K8s 01

1) Setup Minikube in your local machine.

→Follow the below steps to set up Minikube:

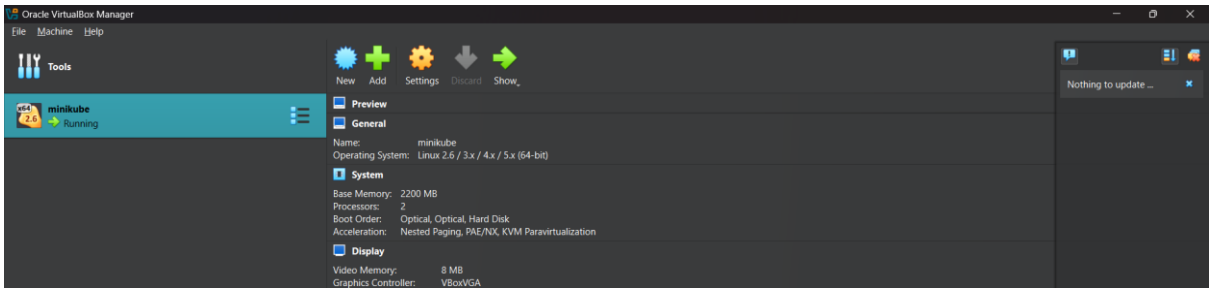
1) **Install Docker on windows.**

```
naren@narendar MINGW64 ~ (master)
$ docker --version
Docker version 28.0.1, build 068a01e
```

https://docs.docker.com/desktop/install/windows-install/

2) **Install Oracle box on windows**

https://adamtheautomator.com/install-virtualbox-on-windows-10/



3) **Check system info in CMD**

Systeminfo

```
C:\Users\naren>systeminfo

Host Name:                     NARENDAR
OS Name:                       Microsoft Windows 11 Home Single Language
OS Version:                    10.0.26100 N/A Build 26100
OS Manufacturer:               Microsoft Corporation
```

```
Virtualization-based security: Status: Running
                               Required Security Properties:
                               Available Security Properties:
                                     Base Virtualization Support
                                     Secure Boot
                                     DMA Protection
                                     UEFI Code Readonly
                                     Mode Based Execution Control
                                     APIC Virtualization
                               Services Configured:
                               Services Running:
                               App Control for Business policy: Enforced
                               App Control for Business user mode policy: Off
                               Security Features Enabled:
Hyper-V Requirements:          A hypervisor has been detected. Features required for Hyper-V will not be displayed.
```

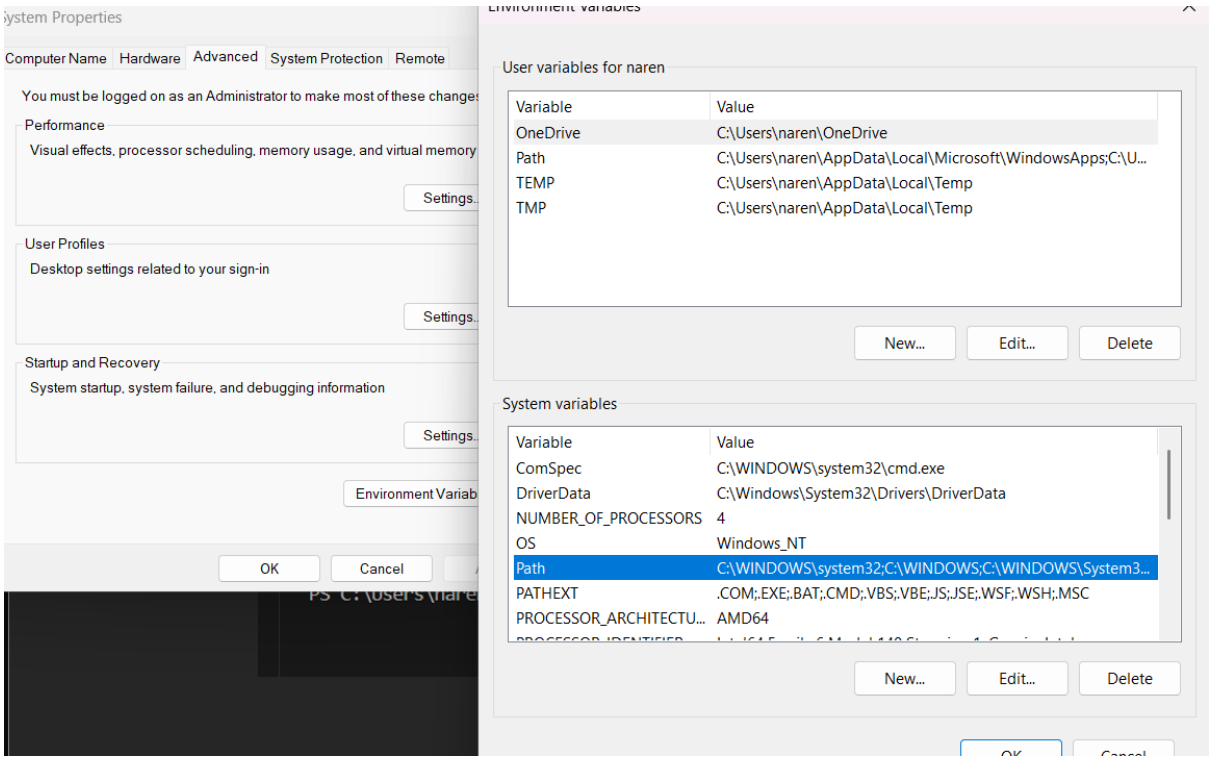
4) Install Minikube using powershell

https://minikube.sigs.k8s.io/docs/start/

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
                                                                                    Install the latest PowerShell for ne
w features and improvements! https://aka.ms/PSWindows
                                                    PS C:\Users\naren> wsl --set-defaut -version2
            Invalid command line argument: --set-defaut
Please use 'wsl.exe --help' to get a list of supported arguments.
PS C:\Users\naren> New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11-04-2025     11:13                minikube


PS C:\Users\naren> Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-wind
ows-amd64.exe' -UseBasicParsing
PS C:\Users\naren>
```

```
PS C:\Users\naren> Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-wind
ows-amd64.exe' -UseBasicParsing
PS C:\Users\naren> $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
PS C:\Users\naren> if ($oldPath.Split(';') -inotcontains 'C:\minikube'){
>>   [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)
```

5) Set the minikube path to environmental variable



6) Restart powershell and start minikube

minikube start

```
PS C:\Users\naren> minikub start
```

To check minikube ip use : minikube ip

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS        powershell

PS C:\Users\naren> minikube ip
192.168.59.100
PS C:\Users\naren> minikub start
```

To ssh to minikube: minikube ssh

```
PS C:\Users\naren> minikube ssh
                         _             _
            _         _ ( )           ( )
  ___ ___  (_)  ___  (_)| |/')  _   _ | |_      __
/' _ ` _ `\| |/' _ `\| || , <  ( ) ( )| '_`\  /'__`\
| ( ) ( ) || || ( ) || || |\`\ | (_) || |_) )(  ___/
(_) (_) (_)(_)(_) (_)(_)(_) (_)`\___/'(_,__/'`\____)
```

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

→launch 3 ec2 instance as master and worker nodes by following below:

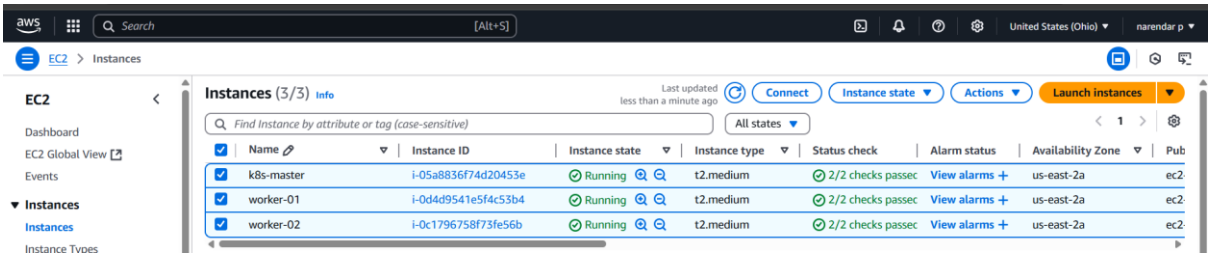
-ubuntu AMI

-type-t2.medum

-volume-20 gb

-enable public ip

-key pair create-k8s



→connect now master:

-- swap memory should be zero check it

```
ubuntu@master:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       417Mi       3.3Gi       892Ki       384Mi       3.4Gi
Swap:             0B          0B          0B
ubuntu@master:~$
```

→change hostname

```
ubuntu@ip-172-31-1-77:~$ sudo echo $hostname

ubuntu@ip-172-31-1-77:~$ sudo hostname master
```

Follow above then do refresh

```
ubuntu@master:~$
```

```
Last login: Fri Apr 11 12:39:17 2025 from 3.16.146.5
ubuntu@worker-02:~$ sudo -i
root@worker-02:~#
```

```
Last login: Fri Apr 11 12:37:07 2025 from 3.16.146.5
ubuntu@worker-01:~$ sudo -i
root@worker-01:~#
```

→install docker in all machines

sudo apt update && sudo apt upgrade -y

apt install docker

systemctl start docker

systemctl enable docker

systemctl status docker

```
ubuntu@master:~$ sudo systemctl start docker
ubuntu@master:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@master:~$ sudo -i
root@master:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-11 14:54:15 UTC; 3min 28s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 20192 (dockerd)
      Tasks: 9
     Memory: 20.2M (peak: 21.4M)
        CPU: 302ms
     CGroup: /system.slice/docker.service
             └─20192 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
ubuntu@worker-01:~$ sudo systemctl start docker
ubuntu@worker-01:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@worker-01:~$ sudo -i
root@worker-01:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-11 14:54:03 UTC; 2min 28s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 19150 (dockerd)
      Tasks: 9
     Memory: 20.1M (peak: 20.5M)
        CPU: 296ms
     CGroup: /system.slice/docker.service
             └─19150 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
ubuntu@worker-02:~$ sudo systemctl start docker
ubuntu@worker-02:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@worker-02:~$ sudo -i
root@worker-02:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-04-11 14:54:09 UTC; 3min 1s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 19124 (dockerd)
      Tasks: 10
     Memory: 20.2M (peak: 21.4M)
        CPU: 302ms
     CGroup: /system.slice/docker.service
             └─19124 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/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:/v1.32/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Initiliaze with CIDR in master machine use command below

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

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

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

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

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

  export KUBECONFIG=/etc/kubernetes/admin.conf

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

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

kubeadm join 172.31.24.93:6443 --token 4dj1nh.9co81jfzxo34o205 \
        --discovery-token-ca-cert-hash sha256:69e19acd2f14a891ed7d1c75b34402e3f942c7deed46004e910ffde3c99c30c2
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

```
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 172.31.24.93:6443 --token 4dj1nh.9co81jfzxo34o205 \
        --discovery-token-ca-cert-hash sha256:69e19acd2f14a891ed7d1c75b34402e3f942c7deed46004e910ffde3c99c30c2
root@master:~# mkdir -p $HOME/.kube
root@master:~# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@master:~#  sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@master:~# |
```

```
root@master:~# kubectl get nodes
NAME      STATUS     ROLES           AGE    VERSION
master    NotReady   control-plane   5m1s   v1.32.3
```

→copy the token and give it on both worker machines

kubeadm join 172.31.24.93:6443 --token 4dj1nh.9co81jfzxo34o205 \
--discovery-token-ca-cert-hash sha256:69e19acd2f14a891ed7d1c75b34402e3f942c7deed46004e910ffde3c99c30c2

```
root@master:~# kubectl get nodes
NAME        STATUS     ROLES           AGE     VERSION
master      NotReady   control-plane   6m15s   v1.32.3
worker-01   NotReady   <none>          14s     v1.32.3
worker-02   NotReady   <none>          18s     v1.32.3
```

we need to start these nodes by flannel

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

→now the pods are ready:

```
root@master:~# kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@master:~# kubectl get nodes
NAME        STATUS     ROLES           AGE     VERSION
master      Ready      control-plane   13m     v1.32.3
worker-01   Ready      <none>          7m52s   v1.32.3
worker-02   NotReady   <none>          7m56s   v1.32.3
root@master:~# kubectl get nodes
NAME        STATUS   ROLES           AGE     VERSION
master      Ready    control-plane   14m     v1.32.3
worker-01   Ready    <none>          8m49s   v1.32.3
worker-02   Ready    <none>          8m53s   v1.32.3
root@master:~# |
```

3) Run one nginx pod.

→in minikube:

kubectl run firstpod --image=nginx

```
PS C:\Users\naren> kubectl run firstpod --image=nginx
pod/firstpod created
```

→in k8s setup

--check if any pods are running

-- to run nginx pod use command:

kubectl run firstpod --image=nginx

-- check if pod is running or notUse command:

kubectl get pods

```
root@master:~# kubectl get pods
No resources found in default namespace.
root@master:~# kubectl run firstpod --image=nginx
error: required flag(s) "image" not set
root@master:~# kubectl run firstpod --image=nginx
error: required flag(s) "image" not set
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          67s
```

-- to access in browser

First we need to expose to port number by using below command

kuberctl expose pod firstpod --port=80 --type=NodePort

```
root@master:~# kubectl expose pod firstpod --port=80 --type=NodePort
service/firstpod exposed
```

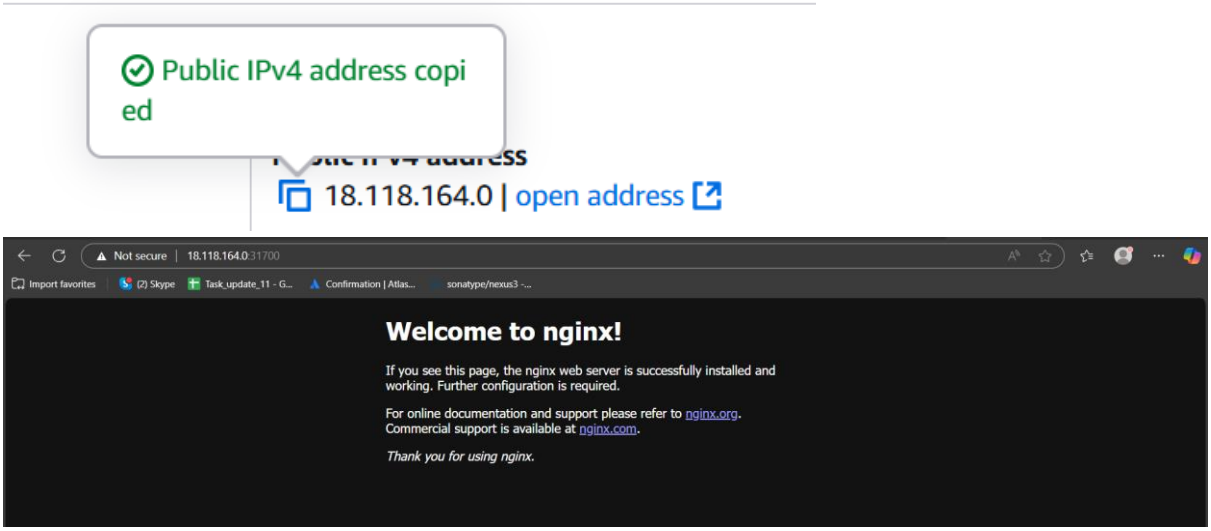
```
root@master:~# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE        NOMINATED NODE   READINESS GATES
firstpod    1/1     Running   0          30m   10.10.171.4  worker-01   <none>           <none>
```

--use this command to show port number

kubectl get svc firstpod

```
root@master:~# kubectl get svc firstpod
NAME        TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
firstpod    NodePort   10.105.174.181   <none>        80:31700/TCP   3m33s
```

--now access with worker-01 node pubip:31700 on browser



4) Mugup Master and slave components on k8s.

→**Noted**