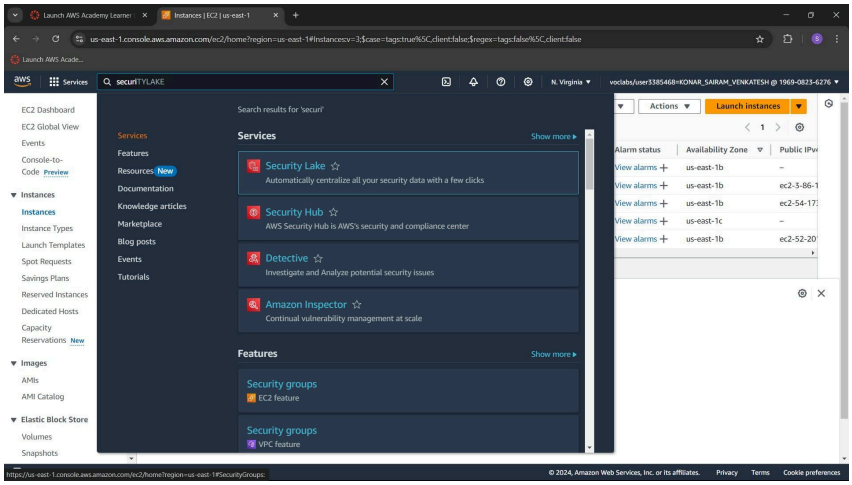
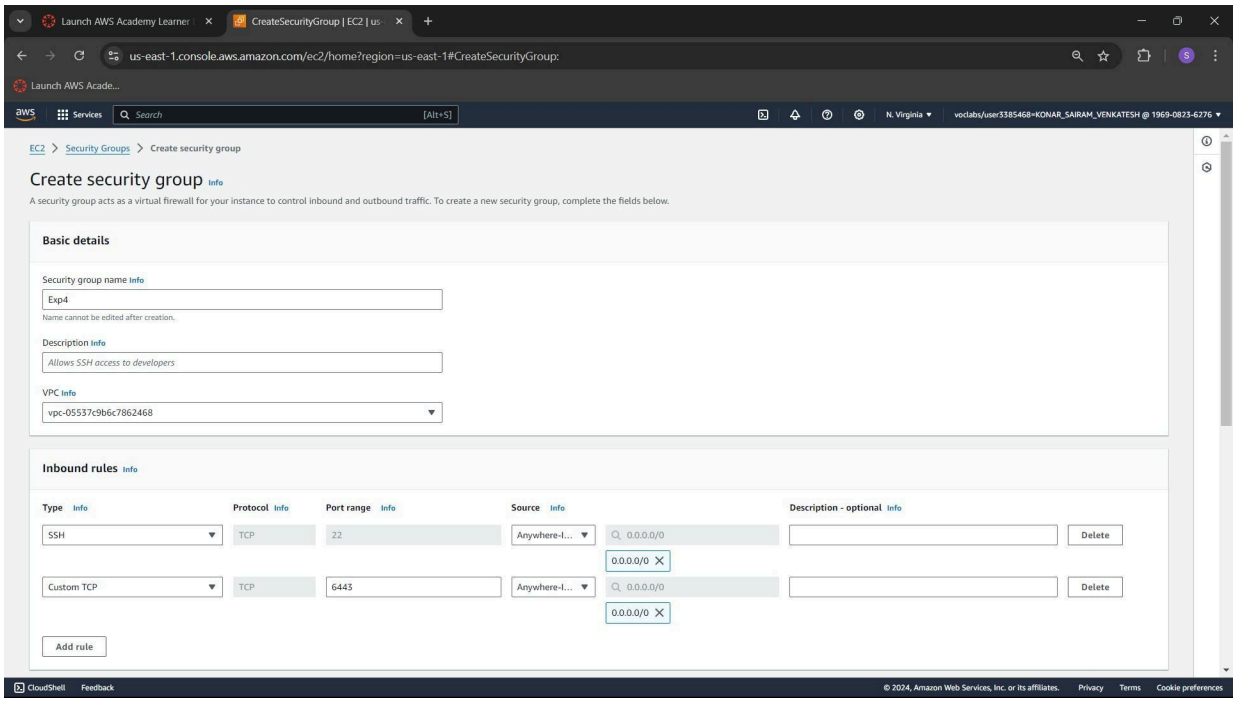
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Prerequisites:

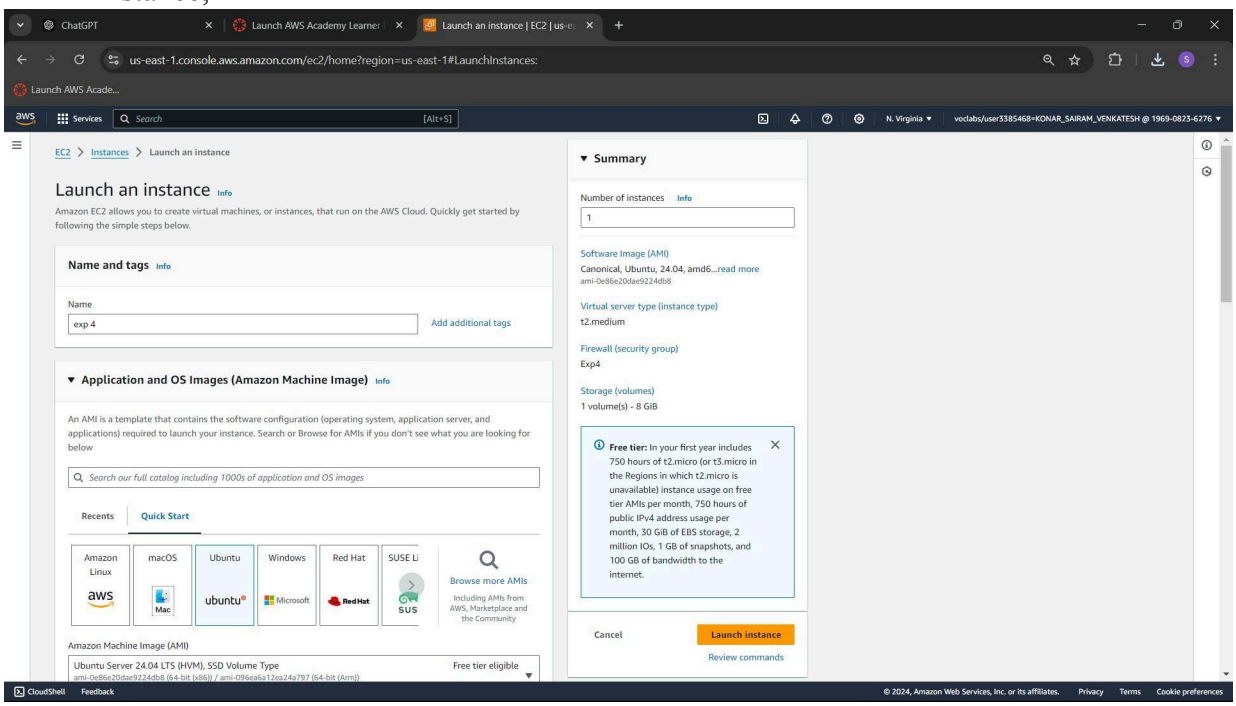
Create a security group by navigating to Search → Security Groups.



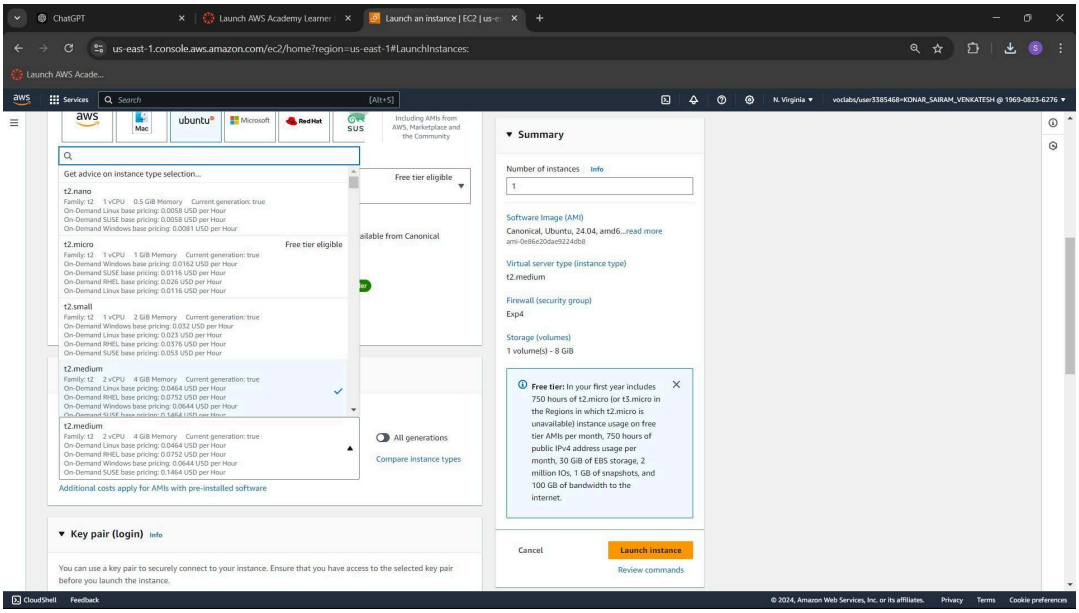


Step 1: Creation of instances.

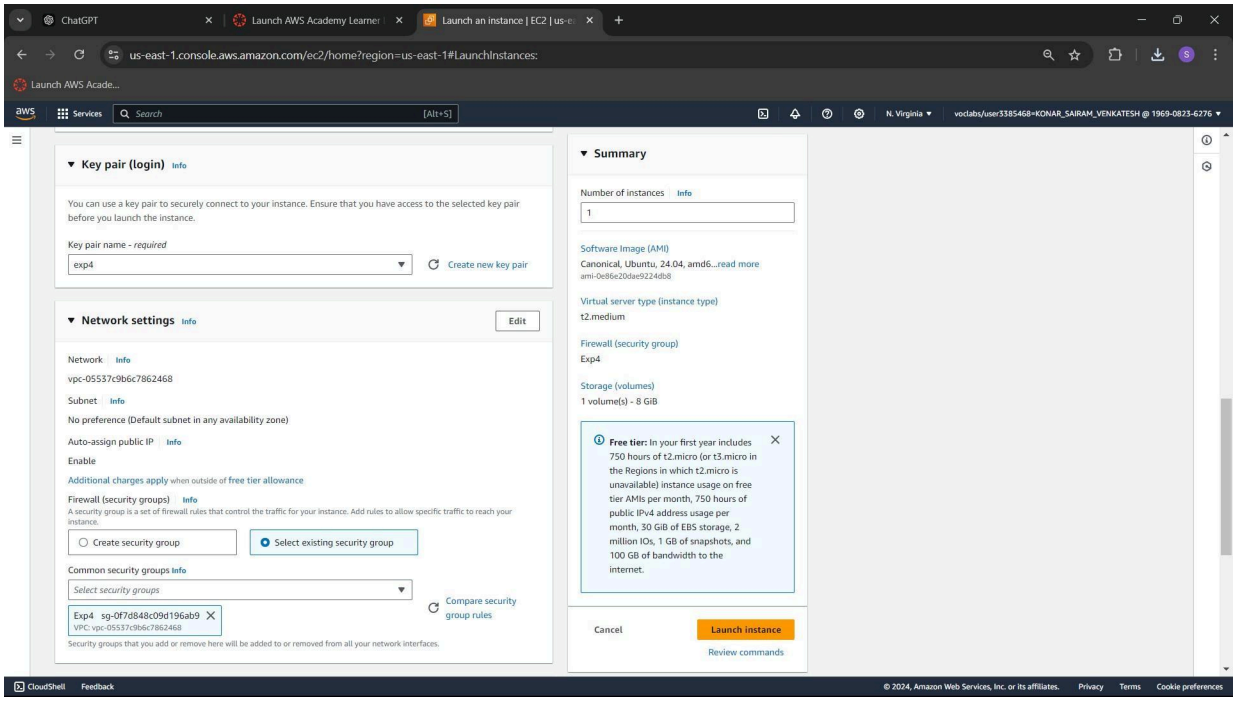
1) Go to the AWS Dashboard. In Services, search for EC2. Open it. Click on Launch instance,



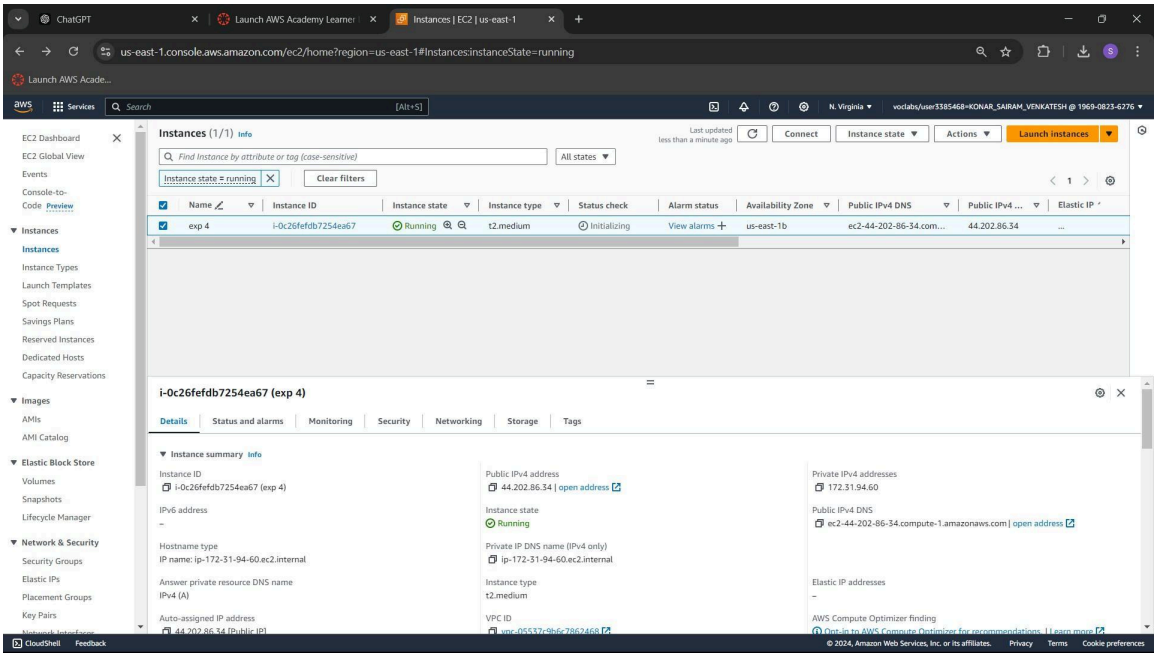
2) Select ubuntu 22.04 as you OS image.
IMPORTANT: The default instance type and free one provided by AWS is t2.micro, which provides only 1CPU and 1 GiB of memory. For running Kubernetes, a minimum of 2 CPUs and 2GiB of RAM is required, hence change **t2.micro** to **t2.medium.**



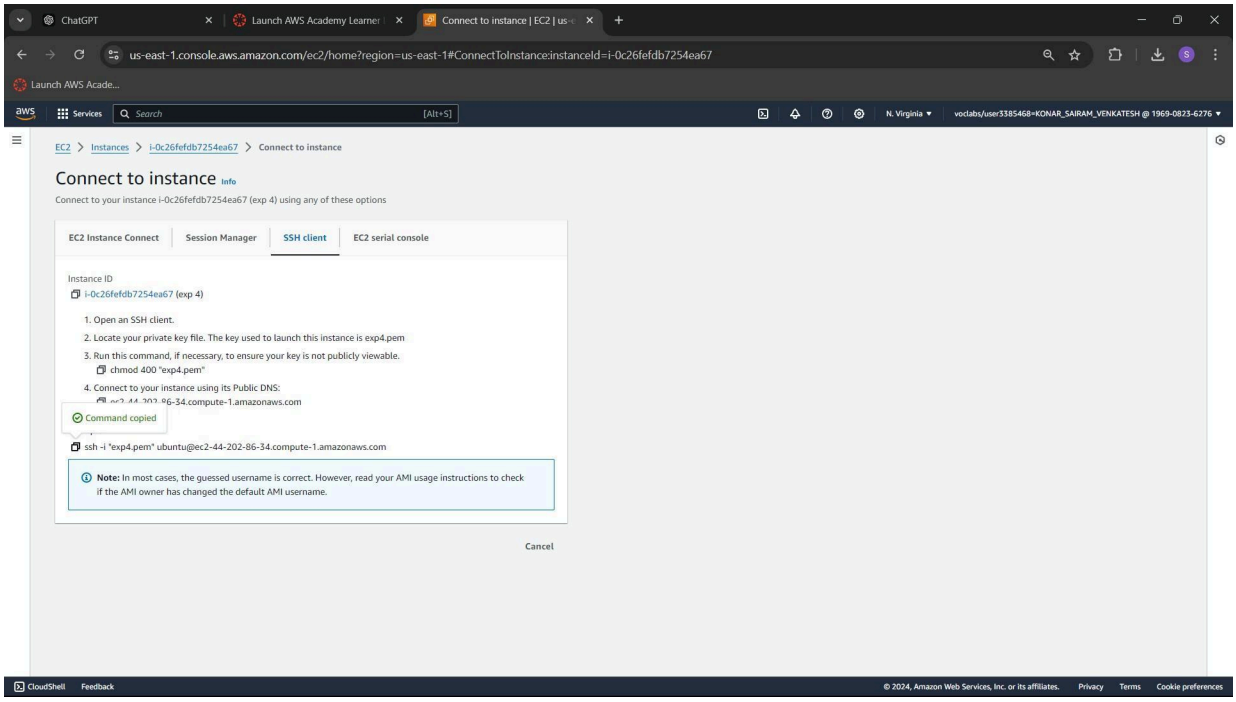
3) Create a key pair as you need the .pem file (private key file) on your system.
4) Click on 'Select existing security group' and select the exp4 group for the instance.



5) The instance is created. Click on the instance id, then click on connect.

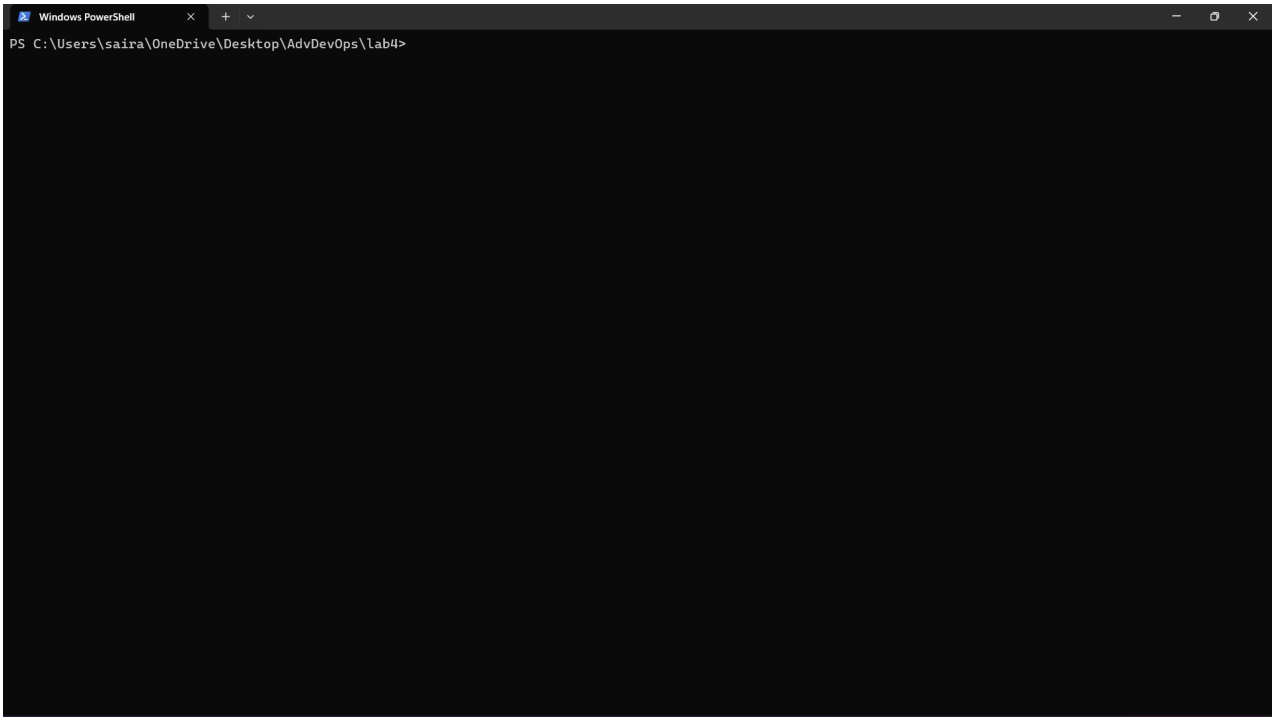


6) Click on Connect. This directs you to a connect dashboard. Click on SSH client, you get a SSH command. Use this command to access the instance terminal on your local



system.

7) Go to the folder where your private key file (.pem file) is installed. Right click → Open in terminal.



Paste the SSH command here and run it.

You might get the error of UNPROTECTED KEY FILE. This is because the .pem access Is with all users of teh system. Run the following commands to change the access to only the current user.

- icacls "C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab 3 4\tester.pem" /inheritance:r
- icacls "C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab 3 4\tester.pem" /grant:r "%USERNAME%:F"

Now rerun the SSH command.

```
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab4> ssh -i "exp4.pem" ubuntu@ec2-44-203-88-254.compute-1.amazonaws.com
The authenticity of host 'ec2-44-203-88-254.compute-1.amazonaws.com (44.203.88.254)' can't be established.
ED25519 key fingerprint is SHA256:4a0YwFPIjO+zy4hJHl4dpR++zJGgOkyYRFICt3/gbBI.
This host key is known by the following other names/addresses:
    C:\Users\saira/.ssh/known_hosts:47: ec2-44-202-86-34.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-203-88-254.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 Mon Sep 30 05:19:42 UTC 2024

  System load:  1.08              Processes:             137
  Usage of /:   55.5% of 6.71GB   Users logged in:       0
  Memory usage: 13%               IPv4 address for enX0: 172.31.94.60
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

147 updates can be applied immediately.
41 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: Mon Sep 30 04:42:33 2024 from 103.197.221.172
ubuntu@ip-172-31-94-60:~$
```

Step 2: Setup Docker

1) We have to install and setup Docker. Run these commands

- 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-172-31-94-60:~$ 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
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-94-60:~$ 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
W: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
ubuntu@ip-172-31-94-60:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/dock
er.gpg > /dev/null
ubuntu@ip-172-31-94-60:~$
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:45 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:46 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:47 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
Get:48 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [10.9 kB]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [2808 B]
Get:50 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [344 B]
Fetched 29.1 MB in 4s (7836 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/dock
er.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports 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: The key(s) in the keyring /etc/apt/trusted.gpg.d/dock
er.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
```

```
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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 3: Set up 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-172-31-94-60:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /et
c/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-94-60:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/s
table:/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-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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
```

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

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-94-60:~$
```

Step 4: Initialise the kubecluster

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

```
ubuntu@ip-172-31-94-60:~$ 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
W0930 04:29:33.758696    4337 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
```

Here, we encounter an error as a few of the dependencies for running the command are not installed. So, run the following commands

- sudo apt-get install -y containerd

```
ubuntu@ip-172-31-94-60:~$ 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 143 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
```

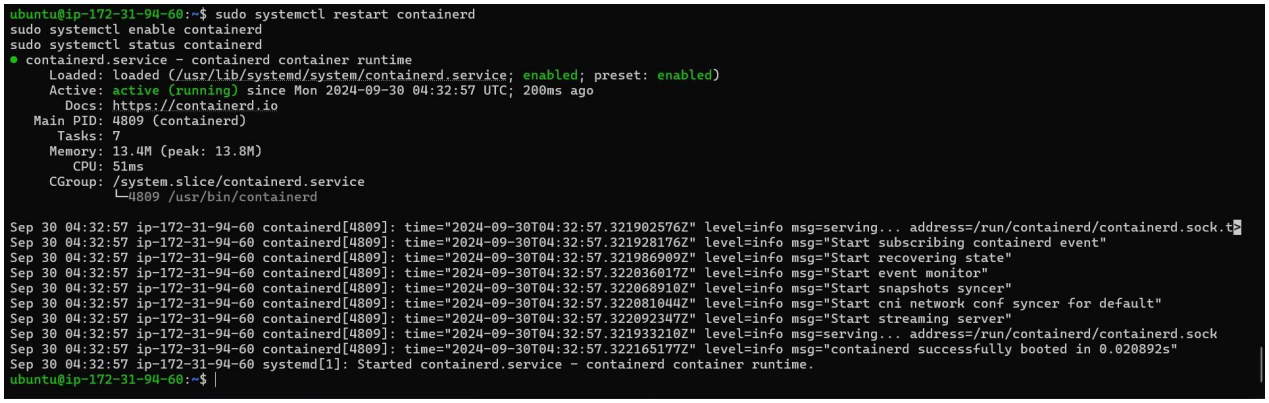
```
Running kernel seems to be up-to-date.

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-172-31-94-60:~$
```

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

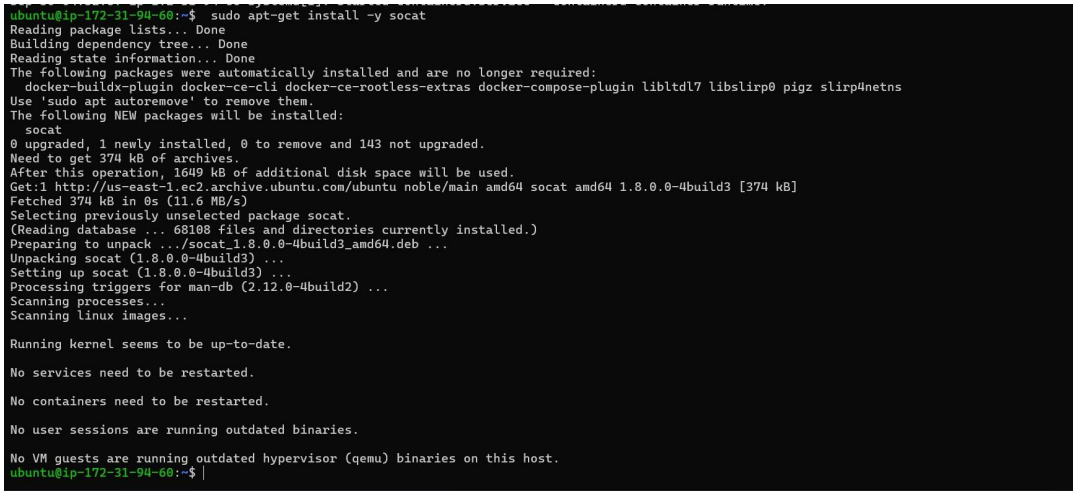
```
ubuntu@ip-172-31-94-60:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

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

```
[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-172-31-94-60:~$
```

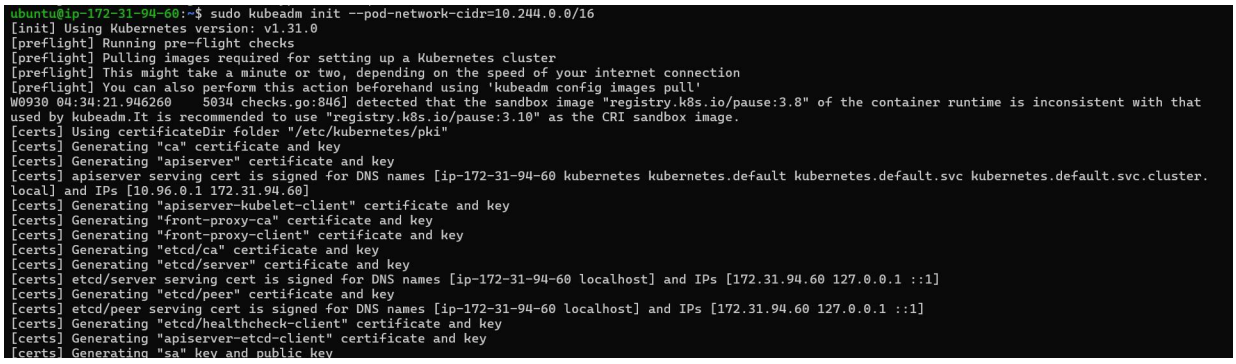
- sudo systemctl restart containerd
- sudo systemctl enable containerd
- sudo systemctl status containerd



- sudo apt-get install -y socat



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



```
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 172.31.94.60:6443 --token bhzgoy.c4u84a2hc4bf7u19 \
        --discovery-token-ca-cert-hash sha256:9d14667afdeed67b83ac64ae48deb6b315e62070260dd78c7d18b1db4c94b195
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ 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
ubuntu@ip-172-31-94-60:~$
```

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

```
ubuntu@ip-172-31-94-60:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

Step 5: Connect Ngix server to pod.

- kubectl get pods

```
ubuntu@ip-172-31-94-60:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4cbnz    0/1     Pending   0          28s
nginx-deployment-d556bf558-zk989    0/1     Pending   0          28s
ubuntu@ip-172-31-94-60:~$
```

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

- kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-94-60:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-94-60:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-94-60:~$
```

Here, another error is generated as the pods are not ready. To make them ready, we need to untaint them. Run the following commands.

- kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-

- kubectl get nodes

```
ubuntu@ip-172-31-94-60:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-94-60 untainted
ubuntu@ip-172-31-94-60:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-94-60   Ready    control-plane   5m36s   v1.31.1
ubuntu@ip-172-31-94-60:~$
```

- kubectl get pods

```
ubuntu@ip-172-31-94-60:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-4cbnz    1/1     Running   0          4m36s
nginx-deployment-d556bf558-zk989    1/1     Running   0          4m36s
ubuntu@ip-172-31-94-60:~$
```

Now rerun the following command.

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

```
ubuntu@ip-172-31-94-60:~$ 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
```

Open another terminal and connect ssh to that terminal as well.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\saira> cd '.\OneDrive\Desktop\AdvDevOps\lab4'
PS C:\Users\saira\OneDrive\Desktop\AdvDevOps\lab4> ssh -i "exp4.pem" ubuntu@ec2-44-202-86-34.compute-1.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

 System information as of Mon Sep 30 04:42:33 UTC 2024

  System load:  0.12              Processes:             153
  Usage of /:   55.3% of 6.71GB   Users logged in:       1
  Memory usage: 19%               IPv4 address for enX0: 172.31.94.60
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

147 updates can be applied immediately.
41 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: Mon Sep 30 04:22:55 2024 from 103.197.221.172
ubuntu@ip-172-31-94-60:~$
```

Run the command curl --head http://127.0.0.1:8080

```
ubuntu@ip-172-31-94-60:~$  curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Mon, 30 Sep 2024 04:43:48 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
```

If it shows status code of 200 OK, it means that the deployment of nginx server was successful.

Go back to the previous terminal. After running curl, the output for port forward looks like this

```
ubuntu@ip-172-31-94-60:~$ 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
Handling connection for 8080
```

Conclusion:

In this experiment, we have successfully installed kubectl and deployed our first Kubernetes Application. After setting up the instance, we connect it to our local terminal using SSH. The application we deployed was an nginx server. To deploy this, we had to install flannel, a common networking plugin. There were a few errors. One being while initializing the cluster, which was solved by installing the missing dependencies. The second one was when we checked the pods that were created. These pods were in the not ready state as they had some issues. To solve this, we had to untaint them. Once all the errors were resolved, we run the port command to initiate the connection. When we run the curl command, we get a OK status.
Hence the deployment was complete.