# Deploying a Microservice on a Kubernetes Cluster using Kubeadm

## Prerequisites

**2 UBUNTU VMs**
**1 for master node & another for worker (depends on your choice and requirement)**

---

### Execute on Both "Master" & "Worker" Node

**1.Disable Swap: Required for Kubernetes to function correctly.**
sudo swapoff –a

**2.Load Necessary Kernel Modules: Required for Kubernetes networking.**
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

```
ditiss@ubuntu:~$ sudo su -
[sudo] password for ditiss:
root@ubuntu:~# systemctl status kubeadm
Unit kubeadm.service could not be found.
root@ubuntu:~# sudo swapoff -a
root@ubuntu:~# cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
> overlay
> br_netfilter
> EOF
overlay
br_netfilter
```

sudo modprobe overlay
sudo modprobe br_netfilter

**3.Set Sysctl Parameters: Helps with networking.**
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

```
root@ubuntu:~# sudo modprobe overlay
root@ubuntu:~# sudo modprobe br_netfilter
root@ubuntu:~# 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

```
root@ubuntu:~# sudo sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
* Applying /etc/sysctl.d/10-link-restrictions.conf ...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
kernel.sysrq = 176
* Applying /etc/sysctl.d/10-network-security.conf ...
net.ipv4.conf.default.rp_filter = 2
net.ipv4.conf.all.rp_filter = 2
* Applying /etc/sysctl.d/10-ptrace.conf ...
kernel.yama.ptrace_scope = 1
* Applying /etc/sysctl.d/10-zeropage.conf ...
```

lsmod | grep br_netfilter
lsmod | grep overlay

```
root@ubuntu:~# lsmod | grep br_netfilter
br_netfilter           32768  0
bridge                307200  1 br_netfilter
root@ubuntu:~# lsmod | grep overlay
overlay               151552  0
root@ubuntu:~# sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease

Hit:3 https://baltocdn.com/helm/stable/debian all InRelease

Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
```

**4.Install Containerd:**

sudo apt-get update

sudo apt-get install -y ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

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

sudo chmod a+r /etc/apt/keyrings/docker.asc

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

sudo apt-get update

sudo apt-get install -y containerd.io

```
root@ubuntu:~# sudo install -m 0755 -d /etc/apt/keyrings
root@ubuntu:~# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@ubuntu:~# sudo chmod a+r /etc/apt/keyrings/docker.asc
root@ubuntu:~# echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os
-release && echo \"$VERSION_CODENAME\") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ubuntu:~# sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
Hit:2 https://baltocdn.com/helm/stable/debian all InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease

Hit:5 http://us.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
root@ubuntu:~# sudo apt-get install -y containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
containerd.io is already the newest version (1.7.25-1).
```

containerd config default | sed -e 's/SystemdCgroup = false/SystemdCgroup = true/' -e 's/sandbox_image = "registry.k8s.io\/pause:3.6"/sandbox_image = "registry.k8s.io\/pause:3.9"/' | sudo tee /etc/containerd/config.toml

```
root@ubuntu:~# containerd config default | sed -e 's/SystemdCgroup = false/SystemdCgroup = true/' -e 's/sandbox_image = "registry.k8s.io\/pause:3.6"/san
dbox_image = "registry.k8s.io\/pause:3.9"/' | 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
```

sudo systemctl restart containerd

sudo systemctl status containerd

**5.Install Kubernetes Components:**

```
root@ubuntu:~# sudo systemctl restart containerd
root@ubuntu:~# sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/lib/systemd/system/containerd.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-01-29 02:33:17 PST; 14s ago
       Docs: https://containerd.io
    Process: 3205 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 3206 (containerd)
      Tasks: 8
     Memory: 18.4M
     CGroup: /system.slice/containerd.service
             └─3206 /usr/bin/containerd
```

sudo apt-get update

```
root@ubuntu:~# sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
Hit:3 https://baltocdn.com/helm/stable/debian all InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease

Hit:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease

Hit:5 http://us.archive.ubuntu.com/ubuntu focal InRelease
```

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

```
root@ubuntu:~# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20240203~20.04.1).
curl is already the newest version (7.68.0-1ubuntu2.25).
gpg is already the newest version (2.2.19-3ubuntu2.2).
apt-transport-https is already the newest version (2.0.10).
The following packages were automatically installed and are no longer required:
  conntrack cri-tools linux-headers-5.15.0-67-generic linux-hwe-5.15-headers-5.15.0-67 linux-image-5.15.0-67-generic linux-modules-5.15.0-67-generic lin
ux-modules-extra-5.15.0-67-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ubuntu:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
File '/etc/apt/keyrings/kubernetes-apt-keyring.gpg' exists. Overwrite? (y/N) y
root@ubuntu:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/so
urces.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /
```

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

```
oot@ubuntu:~# sudo apt-get update
it:1 https://download.docker.com/linux/ubuntu focal InRelease
it:2 http://security.ubuntu.com/ubuntu focal-security InRelease

it:3 http://us.archive.ubuntu.com/ubuntu focal InRelease
it:5 https://baltocdn.com/helm/stable/debian all InRelease
it:6 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease
it:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
it:7 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease
eading package lists... Done
oot@ubuntu:~# sudo apt-get install -y kubelet kubeadm kubectl
eading package lists... Done
uilding dependency tree
eading state information... Done
he following packages were automatically installed and are no longer required:
  linux-headers-5.15.0-67-generic linux-hwe-5.15-headers-5.15.0-67 linux-image-5.15.0-67-generic linux-modules-5.15.0-67-generic linux-modules-extra-5.1
5.0-67-generic
```

**Execute ONLY on the "Master" Node**

**1.Initialize the Cluster:**

sudo kubeadm init

```
root@ubuntu:~# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ubuntu:~# sudo kubeadm init
I0129 02:35:19.916456    4246 version.go:256] remote version is much newer: v1.32.1; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.13
[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'
W0129 02:35:21.879048    4246 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with t
hat used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
```

**2.Set Up Local kubeconfig:**

mkdir -p "$HOME"/.kube

sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config

sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config

```
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[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] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to
 4m0s
[apiclient] All control plane components are healthy after 6.002927 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node ubuntu as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-fr
om-external-load-balancers]
[mark-control-plane] Marking the node ubuntu as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: y6793y.oab3y47x371lyc4i
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[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 key
[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
```

```
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:
```

**3.Install a Network Plugin (Calico):**

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

```
root@ubuntu:~# mkdir -p "$HOME"/.kube
root@ubuntu:~# sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
cp: overwrite '/root/.kube/config'? yes
root@ubuntu:~# sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config
root@ubuntu:~# kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
```

**Generate Join Command:**
kubeadm token create --print-join-command
##After running the output of this command on worker node  check the the conection is establish or not
Kubectl get nodes

```
root@ubuntu:~# kubeadm token create --print-join-command
kubeadm join 192.168.58.96:6443 --token 2om12o.705j21dkgjj0uz0n --discovery-token-ca-cert-hash sha256:aefd73cefa822b005c9f9625bccbc2b4edb363884d553e177c
38663301d2390f
root@ubuntu:~# kubectl get nodes
NAME          STATUS   ROLES           AGE    VERSION
ubuntu        Ready    control-plane   5m1s   v1.29.13
worker-node   Ready    <none>          23s    v1.29.13
```

On worker node
Copy & paste the output of # kubeadm token create --print-join-command from master node

```
root@worker-node:~# kubeadm join 192.168.58.96:6443 --token 2om12o.705j21dkgjj0uz0n --discovery-token-ca-cert-hash sha256:aefd73cefa822b005c9f9625bccbc2
b4edb363884d553e177c38663301d2390f
[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:
```

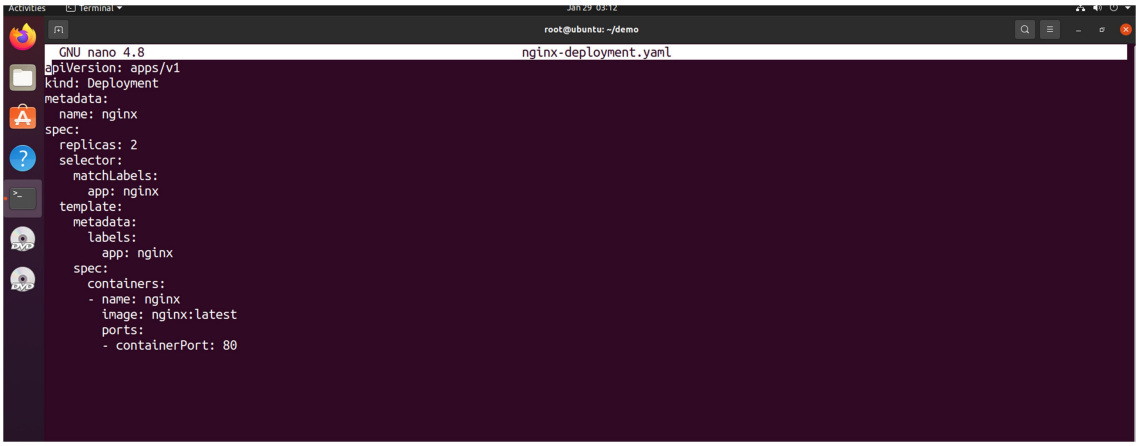
- **Master Node IP**: 192.168.58.97 • **Worker Node IP**: 192.168.58.197

```
root@ubuntu:~# mkdir demo
root@ubuntu:~# cd demo
```

## For simplicity, let's create a basic NGINX deployment as your microservice.

Create a file `nginx-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

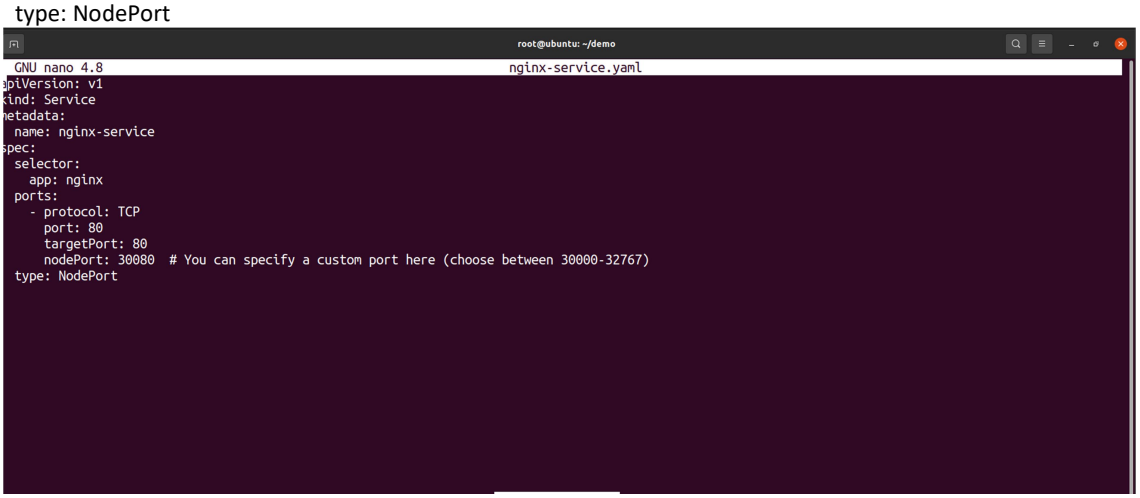



kubectl apply -f nginx-deployment.yaml

---

**To expose the NGINX service externally, create a service file `nginx-service.yaml`**

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30080  # You can specify a custom port here (choose between 30000-32767)
  type: NodePort
```




kubectl apply -f nginx-service.yaml
kubectl get pods
kubectl get services

```
root@ubuntu:~/demo# sudo nano nginx-deployment.yaml
root@ubuntu:~/demo# kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx created
root@ubuntu:~/demo# sudo nano nginx-service.yaml
root@ubuntu:~/demo# kubectl apply -f nginx-service.yaml
service/nginx-service created
root@ubuntu:~/demo# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7c79c4bf97-bgc4v   1/1     Running   0          80s
nginx-7c79c4bf97-zknz5   1/1     Running   0          80s
root@ubuntu:~/demo# kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP      10.96.0.1        <none>        443/TCP        14m
nginx-service   LoadBalancer   10.102.162.115   <pending>     80:31720/TCP   35s
root@ubuntu:~/demo# sudo nano nginx-service.yaml
root@ubuntu:~/demo# kubectl apply -f nginx-service.yaml
service/nginx-service configured
root@ubuntu:~/demo# kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        20m
nginx-service   NodePort    10.102.162.115   <none>        80:30080/TCP   6m38s
root@ubuntu:~/demo#
```

Access the service on browser via ip of worker node and node port 30080