

Setting Up a Kubernetes Cluster with Kubeadm on Azure Virtual Machines

Kubernetes has become the go-to platform for running and scaling containerized applications in the cloud. In this guide, we'll walk through setting up a **production-ready Kubernetes cluster on Azure** using **kubeadm**.

We'll build out one master node and two worker nodes, all provisioned through the **Azure Portal**. To connect and configure the VMs, we'll use **PuTTY (on Windows)** or **SSH (on Mac/Linux)**.

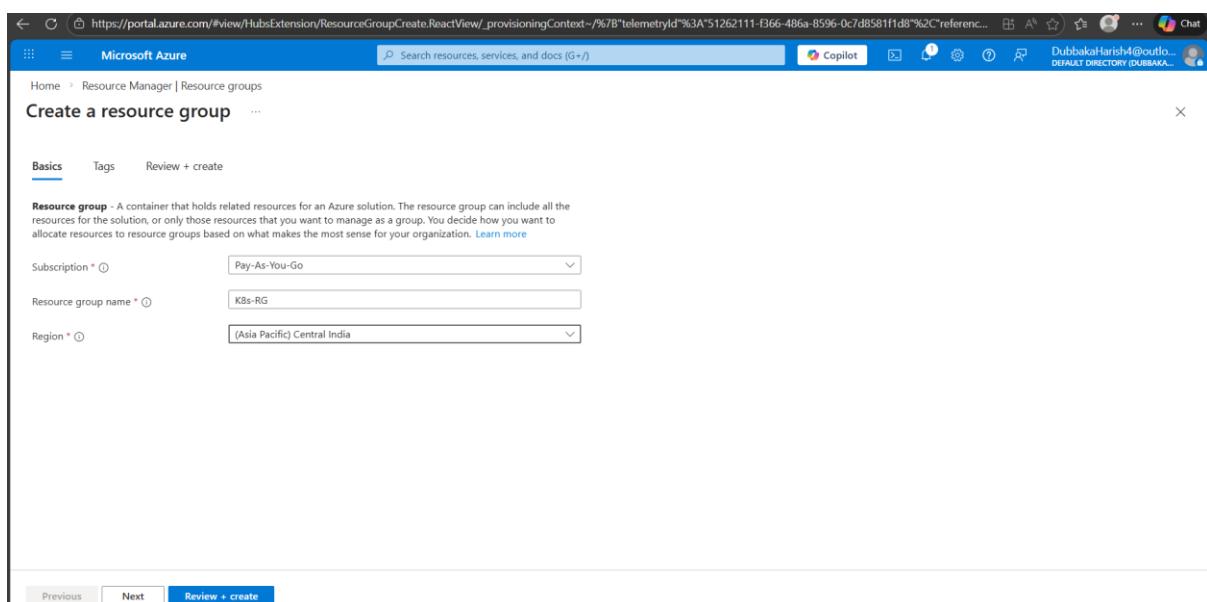
Part 1: Creating Virtual Machines in Azure

Step 1: Log into Azure Portal

👉 Head to portal.azure.com and sign in with your Azure account.

Step 2: Create a Resource Group

1. In the left menu, click **Resource groups**
2. Hit **+ Create**
3. Fill in the details:
 - **Subscription:** pick yours
 - **Resource group name:** choose something neat (e.g., **K8s-RG**)
 - **Region:** select one near you (e.g., Central India)
4. Click **Review + create → Create**



Step 3: Create a Virtual Network (VNet)

1. Search for **Virtual networks** in the portal
2. Click **+ Create**
3. Configure basics:

Microsoft Azure

Create virtual network

Basics Security IP addresses Tags Review + create

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Pay-As-You-Go

Resource group * K8s-RG

Instance details

Virtual network name * harish-vnet

Region * (Asia Pacific) Central India

Deploy to an Azure Extended Zone

Previous Next Review + create Give feedback

Microsoft Azure

Create virtual network

Basics Security IP addresses Tags Review + create

Define the address space of your virtual network with one or more IPv4 or IPv6 address ranges. Create subnets to segment the virtual network address space into smaller ranges for use by your applications. When you deploy resources into a subnet, Azure assigns the resource an IP address from the subnet. [Learn more](#)

Allocate using IP address pools. [Learn more](#)

+ Add a subnet

Subnets	IP address range	Size	NAT gateway
default	10.0.0.0 - 10.0.0.255	/24 (256 addresses)	-

Delete address space

10.0.0.0/16

10.0.0.0 /16 65,536 addresses

10.0.0.0 - 10.0.255.255

Previous Next Review + create Give feedback

- Name: e.g., **harish-vnet**
- Address space: **10.0.0.0/16**

- Add subnet: default → 10.0.0.0/24

Step 4: Create the Master Node VM

1. From the homepage, click + Create a resource → search for **Ubuntu Server 22.04 LTS**

2. Configure:

- VM name: **harish-master**

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure, specifically the 'Basics' tab. The page title is 'Create a virtual machine'. At the top, there are three buttons: 'Help me create a low cost VM', 'Help me choose the right VM size for my workload', and 'Help me create a VM optimized for high availability'. Below these buttons, there is a warning message: '⚠️ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.' A horizontal bar with three buttons follows: 'Help me create a low cost VM', 'Help me create a VM optimized for high availability', and 'Help me choose the right VM size for my workload'. The main content area is divided into sections: 'Project details', 'Subscription *' (Pay-As-You-Go), 'Resource group *' (K8s-RG), 'Virtual machine name *' (harish-master), and 'Region *' ((Asia Pacific) Central India). To the right, there is a sidebar titled 'Estimated monthly cost' showing '₹0.00 / month'.

Size: Standard_D2s_v3 (2 vcpus, 8 GiB RAM)

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure, specifically the 'Size' tab. The page title is 'Create a virtual machine'. At the top, there are three buttons: 'Help me create a low cost VM', 'Help me choose the right VM size for my workload', and 'Help me create a VM optimized for high availability'. Below these buttons, there is a warning message: '⚠️ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.' A horizontal bar with three buttons follows: 'Help me create a low cost VM', 'Help me create a VM optimized for high availability', and 'Help me choose the right VM size for my workload'. The main content area is divided into sections: 'Availability zone *' (Zone 1), 'Security type' (Trusted launch virtual machines), 'Image *' (Ubuntu Server 24.04 LTS - x64 Gen2), 'VM architecture' (x64), 'Run with Azure Spot discount' (unchecked), 'Size *' (Standard_B2ms - 2 vcpus, 8 GiB memory (₹5,441.60/month)), and 'Enable Hibernation' (unchecked). To the right, there is a sidebar titled 'Estimated monthly cost' showing '₹0.00 / month'.

Auth: Password → Harish

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure, specifically the 'Authentication' tab. The page title is 'Create a virtual machine'. At the top, there are three buttons: 'Help me create a low cost VM', 'Help me choose the right VM size for my workload', and 'Help me create a VM optimized for high availability'. Below these buttons, there is a warning message: '⚠️ Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.' A horizontal bar with three buttons follows: 'Help me create a low cost VM', 'Help me create a VM optimized for high availability', and 'Help me choose the right VM size for my workload'. The main content area is divided into sections: 'Administrator account', 'Authentication type' (Password selected), 'Username *' (Harish), 'Password *' (redacted), 'Confirm password *' (redacted), and 'Inbound port rules'. Under 'Inbound port rules', it says 'Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.' There are two radio buttons: 'None' and 'Allow selected ports' (selected). Below this is a dropdown menu 'Select inbound ports *' with 'SSH (22)' selected. A note below the dropdown says: '⚠️ This will allow all IP addresses to access your virtual machine. This is only...'. To the right, there is a sidebar titled 'Estimated monthly cost' showing '₹0.00 / month'.

Networking: Attach to your VNet, allow SSH (22)

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal. The 'Networking' tab is selected. The configuration details are as follows:

- Virtual network:** harish-vnet (X8s-RG)
- Subnet:** (New) snet-centralindia-1
- Public IP:** (new) harish-master-ip
- NIC network security group:** Basic

The estimated monthly cost is ₹1,218.80 / month.

Tag: Role=master

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal. The 'Tags' tab is selected. A single tag is defined:

Name	Value
Role	Master

The estimated monthly cost is ₹1,218.80 / month.

3. Hit Review + create → Create

The screenshot shows the Microsoft Azure Deployment Overview page for a completed deployment. The deployment name is 'CreateVm-canonical.ubuntu-24_04-lts-server-20260224054618'. Key details include:

- Deployment name:** CreateVm-canonical.ubuntu-24_04-lts-server-2...
- Subscription:** Pay-As-You-Go
- Resource group:** K8s-RG
- Start time:** 2/24/2026, 5:49:16 AM
- Correlation ID:** d92986e6-4136-4bfc-bc91-333ce5bc1e5c

The 'Deployment details' section shows the deployment is complete. The 'Next steps' section includes recommendations for auto-shutdown, monitoring, and running scripts.

Step 5: Create Worker Node VMs

Repeat Step 4 for two more VMs:

- Names: **bantu-worker1** and **chintu-worker2**
- Use **Password** Auth

Tag: **Role=worker**

The screenshot shows the 'Create a virtual machine' wizard on the 'Tags' step. A single tag 'Role=worker' is selected. To the right, the 'Estimated monthly costs' table is displayed, showing estimated costs for various Azure services based on the selected configuration.

Category	Cost
Basics	₹0.00
Disks	₹0.00
Networking	₹1,218.80
Management	₹0.00
Monitoring	₹0.00
Estimated monthly cost	₹1,218.80

Home

CreateVm-canonical.ubuntu-24_04-lts-server-20260224055102 | Overview

Deployment

Search x << Delete Cancel Redeploy Download Refresh

Overview

Inputs Outputs Template

Your deployment is complete

Deployment name: CreateVm-canonical.ubuntu-24_04-lts-server-2... Start time: 2/24/2026, 5:54:20 AM
Subscription: Pay-As-You-Go Correlation ID: cbb99f16-00c0-4164-a74a-487727eecc30

Deployment details

Next steps

- Setup auto-shutdown Recommended
- Monitor VM health, performance and network dependencies Recommended
- Run a script inside the virtual machine Recommended

Go to resource Create another VM

Give feedback Tell us about your experience with deployment

Home

CreateVm-canonical.ubuntu-24_04-lts-server-2026022405536 | Overview

Deployment

Search x << Delete Cancel Redeploy Download Refresh

Overview

Inputs Outputs Template

Your deployment is complete

Deployment name: CreateVm-canonical.ubuntu-24_04-lts-server-2... Start time: 2/24/2026, 5:57:04 AM
Subscription: Pay-As-You-Go Correlation ID: 8b06ca6d-d713-4fe9-b1b1-0a002460f651

Deployment details

Next steps

- Setup auto-shutdown Recommended
- Monitor VM health, performance and network dependencies Recommended
- Run a script inside the virtual machine Recommended

Go to resource Create another VM

Give feedback Tell us about your experience with deployment

Cost Management
Get notified prevent unexpected costs [Set up cost monitoring](#)

 Microsoft Defender
Secure your organization [Go to Microsoft Defender](#)

 Free Microsoft 365
Start learning [Start learning](#)

 Work with experts
Azure experts who can help you [Find an expert](#)

Virtual machines Get started

+ Create Reservations Manage view Refresh Export to CSV Open query Assign tags Start Restart Stop ... Group by none

Filter for any field... Subscription equals all Type equals all Resource Group equals all Location equals all Add filter

<input type="checkbox"/>	Name ↑	Subscription	Resource Group	Location	Status	Operating system	Size	Public IP address	Disk	
<input type="checkbox"/>	bantu-worker1	...	Pay-As-You-Go	K8s-RG	Central India	Running	Linux	Standard_B2ms	4.240.112.174	1
<input type="checkbox"/>	chintu-worker2	...	Pay-As-You-Go	K8s-RG	Central India	Running	Linux	Standard_B2ms	40.81.232.201	1
<input type="checkbox"/>	harish-master	...	Pay-As-You-Go	K8s-RG	Central India	Running	Linux	Standard_B2ms	40.81.232.34	1

Step 6: Open Kubernetes Ports

Inside the **Network Security Group (NSG)** of your master node & worker nodes:

- Allow TCP **6443** (Kubernetes API server)
- Allow internal traffic in **10.0.0.0/16** for cluster communication on

The screenshot shows the Azure portal interface for managing network security groups. The left sidebar is collapsed, and the main area displays a table of NSGs. The table columns are Name, Resource Group, Location, Subscription, and Flow log. Three NSGs are listed: bantu-worker1-nsg, chintu-worker2-nsg, and harish-master-nsg, all associated with the Kbs-RG resource group and located in Central India under the Pay-As-You-Go subscription.

Name	Resource Group	Location	Subscription	Flow log
bantu-worker1-nsg	Kbs-RG	Central India	Pay-As-You-Go	
chintu-worker2-nsg	Kbs-RG	Central India	Pay-As-You-Go	
harish-master-nsg	Kbs-RG	Central India	Pay-As-You-Go	

The screenshot shows the Azure portal interface for managing network settings of a virtual machine named bantu-worker1. The left sidebar shows the navigation menu with Network settings selected. The main area displays the network settings for the VM, including the attached network interface bantu-worker128_z1 and its associated NSG bantu-worker1-nsg. The NSG rules section shows five inbound port rules and three outbound port rules. The inbound rules allow traffic on ports 22, 310, 65000, 65001, and 65500. The outbound rules include AllowVnetInBound, AllowAzureLoadBalancerInBound, and DenyAllInBound.

Priority ↑	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
310	nodeportservices	30000-32767	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

chintu-worker2 | Network settings

Network security group chintu-worker2-nsg (attached to networkInterface: chintu-worker269_z1)

Inbound port rules (5)

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
310	nodeportservices	30000-32767	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Outbound port rules (3)

harish-master | Network settings

Network security group harish-master-nsg (attached to networkInterface: harish-master67_z1)

Inbound port rules (6)

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
390	kubernetes_api_server	6443	TCP	Any	Any	Allow
400	internalcluster_communication	Any	TCP	106.51.175.107	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Outbound port rules (4)

Step 7: Note Down IPs

Public IPs

- Each VM (master and workers) will have a **public IP** assigned by your cloud provider (Azure in your case).
- These are used for **SSH access** from your local machine.
- Example:
- harish-master → 40.xx.xx.xx
- bantu-worker1 → 40.xx.xx.xx
- chintu-worker2 → 40.81.232.201

Step 8: Master VM Setup Process

1. SSH into Master VM

- Connect to harish-master.
 - This connects you to the master node where you'll initialize Kubernetes.
-

2. Disable Swap

```
swapoff -a  
sudo sed -i '/ swap / s/^(\.*\)$/#\1/g' /etc/fstab
```

- Kubernetes requires swap disabled for predictable scheduling.
-

3. Enable Kernel Modules & Sysctl

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf  
overlay  
br_netfilter  
EOF
```

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

- `overlay` → container storage.
- `br_netfilter` → iptables visibility for bridged traffic.

Configure sysctl:

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf  
net.bridge.bridge-nf-call-iptables = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
net.ipv4.ip_forward = 1  
EOF
```

```
sudo sysctl --system
```

- Ensures pod/service traffic forwarding works.
-

4. Install Container Runtime (containerd)

- Install containerd, configure SystemdCgroup = true, and enable service:

```
systemctl status containerd
```

5. Install runc

```
curl -LO https://github.com/opencontainers/runc/releases/download/v1.1.12/runc.amd64
```

```
sudo install -m 755 runc.amd64 /usr/local/sbin/runc
```

- runc is the low-level runtime used by containerd.
-

6. Install CNI Plugins

```
curl -LO https://github.com/containernetworking/plugins/releases/download/v1.5.0/cni-plugins-linux-amd64-v1.5.0.tgz
```

```
sudo mkdir -p /opt/cni/bin
```

```
sudo tar Cxvf /opt/cni/bin cni-plugins-linux-amd64-v1.5.0.tgz
```

- Provides pod networking binaries.
-

7. Install Kubernetes Binaries

```
sudo apt-get install -y kubelet kubeadm kubectl
```

- Version pinned to 1.29.6 for later upgrade to 1.30.
-

8. Configure crictl

```
sudo crictl config runtime-endpoint unix:///var/run/containerd/containerd.sock
```

- Ensures crictl talks to containerd.
-

9. Initialize Control Plane (⚠ Use eth0 IP!)

1. Check eth0 IP:
2. ip addr show eth0

Example: inet 10.0.0.4/24.

3. Run init with that IP:
4. sudo kubeadm init \
5. --pod-network-cidr=192.168.0.0/16 \
6. --apiserver-advertise-address=10.0.0.4 \
7. --node-name master

⚠ Why:

- 127.0.0.1 would isolate the API server to localhost.
- Using 10.0.0.4 makes it reachable by worker nodes (10.0.1.4, 10.0.2.4).

⚠ After you run `kubeadm init` on the **master node**, it will generate a **kubeadm join command**. That command is what you'll use to connect each worker node to the cluster.

- It looks something like:

```
kubeadm join 10.0.0.4:6443 --token <token> \  
--discovery-token-ca-cert-hash sha256:<hash>
```

-

10. Configure kubeconfig

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

- Lets you run `kubectl` as your user.

11. Install Calico (CNI plugin)

```
kubectl create -f  
https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml
```

```
curl -O https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/custom-resources.yaml
```

```
kubectl apply -f custom-resources.yaml
```

- Enables pod networking across nodes.
 - Without this, pods can't communicate between worker nodes.
-

 **Note:** After completing all the above steps on your **harish-master VM**, the Kubernetes **control plane is ready**.

- The API server is running and bound to your master's eth0 IP (10.0.0.4).
- kubectl is configured, so you can manage the cluster directly from the master.
- Calico networking is installed, enabling pod communication across nodes.
- A kubeadm join command has been generated — this must be run on each worker node (as root) to connect them to the cluster.

Your master node is now fully prepared to accept worker nodes and orchestrate workloads.

Step-by-Step Worker Node Setup Workflow — (bantu-worker1, chintu-worker2)

Worker VM Setup Process

1. SSH into Worker VM

```
ssh <username>@<public-ip-of-worker>
```

- Example: ssh harish@40.81.232.201
 - Do this for each worker node.
-

2. Disable Swap

```
swapoff -a
```

```
sudo sed -i '/ swap / s/^.*$/#/g' /etc/fstab
```

- Required for stable Kubernetes scheduling.
-

3. Enable Kernel Modules & Sysctl

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
```

```
overlay  
br_netfilter  
EOF
```

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

```
Configure sysctl:
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf  
net.bridge.bridge-nf-call-iptables = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
net.ipv4.ip_forward = 1  
EOF
```

```
sudo sysctl --system
```

```
Verify:
```

```
lsmod | grep br_netfilter
```

```
lsmod | grep overlay
```

```
sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables  
net.ipv4.ip_forward
```

4. Install Container Runtime (containerd)

```
curl -LO https://github.com/containerd/containerd/releases/download/v1.7.14/containerd-  
1.7.14-linux-amd64.tar.gz
```

```
sudo tar Cxzvf /usr/local containerd-1.7.14-linux-amd64.tar.gz
```

```
curl -LO https://raw.githubusercontent.com/containerd/containerd/main/containerd.service
```

```
sudo mkdir -p /usr/local/lib/systemd/system/
```

```
sudo mv containerd.service /usr/local/lib/systemd/system/
```

```
sudo mkdir -p /etc/containerd
```

```
containerd config default | sudo tee /etc/containerd/config.toml
```

```
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml  
sudo systemctl daemon-reload  
sudo systemctl enable --now containerd  
systemctl status containerd
```

5. Install runc

```
curl -LO https://github.com/opencontainers/runc/releases/download/v1.1.12/runc.amd64  
sudo install -m 755 runc.amd64 /usr/local/sbin/runc
```

6. Install CNI Plugins

```
curl -LO https://github.com/containernetworking/plugins/releases/download/v1.5.0/cni-  
plugins-linux-amd64-v1.5.0.tgz  
sudo mkdir -p /opt/cni/bin  
sudo tar Cxvf /opt/cni/bin cni-plugins-linux-amd64-v1.5.0.tgz
```

7. Install Kubernetes Binaries

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

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/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:/v1.29/deb/ /' | sudo tee  
/etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update  
sudo apt-get install -y kubelet=1.29.6-1.1 kubeadm=1.29.6-1.1 kubectl=1.29.6-1.1 --allow-  
downgrades --allow-change-held-packages  
sudo apt-mark hold kubelet kubeadm kubectl
```

Verify versions:

kubeadm version

kubelet --version

kubectl version --client

8. Configure crictl

```
sudo crictl config runtime-endpoint unix:///var/run/containerd/containerd.sock
```

9. Join Worker Nodes to Cluster (which was generated from Masternode)

Run the **join command** generated on the master node after kubeadm init. It looks like:

```
sudo kubeadm join 10.0.0.4:6443 --token <token> \  
--discovery-token-ca-cert-hash sha256:<hash>
```

- 10.0.0.4 → Master's eth0 IP.
- Run this on **both worker nodes** as root.

```
Harish@bantu-worker1:~$ sudo kubeadm join 10.0.0.4:6443 --token lww7rf.lsc5awo22hgqet27 \  
--discovery-token-ca-cert-hash sha256:f69176f866538fa2ca41b7d30e4elf09095f689388a82c669e04236e417c678f  
[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-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.
```

```
Harish@chintu-worker2:~$ sudo kubeadm join 10.0.0.4:6443 --token lww7rf.lsc5awo22hgqet27 \  
--discovery-token-ca-cert-hash sha256:f69176f866538fa2ca41b7d30e4elf09095f689388a82c669e04236e417c678f  
[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-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.
```

If you forgot the command, regenerate it on the master:

```
kubeadm token create --print-join-command
```

Part 5: Verify & Validation Cluster

If all the above steps were completed, you should be able to run kubectl get nodes on the master node, and it should return all the 3 nodes in ready status.

```
Harish@harish-master:~$ kubectl get nodes -o wide
NAME           STATUS   ROLES      AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE        KERNEL-VERSION   CONTAINER-RUNTIME
bantu-worker1  Ready    <none>    3m4s   v1.29.6  10.0.1.4       <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
chintu-worker2  Ready    <none>    2m58s  v1.29.6  10.0.2.4       <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
master         Ready    control-plane  5m42s  v1.29.6  10.0.0.4       <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
Harish@harish-master:~$
```

Also, make sure all the pods are up and running by using the command as below:

```
kubectl get pods -A
```

```
no items found in default namespace
Harish@harish-master:~$ kubectl get pods -A
NAMESPACE     NAME          READY   STATUS    RESTARTS   AGE
calico-apiserver  calico-apiserver-77c47c897c-c28dg  1/1     Running   0          22s
calico-apiserver  calico-apiserver-77c47c897c-mrkwp  1/1     Running   0          22s
calico-system    calico-kube-controllers-7d57969c98-m7vmz  1/1     Running   0          2m11s
calico-system    calico-node-8s75d   1/1     Running   0          100s
calico-system    calico-node-mw6jr   1/1     Running   0          2m11s
calico-system    calico-node-pcbgv   1/1     Running   0          94s
calico-system    calico-typha-59b5848dc7-8p6hc  1/1     Running   0          2m11s
calico-system    calico-typha-59b5848dc7-srxtp  1/1     Running   0          92s
calico-system    csi-node-driver-d7cll   2/2     Running   0          100s
calico-system    csi-node-driver-hj887   2/2     Running   0          2m11s
calico-system    csi-node-driver-jmblt   2/2     Running   0          94s
kube-system      coredns-76f75df574-4hc6q  1/1     Running   0          4m
kube-system      coredns-76f75df574-kr44f  1/1     Running   0          4m
kube-system      etcd-master      1/1     Running   0          4m14s
kube-system      kube-apiserver-master  1/1     Running   0          4m15s
kube-system      kube-controller-manager-master  1/1     Running   1          4m14s
kube-system      kube-proxy-cprd5    1/1     Running   0          100s
kube-system      kube-proxy-vpshr   1/1     Running   0          4m
kube-system      kube-proxy-x4jzv    1/1     Running   0          94s
kube-system      kube-scheduler-master  1/1     Running   1          4m14s
tigera-operator  tigera-operator-76c4974c85-cf7j4  1/1     Running   0          2m19s
```

Step 13: Test with NGINX

```
kubectl create deployment nginx --image=nginx
```

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

```
kubectl get svc nginx
```

```
Harish@harish-master:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
Harish@harish-master:~$ kubectl expose deployment nginx --port=80 --type=NodePort
service/nginx exposed
Harish@harish-master:~$ kubectl get svc nginx
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx    NodePort  10.96.159.65  <none>        80:30407/TCP  13s
Harish@harish-master:~$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nginx-7854ff8877-mnstj  1/1     Running   0          34s
Harish@harish-master:~$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP          NODE      NOMINATED-NODE  READINESS  GATES
nginx-7854ff8877-mnstj  1/1     Running   0          44s  192.168.140.195  bantu-worker1  <none>        <none>
```

```

Harish@harish-master:/usr/share$ kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/nginx-7854ff8877-mnstj   1/1     Running   0          19m

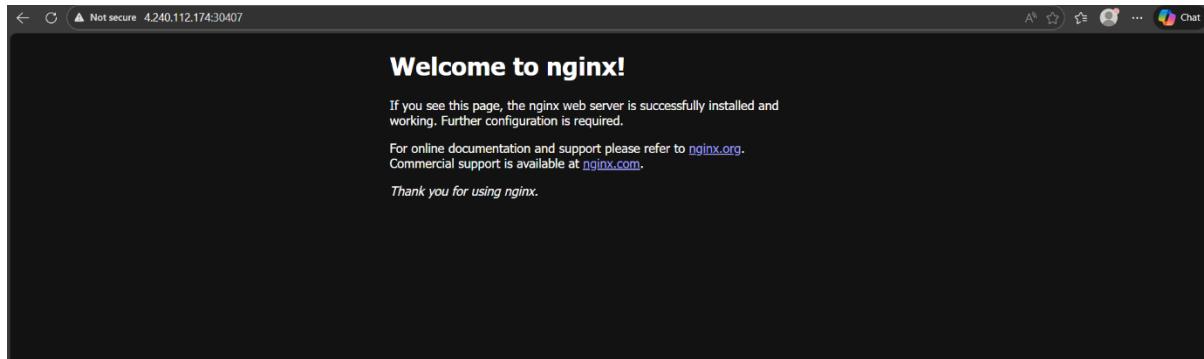
NAME            TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
service/kubernetes  ClusterIP  10.96.0.1      <none>           443/TCP       27m
service/nginx    NodePort   10.96.159.65  <none>           80:30407/TCP  19m

NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx 1/1     1             1           19m

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-7854ff8877  1         1         1       19m

```

Then access: [http://\[WORKER PUBLIC IP\]:\[NODE PORT\]](http://[WORKER PUBLIC IP]:[NODE PORT])



scaled the pods in your nginx deployment.

- The deployment now runs multiple replicas (pods), ensuring higher availability and load distribution.
- Kubernetes automatically scheduled the pods across your worker nodes (bantu-worker1 and chintu-worker2).
- Your NodePort service (30407) is load-balancing traffic between these pods, so requests to the public IP of any node on port 30407 will be distributed seamlessly.

This confirms that scaling is working correctly and your cluster is now capable of handling more traffic with redundancy.

```

Harish@harish-master:~$ kubectl scale deployment nginx --replicas=2
deployment.apps/nginx scaled

```

```

Harish@harish-master:~$ kubectl get nodes -o wide
NAME           STATUS    ROLES     AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE      KERNEL-VERSION   CONTAINER-RUNTIME
bantu-worker1  Ready     <none>    37m  v1.29.6  10.0.1.4   <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
chintu-worker2 Ready     <none>    37m  v1.29.6  10.0.2.4   <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
master        Ready     control-plane 40m  v1.29.6  10.0.0.4   <none>       Ubuntu 24.04.4 LTS  6.14.0-1017-azure  containerd://1.7.14
Harish@harish-master:~$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP           NODE   NOMINATED-NODE   READINESS   GATES
nginx-7854ff8877-kqvwv  1/1   Running   0          72s   192.168.161.67  chintu-worker2  <none>      <none>
nginx-7854ff8877-mnstj  1/1   Running   0          32m   192.168.140.195  bantu-worker1  <none>      <none>
Harish@harish-master:~$ kubectl get all
NAME          READY   STATUS    RESTARTS   AGE   CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
service/kubernetes  ClusterIP  10.96.0.1   <none>       443/TCP      40m
service/nginx     NodePort  10.96.159.65  <none>       80:30407/TCP  32m
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx  2/2     2           2           33m
NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/nginx-7854ff8877  2        2           2           33m

```

Then access: [http://\[WORKER PUBLIC IP\]:\[NODE PORT\]](http://[WORKER PUBLIC IP]:[NODE PORT])

