MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE,
ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS
KHANDWALA COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
(EMPOWERED AUTONOMOUS)

(REACCREDITED 'A' Grade by NAAC)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
(ISO 9001:2015)

CERTIFICATE

Name: Gauri Chandrasekar

Roll No: 01 **Programme:** Containerization Using Docker Kubernetes **Semester: IV**

This is certified to be a bonafide record of practical work done by the above student in the college laboratory for the **Containerization Using Docker Kubernetes** (Course Code: **PSCS404CDK24OE**) course for the partial fulfilment of **Semester IV** of **M.Sc.** with specialization in **Computer Science** during the academic year 2024-2025.

The journal is the original study work duly approved by the undersigned in 2024-2025.

External Examiner **(Subject-In-Charge)**

Prof. Rashid Patel

Date of Examination **(College Stamp)**

INDEX

SUBJECT: CONTAINERIZATION USING DOCKER KUBERNETES

CLASS: SY MSC (CS)

SEMESTER: IV

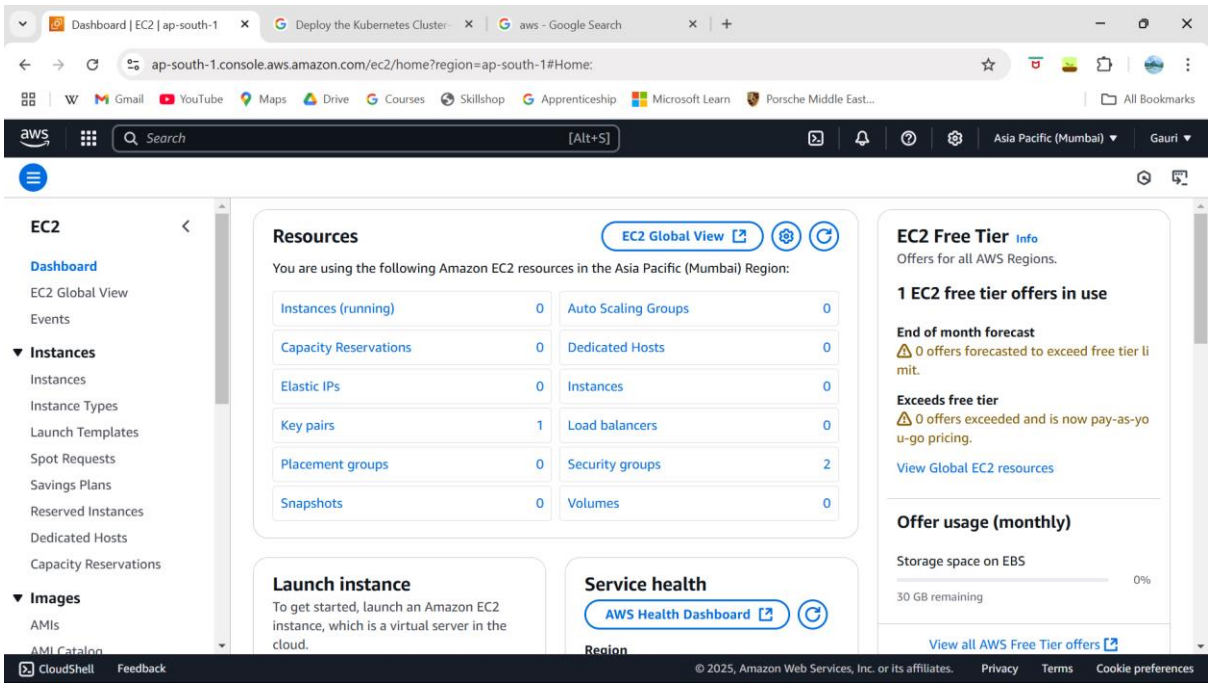
ROLL NO: 01

SR NO.	DATE	PRACTICAL AIM	SIGN
1		Deploy the Kubernetes Cluster – Master Node on EC2	
2		Deploy the Kubernetes Cluster – Worker Node 1 on EC2	
3		Deploy the Kubernetes Cluster – Worker Node 2 on EC2	
4		Deploy the Kubernetes Cluster on Windows using VirtualBox	
5		Deploy the Kubernetes Cluster on Linux Ubuntu	
6		Deploy an Azure Kubernetes Service (AKS) cluster using Azure portal	
7		Kubernetes deployment using YAML – NGINX Containers	
8		Netflix's Challenges on AWS Cloud: A Case Study	

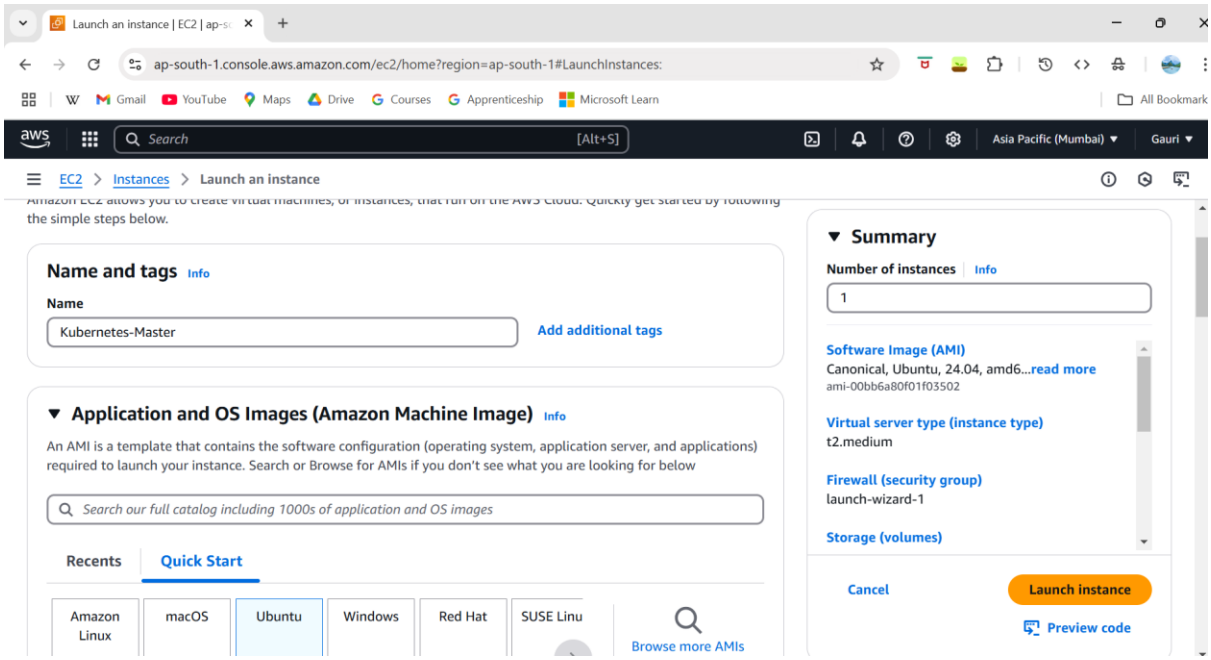
Practical No. 01

Aim: Deploy the Kubernetes Cluster – Master Node on EC2

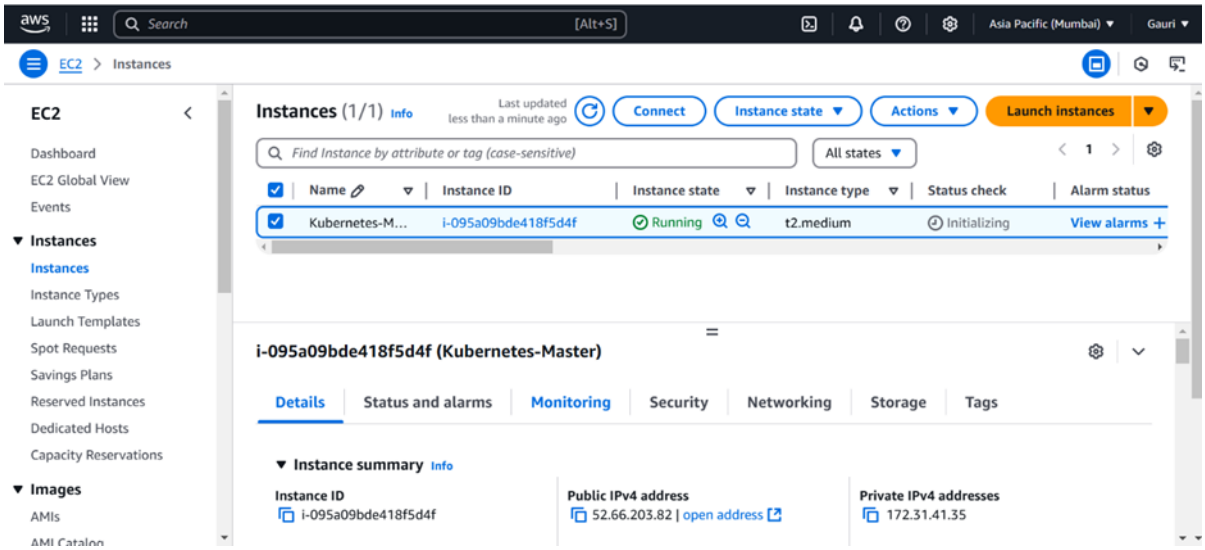
Step 1: Create an account on Cloud Computing Services – Amazon Web Services (AWS). After this, log into your account where dashboard will appear like this.



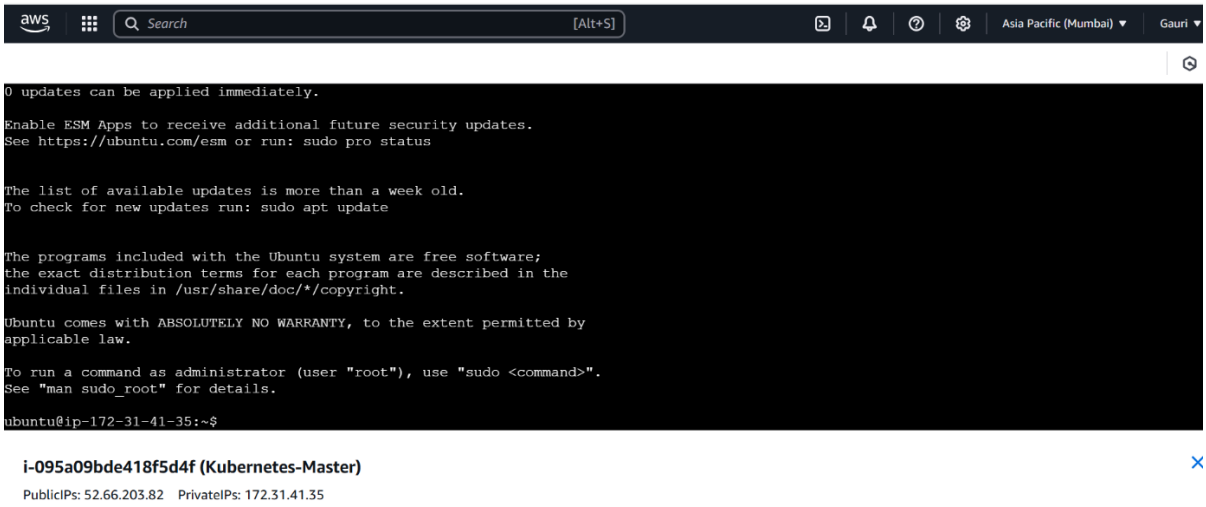
Step 2: Create an instance on EC2 to deploy a master node.



Step 3: After creating, click on the connect to start the instance.



Step 4: It will redirect you to the EC2 terminal that you created.



Step 5: Update the packages.
Command: sudo su
sudo apt-get update

Step 6: Write a command that is often used as a prerequisite when setting up external repositories or downloading packages securely from HTTPS sources.
Command: sudo apt install apt-transport-https curl -y

```
root@ip-172-31-41-35:/home/ubuntu# sudo apt install apt-transport-https curl -y
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.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 134 not upgraded.
```

Step 7: Type in these commands

sudo mkdir -p /etc/apt/keyrings

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

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

```
root@ip-172-31-41-35:/home/ubuntu# sudo mkdir -p /etc/apt/keyrings
root@ip-172-31-41-35:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docke
r.gpg
root@ip-172-31-41-35:/home/ubuntu# echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.d
ocker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-172-31-41-35:/home/ubuntu# sudo apt-get 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
```

Step 8: Install containerd, which is essential for running containers.
Command: sudo apt-get install containerd.io -y

```
root@ip-172-31-41-35:/home/ubuntu# sudo apt-get install containerd.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 134 not upgraded.
Need to get 29.6 MB of archives.
After this operation, 121 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.25-1 [29.6 MB]
```

Step 9: sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
root@ip-172-31-41-35:/home/ubuntu# sudo mkdir -p /etc/containerd
root@ip-172-31-41-35:/home/ubuntu# 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
```

Step 10: Type in these commands

sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/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:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
root@ip-172-31-41-35:/home/ubuntu# sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
root@ip-172-31-41-35:/home/ubuntu# sudo systemctl restart containerd
root@ip-172-31-41-35:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/key
rings/kubernetes-apt-keyring.gpg
root@ip-172-31-41-35:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1
.30/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:/v1.30/deb/ /
```

Step 11: Install Kubernetes Components
Commands: sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl

```
root@ip-172-31-41-35:/home/ubuntu# 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 134 not upgraded.
Need to get 93.5 MB of archives.
After this operation, 341 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  cri-tools 1.30.1-1.1 [21.3 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubeadm 1.30.10-1.1 [10.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubectl 1.30.10-1.1 [10.8 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  kubernetes-cni 1.4.0-1.1 [32.9 MB]
```

Step 12: Prevent Automatic Updates
Command: sudo apt-mark hold kubelet kubeadm kubectl

```
root@ip-172-31-41-35:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-41-35:/home/ubuntu#
```

Step 13: Enable and Start Kubelet
Command: sudo systemctl enable --now kubelet
sudo swapoff -a
sudo modprobe br_netfilter
sudo sysctl -w net.ipv4.ip_forward=1

```
root@ip-172-31-41-35:/home/ubuntu# sudo systemctl enable --now kubelet
root@ip-172-31-41-35:/home/ubuntu# sudo swapoff -a
root@ip-172-31-41-35:/home/ubuntu# sudo modprobe br_netfilter
root@ip-172-31-41-35:/home/ubuntu# sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@ip-172-31-41-35:/home/ubuntu#
```

Step 14: Use the following command to initialize the cluster
Command: sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
root@ip-172-31-41-35:/home/ubuntu# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0303 19:54:37.679890    3273 version.go:256] remote version is much newer: v1.32.2; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.10
[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 in beforehand using 'kubeadm config images pull'
W0303 19:54:38.341447    3273 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.8" of the c
sistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sandbox image.
```

Step 15: Create a .kube directory in your home directory
Command: mkdir -p $HOME/.kube
Copy the Kubernetes configuration file to your home directory
Command: sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
Change ownership of the file:
Command: sudo chown (id−u):(id -g) $HOME/.kube/config

kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

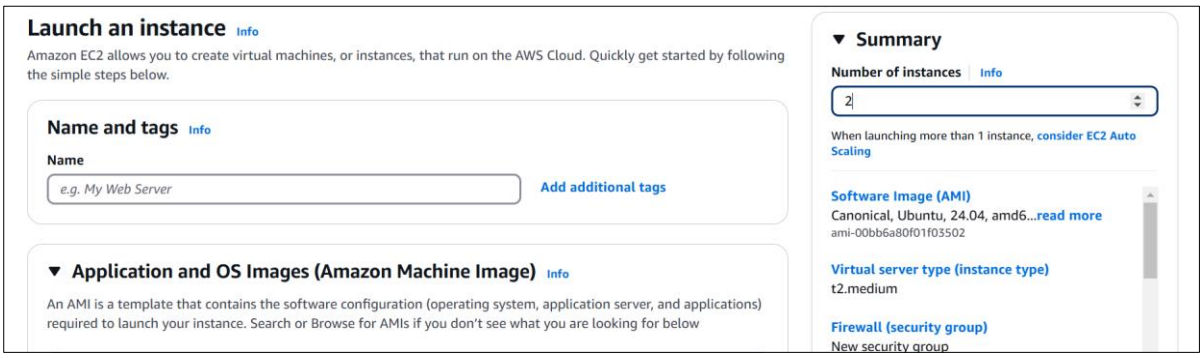
```
root@ip-172-31-41-35:/home/ubuntu# mkdir -p $HOME/.kube
root@ip-172-31-41-35:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@ip-172-31-41-35:/home/ubuntu# sudo chown (id-u):(id -g) $HOME/.kube/config
bash: syntax error near unexpected token `('
root@ip-172-31-41-35:/home/ubuntu# ^C
root@ip-172-31-41-35:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@ip-172-31-41-35:/home/ubuntu# kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

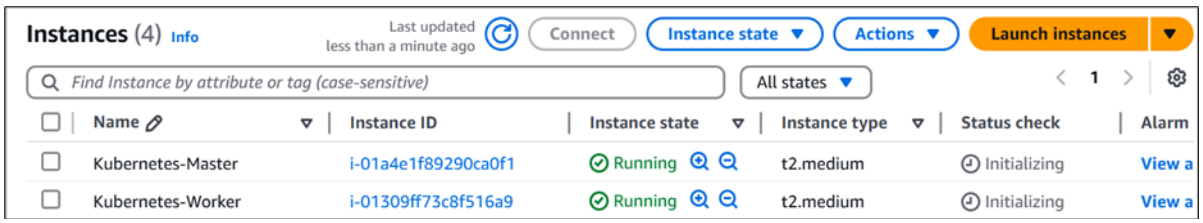
Step 15: Verify that all the pods are up and running
Commands: kubectl get pods --all-namespaces
kubectl get nodes

```
root@ip-172-31-41-35:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE      NAME                                       READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-xbr7x                      1/1     Running   0          36s
kube-system    coredns-55cb58b774-hhtkj                   1/1     Running   0          2m43s
kube-system    coredns-55cb58b774-r842p                   1/1     Running   0          2m43s
kube-system    etcd-ip-172-31-41-35                       1/1     Running   0          2m59s
kube-system    kube-apiserver-ip-172-31-41-35             1/1     Running   0          2m59s
kube-system    kube-controller-manager-ip-172-31-41-35    1/1     Running   0          2m59s
kube-system    kube-proxy-5xdbk                           1/1     Running   0          2m43s
kube-system    kube-scheduler-ip-172-31-41-35             1/1     Running   0          2m59s
root@ip-172-31-41-35:/home/ubuntu# kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-41-35   Ready    control-plane   3m16s   v1.30.10
root@ip-172-31-41-35:/home/ubuntu#
```

Practical No. 02

Aim: Deploy the Kubernetes Cluster – Worker Node 1 on EC2





BOTH MASTER AND WORKER

Commands:
sudo su
sudo apt-get update
sudo apt install apt-transport-https curl -y

```
ubuntu@ip-172-31-33-93:~$ sudo su
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
```

Commands:
sudo mkdir -p /etc/apt/keyrings

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

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
root@ip-172-31-33-93:/home/ubuntu# sudo mkdir -p /etc/apt/keyrings
root@ip-172-31-33-93:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docke
r.gpg
root@ip-172-31-33-93:/home/ubuntu# echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.d
ocker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-172-31-33-93:/home/ubuntu#
```

Commands:
sudo apt-get update
sudo apt-get install containerd.io -y

```
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get install containerd.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 134 not upgraded.
Need to get 29.6 MB of archives.
After this operation, 121 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.25-1 [29.6 MB]
Fetched 29.6 MB in 0s (77.3 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 70614 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.7.25-1_amd64.deb ...
Unpacking containerd.io (1.7.25-1) ...
Setting up containerd.io (1.7.25-1) ...
```

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

```
root@ip-172-31-33-93:/home/ubuntu# sudo mkdir -p /etc/containerd
root@ip-172-31-33-93:/home/ubuntu# 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
```

Commands:
sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/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:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update

```
root@ip-172-31-33-93:/home/ubuntu# sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
root@ip-172-31-33-93:/home/ubuntu# sudo systemctl restart containerd
root@ip-172-31-33-93:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/key
rings/kubernetes-apt-keyring.gpg
root@ip-172-31-33-93:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1
.30/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:/v1.30/deb/ /
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get 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 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1189 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [15.3 kB]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 16.5 kB in 1s (28.1 kB/s)
Reading package lists... Done
```

Command: sudo apt-get install -y kubelet kubeadm kubectl

```
root@ip-172-31-33-93:/home/ubuntu# 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 134 not upgraded.
Need to get 93.5 MB of archives.
```

Command: sudo apt-mark hold kubelet kubeadm kubectl

```
root@ip-172-31-33-93:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-33-93:/home/ubuntu#
```

Commands:
sudo systemctl enable --now kubelet
sudo swapoff -a
sudo modprobe br_netfilter
sudo sysctl -w net.ipv4.ip_forward=1

```
root@ip-172-31-33-93:/home/ubuntu# sudo systemctl enable --now kubelet
root@ip-172-31-33-93:/home/ubuntu# sudo swapoff -a
root@ip-172-31-33-93:/home/ubuntu# sudo modprobe br_netfilter
root@ip-172-31-33-93:/home/ubuntu# sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@ip-172-31-33-93:/home/ubuntu#
```

INITIALIZE THE CLUSTER (RUN ONLY ON MASTER)

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

```
root@ip-172-31-33-93:/home/ubuntu# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0304 15:18:17.398983    3559 version.go:256] remote version is much newer: v1.32.2; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.10
[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 in beforehand using 'kubeadm config images pull'
W0304 15:18:18.069976    3559 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.8" of the co
sistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
```

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

```
root@ip-172-31-33-93:/home/ubuntu# mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@ip-172-31-33-93:/home/ubuntu#
```

Command:
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

```
root@ip-172-31-33-93:/home/ubuntu# kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Commands:
kubectl get pods --all-namespaces
kubectl get nodes

```
root@ip-172-31-33-93:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE      NAME                                      READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-2qn7b                     1/1     Running   0          70s
kube-system    coredns-55cb58b774-jttl7                  1/1     Running   0          3m53s
kube-system    coredns-55cb58b774-wzsvm                  1/1     Running   0          3m53s
kube-system    etcd-ip-172-31-33-93                      1/1     Running   0          4m8s
kube-system    kube-apiserver-ip-172-31-33-93            1/1     Running   0          4m8s
kube-system    kube-controller-manager-ip-172-31-33-93   1/1     Running   0          4m8s
kube-system    kube-proxy-dwbqp                          1/1     Running   0          3m53s
kube-system    kube-scheduler-ip-172-31-33-93            1/1     Running   0          4m8s
root@ip-172-31-33-93:/home/ubuntu# kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-33-93   Ready    control-plane   4m20s   v1.30.10
```

Note: Copy kubeadm join key from the master node and paste in the worker node.

kubeadm join 172.31.33.93:6443 --token iw26d0.l44aaydskuma0gnd \
--discovery-token-ca-cert-hash
sha256:0e87e11ceed985e317b6f2ae9bb36a637cae2bb4ed479aa97f156f3032ecc8d2

```
root@ip-172-31-35-97:/home/ubuntu# kubeadm join 172.31.33.93:6443 --token iw26d0.l44aaydskuma0gnd \
        --discovery-token-ca-cert-hash sha256:0e87e11ceed985e317b6f2ae9bb36a637cae2bb4ed479aa97f156f3032ecc8d2
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.002639624s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.
```

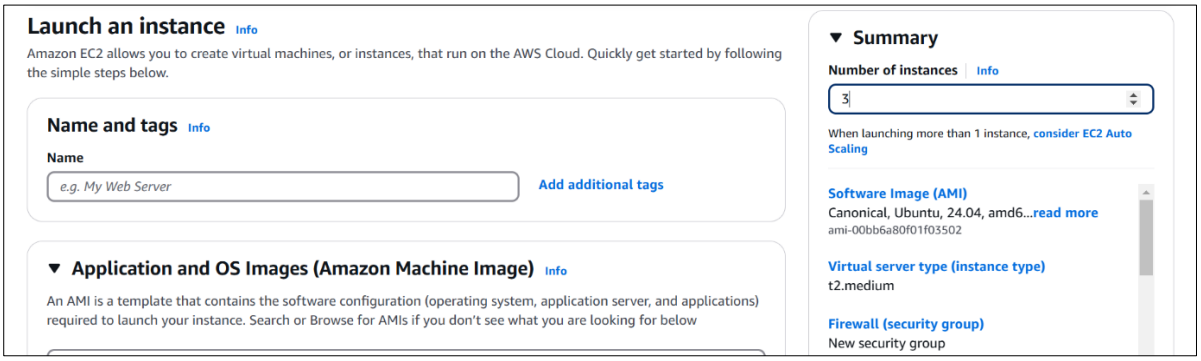
Note: Run the below command for worker node to join the cluster.

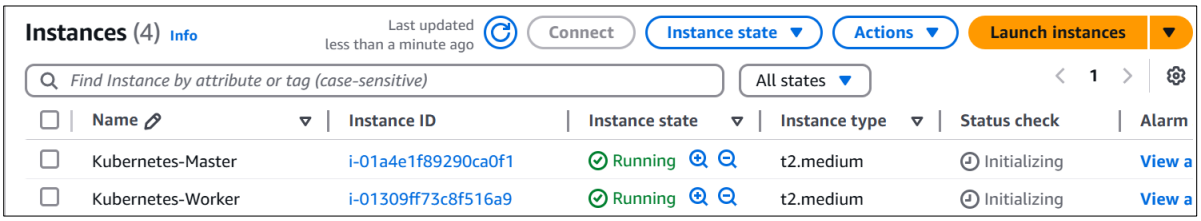
Command: kubectl get nodes

```
root@ip-172-31-33-93:/home/ubuntu# kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-33-93   Ready    control-plane   6m34s   v1.30.10
ip-172-31-35-97   Ready    <none>          49s     v1.30.10
root@ip-172-31-33-93:/home/ubuntu#
```

Practical No. 03

Aim: Deploy the Kubernetes Cluster – Worker Node 2 on EC2





BOTH MASTER AND WORKER

Commands:
sudo su
sudo apt-get update
sudo apt install apt-transport-https curl -y

```
ubuntu@ip-172-31-33-93:~$ sudo su
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
```

Commands:
sudo mkdir -p /etc/apt/keyrings

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

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
root@ip-172-31-33-93:/home/ubuntu# sudo mkdir -p /etc/apt/keyrings
root@ip-172-31-33-93:/home/ubuntu# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docke
r.gpg
root@ip-172-31-33-93:/home/ubuntu# echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.d
ocker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-172-31-33-93:/home/ubuntu#
```

Commands:
sudo apt-get update
sudo apt-get install containerd.io -y

```
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get install containerd.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  containerd.io
0 upgraded, 1 newly installed, 0 to remove and 134 not upgraded.
Need to get 29.6 MB of archives.
After this operation, 121 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.25-1 [29.6 MB]
Fetched 29.6 MB in 0s (77.3 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 70614 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.7.25-1_amd64.deb ...
Unpacking containerd.io (1.7.25-1) ...
Setting up containerd.io (1.7.25-1) ...
```

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

```
root@ip-172-31-33-93:/home/ubuntu# sudo mkdir -p /etc/containerd
root@ip-172-31-33-93:/home/ubuntu# 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
```

Commands:
sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/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:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update

```
root@ip-172-31-33-93:/home/ubuntu# sudo sed -i -e 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
root@ip-172-31-33-93:/home/ubuntu# sudo systemctl restart containerd
root@ip-172-31-33-93:/home/ubuntu# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/key
rings/kubernetes-apt-keyring.gpg
root@ip-172-31-33-93:/home/ubuntu# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1
.30/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:/v1.30/deb/ /
root@ip-172-31-33-93:/home/ubuntu# sudo apt-get 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 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1189 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [15.3 kB]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 16.5 kB in 1s (28.1 kB/s)
Reading package lists... Done
```

Command: sudo apt-get install -y kubelet kubeadm kubectl

```
root@ip-172-31-33-93:/home/ubuntu# 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 134 not upgraded.
Need to get 93.5 MB of archives.
```

Command: sudo apt-mark hold kubelet kubeadm kubectl

```
root@ip-172-31-33-93:/home/ubuntu# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-33-93:/home/ubuntu#
```

Commands:
sudo systemctl enable --now kubelet
sudo swapoff -a
sudo modprobe br_netfilter
sudo sysctl -w net.ipv4.ip_forward=1

```
root@ip-172-31-33-93:/home/ubuntu# sudo systemctl enable --now kubelet
root@ip-172-31-33-93:/home/ubuntu# sudo swapoff -a
root@ip-172-31-33-93:/home/ubuntu# sudo modprobe br_netfilter
root@ip-172-31-33-93:/home/ubuntu# sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@ip-172-31-33-93:/home/ubuntu#
```

INITIALIZE THE CLUSTER (RUN ONLY ON MASTER)

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

```
root@ip-172-31-33-93:/home/ubuntu# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0304 15:18:17.398983    3559 version.go:256] remote version is much newer: v1.32.2; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.10
[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 in beforehand using 'kubeadm config images pull'
W0304 15:18:18.069976    3559 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.8" of the co
sistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
```

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

```
root@ip-172-31-33-93:/home/ubuntu# mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@ip-172-31-33-93:/home/ubuntu#
```

Command:
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

```
root@ip-172-31-33-93:/home/ubuntu# kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

Commands:
kubectl get pods --all-namespaces
kubectl get nodes

```
root@ip-172-31-33-93:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE      NAME                                      READY   STATUS    RESTARTS   AGE
kube-flannel   kube-flannel-ds-2qn7b                     1/1     Running   0          70s
kube-system    coredns-55cb58b774-jttl7                  1/1     Running   0          3m53s
kube-system    coredns-55cb58b774-wzsvm                  1/1     Running   0          3m53s
kube-system    etcd-ip-172-31-33-93                      1/1     Running   0          4m8s
kube-system    kube-apiserver-ip-172-31-33-93            1/1     Running   0          4m8s
kube-system    kube-controller-manager-ip-172-31-33-93   1/1     Running   0          4m8s
kube-system    kube-proxy-dwbqp                          1/1     Running   0          3m53s
kube-system    kube-scheduler-ip-172-31-33-93            1/1     Running   0          4m8s
root@ip-172-31-33-93:/home/ubuntu# kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-33-93   Ready    control-plane   4m20s   v1.30.10
```

Note: Copy kubeadm join key from the master node and paste in the worker node.

kubeadm join 172.31.33.93:6443 --token iw26d0.l44aaydskuma0gnd \
--discovery-token-ca-cert-hash
sha256:0e87e11ceed985e317b6f2ae9bb36a637cae2bb4ed479aa97f156f3032ecc8d2

```
root@ip-172-31-35-97:/home/ubuntu# kubeadm join 172.31.33.93:6443 --token iw26d0.l44aaydskuma0gnd \
        --discovery-token-ca-cert-hash sha256:0e87e11ceed985e317b6f2ae9bb36a637cae2bb4ed479aa97f156f3032ecc8d2
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.002639624s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.
```

Note: Run the below command for worker node to join the cluster.

Command: kubectl get nodes

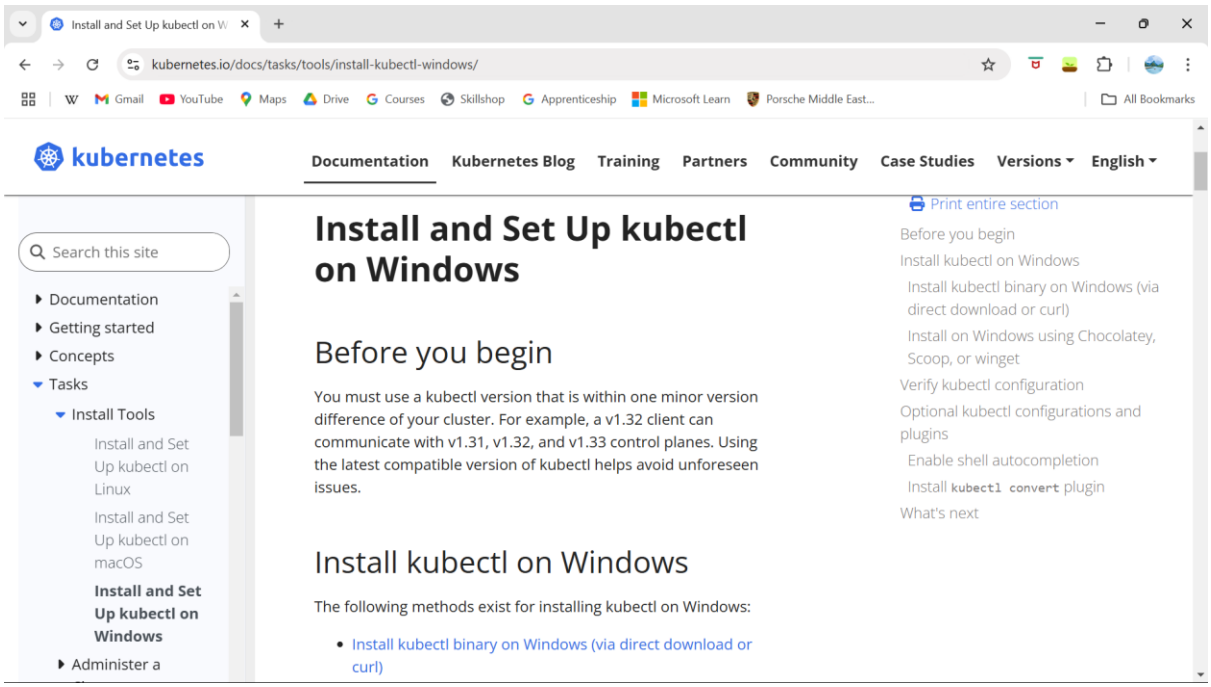
Instead of two, three nodes will be visible.

```
root@ip-172-31-33-93:/home/ubuntu# kubectl get nodes
NAME              STATUS   ROLES           AGE     VERSION
ip-172-31-33-93   Ready    control-plane   6m34s   v1.30.10
ip-172-31-35-97   Ready    <none>          49s     v1.30.10
root@ip-172-31-33-93:/home/ubuntu#
```

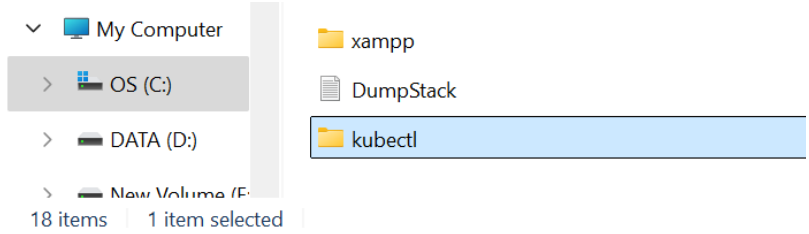
Practical No. 04

Aim: Deploy the Kubernetes Cluster on Windows using VirtualBox

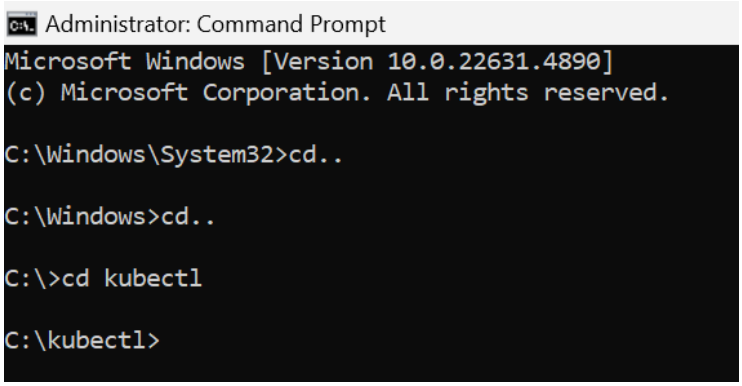
Step 1: Search “install kubectl” on google. Click on the Kubernetes website and go into install tools.



Step 2: Create a folder “kubectl” in your C – Drive.



Step 3: Open Command Prompt and run as an Administrator. Navigate into the kubectl folder created in the C – Drive.



Step 4: Write command from the Kubernetes website on command prompt to install kubectl on windows.
Command: curl.exe -LO https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe

```
C:\kubectl>curl.exe -LO "https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    167      0 --:--:-- --:--:-- --:--:--   168
 23 56.1M   23 13.3M    0     0  1808k      0  0:00:31  0:00:07  0:00:24 2079k
```

Step 5: Type “dir” after the above command to check the directory.

```
C:\kubectl>dir
 Volume in drive C is OS
 Volume Serial Number is 9ABC-B461

 Directory of C:\kubectl

25-02-2025  12:31    <DIR>          .
25-02-2025  12:32        58,857,472 kubectl.exe
               1 File(s)     58,857,472 bytes
               1 Dir(s)  38,700,539,904 bytes free
```

Step 6: Validate the binary by downloading the kubectl checksum file.
Command: curl.exe -LO https://dl.k8s.io/v1.32.0/bin/windows/amd64/kubectl.exe.sha256

```
C:\kubectl>curl.exe -LO "https://dl.k8s.io/v1.32.0/bin/windows/amd64/kubectl.exe.sha256"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    443      0 --:--:-- --:--:-- --:--:--   448
100    64  100    64    0     0    165      0 --:--:-- --:--:-- --:--:--   165
```

Step 7: Use Command Prompt to manually compare CertUtil's output to the checksum file downloaded.
Command: CertUtil -hashfile kubectl.exe SHA256

```
C:\kubectl>CertUtil -hashfile kubectl.exe SHA256
SHA256 hash of kubectl.exe:
3601cb47c4d6a42b033a8f8fca68bc6f24baa99f5a1250fdb138d24a6c7cc749
CertUtil: -hashfile command completed successfully.
```

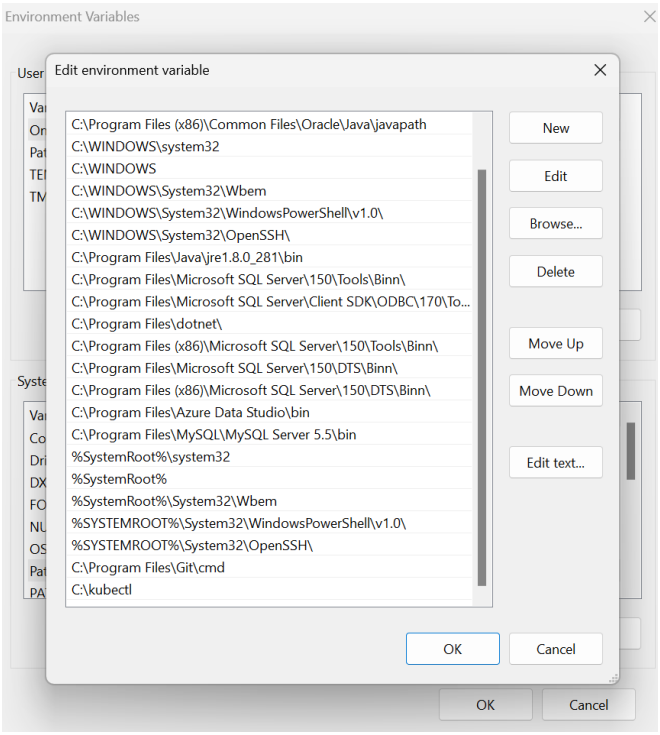
Step 8: Use PowerShell to automate the verification using the -eq operator to get a True or False result.
Command: $(Get-FileHash -Algorithm SHA256 .\kubectl.exe).Hash -eq $(Get-Content .\kubectl.exe.sha256)

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

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

PS C:\WINDOWS\system32> cd..
PS C:\WINDOWS> cd..
PS C:\> cd kubectl
PS C:\kubectl> $(Get-FileHash -Algorithm SHA256 .\kubectl.exe).Hash -eq $(Get-Content .\kubectl.exe.sha256)
True
PS C:\kubectl>
```

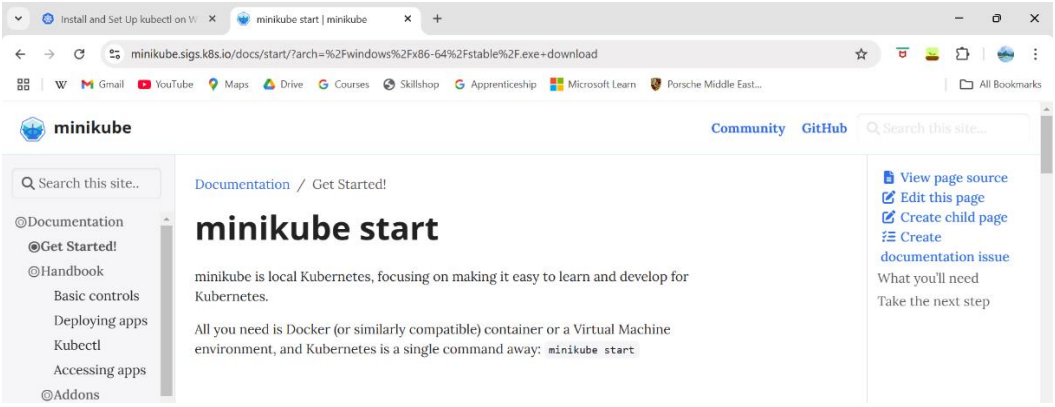
Step 9: Append or prepend the kubectl binary folder to your PATH environment variable.



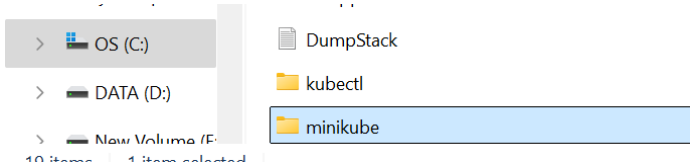
Step 10: Test to ensure the version of kubectl is the same as downloaded.
Command: kubectl version --client

```
C:\kubectl>kubectl version --client
Client Version: v1.32.0
Kustomize Version: v5.5.0
```

Step 11: After this, we have to install minikube from the website “minikube start”



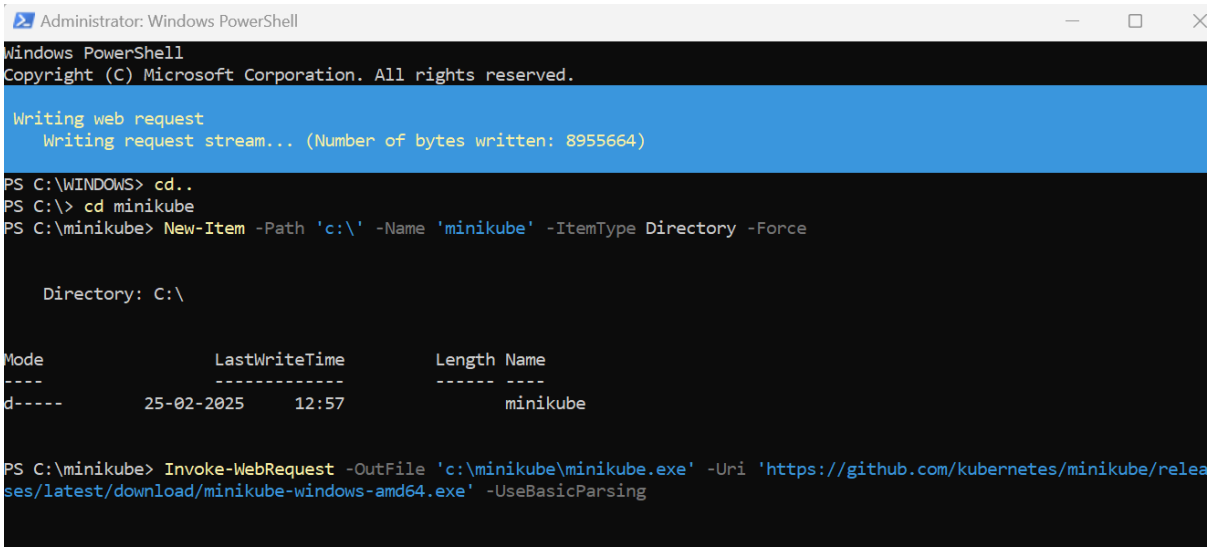
Step 12: Create a folder “minikube” in C – Drive.



Step 13: Open PowerShell and navigate to the minikube folder created

```
PS C:\WINDOWS\system32> cd..
PS C:\WINDOWS> cd..
PS C:\> cd minikube
PS C:\minikube>
```

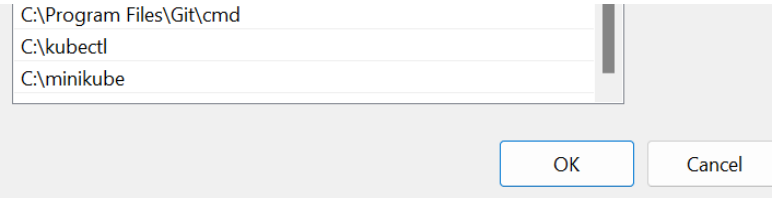
Step 14: Type the command to install minikube
Command: New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing



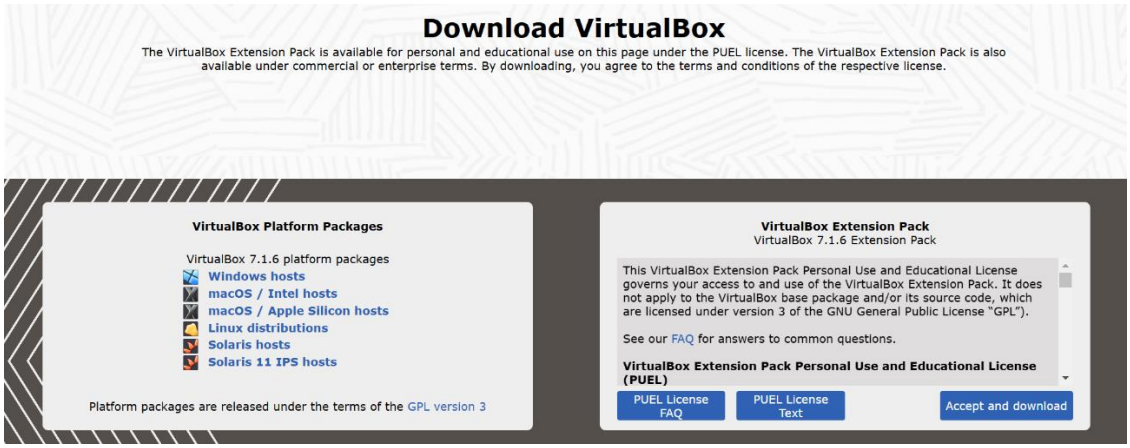
Step 15: Add the minikube.exe binary to your PATH
Command: $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine) if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)}

```
PS C:\minikube> $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
PS C:\minikube> if ($oldPath.Split(';') -inotcontains 'C:\minikube'){
>>   [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)
>> }
PS C:\minikube>
```

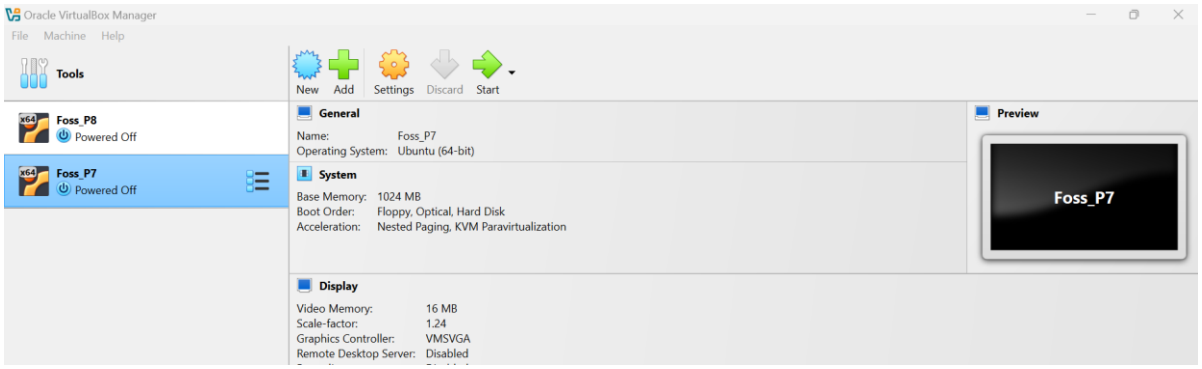
Check the system environment variable if the path is added or not.



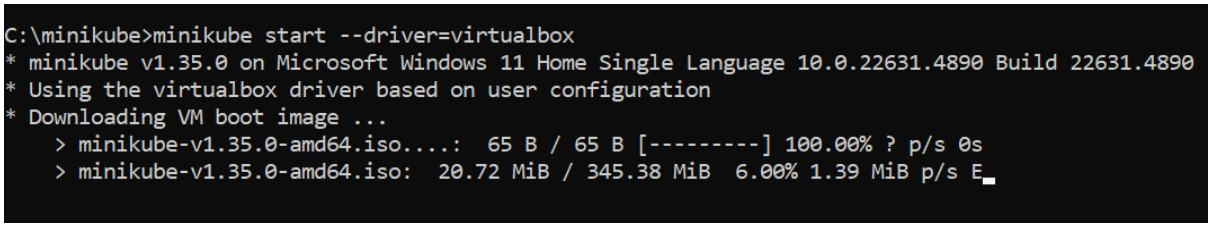
Step 16: One of the requirements of running minikube is installing VirtualBox. Install VirtualBox for Windows



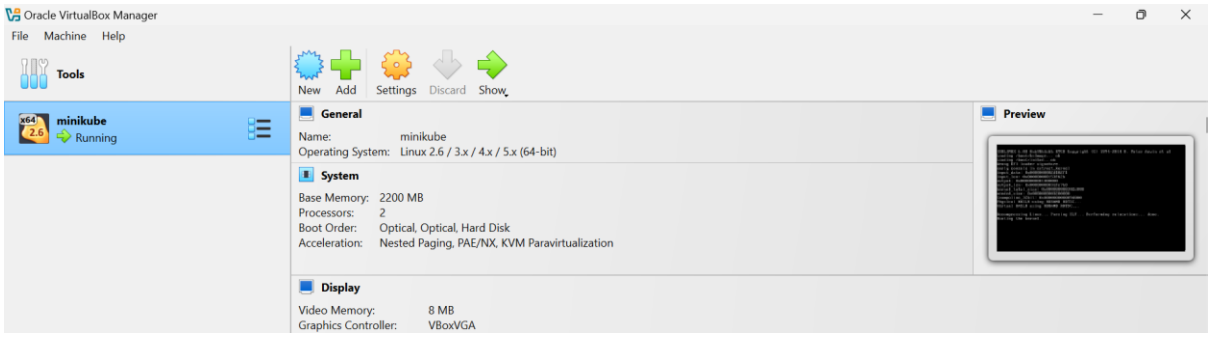
Step 17: Install it with all the default settings.



Step 18: Open Command Prompt as Admin. Start a cluster using the virtualbox driver by typing in the command.
Command: minikube start --driver=virtualbox



This command will start the minikube in the Oracle VirtulBox.



Step 19: Check minikube status if it is running or not
Command: minikube status

```
C:\minikube>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Step 20: Check the nodes if they are working or not. Navigate to kubectl folder
Command: kubectl get nodes

```
C:\>cd kubectl

C:\kubectl>kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   35m   v1.32.0
```

Step 21: type the command that lists all the pods running across all namespaces in your Kubernetes cluster.
Command: kubectl get po -A

```
C:\kubectl>kubectl get po -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-668d6bf9bc-xxl7g           1/1     Running   0             38m
kube-system   etcd-minikube                      1/1     Running   0             38m
kube-system   kube-apiserver-minikube            1/1     Running   0             38m
kube-system   kube-controller-manager-minikube   1/1     Running   0             38m
kube-system   kube-proxy-gnmdv                   1/1     Running   0             38m
kube-system   kube-scheduler-minikube            1/1     Running   0             38m
kube-system   storage-provisioner                1/1     Running   1 (38m ago)   38m
```

Step 22: Create a sample deployment and expose it on port 8080
Command: kubectl create deployment hello-minikube --image=kicbase/echo-server:1.0
kubectl expose deployment hello-minikube --type=NodePort --port=8080

The easiest way to access this service is to let minikube launch a web browser for you.
Command: minikube service hello-minikube

```
C:\kubectl>kubectl create deployment hello-minikube --image=kicbase/echo-server:1.0
deployment.apps/hello-minikube created

C:\kubectl>kubectl expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed

C:\kubectl>minikube service hello-minikube
|-----------|----------------|-------------|---------------------------|
| NAMESPACE |      NAME      | TARGET PORT |            URL            |
|-----------|----------------|-------------|---------------------------|
| default   | hello-minikube |        8080 | http://192.168.59.100:32516 |
|-----------|----------------|-------------|---------------------------|
* Opening service default/hello-minikube in default browser...
```

It takes you to the browser localhost requested by minikube

```
Request served by hello-minikube-ffcbb5874-52xdq

HTTP/1.1 GET /

Host: 192.168.59.100:32516
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
```

Step 23: Type the command to see if the pods are deployed or not.
Command: kubectl get pods

```
C:\kubectl>kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
hello-minikube-ffcbb5874-52xdq    1/1     Running   0          4m7s
```

Practical No. 05

Aim: Deploy the Kubernetes Cluster on Linux Ubuntu

Commands:
sudo apt-get update
sudo apt-get install -y apt-transport-https

```
haifa@haifa-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
haifa@haifa-VirtualBox:~$ sudo apt-get update
[sudo] password for haifa:
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [752 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [591 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [295 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 48x48 Icons [73.8 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 64x64 Icons [148 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages [982 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1,010 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [254 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 48x48 Icons [196 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 64x64 Icons [449 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:16 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [7,920 B]
Fetched 4,923 kB in 30s (164 kB/s)
Reading package lists... Done
haifa@haifa-VirtualBox:~$ sudo apt-get install -y apt-transport-https
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.6.12).
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
```

Command: sudo apt install docker.io

```
haifa@haifa-VirtualBox:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  bridge-utils cgroupfs-mount containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  aufs-tools btrfs-progs debootstrap docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount containerd docker.io git git-man liberror-perl pigz runc ubuntu-fan
0 upgraded, 10 newly installed, 0 to remove and 30 not upgraded.
Need to get 56.9 MB of archives.
After this operation, 291 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Commands:
sudo systemctl start docker
sudo systemctl enable docker

```
haifa@haifa-VirtualBox:~$ sudo systemctl start docker
haifa@haifa-VirtualBox:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
haifa@haifa-VirtualBox:~$
```

Command:
sudo apt-get install curl

```
haifa@haifa-VirtualBox:~$ sudo apt-get install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer req
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl libcurl4
0 upgraded, 2 newly installed, 0 to remove and 30 not upgraded.
Need to get 373 kB of archives.
After this operation, 1,038 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Step: Add key and repository. Change the permission of the file. Update the packages again.

```
haifa@haifa-VirtualBox:~$ sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg| sudo apt-key add
OK
haifa@haifa-VirtualBox:~$ sudo chmod 777 /etc/apt/sources.list.d/
haifa@haifa-VirtualBox:~$ cat /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
haifa@haifa-VirtualBox:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [38.5 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 48x48 Icons [17.6 kB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8,993 B]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 64x64 Icons [41.5 kB]
Get:9 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [29.9 kB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [42.1 kB]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/universe DEP-11 48x48 Icons [16.4 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/universe DEP-11 64x64 Icons [116 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Fetched 566 kB in 6s (97.6 kB/s)
Reading package lists... Done
```

Command:
sudo apt-get install -y kubelet kubectl kubernetes-cni

```
haifa@haifa-VirtualBox:~$ sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  conntrack cri-tools ebtables ethtool socat
The following NEW packages will be installed:
  conntrack cri-tools ebtables ethtool kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 9 newly installed, 0 to remove and 30 not upgraded.
Need to get 54.5 MB of archives.
After this operation, 292 MB of additional disk space will be used.
Get:2 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 conntrack amd64 1:1.4.4+snapshot20161117-6ubuntu2 [30.6
kB]
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.13.0-00 [8,776 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ebtables amd64 2.0.10.4-3.5ubuntu2.18.04.3 [79.9
 kB]
```

Commands:
sudo swapoff -a
sudo kubeadm init

```
haifa@haifa-VirtualBox:~$ sudo swapoff -a
haifa@haifa-VirtualBox:~$ sudo kubeadm init
[init] Using Kubernetes version: v1.16.1
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "
systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[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 in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
```

Step: Start the cluster by copy pasting the command

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

Step: Deploy the pods using the following commands

```
haifa@haifa-VirtualBox:~$ sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentatio
n/kube-flannel.yml
[sudo] password for haifa:
podsecuritypolicy.policy/psp.flannel.unprivileged configured
clusterrole.rbac.authorization.k8s.io/flannel configured
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
serviceaccount/flannel unchanged
configmap/kube-flannel-cfg unchanged
daemonset.apps/kube-flannel-ds-amd64 unchanged
daemonset.apps/kube-flannel-ds-arm64 unchanged
daemonset.apps/kube-flannel-ds-arm unchanged
daemonset.apps/kube-flannel-ds-ppc64le unchanged
daemonset.apps/kube-flannel-ds-s390x unchanged
haifa@haifa-VirtualBox:~$ sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentatio
n/k8s-manifests/kube-flannel-rbac.yml
clusterrole.rbac.authorization.k8s.io/flannel configured
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
haifa@haifa-VirtualBox:~$
```

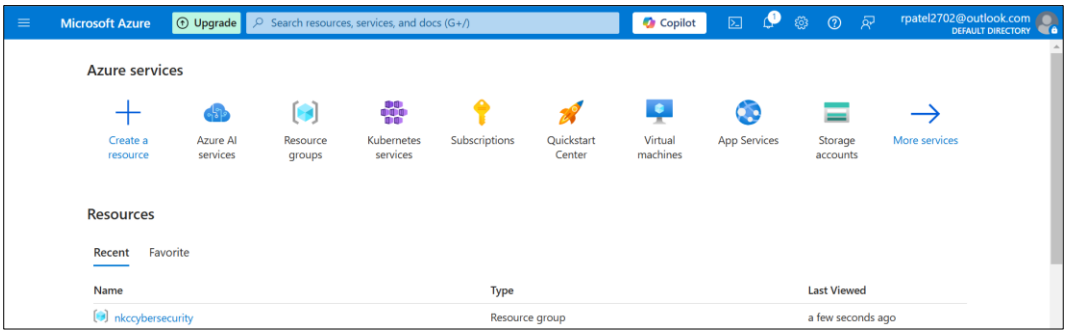
Command: sudo kubectl get pods –all-namespaces

```
haifa@haifa-VirtualBox:~$ sudo kubectl get pods --all-namespaces
NAMESPACE     NAME                                      READY   STATUS              RESTARTS   AGE
kube-system   coredns-5644d7b6d9-d8sc6                  0/1     ContainerCreating   0          9m40s
kube-system   coredns-5644d7b6d9-zlvhb                  0/1     ContainerCreating   0          9m40s
kube-system   etcd-haifa-virtualbox                     1/1     Running             0          8m30s
kube-system   kube-apiserver-haifa-virtualbox           1/1     Running             0          8m43s
kube-system   kube-controller-manager-haifa-virtualbox  1/1     Running             0          8m45s
kube-system   kube-flannel-ds-amd64-z75vk               0/1     CrashLoopBackOff    6          7m51s
kube-system   kube-proxy-x2bng                          1/1     Running             0          9m40s
kube-system   kube-scheduler-haifa-virtualbox           1/1     Running             0          8m32s
haifa@haifa-VirtualBox:~$
```

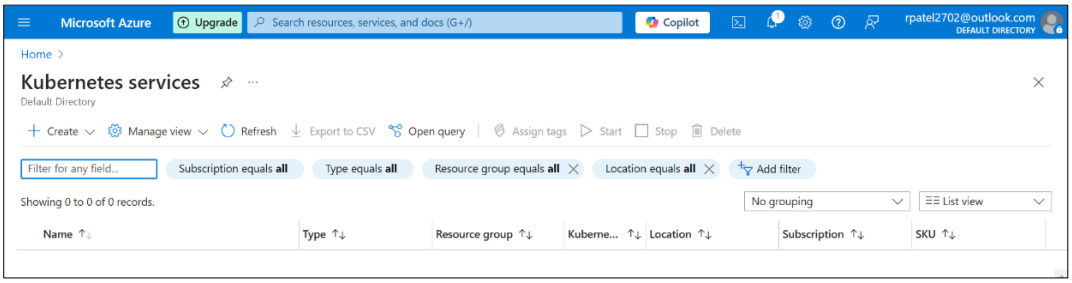
Practical No. 06

Aim: Deploy an Azure Kubernetes Service (AKS) cluster using Azure portal

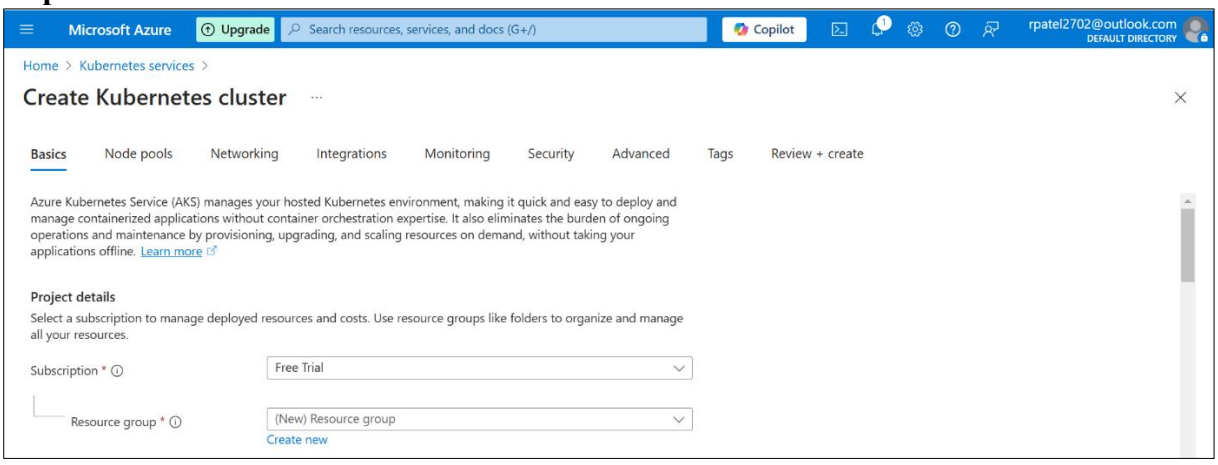
Step 1: Open Microsoft Azure Portal and sign in with your account.



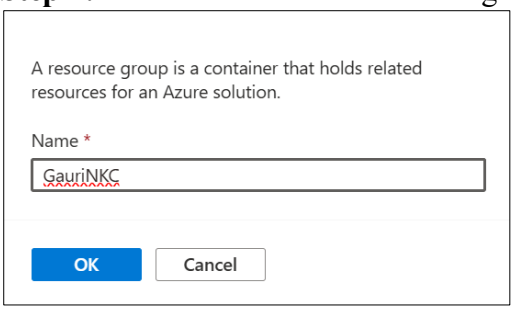
Step 2: Click on Kubernetes Services



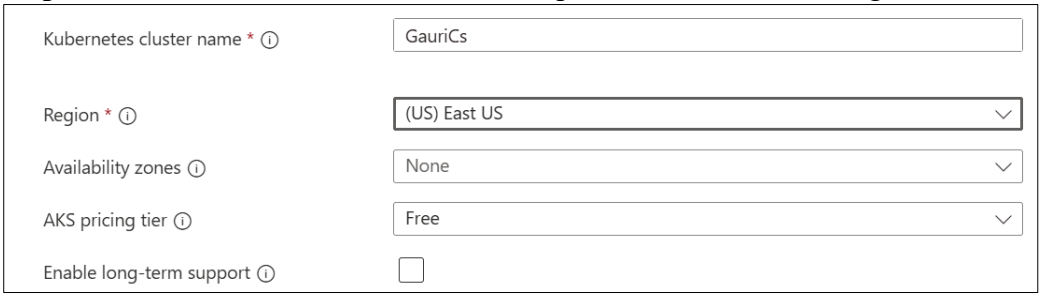
Step 3: Create a Kubernetes cluster.



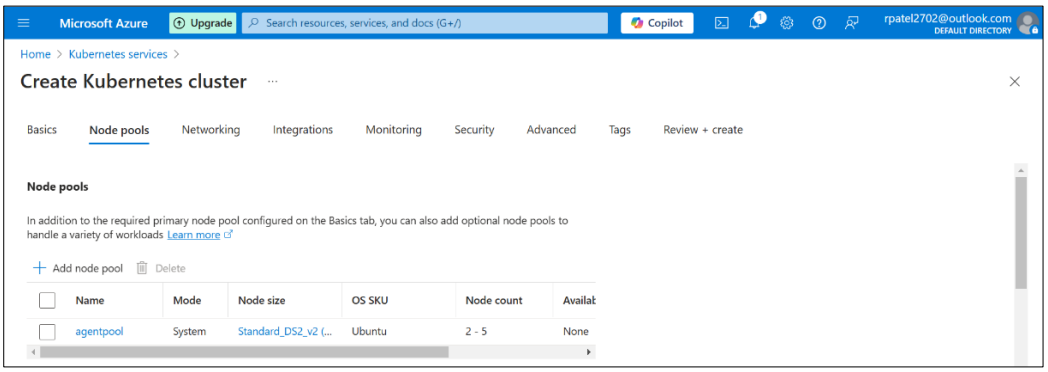
Step 4: Give a name for the resource group.



Step 5: Give a name for the cluster and keep the other default settings as it is.

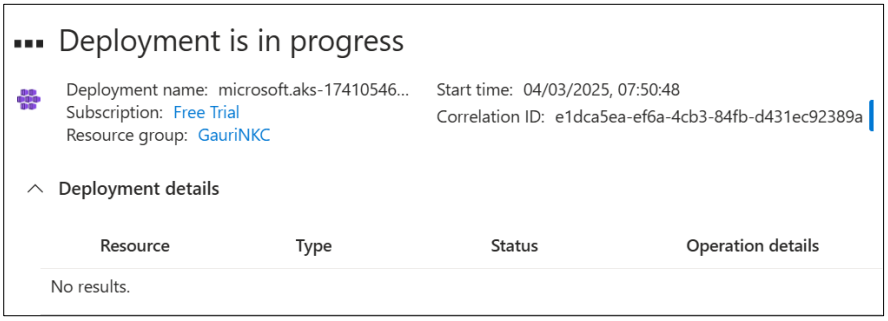


Step 6: Select next and keep the default settings as it is.

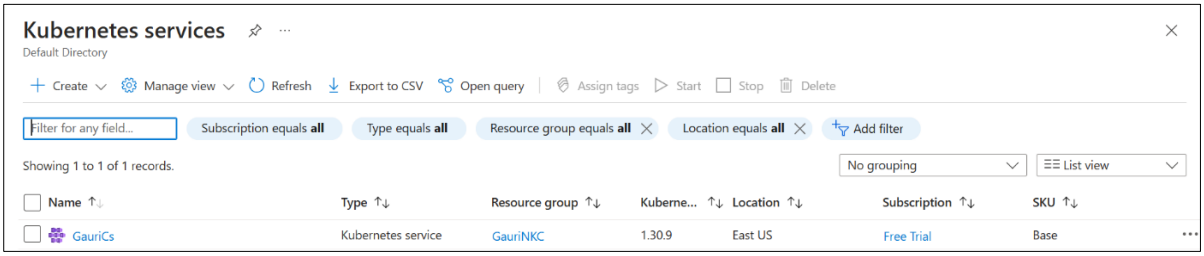


Step 7: Click on Review + create

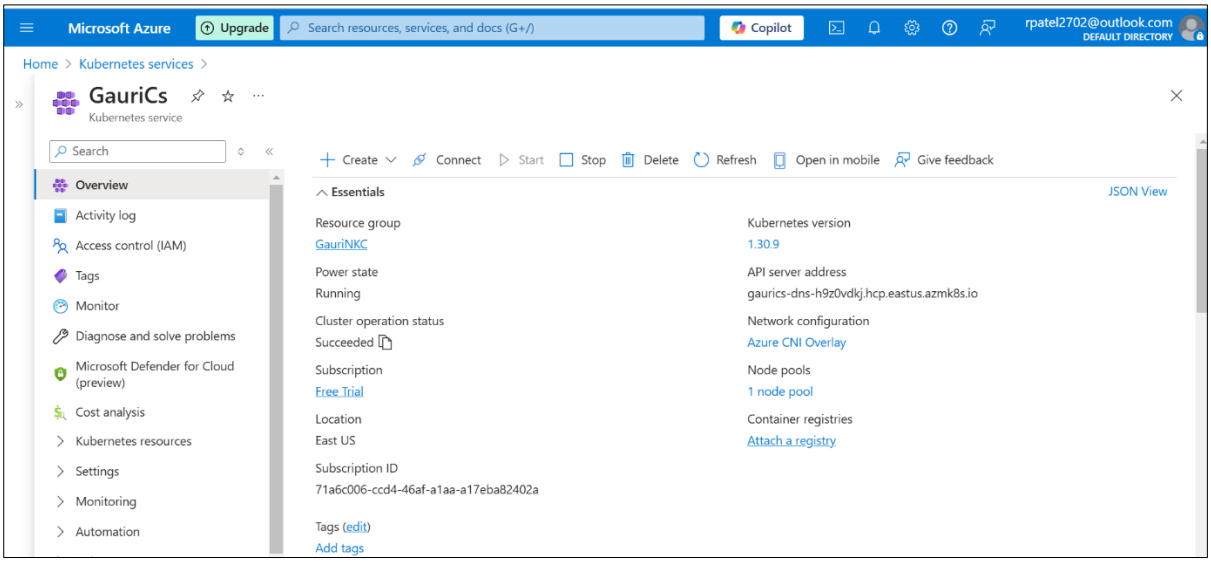
Step 8: The deployment will take some time.



Step 9: The Kubernetes cluster will be created.



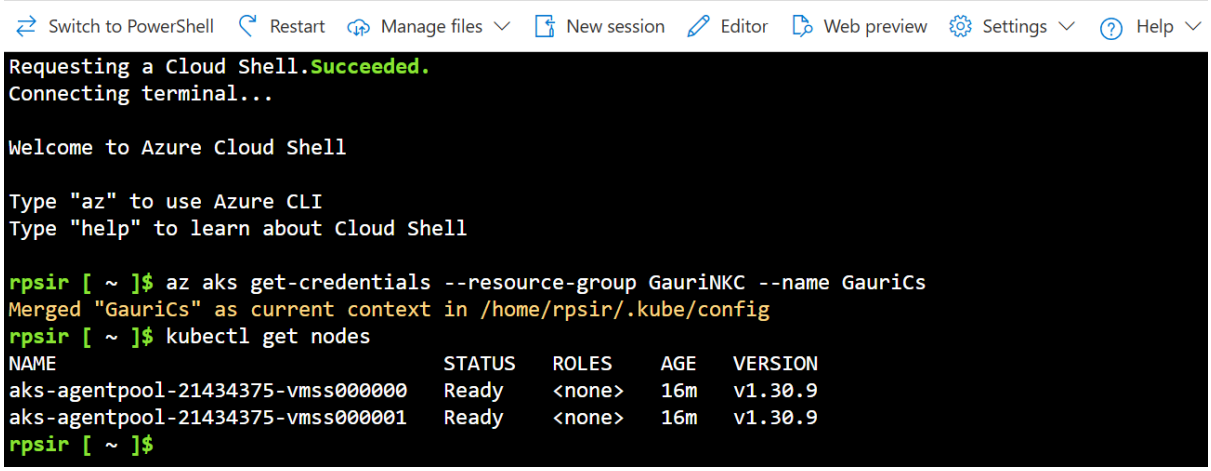
Step 10: Click on the Cluster name to get the details.



Step 11: Open the Cloud Shell and type in the commands:

az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
kubectl get nodes

Note: Provide your Resource Group name and Cluster name in the code.



Practical No. 07

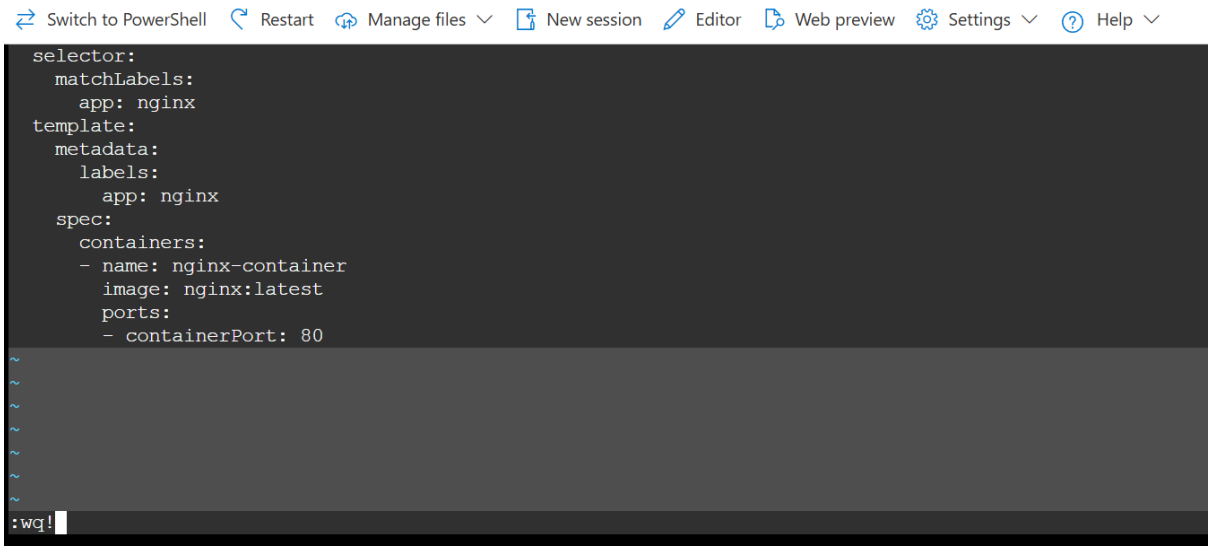
Aim: Kubernetes deployment using YAML – NGINX Containers

Step 1: Continue in the same Cloud Shell as the above practical. We will deploy a nginx pod using YAML.
Command: vi nginx-deployment.yaml

```
rpsir [ ~ ]$ az aks get-credentials --resource-group GauriNKC --name GauriCs
Merged "GauriCs" as current context in /home/rpsir/.kube/config
rpsir [ ~ ]$ kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
aks-agentpool-21434375-vmss000000   Ready    <none>   16m   v1.30.9
aks-agentpool-21434375-vmss000001   Ready    <none>   16m   v1.30.9
rpsir [ ~ ]$ vi nginx-deployment.yaml
```

Step 2: You will be redirect to this interface. Then type in this command:

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



Step 3: To exit the above interface type ":wq!". Deploy the application using the kubectl apply command and specify the name of your YAML manifest.

Command: kubectl apply -f nginx-deployment.yaml

```
rpsir [ ~ ]$ vi nginx-deployment.yaml
rpsir [ ~ ]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
rpsir [ ~ ]$
```

Step 4: Type the command to get the number of pods running.

Command: kubectl get pods

```
rpsir [ ~ ]$ vi nginx-deployment.yaml
rpsir [ ~ ]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
rpsir [ ~ ]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5449cb55b-cjj5c    1/1     Running   0          70s
nginx-deployment-5449cb55b-h65v9    1/1     Running   0          70s
nginx-deployment-5449cb55b-mmwd6    1/1     Running   0          70s
rpsir [ ~ ]$
```

Practical No. 8

Aim: Netflix's Challenges on AWS Cloud: A Case Study

INTRODUCTION:

Netflix, the world's leading streaming platform, delivers billions of hours of content to subscribers globally. The company migrated its infrastructure to Amazon Web Services (AWS) to ensure scalability, reliability, and global reach. However, running such a massive operation on AWS introduced significant challenges. This case study explores Netflix's migration to AWS, the challenges faced, and how the company overcame them.

BACKGROUND:

Netflix started as a DVD rental service but transitioned to a streaming platform in 2007. As its user base grew exponentially, the company faced infrastructure scalability issues. By 2010, Netflix decided to move its infrastructure to AWS to achieve high availability and elasticity. The migration was completed in 2016.

CHALLENGES FACED ON AWS:

1. Scalability and Performance:

Netflix's streaming service operates at an enormous scale, with peak traffic exceeding millions of concurrent users. Managing such a workload required auto-scaling solutions and efficient load balancing.

2. Latency and Global Content Delivery:

With users spread across different continents, content delivery with low latency was a significant challenge. AWS regions and availability zones had to be optimally configured.

3. Cost Optimization:

Operating on AWS comes with significant costs, particularly for data storage, compute resources, and data transfer. Optimizing resources while maintaining service quality was a priority.

4. Resilience and Disaster Recovery:

Ensuring 24/7 availability despite potential outages was critical. AWS outages could affect Netflix's services, requiring robust resilience strategies.

5. Security and Compliance:

Protecting user data and intellectual property from cyber threats while ensuring compliance with regional regulations like GDPR and CCPA was a crucial challenge.

SOLUTIONS IMPLEMENTED:

1. Auto-Scaling and Load Balancing

Netflix leveraged AWS Auto Scaling Groups, Elastic Load Balancing (ELB), and EC2 instances to manage traffic spikes dynamically.

2. Content Delivery with Amazon CloudFront and Open Connect

To reduce latency, Netflix used Amazon CloudFront and developed Open Connect, its own Content Delivery Network (CDN), ensuring efficient data caching and local delivery.

3. Cost Optimization Strategies

Netflix adopted AWS Reserved Instances, Spot Instances, and Compute Savings Plans to optimize costs. Additionally, the company used its own cost-monitoring tools.

4. Resilience through Chaos Engineering

Netflix introduced Chaos Monkey, a tool that randomly shuts down instances to test system resilience. Multi-region replication and backup strategies further strengthened availability.

5. Security Enhancements

Netflix implemented IAM roles, AWS Shield for DDoS protection, and encryption for data at rest and in transit. Compliance automation tools ensured regulatory adherence.

KEY TAKEAWAYS:

- Cloud migration helped Netflix scale and maintain reliability.
- Proactive monitoring and automation enhanced system resilience.
- Custom-built tools like Open Connect and Chaos Monkey improved efficiency.
- Security and compliance measures ensured data protection.

CONCLUSION:

Netflix's journey on AWS showcases how large-scale enterprises can leverage cloud computing to achieve scalability, reliability, and security. Despite challenges, innovative solutions enabled Netflix to build a robust streaming infrastructure that continues to evolve with emerging technologies.