Ques 2 : Create a Kubernetes cluster using kubeadm

SOLN :

Difference Between Minikube and Kubeadm

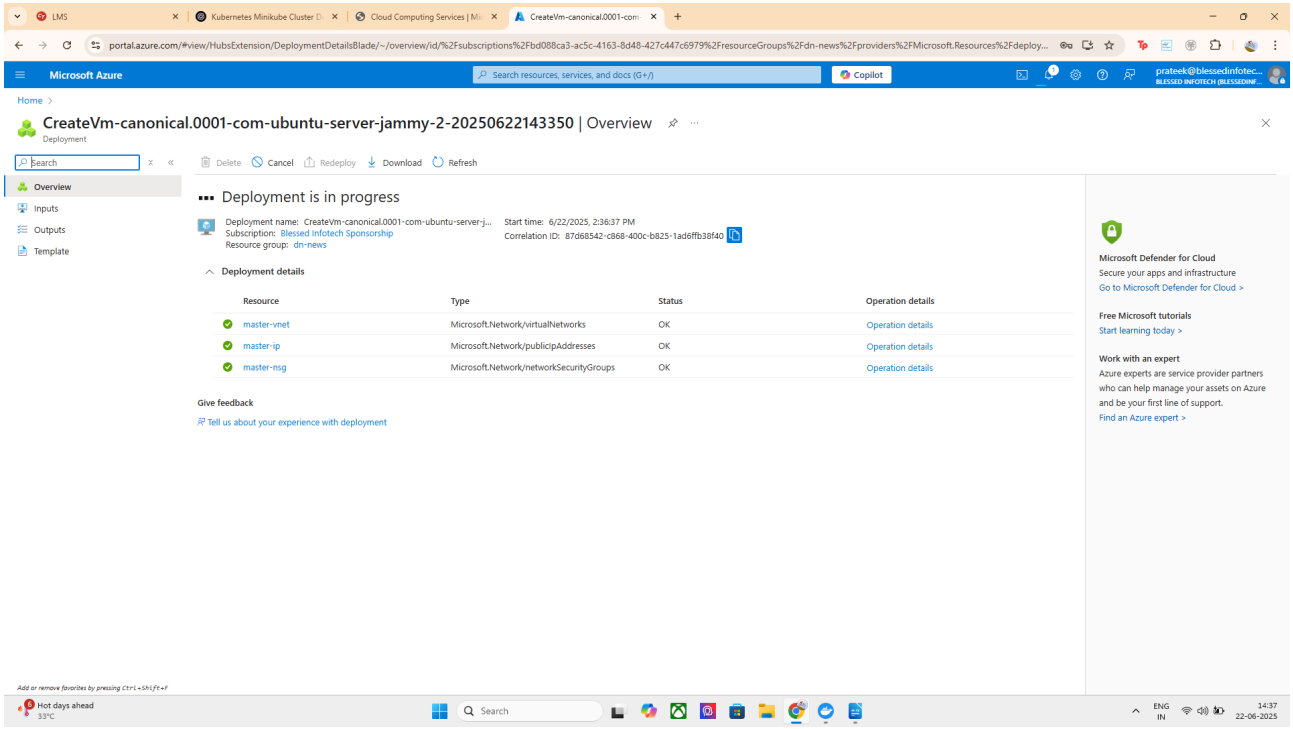
Minikube is a lightweight Kubernetes tool designed for local development and testing on a single machine. In contrast, kubeadm is used to set up multi-node, production-like Kubernetes clusters with fine-grained control over components. While Minikube is ideal for beginners and prototyping, kubeadm simulates real infrastructure, making it better suited for DevOps and cloud environments.

FOR THIS WE NEED MINIMUM 2 MACHINE OR NODES

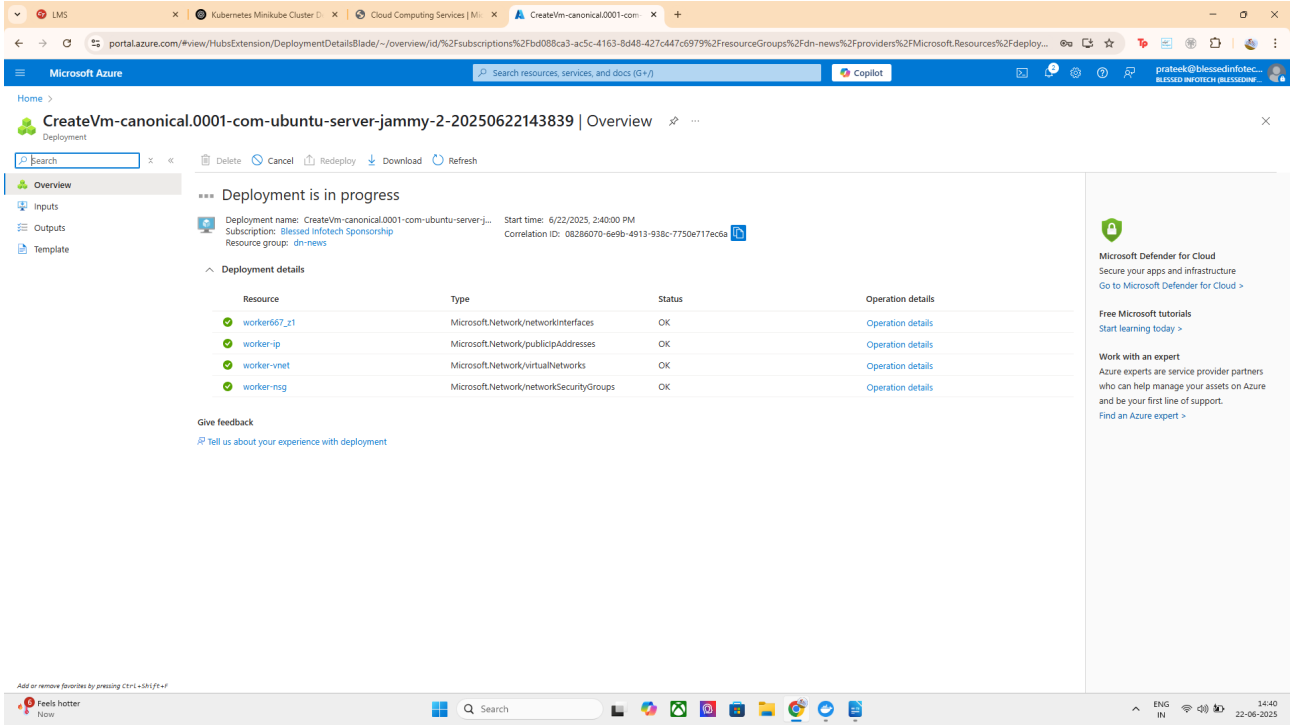
1 MASTER NODE
2 WORKER NODE

SO FOR THIS WE WILL USE VM WITH THE HELP OF AZURE VM INTERFACE TO GET THROUGH IT

MASTER NODE :



WORKER NODE :



Now SSH Into VMs

```
master@master: ~

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

master@master:~$
```

```
worker@worker: ~

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

worker@worker:~$
```

Install Kubernetes & Container Runtime (On Both VMs)

commands used :

Disable swap

sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab

 # Install container runtime (containerd)
sudo apt update
sudo apt install -y containerd

```
master@master: ~
(Reading database ... 62803 files and directories currently installed.)
Preparing to unpack .../runc_1.2.5-0ubuntu1~22.04.1_amd64.deb ...
Unpacking runc (1.2.5-0ubuntu1~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.27-0ubuntu1~22.04.1_amd64.deb ...
Unpacking containerd (1.7.27-0ubuntu1~22.04.1) ...
Setting up runc (1.2.5-0ubuntu1~22.04.1) ...
Setting up containerd (1.7.27-0ubuntu1~22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/sys
tem/containerd.service.
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
master@master:~$
```

```
worker@worker: ~
(Reading database ... 62803 files and directories currently installed.)
Preparing to unpack .../runc_1.2.5-0ubuntu1~22.04.1_amd64.deb ...
Unpacking runc (1.2.5-0ubuntu1~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.27-0ubuntu1~22.04.1_amd64.deb ...
Unpacking containerd (1.7.27-0ubuntu1~22.04.1) ...
Setting up runc (1.2.5-0ubuntu1~22.04.1) ...
Setting up containerd (1.7.27-0ubuntu1~22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/sys
tem/containerd.service.
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
worker@worker:~$
```

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

 # Install kubeadm, kubelet, kubectl
sudo apt update && sudo apt install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
**echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" **
| sudo tee /etc/apt/sources.list.d/kubernetes.list

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

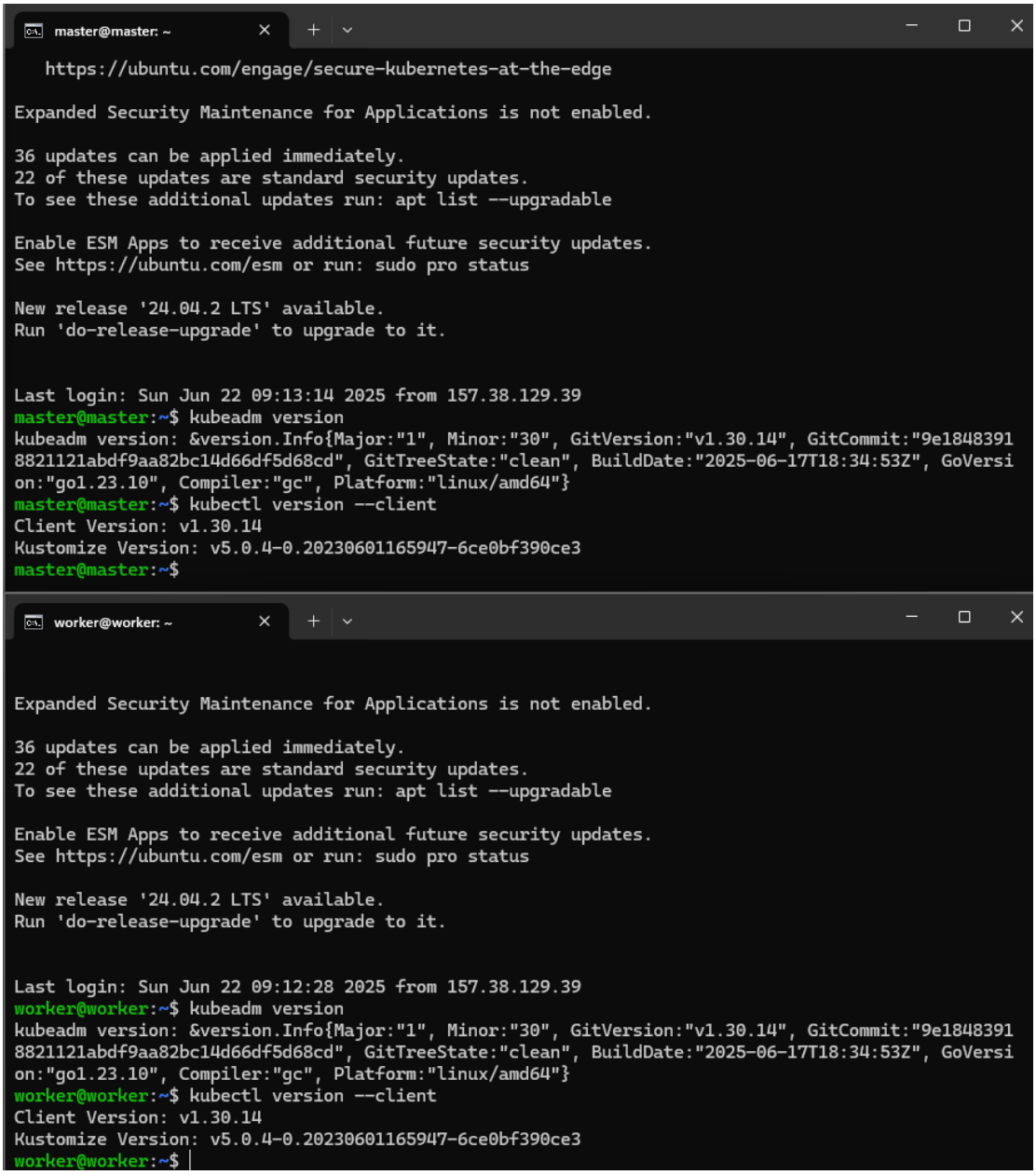
This is the common error i faced on installation

How to switch to the new repo on Ubuntu 22.04 (Jammy)

The commands below install the current stable stream (v1.30).
If you need a different minor version, replace **`v1.30`** everywhere with e.g. **`v1.29`**.

```
# 1) Clean up any old entry that points to apt.kubernetes.io
sudo rm -f /etc/apt/sources.list.d/kubernetes.list

# 2) Create keyring directory (needed on Ubuntu 22.04)
sudo mkdir -p /etc/apt/keyrings

# 3) Add the new signing key
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key \
  | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

# 4) Add the new APT repository
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

# 5) Update indexes and install the tools
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

KUBEADM SUCCESSFULLY INSTALLED ON BOTH THE MASTER AND WORKER VMS



NOW WE HAVE TO INITIALIZE THE Kubernetes (Master Node Only)

Enable IP Forwarding

Run the following command to enable IP forwarding temporarily:

```
sudo sysctl -w net.ipv4.ip_forward=1

echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

COMMAND USED : sudo kubeadm init –pod-network-cidr=192.168.0.0/16

MASTER READY TO JOIN

```
master@master: ~

[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 10.0.0.4:6443 --token 0n9sn8.dquizu21rwl5cfly \
        --discovery-token-ca-cert-hash sha256:68daa32d80e1b3576df9a1ef2af659a63dc3c6bb558747b81df
e85c4632f955b
master@master:~$
```

After success, copy the output that contains the `kubeadm join` command.

Then:

```
mkdir -p $HOME/.kube
sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
```

Install Pod Network (Calico) ON MASTER NODE

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

```
master@master: ~
calico-kube-controllers-5b9b456c66-q6cbw   0/1     ContainerCreating   0          9s
calico-node-wcp9s                          0/1     Init:2/3            0          9s
coredns-55cb58b774-4hn8d                   0/1     ContainerCreating   0          21s
coredns-55cb58b774-nc8s2                   0/1     ContainerCreating   0          21s
etcd-master                                1/1     Running             6          36s
kube-apiserver-master                      1/1     Running             9          36s
kube-controller-manager-master             1/1     Running             12         37s
kube-proxy-78j8t                           1/1     Running             0          21s
kube-scheduler-master                      1/1     Running             7          36s
master@master:~$ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-5b9b456c66-q6cbw   1/1     Running   0          57s
calico-node-wcp9s                          1/1     Running   0          57s
coredns-55cb58b774-4hn8d                   1/1     Running   0          69s
coredns-55cb58b774-nc8s2                   1/1     Running   0          69s
etcd-master                                1/1     Running   6          84s
kube-apiserver-master                      1/1     Running   9          84s
kube-controller-manager-master             1/1     Running   12         85s
kube-proxy-78j8t                           1/1     Running   0          69s
kube-scheduler-master                      1/1     Running   7          84s
master@master:~$ kubectl get nodes
NAME     STATUS   ROLES           AGE    VERSION
master   Ready    control-plane   105s   v1.30.14
master@master:~$
```

NOW LETS JOIN THE WORKER NODE :

COMMAND USED : sudo kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

Fix on Worker Node

Run these **on the worker node**:

```
# Enable temporarily
sudo sysctl -w net.ipv4.ip_forward=1

# Make it persistent
echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

```
worker@worker:~$ sudo kubeadm join 10.0.0.4:6443 --token if5t6m.ejmib9r0zw0llah8 --discovery-token-ca-
cert-hash sha256:c94edda2a9e8503a6ce40ae125ec5a47b1b71b0768724e1a4472a096c902e437
[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.en
v"
[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 4
m0s
[kubelet-check] The kubelet is healthy after 501.533609ms
[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.

worker@worker:~$
```

Verify Nodes (Back on Master)

On master node its connected successfully :

```
master@master:~$ kubectl get nodes
NAME      STATUS     ROLES           AGE   VERSION
master    Ready      control-plane   98m   v1.30.14
worker    NotReady   <none>          14s   v1.30.14
master@master:~$
```

FACED CALIO ISSUSE REINSTALLED IT FROM BEGINNING SO NOW :

BOTH READY

```
master@master:~$ kubectl get nodes
NAME      STATUS   ROLES           AGE    VERSION
master    Ready    control-plane   112m   v1.30.14
worker    Ready    <none>          14m    v1.30.14
master@master:~$
```

NOW , Let's Deploy a Sample App (Nginx)

Step 1: Create the Deployment

COMMAND USED : kubectl create deployment nginx –image=nginx

Step 2: Expose the Deployment via a NodePort service

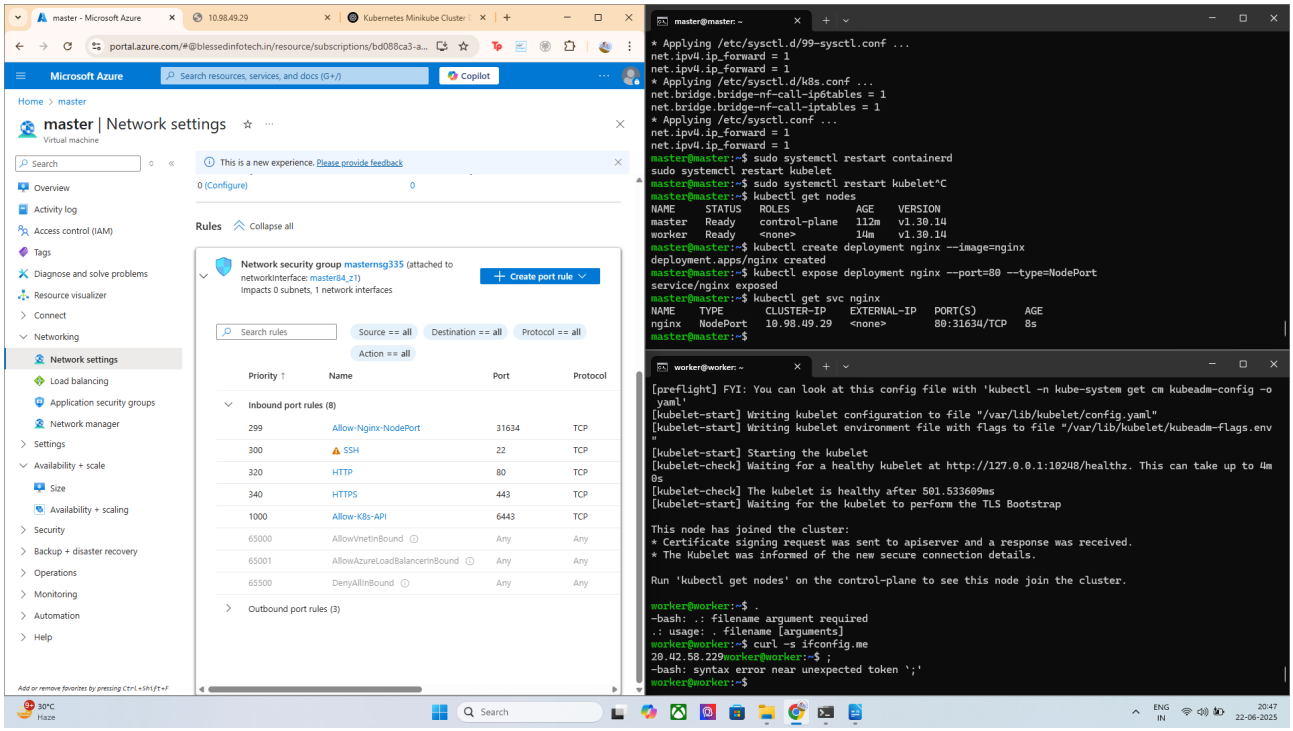
kubectl expose deployment nginx --port=80 --type=NodePort

Step 3: Get the Service Details

kubectl get svc nginx

```
master@master:~$ kubectl get svc nginx
NAME    TYPE       CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
nginx   NodePort   10.98.49.29   <none>        80:31634/TCP   8s
master@master:~$
```

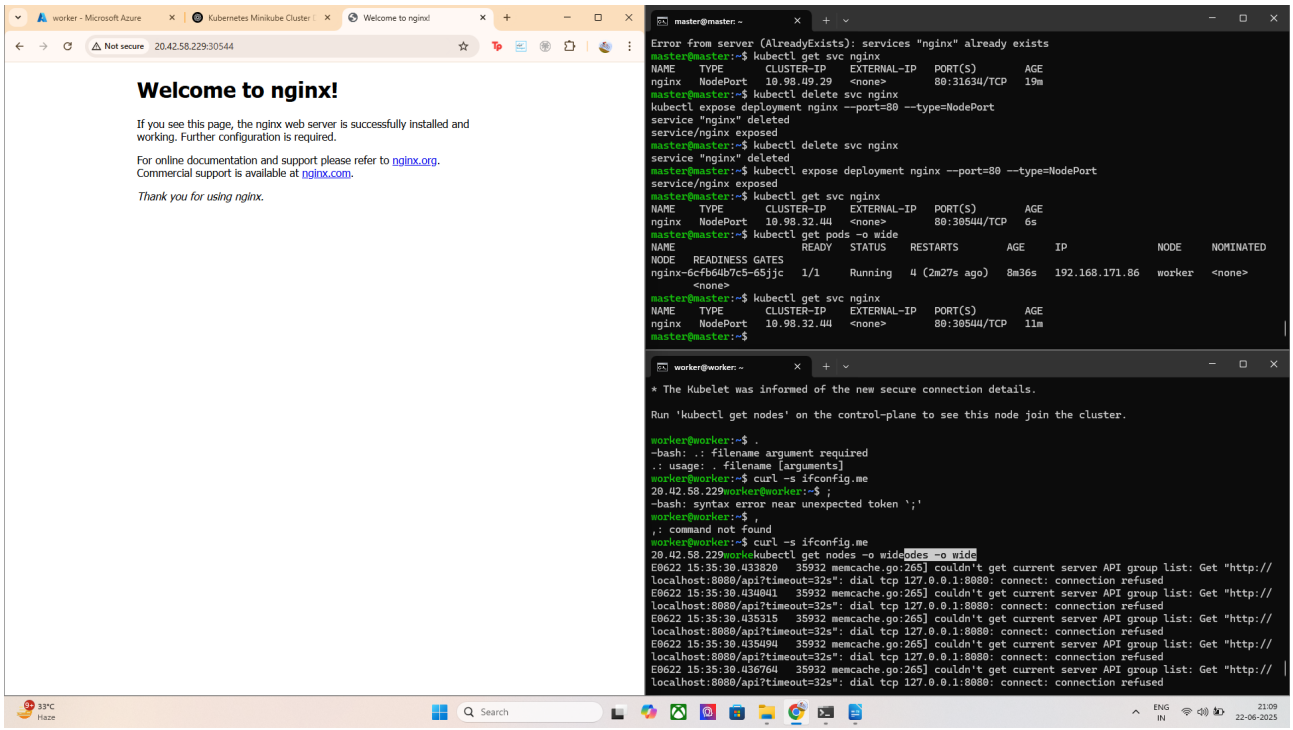
NOW ALLOW THE PORT INTO VMS INBOUND RULE



Access the App

Open your browser and visit:

`http://<ANY_NODE_PUBLIC_IP>:<NodePort>`



most important command : kubectl get pods -o wide

So basically i was concerned why my master not running the application this is what i found so i thought i should share :

Kubernetes master nodes manage the cluster's control plane and are tainted to avoid running application pods.
This ensures high availability and performance of cluster operations.
Only worker nodes are used to run user workloads by default.
To run pods on the master, taints must be removed or tolerated explicitly.

Lets deploy some custom application image on the same using kubeadm

I have used my portfolio website image:

https://hub.docker.com/r/prateek2004/my-frontend

steps :

1 > Create a deployment (replace your-dockerhub-username/your-image:tag with your actual image):

command used : `kubectl create deployment my-app --image=your-dockerhub-username/your-image:tag`

2> Expose the deployment as a service:

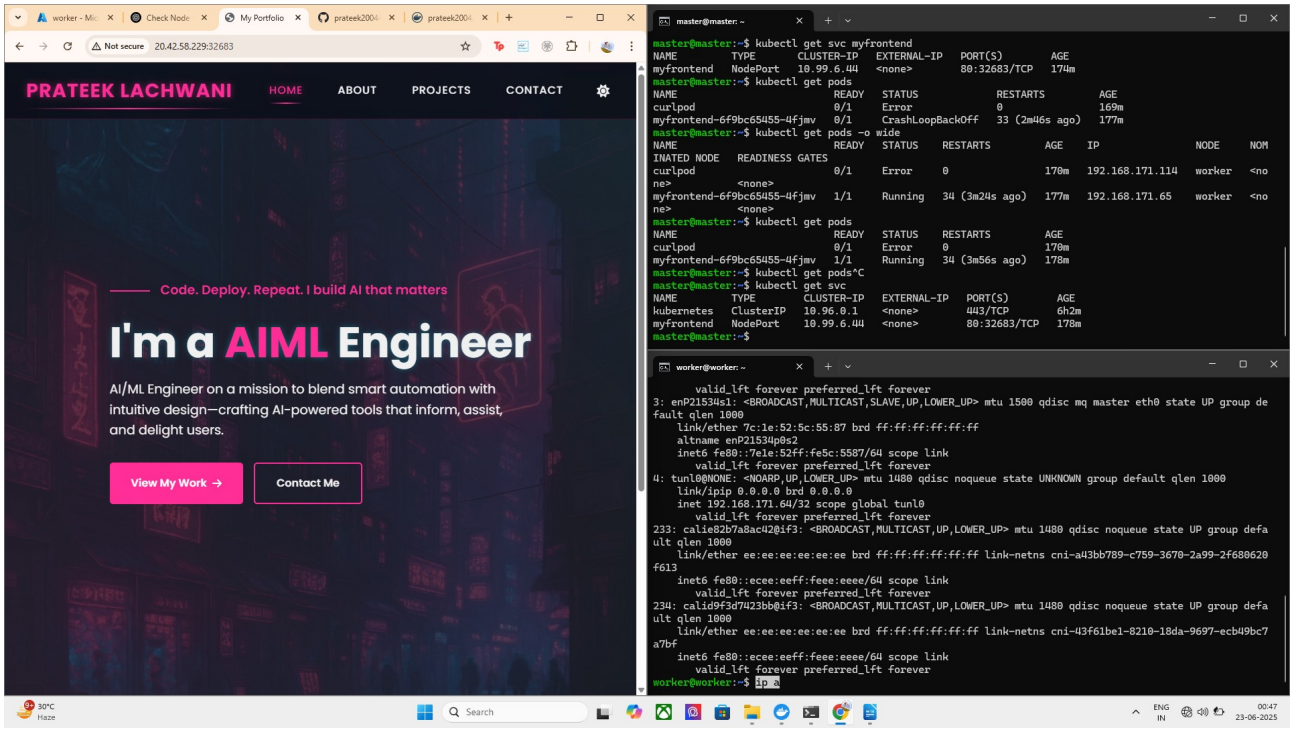
Command used : kubectl expose deployment my-app --type=NodePort –port=80

3> Get the NodePort to access the app:

command used : kubectl get svc my-app

Access using:

http://<your-node-ip>:<node-port>



ALL THE CLUSTERS AND NODES ARE SUCCESSFULLY WORKING AND THE DEPLOYMENT IS SUCCESSFULLY MADE