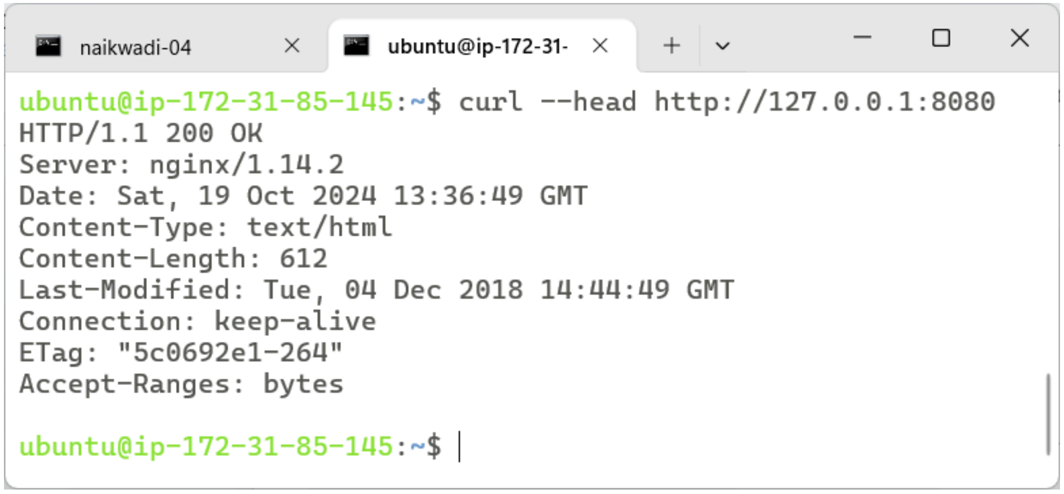
Open a New Terminal and SSH back into your EC2 instance.



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws

C:\Users\Admin\Documents\Labs\advance devops\naikwadi-aws>ssh -i "naikwadi04.pem" ubuntu@ec2-174-129-153-32.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

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

 System information as of Sat Oct 19 13:36:09 UTC 2024

  System load:  0.15              Processes:             154
  Usage of /:   55.6% of 6.71GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for enX0: 172.31.85.145
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

31 updates can be applied immediately.
20 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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


Last login: Sat Oct 19 13:16:53 2024 from 103.251.51.56
ubuntu@ip-172-31-85-145:~$
```

Use Curl to Check Nginx:

curl --head http://127.0.0.1:8080



```
ubuntu@ip-172-31-85-145:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sat, 19 Oct 2024 13:36:49 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

ubuntu@ip-172-31-85-145:~$ 
```

If you see `200 OK`, your Nginx server is successfully running.

Conclusion :

Understanding **kubectl** is crucial for anyone working with Kubernetes, as it serves as the primary interface for managing applications and resources. Through `kubectl`, users can effectively deploy, monitor, and troubleshoot applications, ensuring that they run smoothly in a Kubernetes environment.