Installation of Minikube and Creation of Pods and Deployment

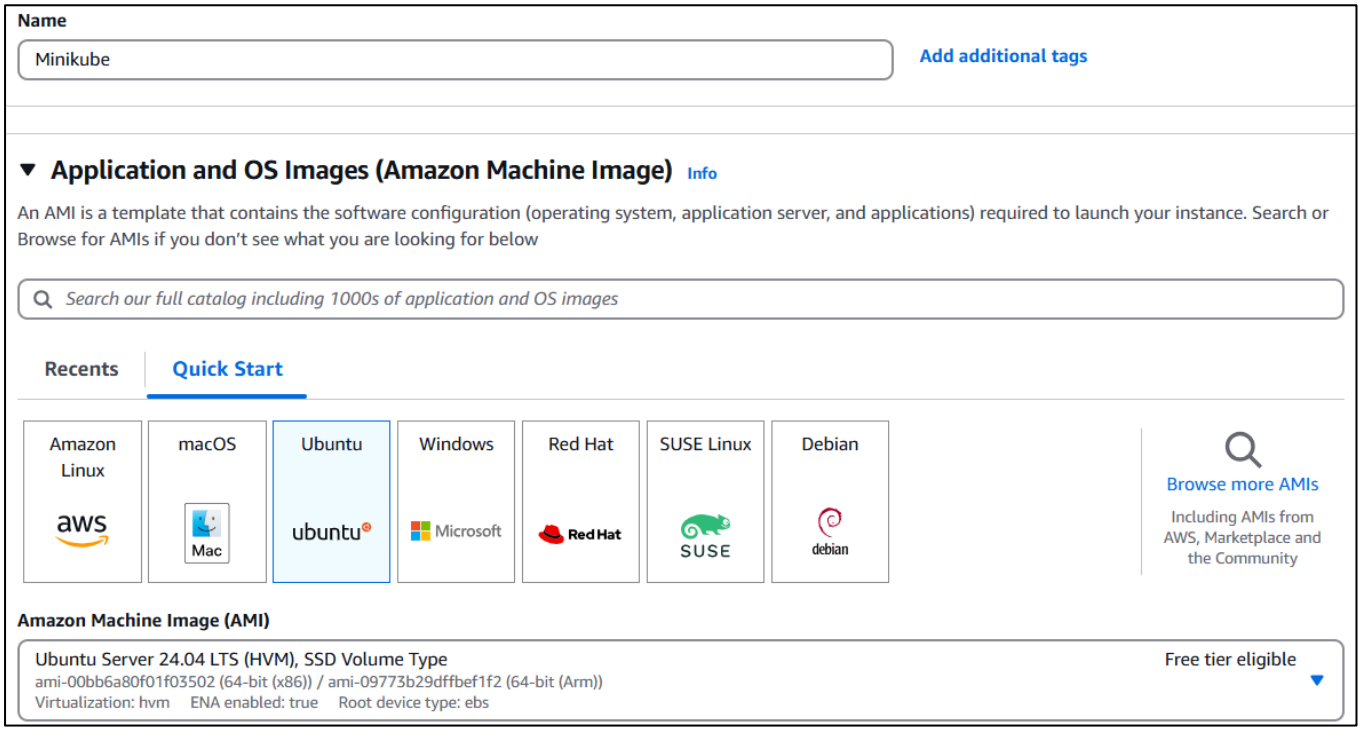
In this Project we will be Installing Minikube and Creating Pods and Deploying Pods

- Log In to your AWS Console.

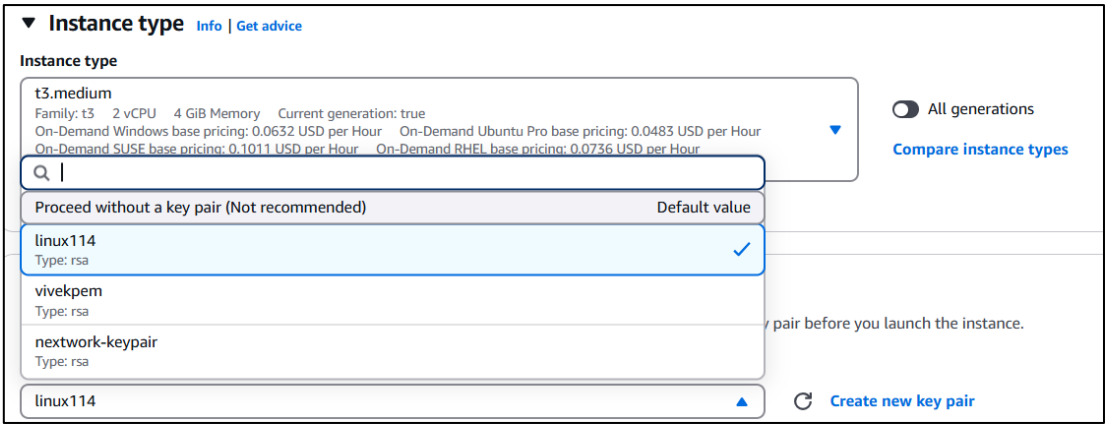
In this step, you're going to:

1. Launch an EC2 instance.
2. Set up Docker inside the EC2 and install Minikube.
3. Set up pods and carry out deployment in Minikube with yaml files.

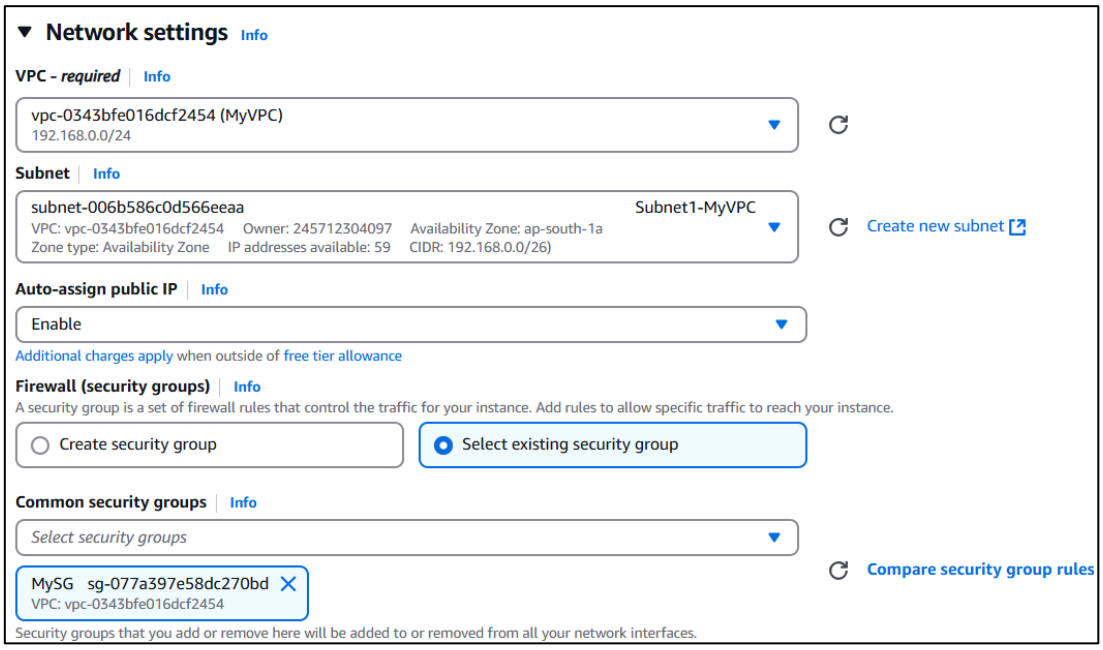
- Launch an EC2 Instance
    - **Enter the Name of the Instance**, eg: **Minikube**
    - Choose **Ubuntu Server 24.04 LTS (HVM)** under **Amazon Machine Image(AMI)**



    - Choose **t3.medium** under **Instance type**.
    - Under **Key pair (login)**, give your key pair name eg:linux114 is my keypair.



- Back to our EC2 instance setup, head to the **Network settings** section and click **Edit.**
- Give Your customised VPC and Public Subnet, **Enable** Auto Assign Public IP and Select your own customised Security Group.



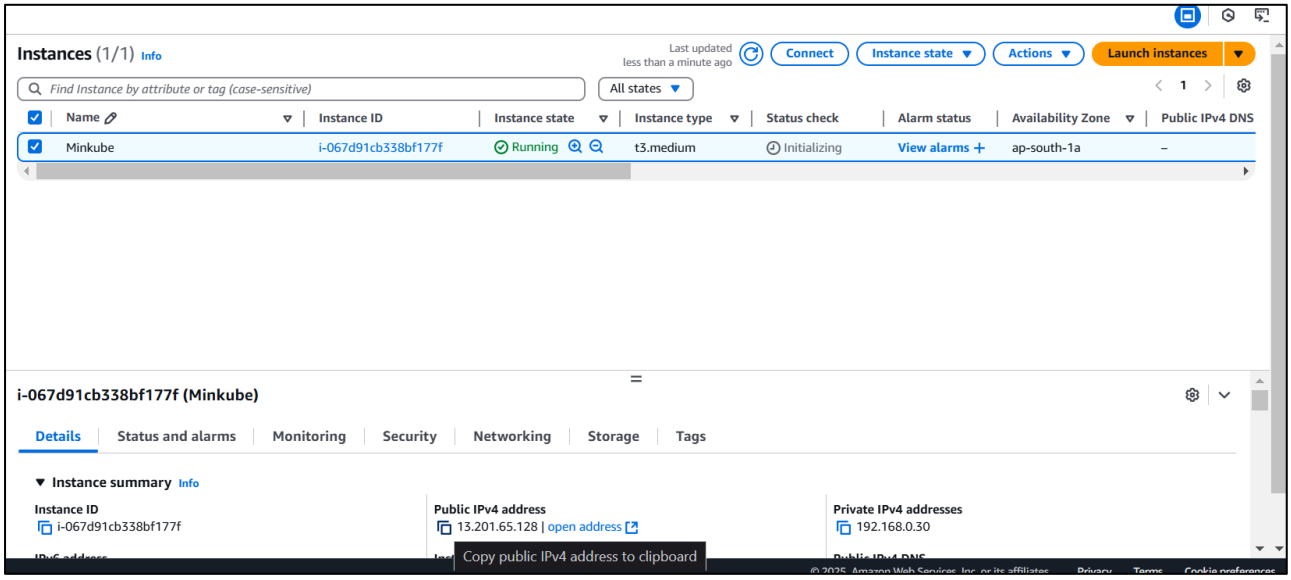
- In **configure storage** Allocate 24Gb.
- **We have allocated the instance type and configured the storage to meet the requirements for installation of Minikube.**

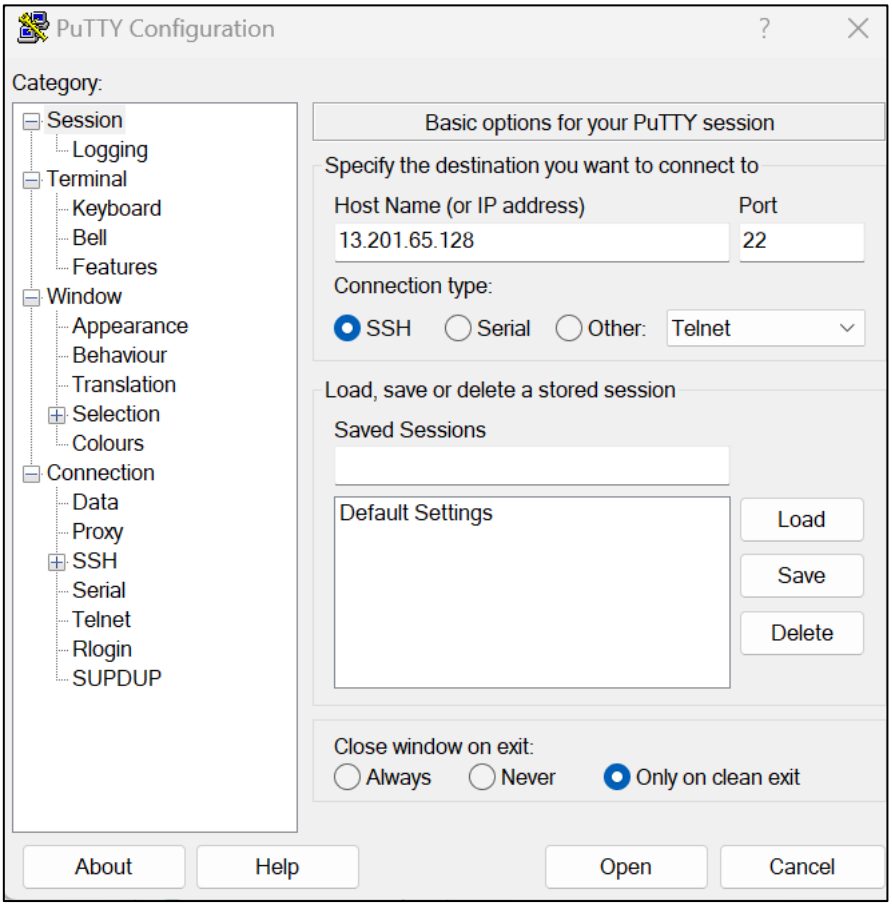
Prerequisites

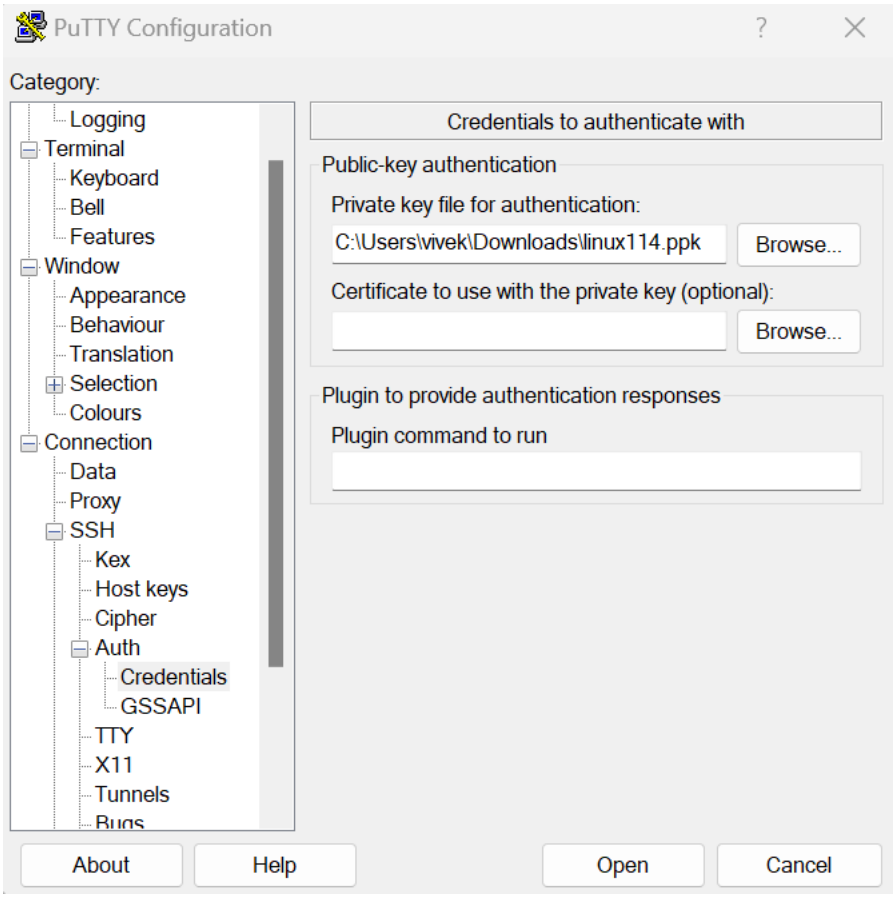
- Pre-Install Ubuntu 24.04
- Sudo User with admin privileges
- 2 GB RAM or more
- 2 CPU / vCPU or more
- 20 GB free hard disk space or more
- Docker / Virtual Machine Manager – KVM & VirtualBox
- Stable Internet Connection

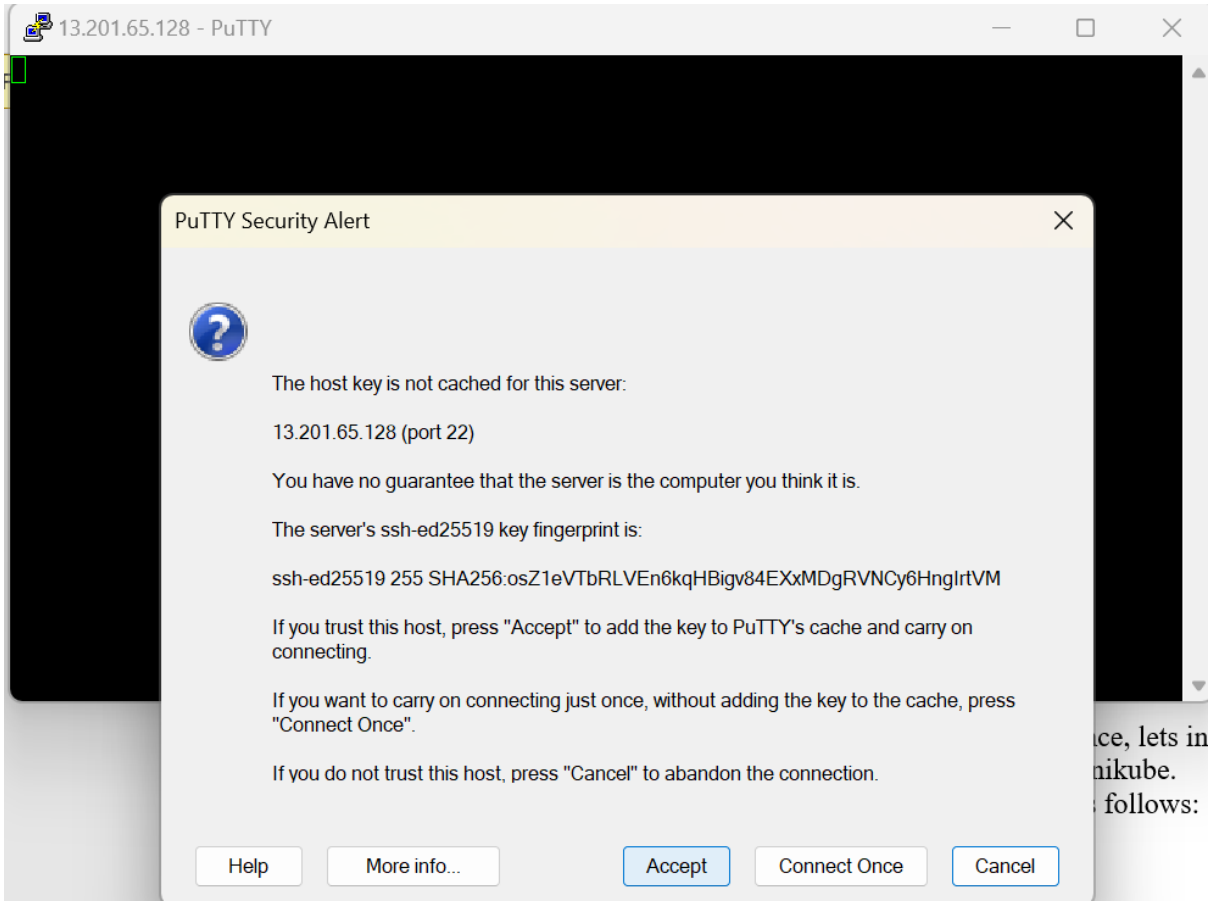
- choose **Launch instance**

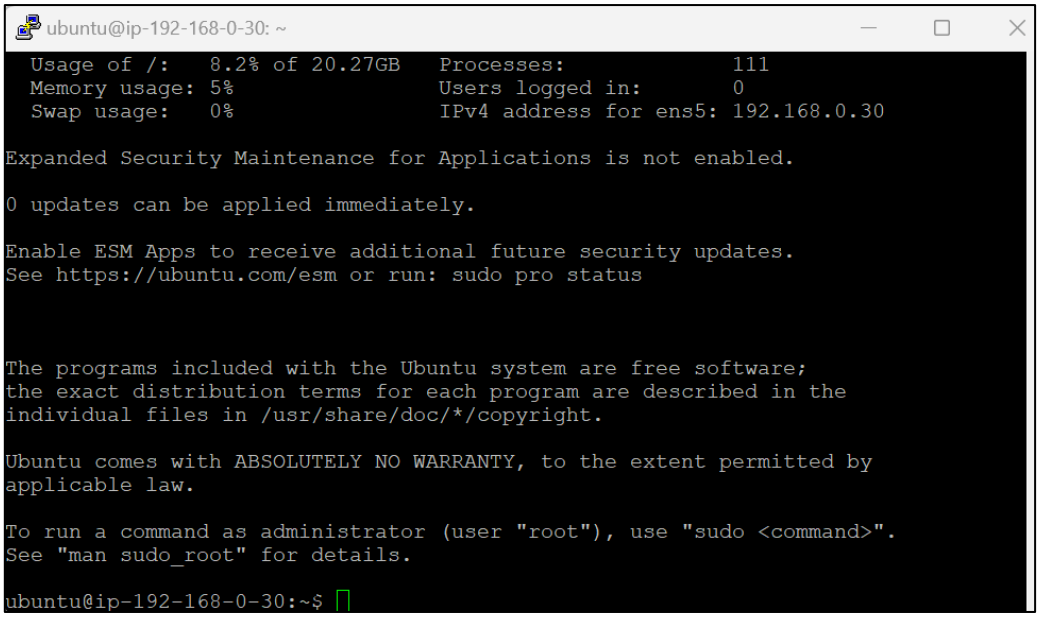
- Connect to Instance via Putty (**note: login as : ubuntu**)





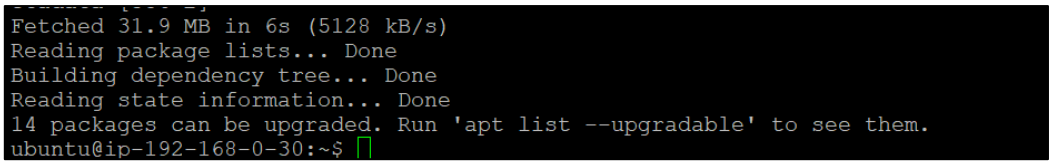

PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Bugs
Credentials to authenticate with
Public-key authentication
Private key file for authentication:
C:\Users\vivek\Downloads\linux114.ppk
Browse...
Certificate to use with the private key (optional):
Browse...
Plugin to provide authentication responses
Plugin command to run
About
Help
Open
Cancel


13.201.65.128 - PuTTY
PuTTY Security Alert
The host key is not cached for this server:
13.201.65.128 (port 22)
You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255 SHA256:osZ1eVTbRLVEn6kqHBigv84EXxMDgRVNCy6HngIrtVM
If you trust this host, press "Accept" to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, press "Connect Once".
If you do not trust this host, press "Cancel" to abandon the connection.
Help
More info...
Accept
Connect Once
Cancel

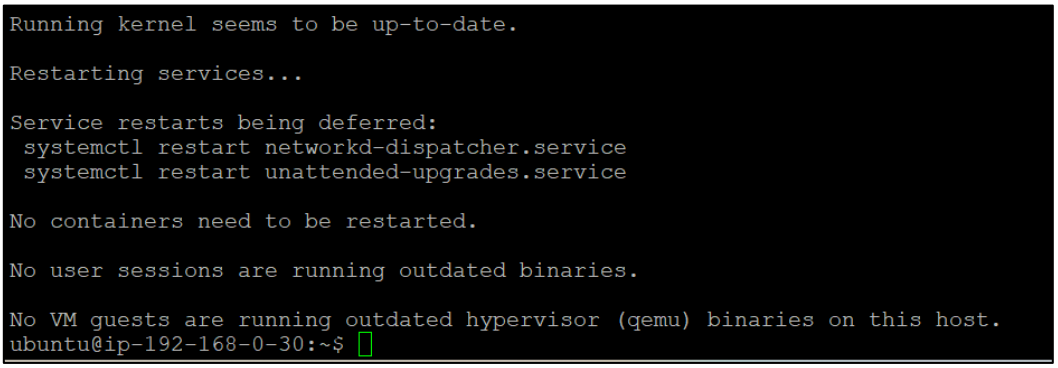


- Now that we have connected to our EC2 Instance, lets install docker in the EC2 to meet the prerequisites for installing Minikube.
- The commands for installation of docker are as follows:

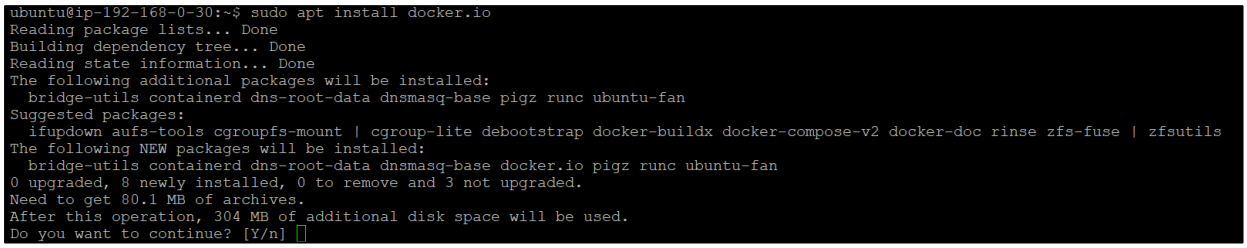
sudo apt update



sudo apt upgrade -y



sudo apt install docker.io



Type Y

```
Unpacking dns-root-data (2023112702~willsync1) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.90-2build2_amd64.deb ...
Unpacking dnsmasq-base (2.90-2build2) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../6-docker.io_26.1.3-0ubuntu1~24.04.1_amd64.deb ...
Unpacking docker.io (26.1.3-0ubuntu1~24.04.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../7-ubuntu-fan_0.12.16_all.deb ...
Unpacking ubuntu-fan (0.12.16) ...
Setting up dnsmasq-base (2.90-2build2) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up dns-root-data (2023112702~willsync1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.19+really1.7.12-0ubuntu4.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (26.1.3-0ubuntu1~24.04.1) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group `docker' (GID 113) ...
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 dbus (1.14.10-4ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...

Service restarts being deferred:
 systemctl restart networkd-dispatcher.service
 systemctl restart unattended-upgrades.service

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-192-168-0-30:~$
```

sudo chmod 666 /var/run/docker.sock

```
ubuntu@ip-192-168-0-30:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-192-168-0-30:~$
```

- Run docker ps command to see the status.

```
ubuntu@ip-192-168-0-30:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-192-168-0-30:~$
```

- Now install theMinikube in your EC2 with the following commands:

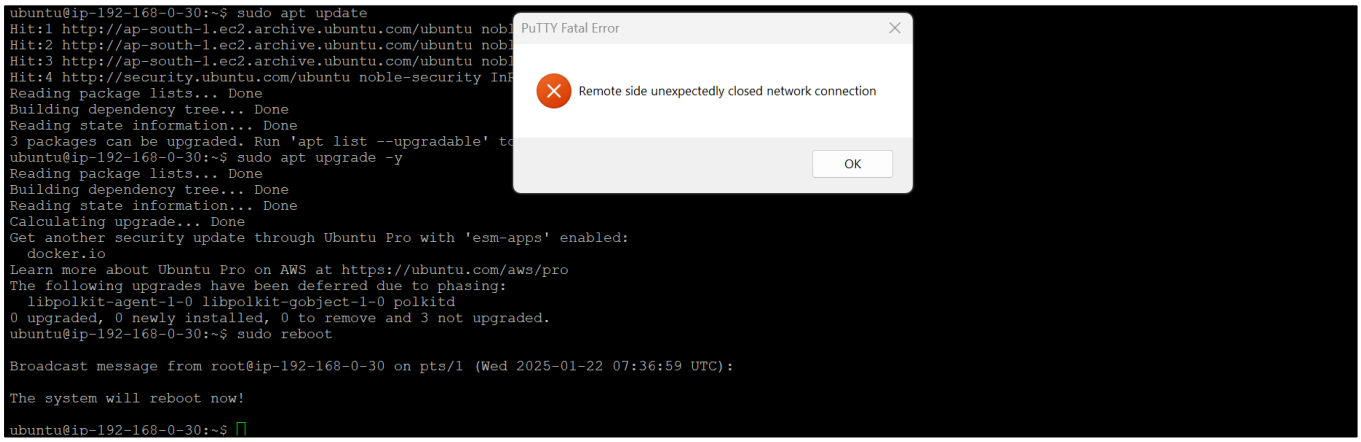
1) Apply Updates

sudo apt update
sudo apt upgrade -y

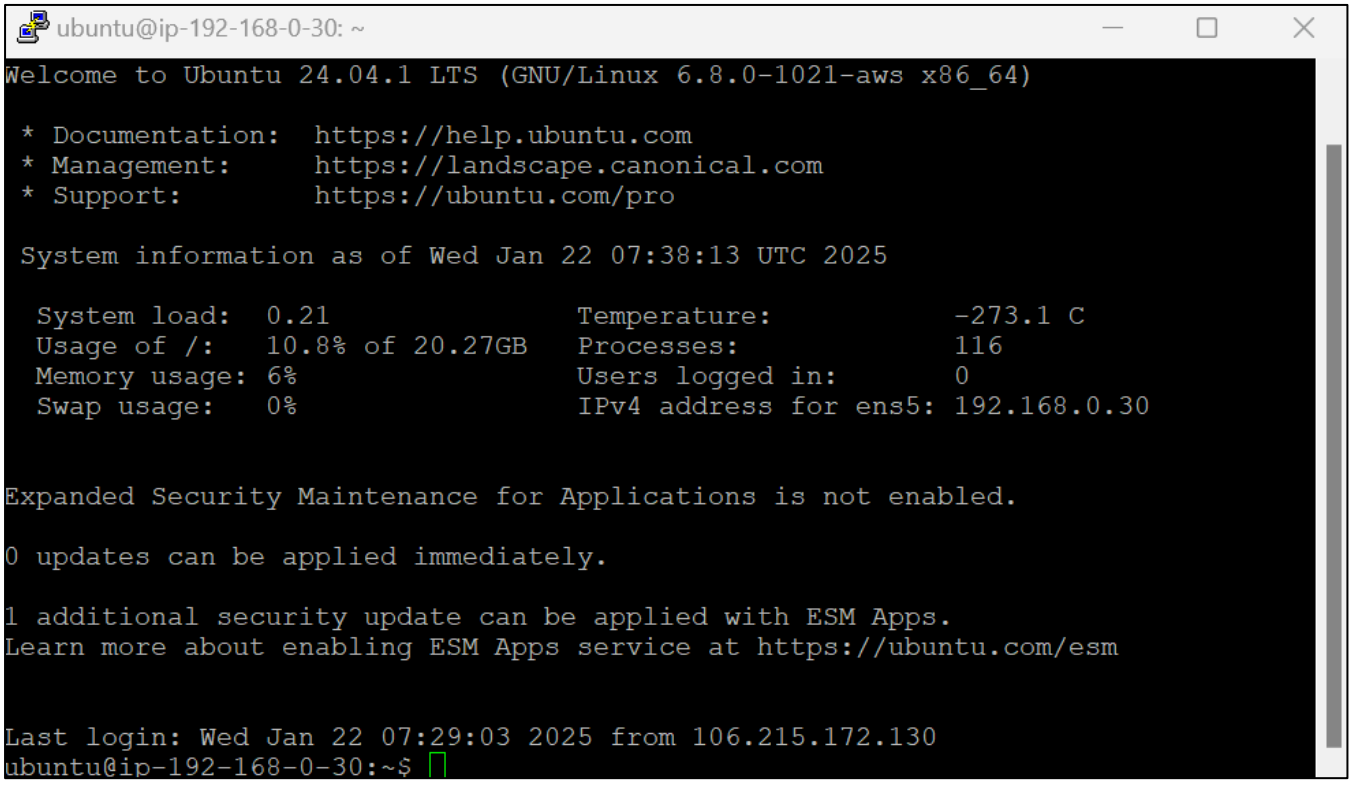
```
ubuntu@ip-192-168-0-30:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-192-168-0-30:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get another security update through Ubuntu Pro with 'esm-apps' enabled:
  docker.io
Learn more about Ubuntu Pro on AWS at https://ubuntu.com/aws/pro
The following upgrades have been deferred due to phasing:
  libpolkit-agent-1-0 libpolkit-gobject-1-0 polkitd
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
ubuntu@ip-192-168-0-30:~$
```

Once all the updates are installed then reboot your system.

```
sudo reboot
```



Now again connect to your EC2 via putty and login as we have rebooted.



2) Install Minikube Dependencies

Run the following to Install minikube dependencies.

```
sudo apt install -y curl wget apt-transport-https
```

```
Progress: [ 20%] [##########.............................................]
Progress: [ 40%] [####################.................................]
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
wget is already the newest version (1.21.4-1ubuntu4.1).
wget set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 3974 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-
ransport-https all 2.7.14build2 [3974 B]
Fetched 3974 B in 0s (222 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 70977 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.7.14build2_all.deb ...
Unpacking apt-transport-https (2.7.14build2) ...
Setting up apt-transport-https (2.7.14build2) ...
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-192-168-0-30:~$
```

3) Download and Install Minikube Binary

Use the following curl command to download latest minikube binary,

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

```
ubuntu@ip-192-168-0-30:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0  12.5M      0  0:00:09  0:00:09 --:--:-- 16.1M
ubuntu@ip-192-168-0-30:~$
```

Once the binary is downloaded then install it under the path /usr/local/bin

sudo install minikube-linux-amd64 /usr/local/bin/minikube

```
ubuntu@ip-192-168-0-30:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
ubuntu@ip-192-168-0-30:~$
```

Verify the minikube version

minikube version

```
ubuntu@ip-192-168-0-30:~$ minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
ubuntu@ip-192-168-0-30:~$
```

4) Install Kubectl Utility

Kubectl is a command line utility which is used to interact with Kubernetes cluster. It is used for managing deployments, service and pods etc. Use below curl command to download latest version of kubectl.

curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl

```
ubuntu@ip-192-168-0-30:~$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/relea
se/stable.txt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 53.7M  100 53.7M    0     0  10.9M      0  0:00:04  0:00:04 --:--:-- 11.8M
ubuntu@ip-192-168-0-30:~$
```

Once kubectl is downloaded then set the executable permissions on kubectl binary and move it to the path /usr/local/bin.

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

```
ubuntu@ip-192-168-0-30:~$ chmod +x kubectl
ubuntu@ip-192-168-0-30:~$ sudo mv kubectl /usr/local/bin/
ubuntu@ip-192-168-0-30:~$
```

Now verify the kubectl version

kubectl version -o yaml

```
ubuntu@ip-192-168-0-30:~$ kubectl version -o yaml
clientVersion:
  buildDate: "2024-08-13T07:37:34Z"
  compiler: gc
  gitCommit: 9edcffcde5595e8a5b1a35f88c421764e575afce
  gitTreeState: clean
  gitVersion: v1.31.0
  goVersion: go1.22.5
  major: "1"
  minor: "31"
  platform: linux/amd64
kustomizeVersion: v5.4.2

The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-192-168-0-30:~$
```

5) Start Minikube

As we are already stated in the beginning that we would be using docker as base for minikube, so start the minikube with the docker driver, run

minikube start --driver=docker

```
ubuntu@ip-192-168-0-30:~$ minikube start --driver=docker
* minikube v1.35.0 on Ubuntu 24.04
* Using the docker driver based on user configuration

X Exiting due to PROVIDER_DOCKER_NEWGRP: "docker version --format <no value>-<no value>:<no value>" exit status 1: permission denied while trying to connect
to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.45/version": dial unix /var/run/docker.sock: connect: p
ermission denied
* Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'
* Documentation: https://docs.docker.com/engine/install/linux-postinstall/

ubuntu@ip-192-168-0-30:~$
```

Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'

```
ubuntu@ip-192-168-0-30:~$ sudo usermod -aG docker $USER && newgrp docker
ubuntu@ip-192-168-0-30:~$
```

- Run following kubectl command to verify the Kubernetes version, node status

  kubectl get nodes

```
ubuntu@ip-192-168-0-4:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   57s   v1.32.0
ubuntu@ip-192-168-0-4:~$
```

- Run docker ps and you will see that the minikube is installed as a container in you EC2

```
ubuntu@ip-192-168-0-4:~$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED         STATUS
         PORTS                                                                                                                                                                                  NAMES
8229c34708a2   gcr.io/k8s-minikube/kicbase:v0.0.46   "/usr/local/bin/entr…"   38 seconds ago   Up 37 s
econds   127.0.0.1:32772->22/tcp, 127.0.0.1:32771->2376/tcp, 127.0.0.1:32770->5000/tcp, 127.0.0.1:3276
9->8443/tcp, 127.0.0.1:32768->32443/tcp   minikube
ubuntu@ip-192-168-0-4:~$
```

To verify the minikube installation, let's try to create a pod and also deploy nginx based deployment.

- Create a folder in your EC2 and enter the folder

  mkdir vivek
  cd vivek

```
ubuntu@ip-192-168-0-4:~$ mkdir vivek
ubuntu@ip-192-168-0-4:~$ ls
minikube-linux-amd64  vivek
ubuntu@ip-192-168-0-4:~$ cd vivek
ubuntu@ip-192-168-0-4:~/vivek$
```

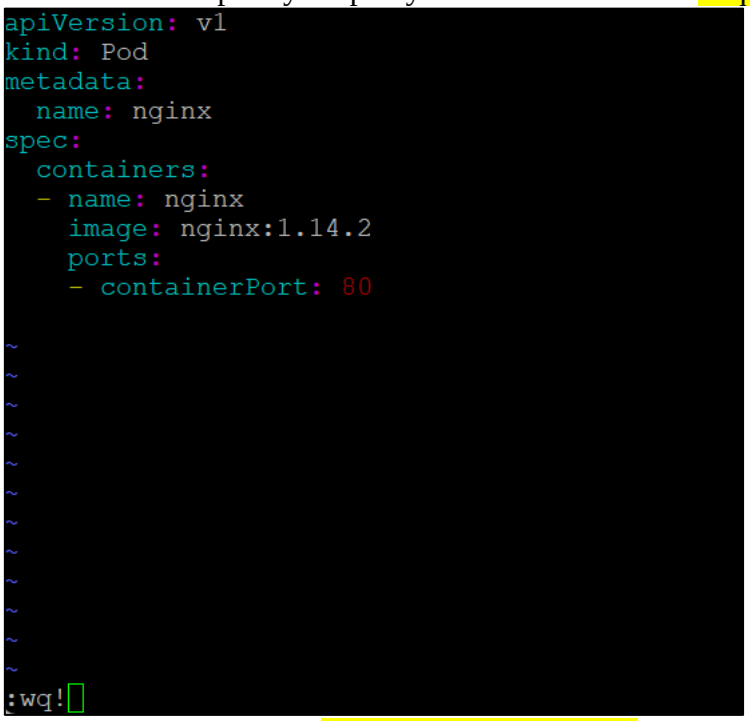
- Now create a yaml file and paste the yaml script obtained form Kubernetes official website for creating a pod,

  vim pod.yml

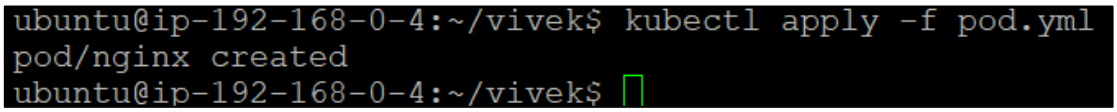
- The YAML Script is as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

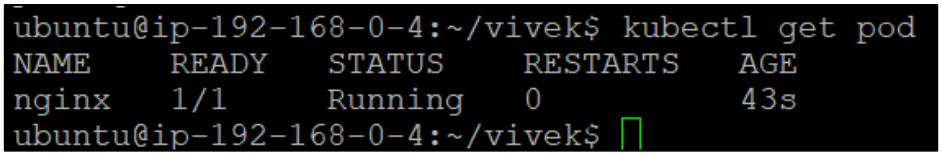
- Paste the above script in your pod.yml file and save with :wq! Command.



- Now execute the command kubectl apply -f pod.yml to create the pod.



- Check the newly created pod with kubectl get pod command.



- Now lets deploy nginx based deployment.
  - create a yaml file and paste the yaml script obtained form Kubernetes official website for creating a pod,

    vim deployment.yml

- The YAML Script is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- Paste the above script in your deployment.yml file and save with :wq! command.

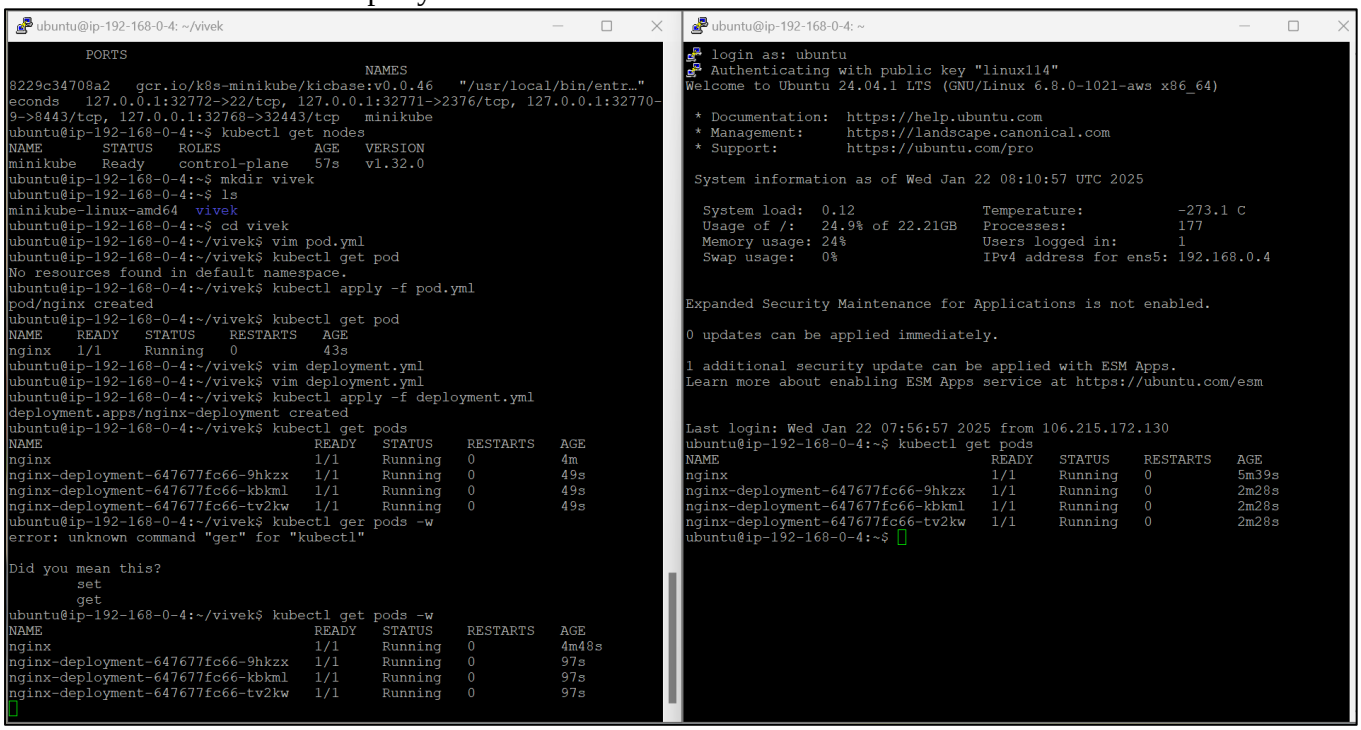
- Now execute the command kubectl apply -f deployment.yml to create the pod.

```
ubuntu@ip-192-168-0-4:~/vivek$ kubectl apply -f deployment.yml
deployment.apps/nginx-deployment created
ubuntu@ip-192-168-0-4:~/vivek$
```

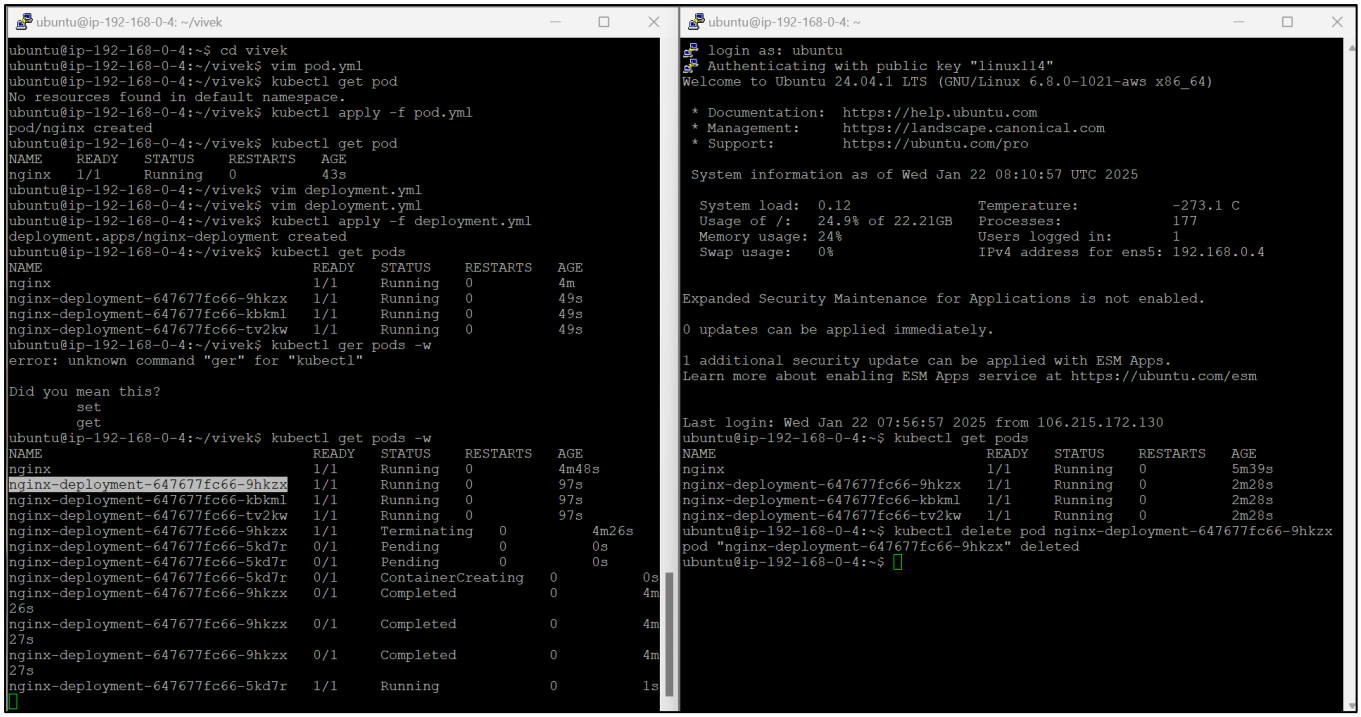
- Check the newly created pods via deployment with kubectl get pods command.

```
ubuntu@ip-192-168-0-4:~/vivek$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx                               1/1     Running   0          4m
nginx-deployment-647677fc66-9hkzx   1/1     Running   0          49s
nginx-deployment-647677fc66-kbkml   1/1     Running   0          49s
nginx-deployment-647677fc66-tv2kw   1/1     Running   0          49s
ubuntu@ip-192-168-0-4:~/vivek$
```

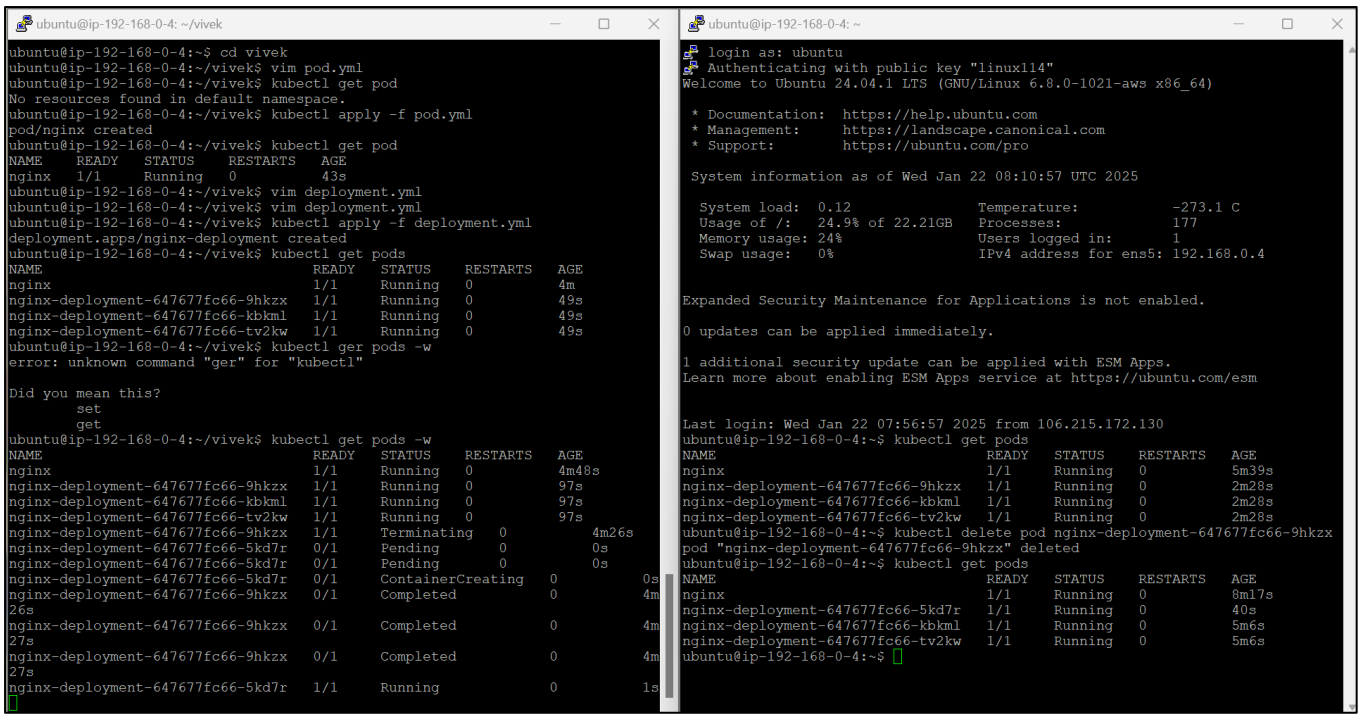
- Now we can see the 3 pods created by deployment.yml file along with one pod created by pod.yml file.
- Now execute the command kubectl get pods -w to monitor the pods and login to a different putty window.



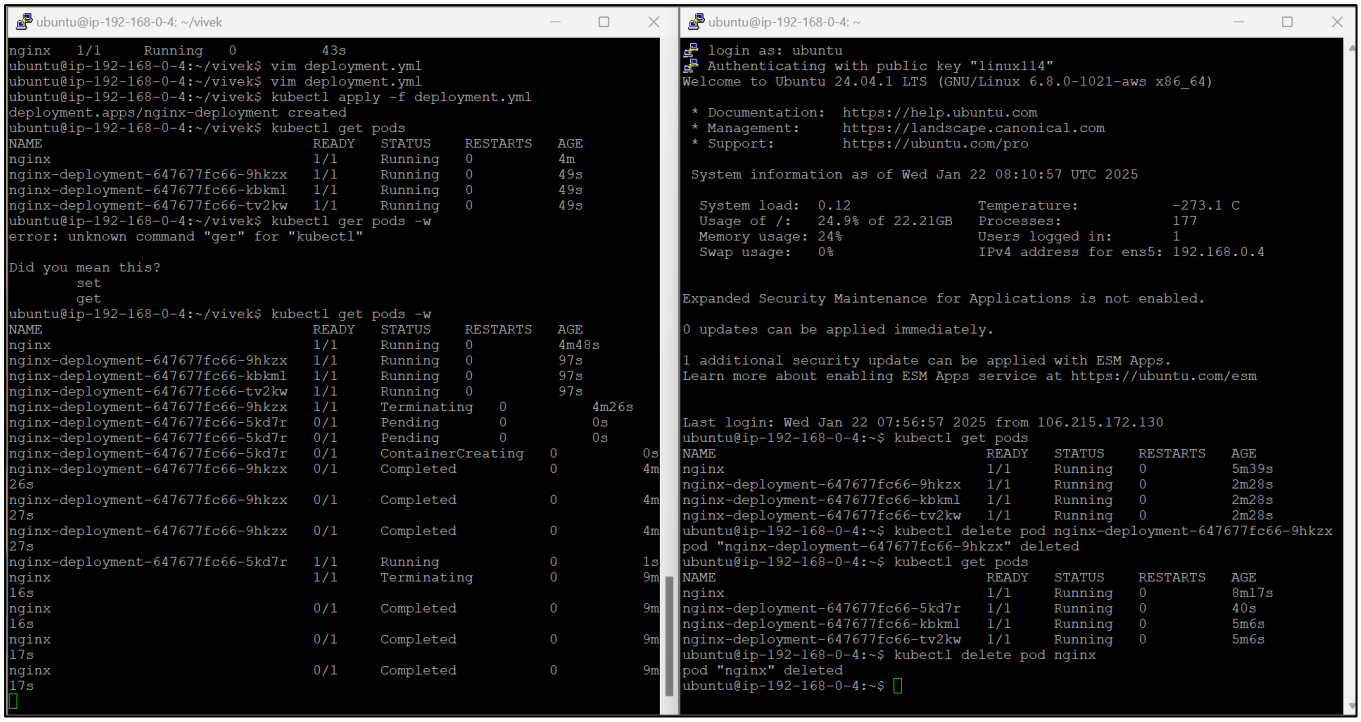
- In the new Putty window, try deleting a deployment-managed pod using the following command: kubectl delete pod <deployment-pod-name> (replace <deployment-pod-name> with the name of the pod created by the deployment).



- When we delete a deployment pod, a new pod is automatically created. This behavior is governed by the deployment's YAML configuration, which specifies a replica count of 3. The Kubernetes scheduler ensures that the desired number of replicas (3) is always maintained by automatically creating a new pod whenever one is deleted.



- When we delete a manually created pod using the command kubectl delete pod <pod-name> (replacing <pod-name> with the name of the pod we manually created), we notice that a new pod is not generated. This is because manually created pods are not managed or overseen by the deployment.




```
ubuntu@ip-192-168-0-4:~$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
nginx-deployment-647677fc66-5kd7r      1/1     Running   0          2m15s
nginx-deployment-647677fc66-kbkml      1/1     Running   0          6m41s
nginx-deployment-647677fc66-tv2kw      1/1     Running   0          6m41s
ubuntu@ip-192-168-0-4:~$
```