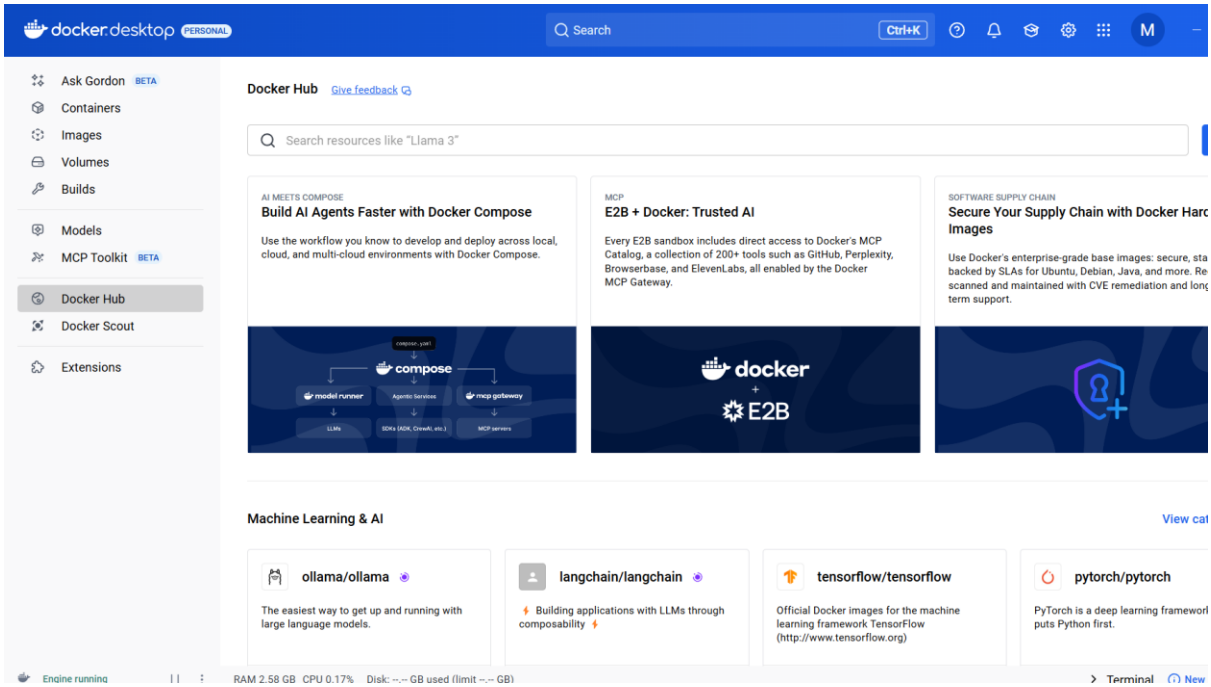
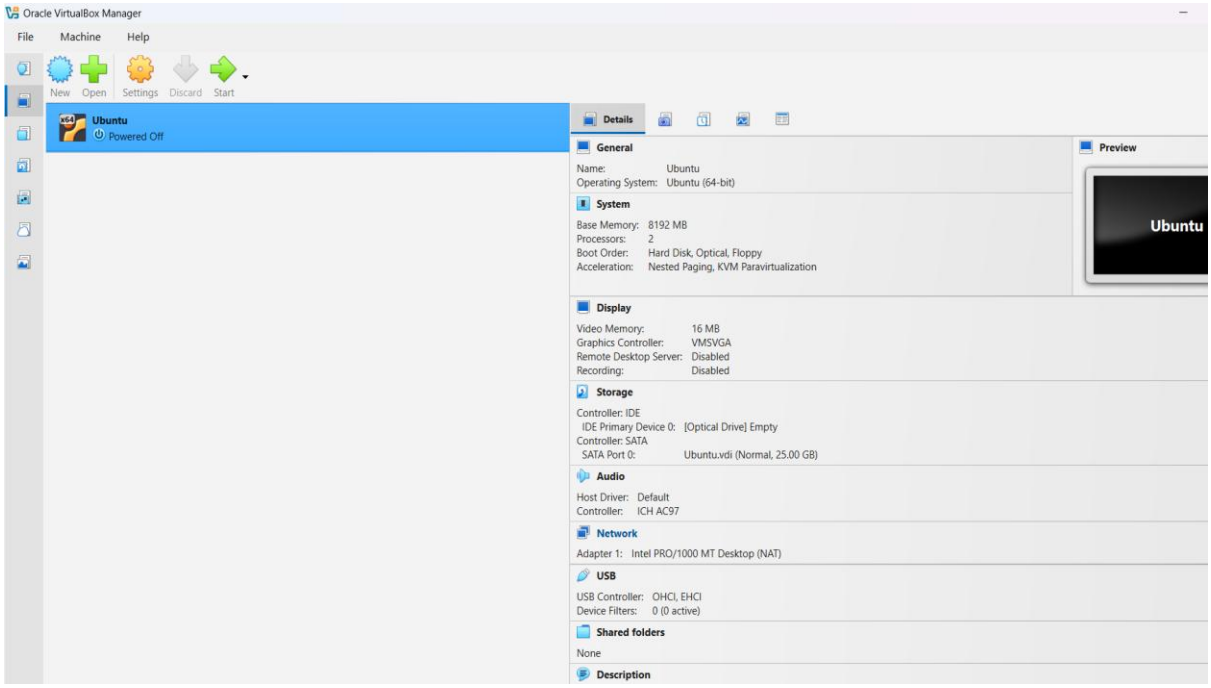
1. Setup Minikube in your local machine.

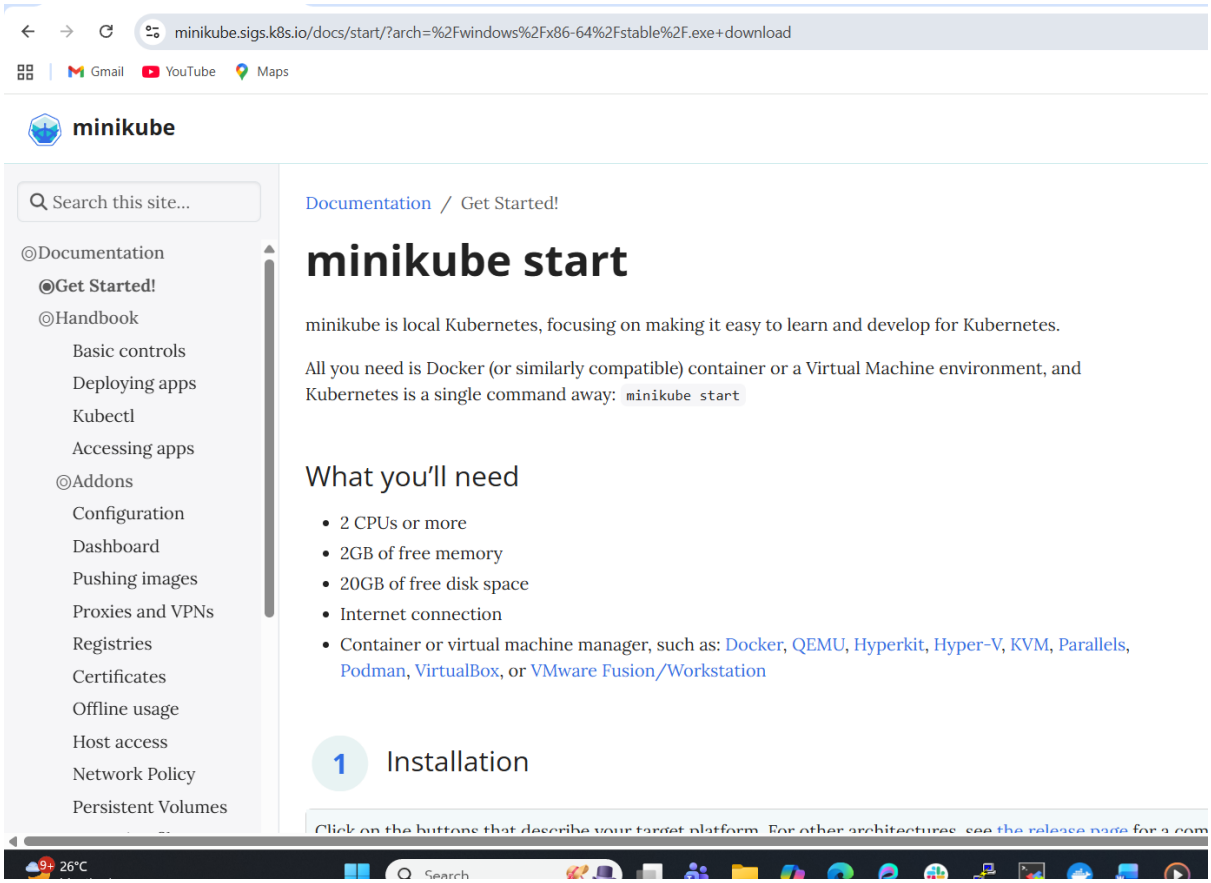
Install docker desktop in your local machine.



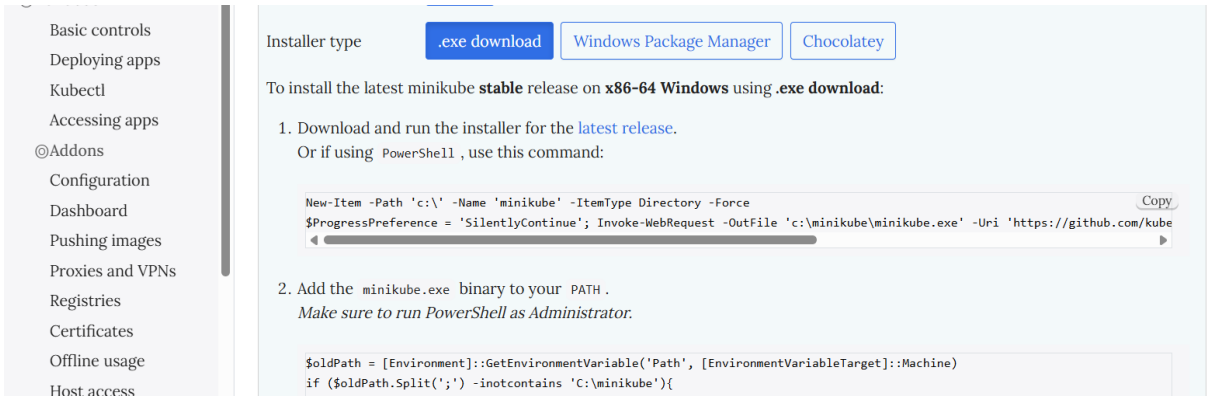
Install oracle virtualbox in your local machine.



Search on browser to install minikube in windows.

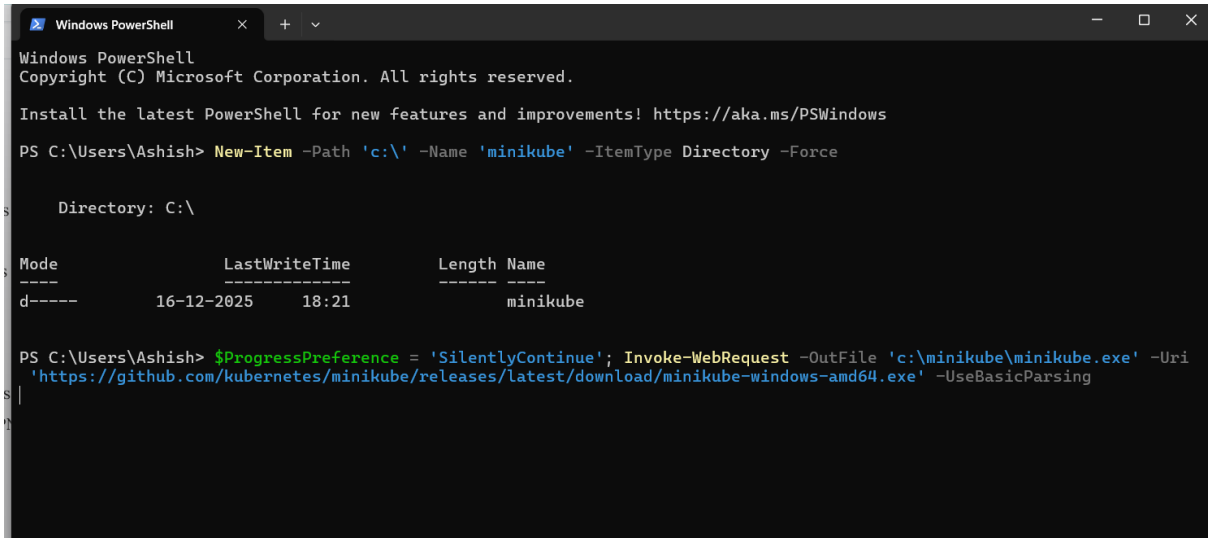


Copy this command this command and give this in powershell.

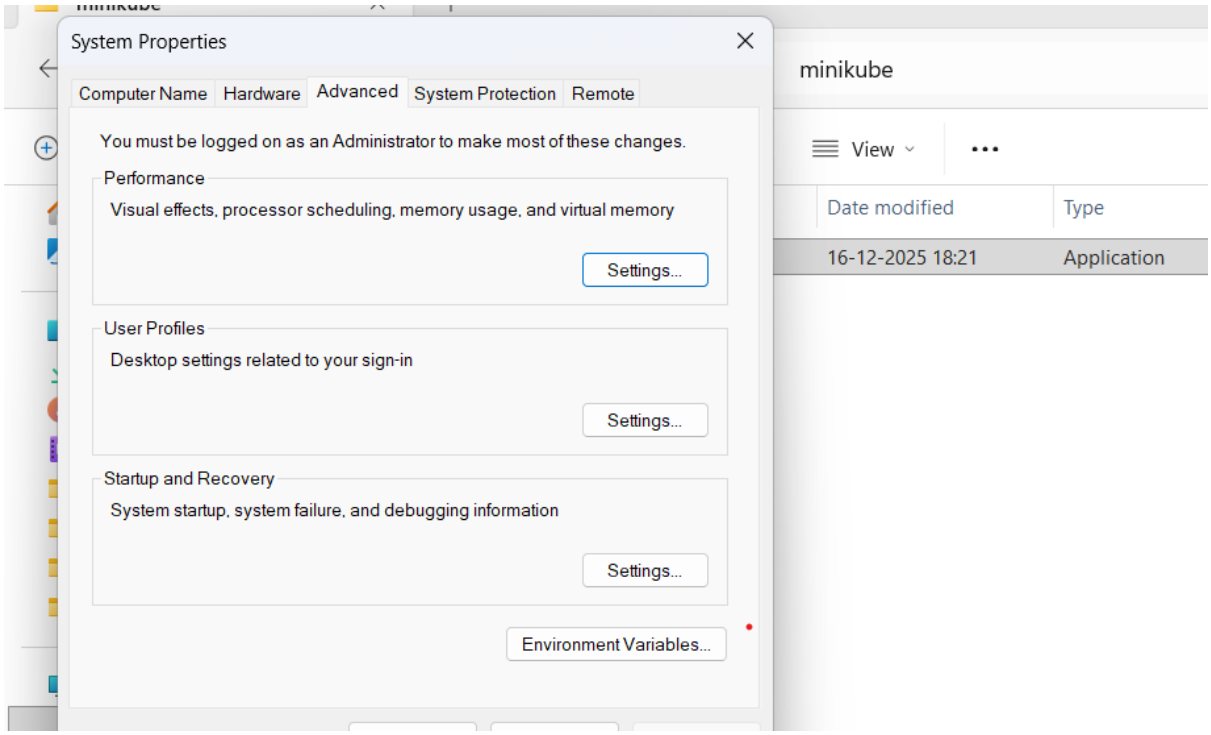


- **New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force$ProgressPreference = 'SilentlyContinue'; Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/lat**

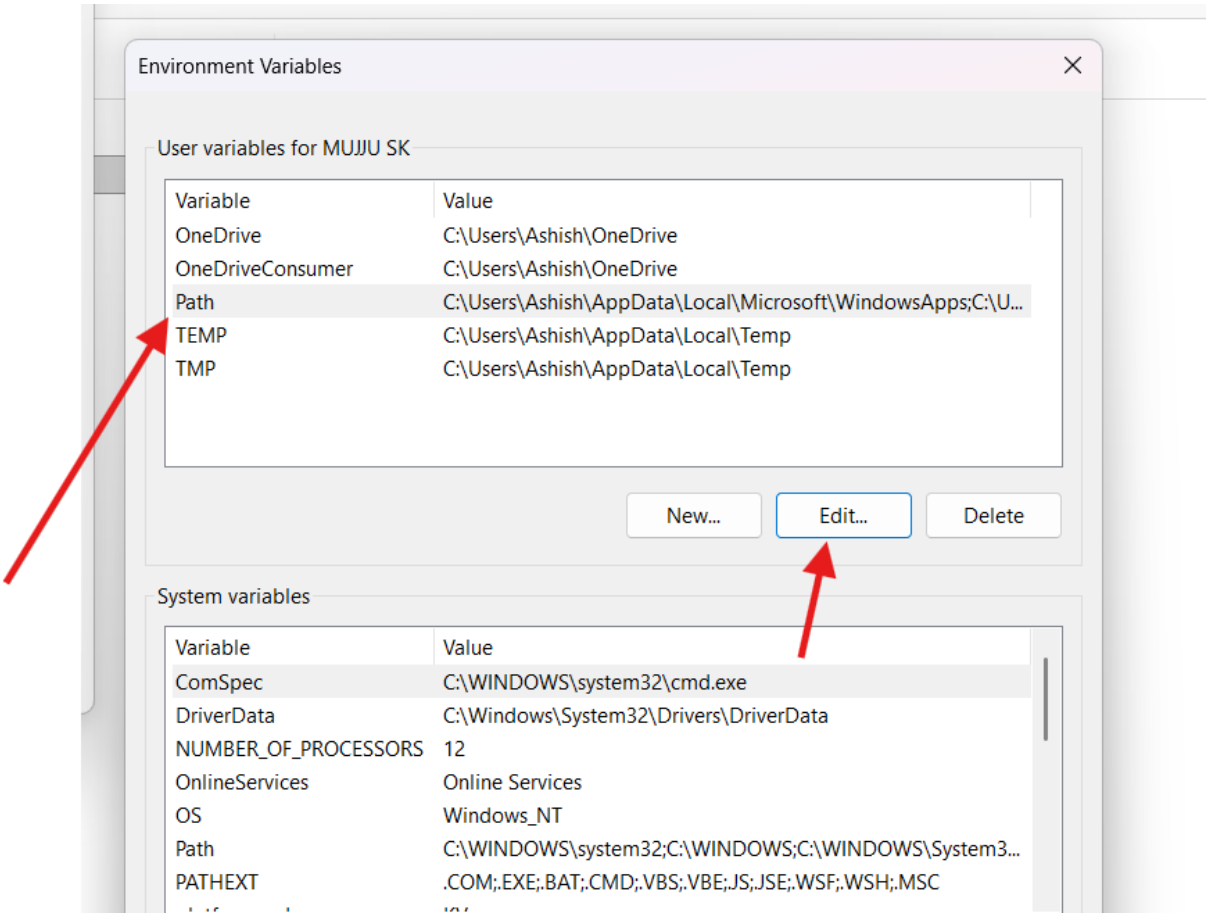
est/download/minikube-windows-amd64.exe' -UseBasicParsing



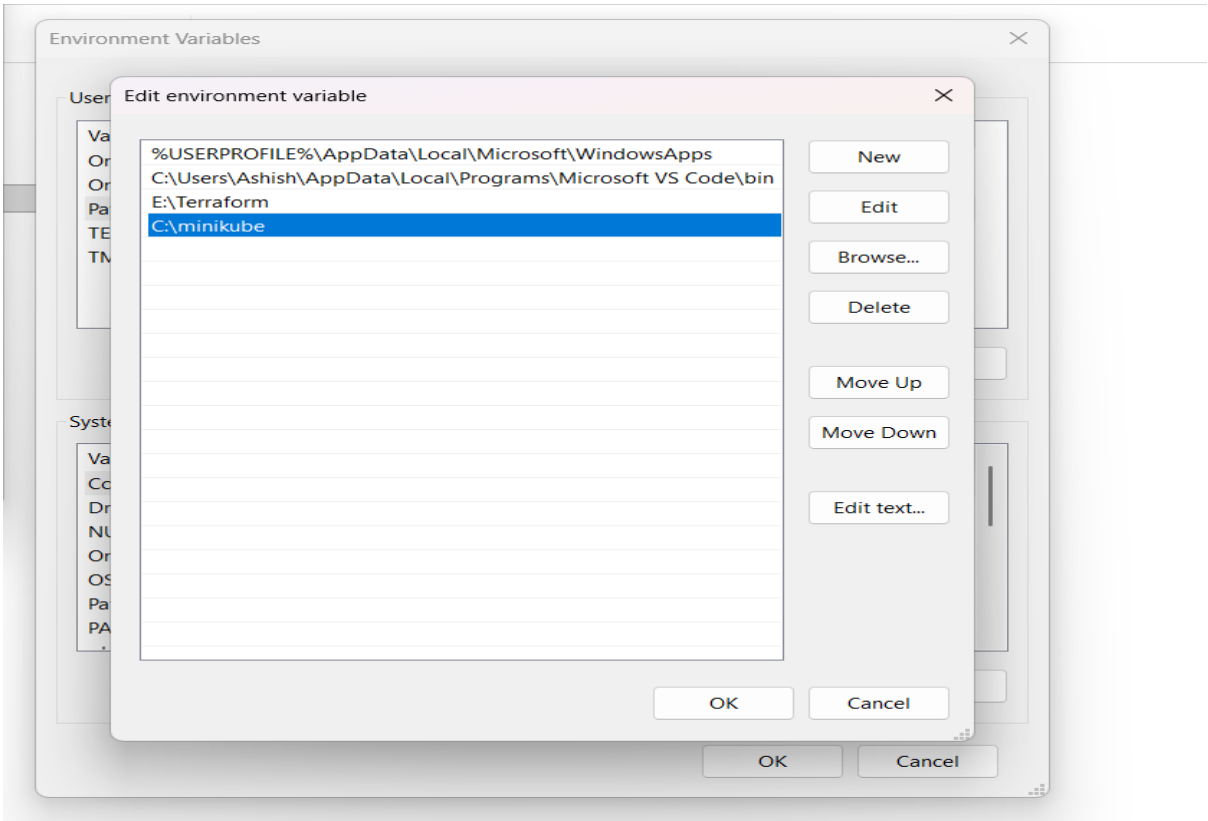
Go to environmental variables and copy the path of minikube and paste in the path



Click on edit.



Paste the copied path.



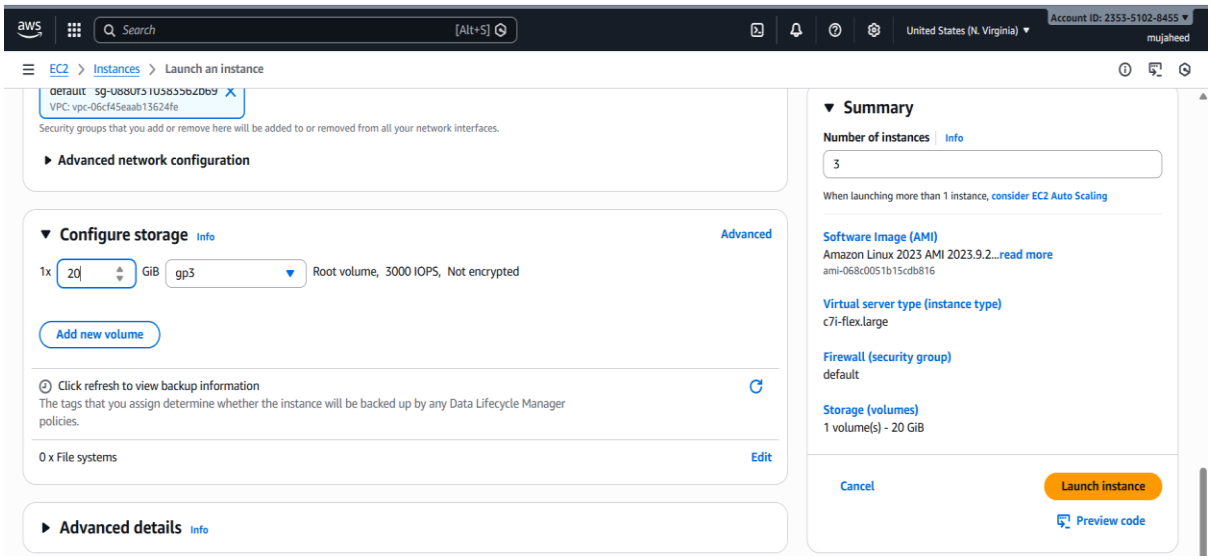
Go to visual studio and execute this command.

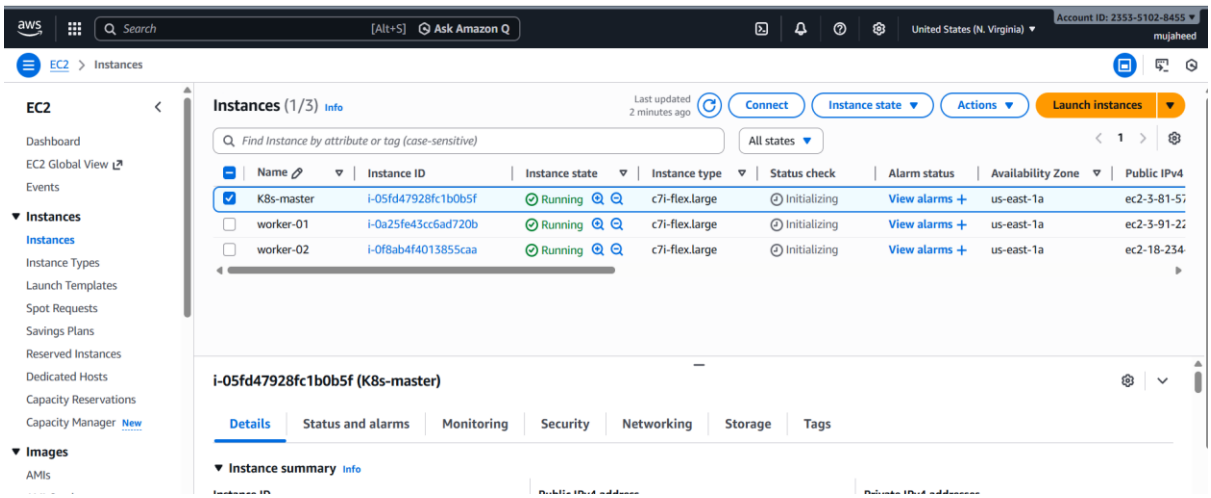
- **minikube start**

```
PS C:\Users\Ashish\Desktop\Terraform basics> minikube start
😄  minikube v1.37.0 on Microsoft Windows 11 Home Single Language 10.0.26200.7462 Build 26200.7462
✨  Automatically selected the docker driver. Other choices: virtualbox, ssh
📌  Using Docker Desktop driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.48 ...
💾  Downloading Kubernetes v1.34.0 preload ...
    > preloaded-images-k8s-v18-v1...:  337.07 MiB / 337.07 MiB  100.00% 9.76 Mi
    > gcr.io/k8s-minikube/kicbase...:  488.52 MiB / 488.52 MiB  100.00% 10.52 M
🔥  Creating docker container (CPUs=2, Memory=3900MB) ...
❗  Failing to connect to https://registry.k8s.io/ from inside the minikube container
💡  To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass

❗  C:\Program Files\Docker\Docker\resources\bin\kubectl.exe is version 1.32.2, which may have incompatibilities with Kubernetes 1.34.0.
    ▪ Want kubectl v1.34.0? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\Ashish\Desktop\Terraform basics> minikube ip
192.168.49.2
PS C:\Users\Ashish\Desktop\Terraform basics>
```

2. Setup k8s master and two worker nodes on ubuntu.

Launch 3 instances with c7 large and memory of 20gb.





Login to master instance.

- **hostname master**
- **echo $ hostname**
- **vi /etc/hostname**

```
[root@ip-172-31-24-182 ~]# hostname master
[root@ip-172-31-24-182 ~]# echo $hostname

[root@ip-172-31-24-182 ~]# vi /etc/hostname
[root@ip-172-31-24-182 ~]#
```

Here we are replacing our ipaddress as master for our identification.

```
master
```

Logout and login again you will get like this.

```
[root@master ~]#
```

Now do for same like this for worker nodes for understanding purpose.

```
[root@ip-172-31-27-50 ~]# hostname worker-01
[root@ip-172-31-27-50 ~]# echo $hostname

[root@ip-172-31-27-50 ~]# vi /etc/hostname
[root@ip-172-31-27-50 ~]#
```

```
worker-01
```

```
[root@worker-01 ~]#
```

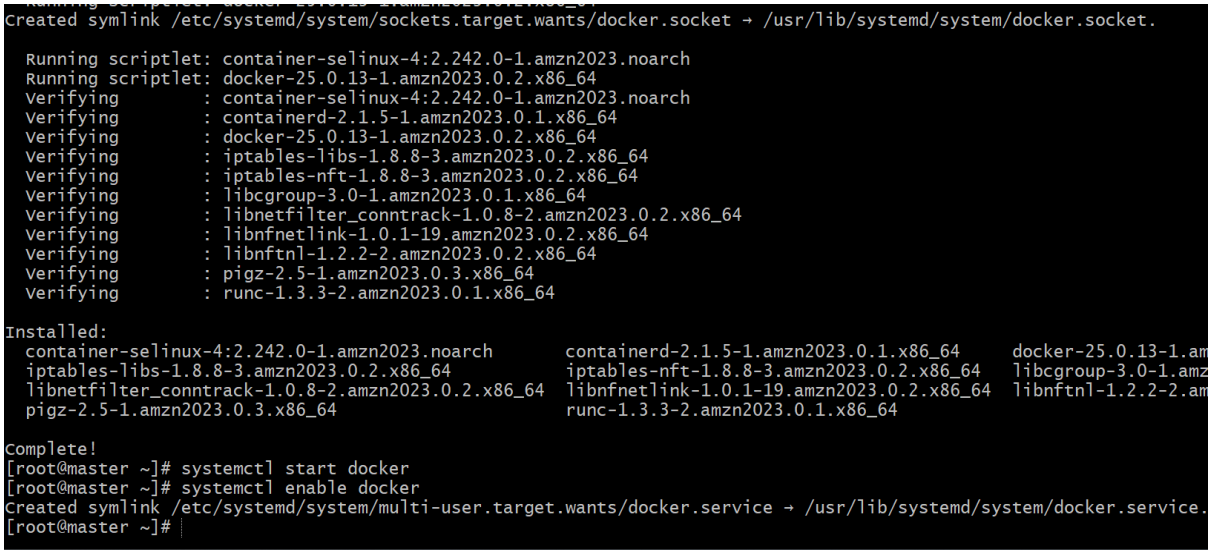
```
[root@ip-172-31-72-26 ~]# hostname worker-02
[root@ip-172-31-72-26 ~]# echo $hostname

[root@ip-172-31-72-26 ~]# vi /etc/hostname
[root@ip-172-31-72-26 ~]#
```

```
worker-02
```

```
[root@worker-02 ~]#
```

Install docker in all 3 servers.

- **yum install docker -y**
- **systemctl start docker**
- **systemctl enable docker**



Search in browser as kubeadm and install this command in 3 servers.

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/

enabled=1

gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key

exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

EOF

```
[root@master ~]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@master ~]#
```

```
[root@worker-01 ~]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@worker-01 ~]#
```

```
[root@worker-02 ~]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@worker-02 ~]#
```

Paste this command in 3 servers,

- **sudo setenforce 0**

- **sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config**

```
[root@master ~]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@master ~]#
```

```
[root@worker-01 ~]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@worker-01 ~]#
```

```
[root@worker-02 ~]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@worker-02 ~]#
```

Install this in 3 servers.

- **sudo yum install -y kubelet kubeadm kubectl --disableexcludes=Kubernetes**

```
[root@master ~]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes
Dependencies resolved.
================================================================================
 Package                 Architecture            Version
================================================================================
Installing:
 kubeadm                 x86_64                  1.34.3-150500.1.1
 kubectl                 x86_64                  1.34.3-150500.1.1
 kubelet                 x86_64                  1.34.3-150500.1.1
Installing dependencies:
 cri-tools               x86_64                  1.34.0-150500.1.1
 kubernetes-cni          x86_64                  1.7.1-150500.1.1

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 52 M
Installed size: 282 M
Downloading Packages:
(1/5): cri-tools-1.34.0-150500.1.1.x86_64.rpm
(2/5): kubectl-1.34.3-150500.1.1.x86_64.rpm
(3/5): kubeadm-1.34.3-150500.1.1.x86_64.rpm
(4/5): kubernetes-cni-1.7.1-150500.1.1.x86_64.rpm
(5/5): kubelet-1.34.3-150500.1.1.x86_64.rpm
--------------------------------------------------------------------------------
```

```
[root@worker-01 ~]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes
Last metadata expiration check: 0:00:01 ago on Tue Dec 16 16:11:07 2025.
Dependencies resolved.
================================================================================
 Package                    Architecture          Version
================================================================================
Installing:
 kubeadm                    x86_64                1.34.3-150500.1.1
 kubectl                    x86_64                1.34.3-150500.1.1
 kubelet                    x86_64                1.34.3-150500.1.1
Installing dependencies:
 cri-tools                  x86_64                1.34.0-150500.1.1
 kubernetes-cni             x86_64                1.7.1-150500.1.1

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 52 M
Installed size: 282 M
Downloading Packages:
(1/5): cri-tools-1.34.0-150500.1.1.x86_64.rpm
(2/5): kubeadm-1.34.3-150500.1.1.x86_64.rpm
(3/5): kubectl-1.34.3-150500.1.1.x86_64.rpm
(4/5): kubelet-1.34.3-150500.1.1.x86_64.rpm
(5/5): kubernetes-cni-1.7.1-150500.1.1.x86_64.rpm
--------------------------------------------------------------------------------
Total
Kubernetes
Importing GPG key 0x9A296436:
 Userid     : "isv:kubernetes OBS Project <isv:kubernetes@build.opensuse.org>"
 Fingerprint: DE15 B144 86CD 377B 9E87 6E1A 2346 54DA 9A29 6436
 From       : https://pkgs.k8s.io/core:/stable:/v1.34/rpm/repodata/repomd.xml.key
```

```
[root@worker-02 ~]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Kubernetes
Last metadata expiration check: 0:00:01 ago on Tue Dec 16 16:11:17 2025.
Dependencies resolved.
================================================================================
 Package                    Architecture          Version
================================================================================
Installing:
 kubeadm                    x86_64                1.34.3-150500.1.1
 kubectl                    x86_64                1.34.3-150500.1.1
 kubelet                    x86_64                1.34.3-150500.1.1
Installing dependencies:
 cri-tools                  x86_64                1.34.0-150500.1.1
 kubernetes-cni             x86_64                1.7.1-150500.1.1

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 52 M
Installed size: 282 M
Downloading Packages:
(1/5): kubeadm-1.34.3-150500.1.1.x86_64.rpm
(2/5): cri-tools-1.34.0-150500.1.1.x86_64.rpm
(3/5): kubectl-1.34.3-150500.1.1.x86_64.rpm
(4/5): kubernetes-cni-1.7.1-150500.1.1.x86_64.rpm
(5/5): kubelet-1.34.3-150500.1.1.x86_64.rpm
--------------------------------------------------------------------------------
Total
Kubernetes
Importing GPG key 0x9A296436:
```

Enable kubeadm in 3 servers.

- **sudo systemctl enable --now kubelet**

```
[root@master ~]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@master ~]#
```

```
[root@worker-01 ~]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@worker-01 ~]#
```

```
[root@worker-02 ~]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@worker-02 ~]#
```

Give pod range as like this command only in master machine.

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

```
[root@master ~]# kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.34.3
[preflight] Running pre-flight checks
        [WARNING Hostname]: hostname "master" could not be reached
        [WARNING Hostname]: hostname "master": lookup master on 172.31.0.2:53:
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your in
[preflight] You can also perform this action beforehand using 'kubeadm config
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.
ster.local master] and IPs [10.96.0.1 172.31.24.182]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] an
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
```

```
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the clus
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 172.31.24.182:6443 --token zy1ouh.pvmh5lkesury66hr \
        --discovery-token-ca-cert-hash sha256:f5c8ca6bbb7f37d1275ed68c3f8fab64700287379194a0655b6ce4dc29e70d19
[root@master ~]#
```

Execute this command in master machine.

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

```
[root@master ~]# mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@master ~]#
```

Execute this command in 3 servers to join into the cluster.

- **kubeadm join 172.31.24.182:6443 --token zy1ouh.pvmh5lkesury66hr \ --discovery-token-ca-cert-hash sha256:f5c8ca6bbb7f37d1275ed68c3f8fab64700287379194a0655b6ce4dc29e70d19**

```
[root@master ~]# kubeadm join 172.31.24.182:6443 --token zy1ouh.pvmh5lkesury66hr \
        --discovery-token-ca-cert-hash sha256:f5c8ca6bbb7f37d1275ed68c3f8fab64700287379194a0655b6ce4dc29e70d19
[preflight] Running pre-flight checks
        [WARNING Hostname]: hostname "master" could not be reached
        [WARNING Hostname]: hostname "master": lookup master on 172.31.0.2:53: no such host
[preflight] Some fatal errors occurred:
        [ERROR FileAvailable--etc-kubernetes-kubelet.conf]: /etc/kubernetes/kubelet.conf already exists
        [ERROR Port-10250]: Port 10250 is in use
        [ERROR FileAvailable--etc-kubernetes-pki-ca.crt]: /etc/kubernetes/pki/ca.crt already exists
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
error: error execution phase preflight: preflight checks failed
To see the stack trace of this error execute with --v=5 or higher
```

```
[root@worker-01 ~]# kubeadm join 172.31.24.182:6443 --token zy1ouh.pvmh5lkesury66hr \
        --discovery-token-ca-cert-hash sha256:f5c8ca6bbb7f37d1275ed68c3f8fab64700287379194a0655b6ce4dc29e70d19
[preflight] Running pre-flight checks
        [WARNING Hostname]: hostname "worker-01" could not be reached
        [WARNING Hostname]: hostname "worker-01": lookup worker-01 on 172.31.0.2:53: no such host
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config kubeadm --config your-config-file' to re-upload it.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/instance-config.yaml"
[patches] Applied patch of type "application/strategic-merge-patch+json" to target "kubeletconfiguration"
[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 501.684629ms
[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.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

```
[root@worker-02 ~]# kubeadm join 172.31.24.182:6443 --token zy1ouh.pvmh5lkesury66hr \
        --discovery-token-ca-cert-hash sha256:f5c8ca6bbb7f37d1275ed68c3f8fab64700287379194a0655b6ce4dc29e70d19
[preflight] Running pre-flight checks
        [WARNING Hostname]: hostname "worker-02" could not be reached
        [WARNING Hostname]: hostname "worker-02": lookup worker-02 on 172.31.0.2:53: no such host
[preflight] Reading configuration from the "kubeadm-config" ConfigMap in namespace "kube-system"...
[preflight] Use 'kubeadm init phase upload-config kubeadm --config your-config-file' to re-upload it.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/instance-config.yaml"
[patches] Applied patch of type "application/strategic-merge-patch+json" to target "kubeletconfiguration"
[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 502.34032ms
[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.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

[root@worker-02 ~]#
```

Go to master node and see the nodes.

- **kubectl get nodes**

```
[root@master ~]# kubectl get nodes
NAME        STATUS     ROLES           AGE     VERSION
master      NotReady   control-plane   15m     v1.34.3
worker-01   NotReady   <none>          2m34s   v1.34.3
worker-02   NotReady   <none>          2m27s   v1.34.3
[root@master ~]#
```

To make status ready execute this command in master machine.

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

```
[root@master ~]# 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
[root@master ~]#
```

- **kubectl get nodes**

```
[root@master ~]# kubectl get nodes
NAME        STATUS   ROLES           AGE     VERSION
master      Ready    control-plane   18m     v1.34.3
worker-01   Ready    <none>          5m40s   v1.34.3
worker-02   Ready    <none>          5m33s   v1.34.3
[root@master ~]#
```

3. Run one nginx pod.

Excute this command in master machine.

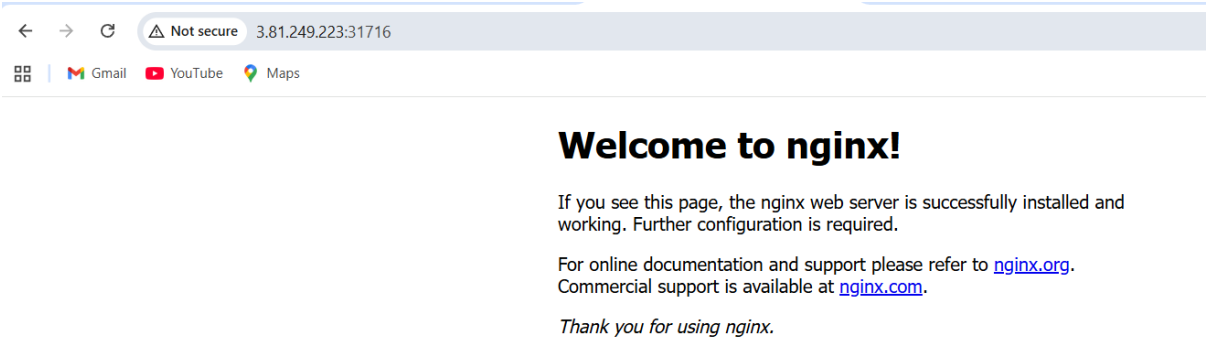
- **kubectl run firstpod --image=nginx**

```
[root@master ~]# kubectl run firstpod  --image=nginx
pod/firstpod created
[root@master ~]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          35s
[root@master ~]#
```

- **kubectl expose pod firstpod --port=80 --type=NodePort**
- **kubectl get svc firstpod**

```
[root@master ~]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          35s
[root@master ~]# kubectl expose pod firstpod --port=80 --type=NodePort
service/firstpod exposed
[root@master ~]# kubectl get svc firstpod
NAME       TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
firstpod   NodePort   10.96.158.54   <none>        80:31716/TCP   20s
[root@master ~]#
```

copy master public ip and port number 31716



4. Mug up Master and slave components on k8s.

MASTER (Control Plane) – "Brain"

4 MAIN COMPONENTS

1. API Server

- Entry point to Kubernetes
- All commands go through this

Think: Gate / Front door

2. etcd

- Stores all cluster data
- Pods, nodes, configs, secrets

Think: Database / Memory

3. Scheduler

- Decides which node a pod runs on

Think: Pod → Node allocator

4. Controller Manager

Kubernetes Task - 01

- Keeps desired state = actual state
- Creates/deletes pods if needed

Think: Auto-corrector

MASTER IN ONE-LINE MUG UP

API - ETCD - Scheduler - Controller

SLAVE / WORKER NODE – “Doer”

3 MAIN COMPONENTS

1. kubelet

- Runs on every node
- Talks to master
- Starts pods

Think: Node manager

2. Container Runtime

- Runs containers
- Example: containerd, CRI-O

Think: Engine

3. kube-proxy

- Handles networking
- Service → Pod traffic

Think: Traffic police

WORKER IN ONE-LINE MUG UP

Kubelet – Runtime – Kube-proxy