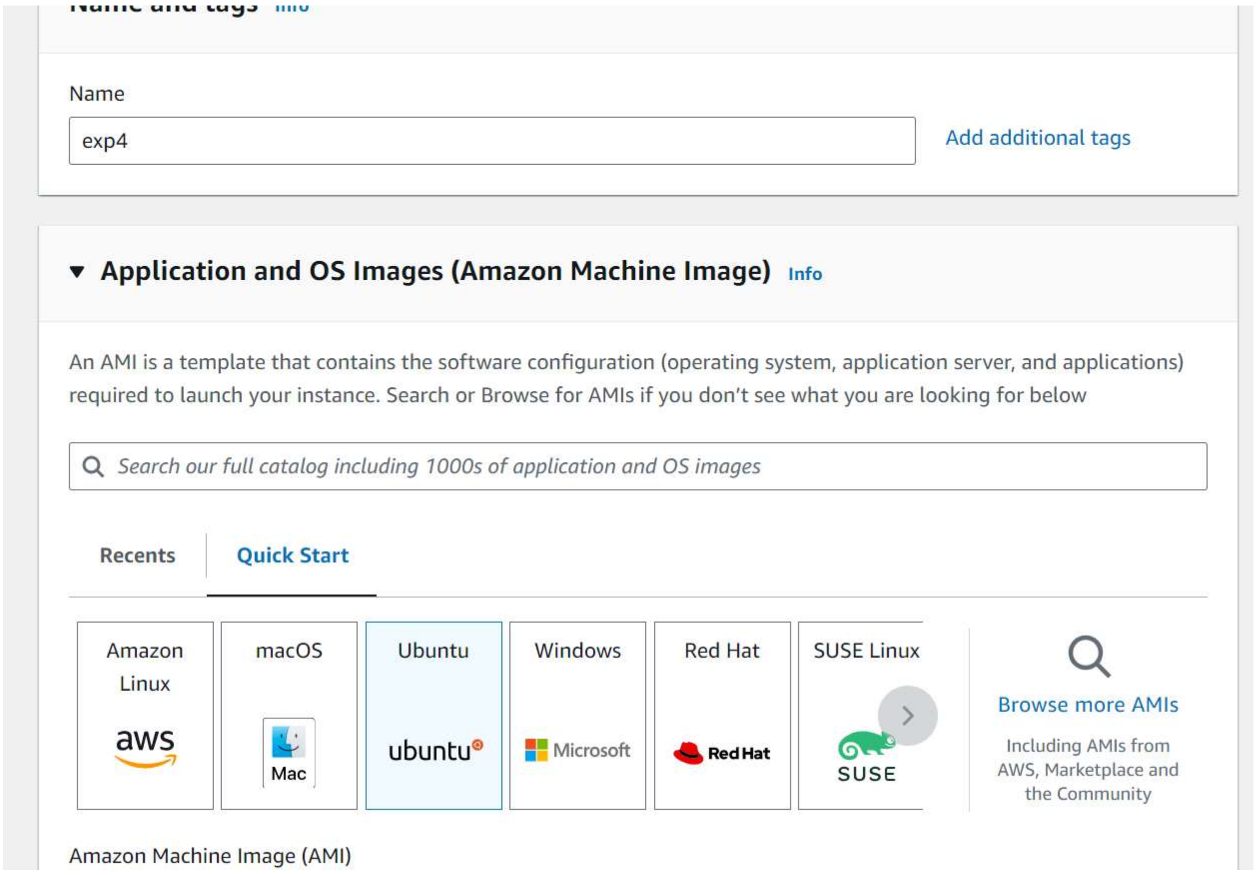
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

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.



Name and tags Info

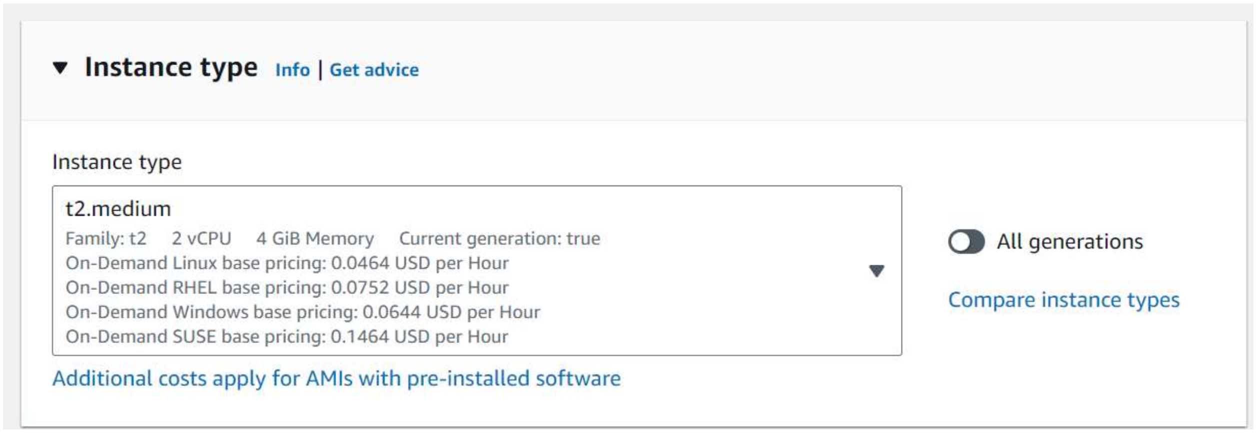
Name: exp4

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

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

Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)

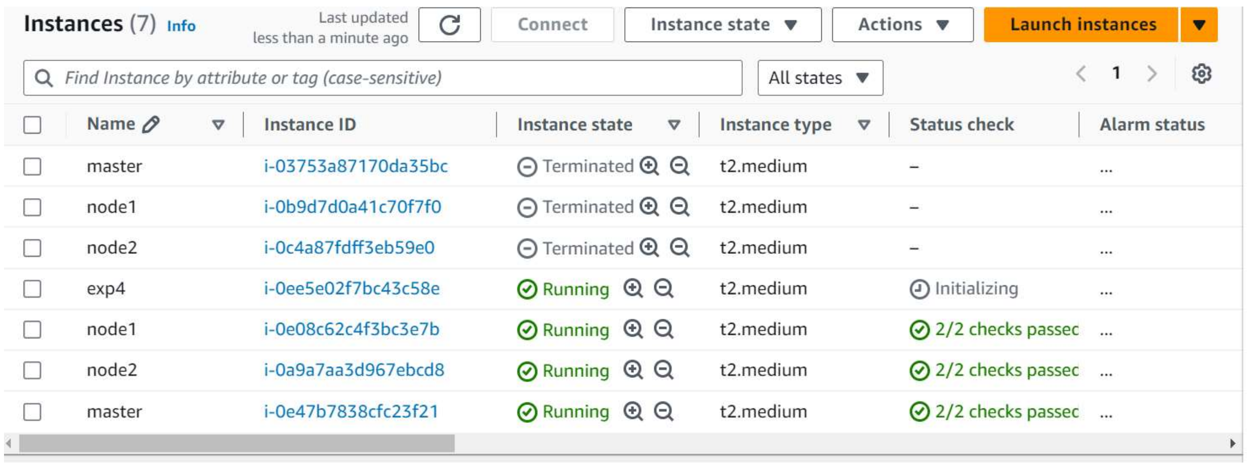


▼ Instance type Info | Get advice

Instance type

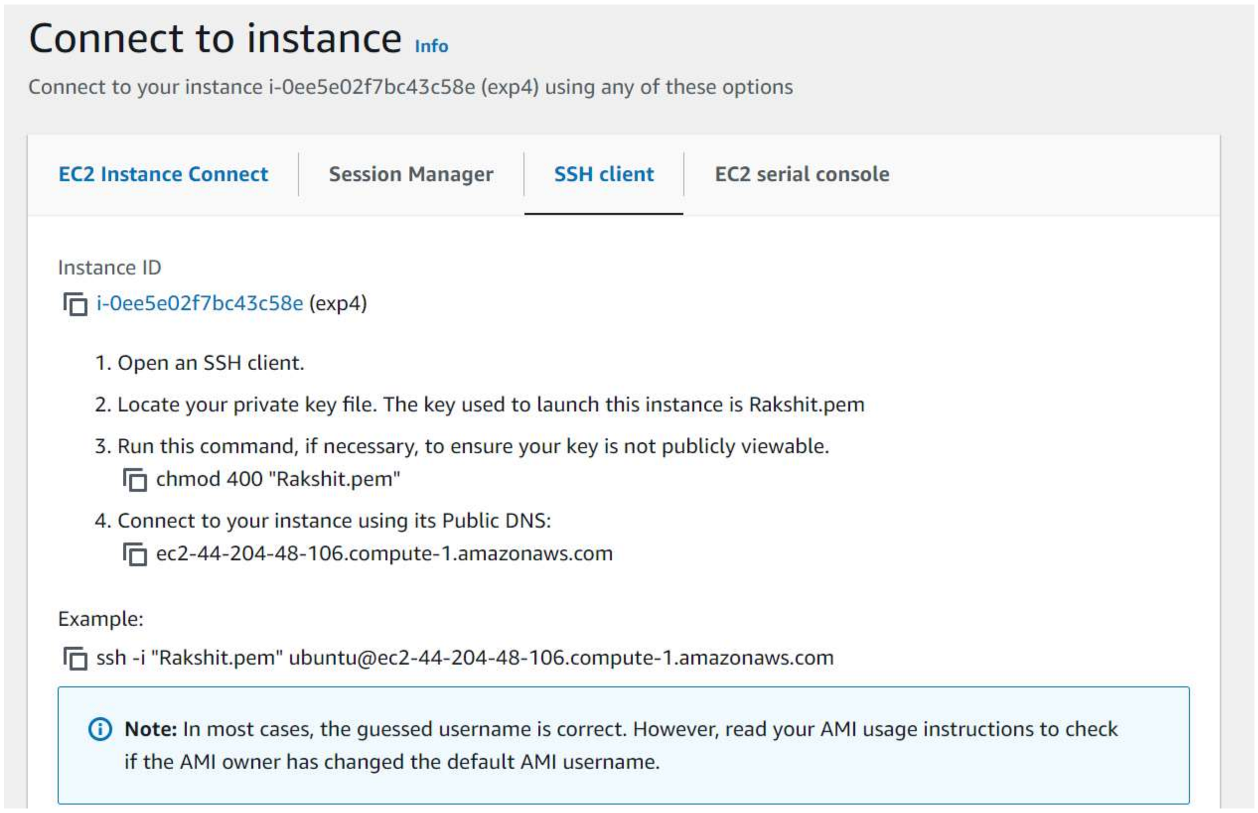
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour

Step 2: After creating the instance click on Connect the instance and navigate to SSH Client.



Instances (7) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
☐	master	i-03753a87170da35bc	Terminated	t2.medium	–	...
☐	node1	i-0b9d7d0a41c70f7f0	Terminated	t2.medium	–	...
☐	node2	i-0c4a87fdff3eb59e0	Terminated	t2.medium	–	...
☐	exp4	i-0ee5e02f7bc43c58e	Running	t2.medium	Initializing	...
☐	node1	i-0e08c62c4f3bc3e7b	Running	t2.medium	2/2 checks passec	...
☐	node2	i-0a9a7aa3d967ebcd8	Running	t2.medium	2/2 checks passec	...
☐	master	i-0e47b7838cfc23f21	Running	t2.medium	2/2 checks passec	...



Connect to instance Info

Connect to your instance i-0ee5e02f7bc43c58e (exp4) using any of these options

Instance ID

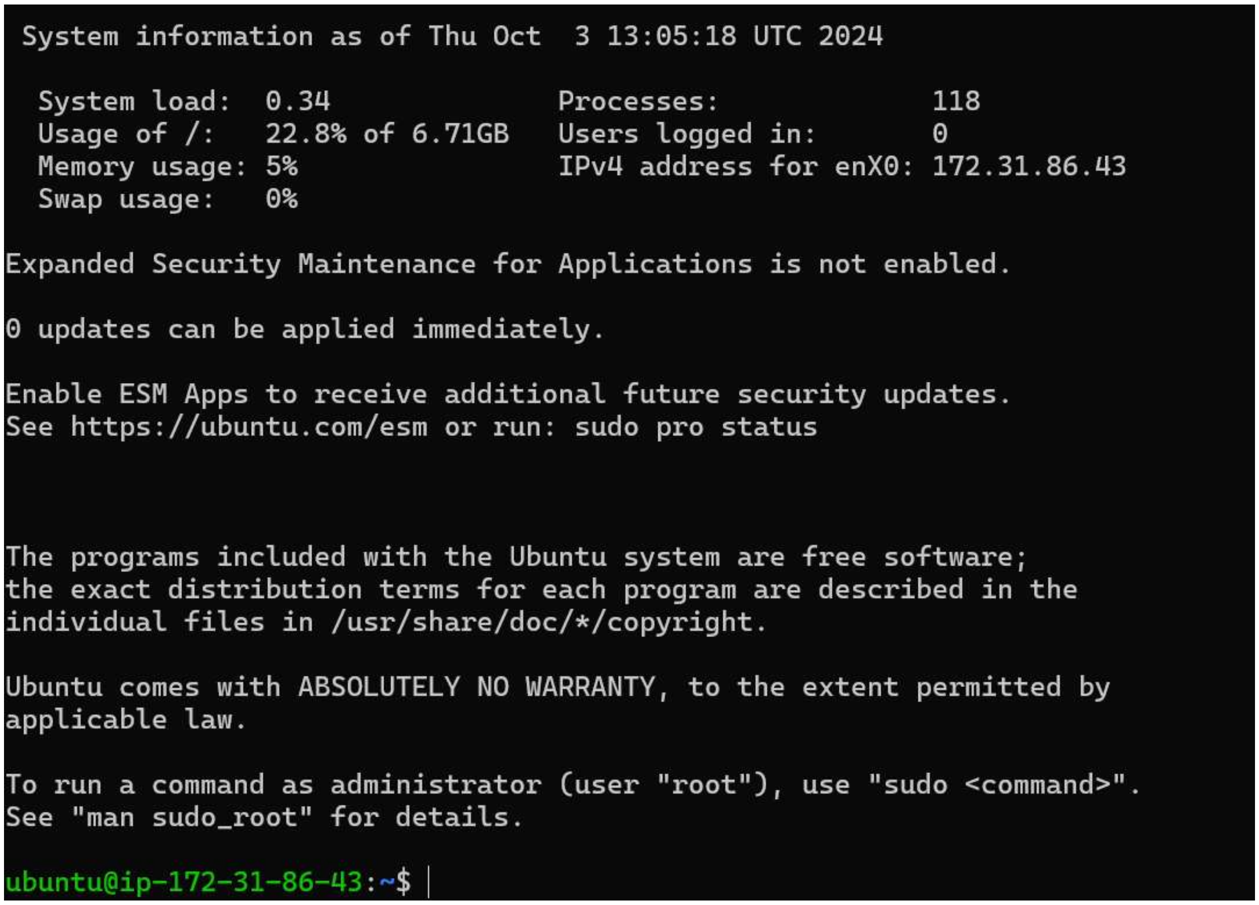
i-0ee5e02f7bc43c58e (exp4)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Rakshit.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
```
chmod 400 "Rakshit.pem"
```
4. Connect to your instance using its Public DNS:
```
ec2-44-204-48-106.compute-1.amazonaws.com
```

Example:
```
ssh -i "Rakshit.pem" ubuntu@ec2-44-204-48-106.compute-1.amazonaws.com
```

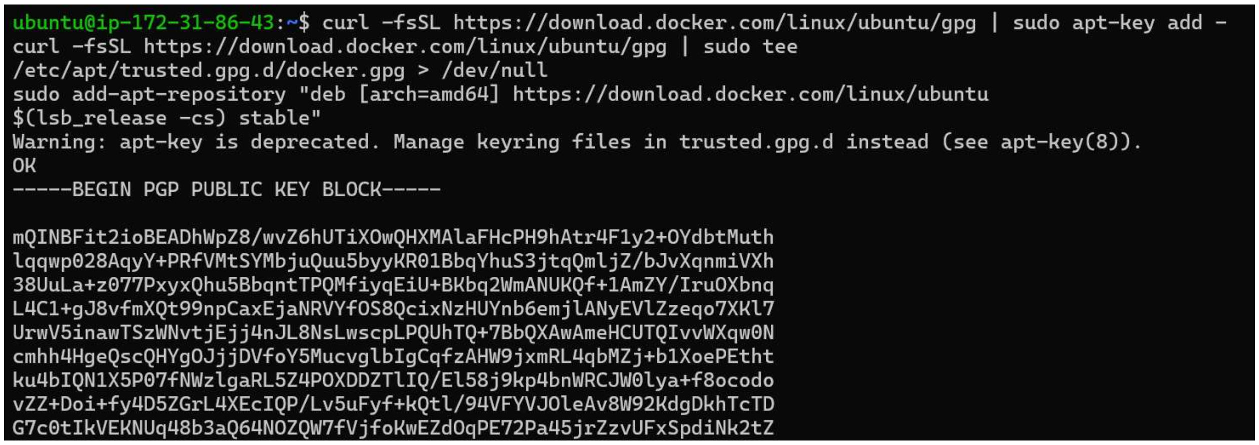
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 3: Now open the folder in the terminal 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)



```
 System information as of Thu Oct  3 13:05:18 UTC 2024

  System load:  0.34              Processes:             118
  Usage of /:   22.8% of 6.71GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for enX0: 172.31.86.43
  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 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-172-31-86-43:~$ 
```

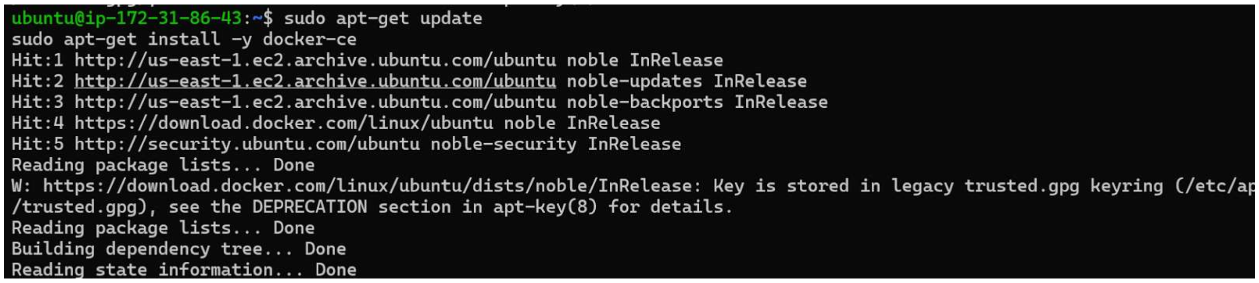
Step 4: Run the below commands to install and setup Docker.
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"



```
ubuntu@ip-172-31-86-43:~$ 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"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
```

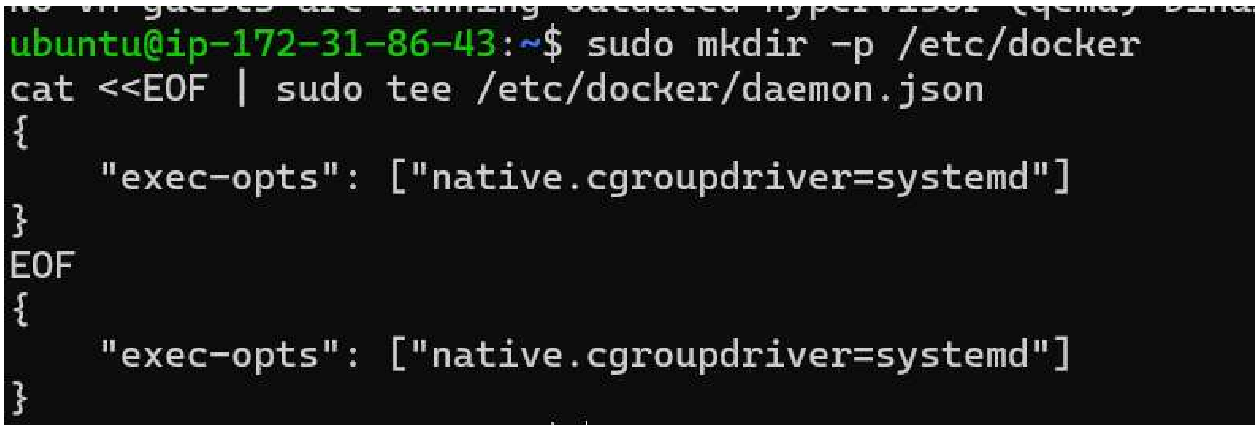
sudo apt-get update

sudo apt-get install -y docker-ce



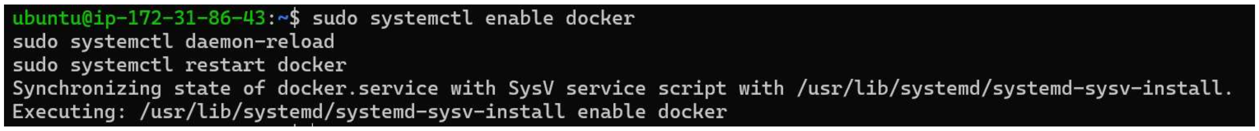
```
ubuntu@ip-172-31-86-43:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/ap
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

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



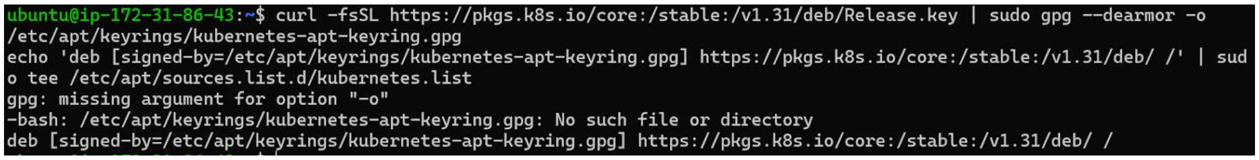
```
ubuntu@ip-172-31-86-43:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
```

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



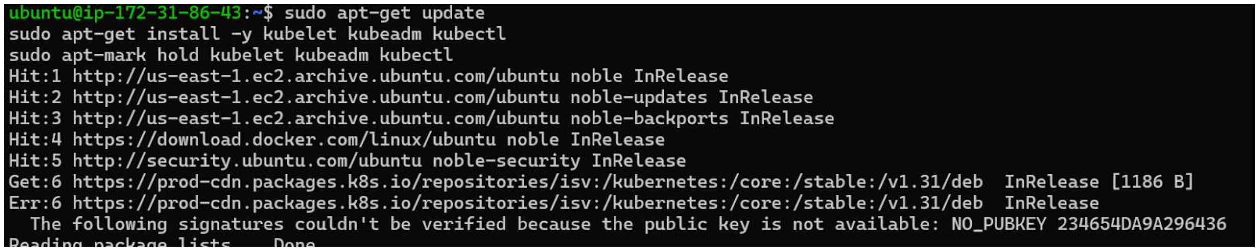
```
ubuntu@ip-172-31-86-43:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

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



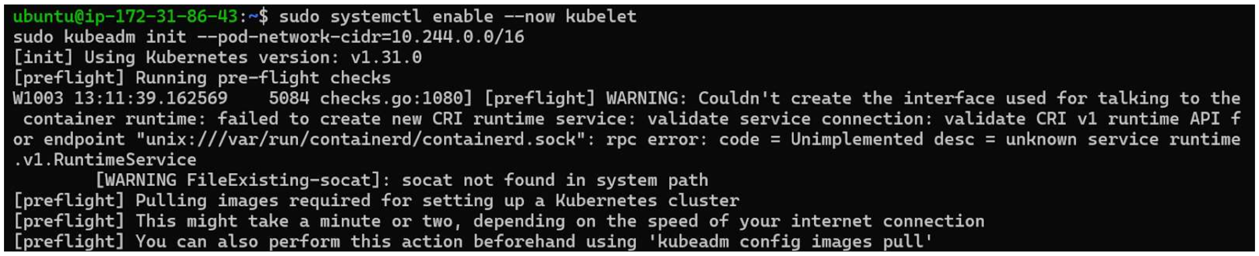
```
ubuntu@ip-172-31-86-43:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sud
o tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

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



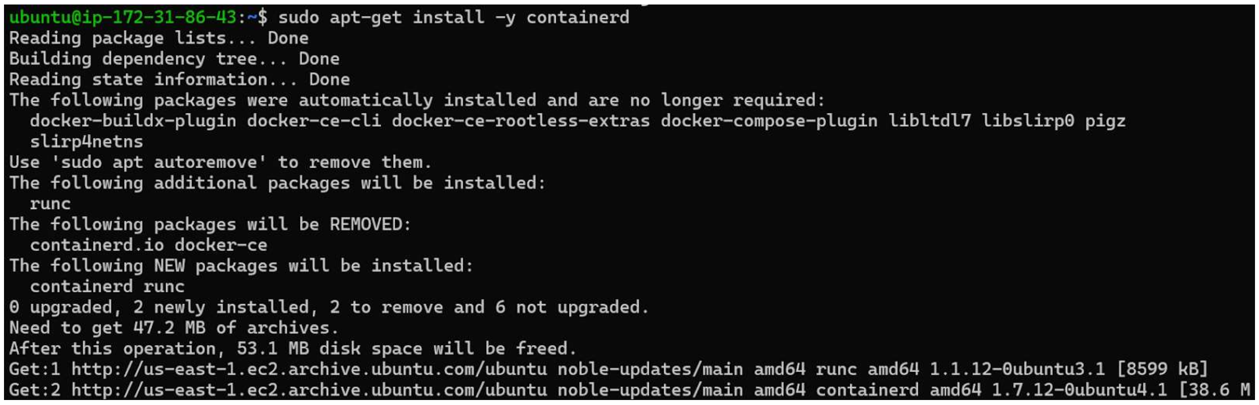
```
ubuntu@ip-172-31-86-43:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Err:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Reading package lists... Done
```

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



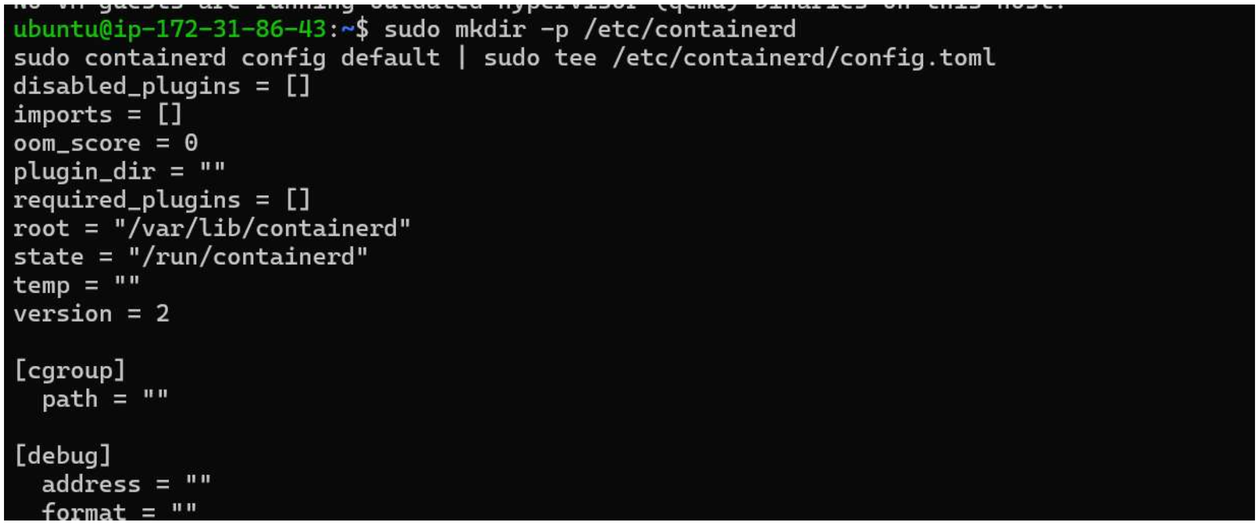
```
ubuntu@ip-172-31-86-43:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W1003 13:11:39.162569    5084 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for 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'
```

Now We have got an error. So we have to perform some additional commands as follow.
sudo apt-get install -y containerd



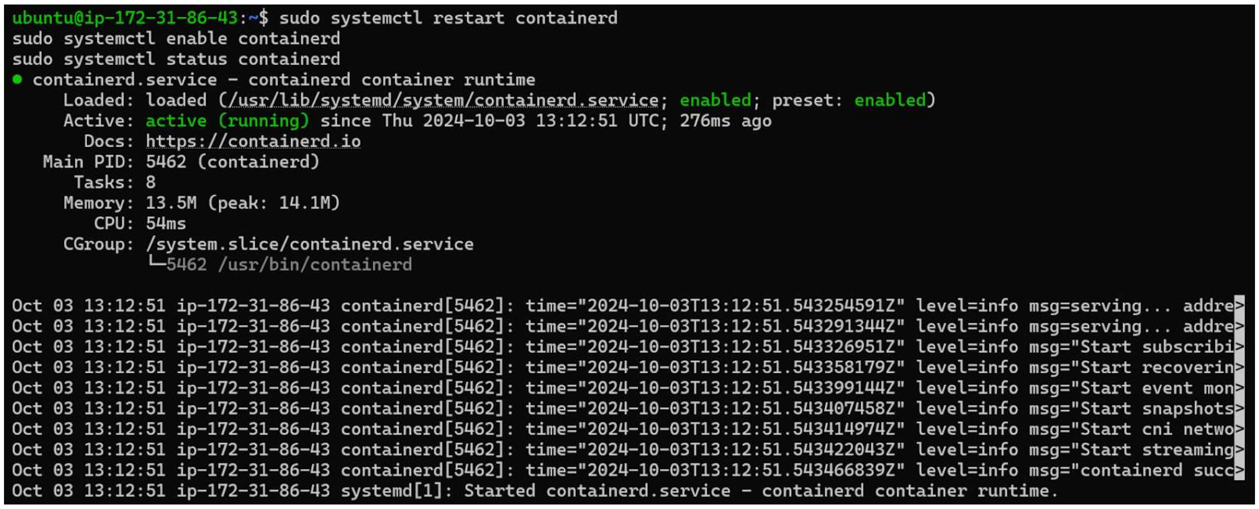
```
ubuntu@ip-172-31-86-43:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras 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 6 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 M
```

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



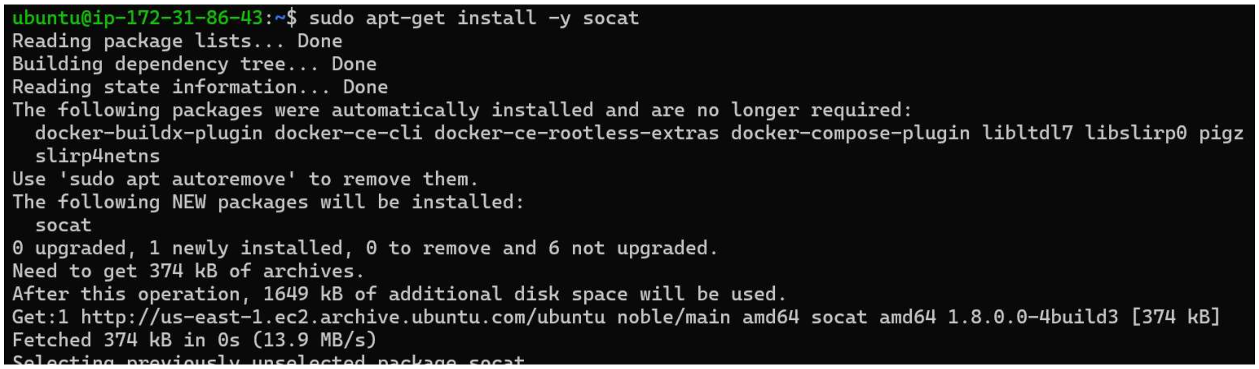
```
ubuntu@ip-172-31-86-43:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
```

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



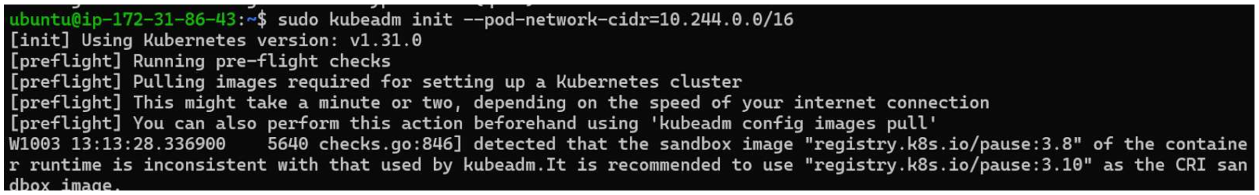
```
ubuntu@ip-172-31-86-43:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-10-03 13:12:51 UTC; 276ms ago
       Docs: https://containerd.io
   Main PID: 5462 (containerd)
      Tasks: 8
     Memory: 13.5M (peak: 14.1M)
        CPU: 54ms
     CGroup: /system.slice/containerd.service
             └─5462 /usr/bin/containerd

Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543254591Z" level=info msg=serving... addre>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543291344Z" level=info msg=serving... addre>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543326951Z" level=info msg="Start subscribi>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543358179Z" level=info msg="Start recoverin>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543399144Z" level=info msg="Start event mon>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543407458Z" level=info msg="Start snapshots>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543414974Z" level=info msg="Start cni netwo>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543422043Z" level=info msg="Start streaming>
Oct 03 13:12:51 ip-172-31-86-43 containerd[5462]: time="2024-10-03T13:12:51.543466839Z" level=info msg="containerd succ>
Oct 03 13:12:51 ip-172-31-86-43 systemd[1]: Started containerd.service - containerd container runtime.
```

sudo apt-get install -y socat



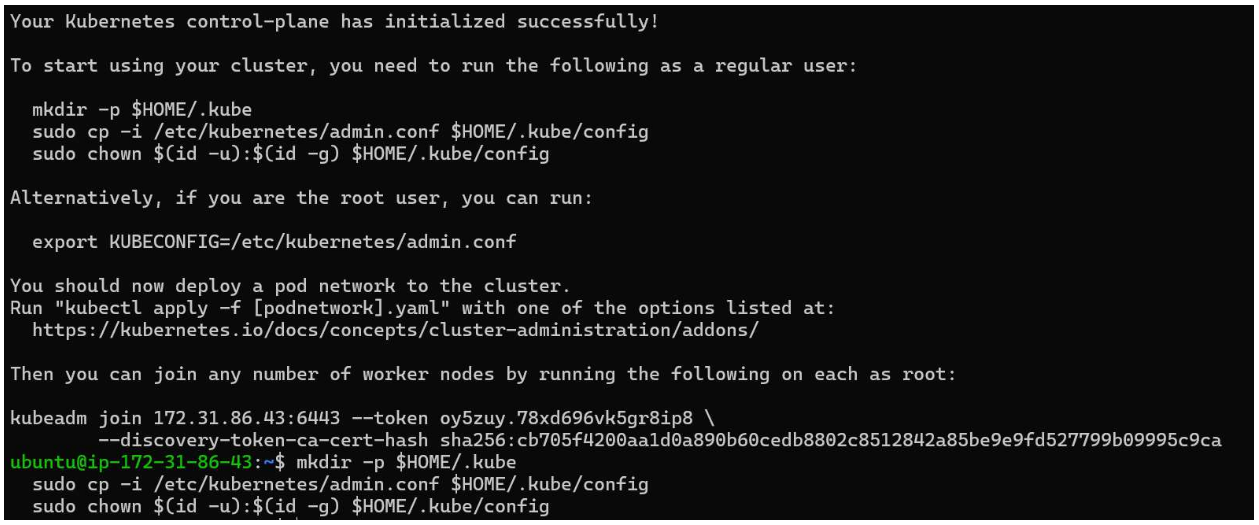
```
ubuntu@ip-172-31-86-43:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin 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 6 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 (13.9 MB/s)
```

Step 6: Initialize the Kubecluster sudo kubeadm init --pod-network-cidr=10.244.0.0/16



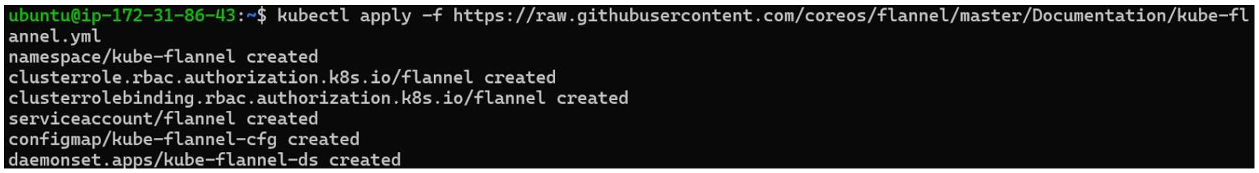
```
ubuntu@ip-172-31-86-43:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W1003 13:13:28.336900    5640 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox image.
```

Copy the mkdir and chown commands from the top and execute them. cat
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config



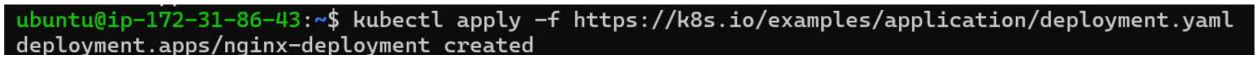
```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.86.43:6443 --token oy5zuy.78xd696vk5gr8ip8 \
        --discovery-token-ca-cert-hash sha256:cb705f4200aa1d0a890b60cedb8802c8512842a85be9e9fd527799b09995c9ca
ubuntu@ip-172-31-86-43:~$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

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



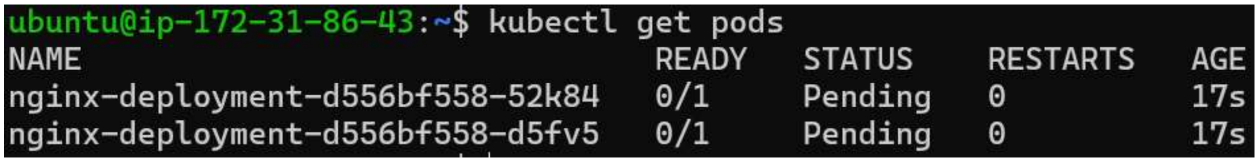
```
ubuntu@ip-172-31-86-43:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-fl
annel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Step 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



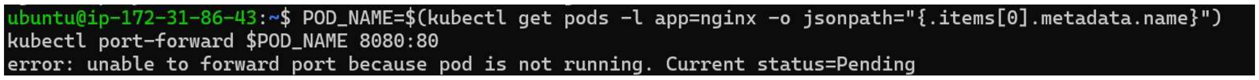
```
ubuntu@ip-172-31-86-43:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods



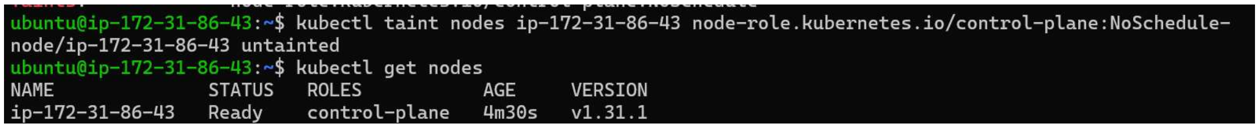
```
ubuntu@ip-172-31-86-43:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-52k84   0/1     Pending   0          17s
nginx-deployment-d556bf558-d5fv5   0/1     Pending   0          17s
```

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



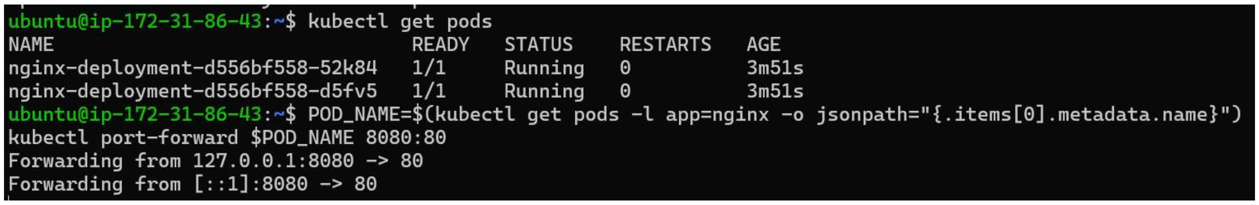
```
ubuntu@ip-172-31-86-43:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

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



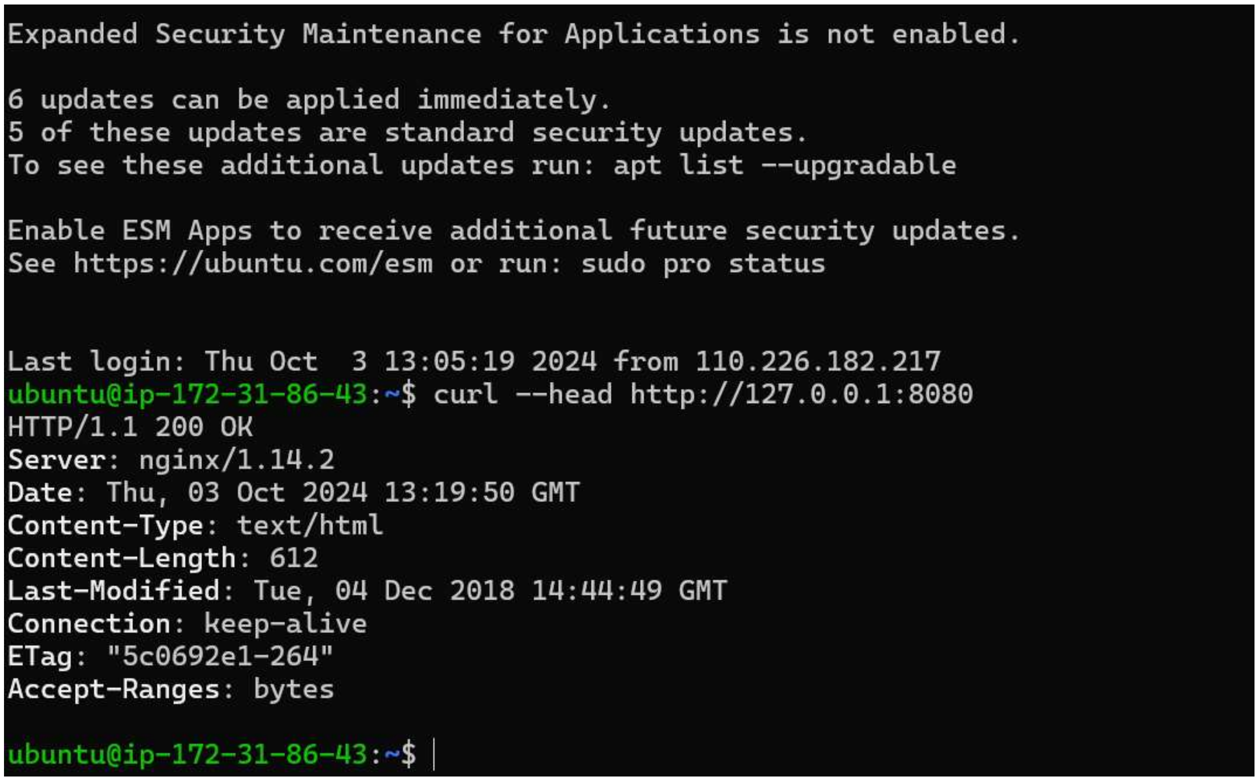
```
ubuntu@ip-172-31-86-43:~$ kubectl taint nodes ip-172-31-86-43 node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-86-43 untainted
ubuntu@ip-172-31-86-43:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-86-43   Ready    control-plane   4m30s   v1.31.1
```

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



```
ubuntu@ip-172-31-86-43:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-52k84   1/1     Running   0          3m51s
nginx-deployment-d556bf558-d5fv5   1/1     Running   0          3m51s
ubuntu@ip-172-31-86-43:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Step 8: Verify your deployment Open up a new terminal and ssh to your EC2 instance. Then, use this curl command to check if the Nginx server is running.



```
Expanded Security Maintenance for Applications is not enabled.

6 updates can be applied immediately.
5 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: Thu Oct  3 13:05:19 2024 from 110.226.182.217
ubuntu@ip-172-31-86-43:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 03 Oct 2024 13:19:50 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-172-31-86-43:~$
```

Conclusion:
In this experiment, we successfully set up a Kubernetes cluster on an AWS EC2 instance using kubectl and deployed a sample nginx application. By installing Docker and Kubernetes, configuring containerd, and initializing the cluster, we ensured smooth cluster operations. After deploying the nginx server, we verified its successful deployment by using kubectl port-forward and confirming access via curl. This process provided practical experience in managing Kubernetes clusters and deploying applications on cloud infrastructure.