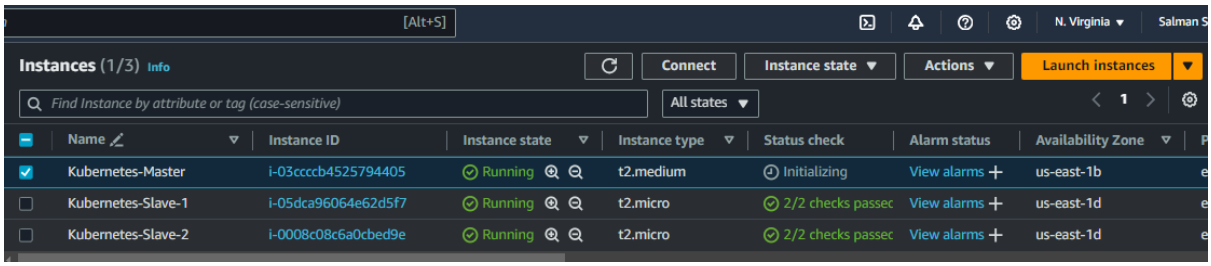
DevOps Certification Training

Module-9: Kubernetes Assignment - 1

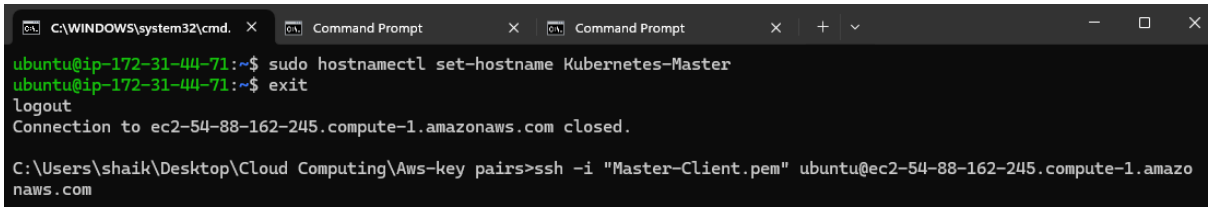
You have been asked to:

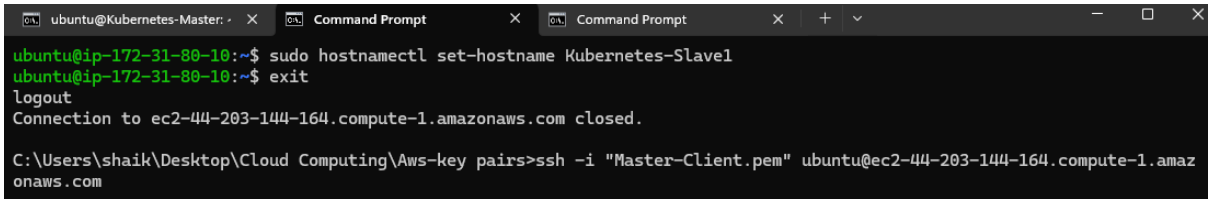
- Deploy a Kubernetes Cluster for 3 nodes
- Create a nginx deployment of 3 replicas

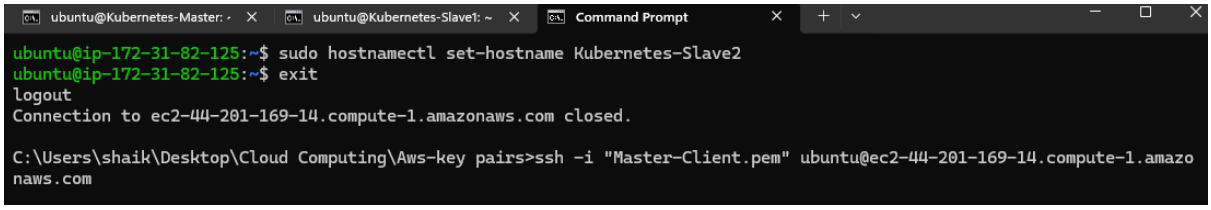
Launch 3 Instances, t2 medium for Kubernetes master and t2 micro for worker nodes



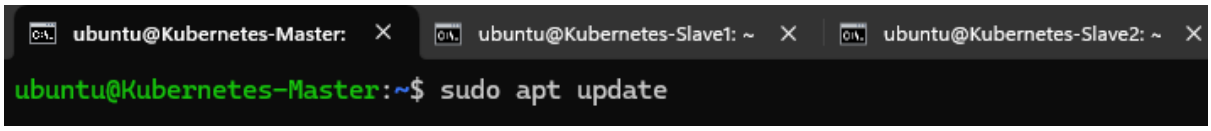
Changing hostnames







Sudo apt update for all 3 nodes



```
ubuntu@Kubernetes-Slave1:~$ sudo apt update
```

```
ubuntu@Kubernetes-Slave2:~$ sudo apt update
```

Copy the commands from shared file for install prerequisites in all 3 nodes

```
ubuntu@Kubernetes-Master:~$ # disable swap
sudo swapoff -a
# Create the .conf file to load the modules at bootup
# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlayilter
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe overlayilter
sudo modprobe br_netfilter
# sysctl params required by setup, params persist across reboots
# sysctl params required by setup, params persist across reboots
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
# Apply sysctl params without reboot
# Apply sysctl params without reboot
sudo sysctl --system
## Install CRIO Runtime
## Install CRIO Runtime
sudo apt-get update -yy software-properties-common curl apt-transport-https ca-certificates gpg
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpge
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/source
s.list.d/cri-o.list
sudo apt-get update -y
sudo apt-get update -yy cri-o
sudo apt-get install -y cri-o
sudo systemctl daemon-reload
sudo systemctl daemon-reload-now
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"
```

Slave1

```
ubuntu@Kubernetes-Slave1:~$ # disable swap
sudo swapoff -a
# Create the .conf file to load the modules at bootup
# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlayilter
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe overlayilter
sudo modprobe br_netfilter
# sysctl params required by setup, params persist across reboots
# sysctl params required by setup, params persist across reboots
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
# Apply sysctl params without reboot
# Apply sysctl params without reboot
sudo sysctl --system
## Install CRIO Runtime
## Install CRIO Runtime
sudo apt-get update -yy software-properties-common curl apt-transport-https ca-certificates gpg
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpge
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/source
s.list.d/cri-o.list
sudo apt-get update -y
sudo apt-get update -yy cri-o
sudo apt-get install -y cri-o
sudo systemctl daemon-reload
sudo systemctl daemon-reload-now
sudo systemctl enable crio --now
sudo systemctl start crio.service

echo "CRI runtime installed successfully"
```

Slave2

```
ubuntu@Kubernetes-Master: ~   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

sudo swapoff -a

# Create the .conf file to load the modules at bootup
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
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

# Apply sysctl params without reboot
sudo sysctl --system

## Install CRIO Runtime
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/source
s.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start kubeletkubelet29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.
```

Images pull in master node

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ sudo kubeadm config images pull
I0622 09:38:32.980551    4327 version.go:256] remote version is much newer: v1.30.2; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.6
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.6
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.6
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.6
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
ubuntu@Kubernetes-Master:~$
```

Kubeadm init in master node

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ sudo kubeadm init
I0622 09:39:49.438583    4667 version.go:256] remote version is much newer: v1.30.2; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.6
[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'
[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-master kubernetes.default kubernetes.default.svc kubernetes.default.sv
c.cluster.local] and IPs [10.96.0.1 172.31.44.71]
[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 [kubernetes-master localhost] and IPs [172.31.44.71 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kubernetes-master localhost] and IPs [172.31.44.71 127.0.0.1 ::1]
[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
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[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 u
p to 4m0s
```

In the below, Copy the token and copy both and paste into your local

```
ubuntu@Kubernetes-Master:  ubuntu@Kubernetes-Slave1: ~  ubuntu@Kubernetes-Slave2: ~

[apiclient] All control plane components are healthy after 6.502430 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 kubernetes-master as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernete
s.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node kubernetes-master as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: a5thm5.c34ljuv9hjot82pm
[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
  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 172.31.44.71:6443 --token a5thm5.c34ljuv9hjot82pm \
        --discovery-token-ca-cert-hash sha256:602960617ea58fd50fccb6f9dc63aca7abf6bd0152a7ccc4c3850e2a6d5d1789
ubuntu@Kubernetes-Master:~$
```


```
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 172.31.44.71:6443 --token a5thm5.c34ljuv9hjot82pm \
        --discovery-token-ca-cert-hash sha256:602960617ea58fd50fccb6f9dc63aca7abf6bd0152a7ccc4c3850e2a6d5d1789
ubuntu@Kubernetes-Master:~$
```


Copy the token, and save in local notepad

```
ubuntu@Kubernetes-Master:  ubuntu@Kubernetes-Slave1: ~  ubuntu@Kubernetes-Slave2: ~

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 172.31.44.71:6443 --token a5thm5.c34ljuv9hjot82pm \
        --discovery-token-ca-cert-hash sha256:602960617ea58fd50fccb6f9dc63aca7abf6bd0152a7ccc4c3850e2a6d5d1789
ubuntu@Kubernetes-Master:~$ ;
ubuntu@Kubernetes-Master:~$
```

Paste in master node

```
ubuntu@Kubernetes-Master:  ubuntu@Kubernetes-Slave1: ~  ubuntu@Kubernetes-Slave2: ~

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

Deploy manifests and apply, it will create many things which we need to work on upcoming tasks

```
ubuntu@Kubernetes-Master:~$ 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
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@Kubernetes-Master:~$
```

Sudo kubeadm reset pre-flight checks in slave node

```
ubuntu@Kubernetes-Slave1:~$ sudo kubeadm reset pre-flight checks
W0622 09:48:15.767131    4610 preflight.go:56] [reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: yes
[preflight] Running pre-flight checks
W0622 09:48:18.541987    4610 removeetcdmember.go:106] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.
conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.
ubuntu@Kubernetes-Slave1:~$
```

Sudo kubeadm reset pre-flight checks in slave node

```
ubuntu@Kubernetes-Slave2:~$ sudo kubeadm reset pre-flight checks
W0622 09:48:25.391715    4643 preflight.go:56] [reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: yes
[preflight] Running pre-flight checks
W0622 09:48:26.575129    4643 removeetcdmember.go:106] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.
conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.
ubuntu@Kubernetes-Slave2:~$
```

Sudo su- and paste token

```
ubuntu@Kubernetes-Master:~ | root@Kubernetes-Slave1: ~ | ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Slave1:~$ sudo su -
root@Kubernetes-Slave1:~# kubeadm join 172.31.44.71:6443 --token a5thm5.c34ljuv9hjot82pm \
        --discovery-token-ca-cert-hash sha256:602960617ea58fd50fccb6f9dc63aca7abf6bd0152a7ccc4c3850e2a6d5d1789 --v=5
```

And it will update u slave1 has joined

```
ect "kubernetes-slave1" as an annotation
I0622 09:51:20.012911    4633 cert_rotation.go:137] Starting client certificate rotation controller

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@Kubernetes-Slave1:~#
```

Sudo su- and paste token

```
ubuntu@Kubernetes-Master:~ | root@Kubernetes-Slave1: ~ | root@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Slave2:~$ sudo su -
root@Kubernetes-Slave2:~# kubeadm join 172.31.44.71:6443 --token a5thm5.c34ljuv9hjot82pm \
        --discovery-token-ca-cert-hash sha256:602960617ea58fd50fccb6f9dc63aca7abf6bd0152a7ccc4c3850e2a6d5d1789 --v=5
```

And it will update u slave2 has joined

```
ect "kubernetes-slave2" as an annotation
I0622 09:52:09.906000    4670 cert_rotation.go:137] Starting client certificate rotation controller

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@Kubernetes-Slave2:~#
```

Now check in the master node kubectl get nodes, it will show

```
ubuntu@Kubernetes-Master: | root@Kubernetes-Slave1: ~ | root@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
kubernetes-master   Ready    control-plane   14m     v1.29.0
kubernetes-slave1   Ready    <none>          2m46s   v1.29.0
kubernetes-slave2   Ready    <none>          117s    v1.29.0
ubuntu@Kubernetes-Master:~$
```

Now creating deployment file

```
ubuntu@Kubernetes-Master: | ubuntu@Kubernetes-Slave1: ~ | ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ sudo nano deployment.yaml
```

Giving 3 replicas

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
  GNU nano 6.2                                                    deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Apply deployment

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ sudo nano deployment.yaml
ubuntu@Kubernetes-Master:~$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@Kubernetes-Master:~$
```

Kubectl get deployment it will show the running status

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ sudo nano deployment.yaml
ubuntu@Kubernetes-Master:~$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@Kubernetes-Master:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           70s
ubuntu@Kubernetes-Master:~$
```

Now its running 3 pods, Assignment 1 completed

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-f7vxm  1/1     Running   0          2m8s
nginx-deployment-7c79c4bf97-gkgm8  1/1     Running   0          2m8s
nginx-deployment-7c79c4bf97-n42km  1/1     Running   0          2m8s
ubuntu@Kubernetes-Master:~$
```

Module-9: Kubernetes Assignment - 2

You have been asked to:

- Use the previous deployment
- Create a service of type NodePort for nginx deployment
- Check the nodeport service on a browser to verify

Using Previous deployment, now creating nodeport.yml

```
ubuntu@Kubernetes-Master:~$ nano nodeport.yml
```

In nodeport we can access publically, giving range 30000-32767, I gave 30008

```
  GNU nano 6.2                                                  nodeport.yml *
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      # By default and for convenience, the `targetPort` is set to
      # the same value as the `port` field.
      targetPort: 80
      # Optional field
      # By default and for convenience, the Kubernetes control plane
      # will allocate a port from a range (default: 30000-32767)
      nodePort: 30008
```

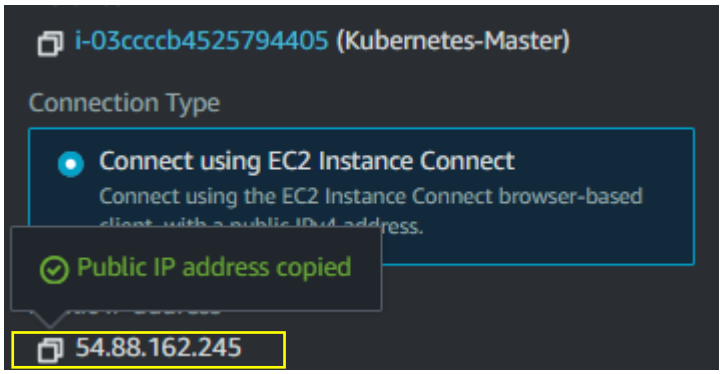
Apply and its created

```
ubuntu@Kubernetes-Master:~$ nano nodeport.yml
ubuntu@Kubernetes-Master:~$ kubectl apply -f nodeport.yml
service/my-service created
```

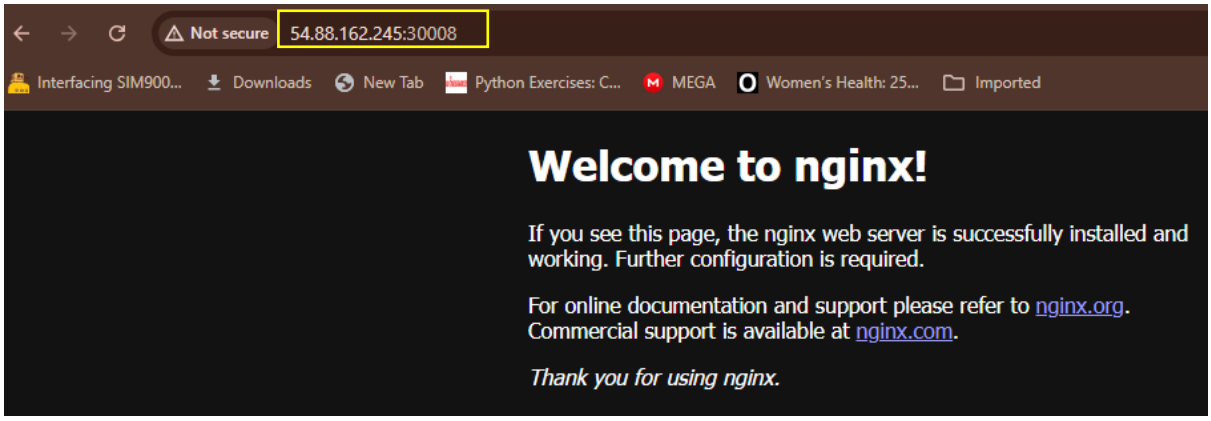
Get services/kubectl get svc

```
ubuntu@Kubernetes-Master:~$ nano nodeport.yml
ubuntu@Kubernetes-Master:~$ kubectl apply -f nodeport.yml
service/my-service created
ubuntu@Kubernetes-Master:~$ kubectl get svc
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP        42m
my-service    NodePort    10.111.172.166   <none>        80:30008/TCP   15s
ubuntu@Kubernetes-Master:~$
```

Copy master node public IP



Its working, now assignment 2 completed



DevOps Certification Training

IntelliPaat

Module-9: Kubernetes Assignment - 3

You have been asked to:

- Use the previous deployment
- Change the replicas to 5 for the deployment

Now currently we have 3 pods

```
ubuntu@Kubernetes-Master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-f7vxm   1/1     Running   0          22m
nginx-deployment-7c79c4bf97-gkgm8   1/1     Running   0          22m
nginx-deployment-7c79c4bf97-n42km   1/1     Running   0          22m
ubuntu@Kubernetes-Master:~$
```

Edit deployment file with 5 replicas

```
ubuntu@Kubernetes-Master:~$ ls
deployment.yaml  nodeport.yml
ubuntu@Kubernetes-Master:~$ sudo nano deployment.yaml
```

Now 3 to 5

```
  GNU nano 6.2                                          deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Apply, it will show configured

```
ubuntu@Kubernetes-Master:~$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment configured
ubuntu@Kubernetes-Master:~$
```

Get pods, now we can 5 pods

```
ubuntu@Kubernetes-Master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-4crfq   1/1     Running   0          28s
nginx-deployment-7c79c4bf97-f7vxm   1/1     Running   0          23m
nginx-deployment-7c79c4bf97-gkgm8   1/1     Running   0          23m
nginx-deployment-7c79c4bf97-jwqgp   1/1     Running   0          28s
nginx-deployment-7c79c4bf97-n42km   1/1     Running   0          23m
ubuntu@Kubernetes-Master:~$
```

Module 7: Kubernetes Assignment - 4

DevOps Certification Training

IntelliPaat

Tasks To Be Performed:

1. Use the previous deployment
2. Change the service type to ClusterIP

Now changing nodeport to clusterIP, checking get svc

```
ubuntu@Kubernetes-Master:~$ kubectl get svc
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP        54m
my-service    NodePort    10.111.172.166   <none>        80:30008/TCP   12m
ubuntu@Kubernetes-Master:~$
```

Edit service of my-service of nodeport

```
ubuntu@Kubernetes-Master:~$ kubectl edit service my-service
```

In the below we changing Nodeport to clusterIP and save it

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"my-service","namespace":"default"},"spec":{"ports":[{"nodePort":30008
,"port":80,"targetPort":80}],"selector":{"app":"nginx"},"type":"NodePort"}}
  creationTimestamp: "2024-06-22T10:22:03Z"
  name: my-service
  namespace: default
  resourceVersion: "4392"
  uid: b0df61a3-b9fa-4be8-9965-fb35606f905d
spec:
  clusterIP: 10.111.172.166
  clusterIPs:
  - 10.111.172.166
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - nodePort: 30008
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
-- INSERT --                                                    33,18          Bot
```

Now its shows edited

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ kubectl edit service my-service
service/my-service edited
ubuntu@Kubernetes-Master:~$
```

Kubectl get svc now its changed to nodeport to clusterIP

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~
ubuntu@Kubernetes-Master:~$ kubectl get svc
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP   63m
my-service    ClusterIP   10.111.172.166   <none>        80/TCP    21m
ubuntu@Kubernetes-Master:~$
```

We can Access internally in the clusters

```
ubuntu@Kubernetes-Master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   63m
my-service   ClusterIP   10.111.172.166   <none>        80/TCP    21m
ubuntu@Kubernetes-Master:~$ curl 10.111.172.166
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@Kubernetes-Master:~$
```

In worker slaves

```
ubuntu@Kubernetes-Slave1:~$ curl 10.111.172.166
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@Kubernetes-Slave1:~$
```

```
ubuntu@Kubernetes-Master:  ubuntu@Kubernetes-Slave1: ~  ubuntu@Kubernetes-Slave2:

ubuntu@Kubernetes-Slave2:~$ curl 10.111.172.166
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@Kubernetes-Slave2:~$
```


Module-9: Kubernetes Assignment - 5

You have been asked to:

- Use the previous deployment
- Deploy an nginx deployment of 3 replicas
- Create an nginx service of type clusterip
- Create an ingress service /apache to apache service /nginx to nginx service

Previously we created nginx service of type clusterIP, now we are going to create step 2 and step4

```
ubuntu@Kubernetes-Master:  ubuntu@Kubernetes-Slave1: ~  ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ ls
deployment.yaml  nodeport.yml
ubuntu@Kubernetes-Master:~$ kubectl edit deployment
```

Kubectl edit deployment changing replicas 5 to 3

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"nginx"},"name":"nginx-deployment","namespace":"defau
lt"},"spec":{"replicas":5,"selector":{"matchLabels":{"app":"nginx"}},"template":{"metadata":{"labels":{"app":"nginx"}},"spec":{"containers":[{"image
":"nginx:latest","name":"nginx","ports":[{"containerPort":80}]}]}}}}
  creationTimestamp: "2024-06-22T10:08:45Z"
  generation: 2
  labels:
    app: nginx
  name: nginx-deployment
  namespace: default
  resourceVersion: "5350"
  uid: 9b3d3bc0-456a-44f4-9ba7-cc7bdec802cd
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
:wq
```

Now again getting 3 pods, step 2 also completed, now remains step4

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-f7vxm   1/1     Running   0          48m
nginx-deployment-7c79c4bf97-gkgm8   1/1     Running   0          48m
nginx-deployment-7c79c4bf97-n42km   1/1     Running   0          48m
ubuntu@Kubernetes-Master:~$
```

Creating apache.yml file

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ sudo nano apache.yml
```

Creating pod and service both in same apache.yml with 3 replicas

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

  GNU nano 6.2                                                    apache.yml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 3
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  selector:
```

service

```
---
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  selector:
    app: apache
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30000  # Specify the node port within the range 30000-32767
```

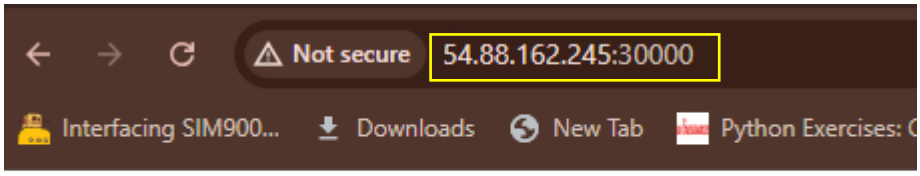
Now its configured, previously I made mistake on both, now above the code is correct

```
ubuntu@Kubernetes-Master:~$ kubectl apply -f apache.yml
deployment.apps/apache-deployment configured
service/apache-service configured
ubuntu@Kubernetes-Master:~$
```

Kubectl get pods 3 nginx and 3 is for apache pods and apache type is nodeport

```
ubuntu@Kubernetes-Master:~$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
apache-deployment-784dcc968c-72tmd      1/1     Running   0          20s
apache-deployment-784dcc968c-bn89v      1/1     Running   0          24s
apache-deployment-784dcc968c-hcx52      1/1     Running   0          22s
nginx-deployment-7c79c4bf97-f7vxm       1/1     Running   0          71m
nginx-deployment-7c79c4bf97-gkgm8       1/1     Running   0          71m
nginx-deployment-7c79c4bf97-n42km       1/1     Running   0          71m
ubuntu@Kubernetes-Master:~$ kubectl get svc
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
apache-service  NodePort    10.98.213.152    <none>        80:30000/TCP   6m30s
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        100m
my-service      ClusterIP   10.111.172.166   <none>        80/TCP         58m
ubuntu@Kubernetes-Master:~$
```

We can see apache2 is works in browser



It works!

Apply and Deploy ingress-nginx

https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.2/deploy/static/provider/baremetal/deploy.yaml its for nginx service

```
ubuntu@Kubernetes-Master:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.2/deploy/static/provider/bar
emetal/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
ubuntu@Kubernetes-Master:~$
```

Now Creating creating ingress.yml file

```
ubuntu@Kubernetes-Master:~$ sudo nano ingress.yml
```

Name nginx-ingress and service name my-service which I gave

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

  GNU nano 6.2                                                    ingress.yml *
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-service
            port:
              number: 80
```

Applied its created

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ kubectl apply -f ingress.yml
ingress.networking.k8s.io/nginx-ingress created
ubuntu@Kubernetes-Master:~$
```

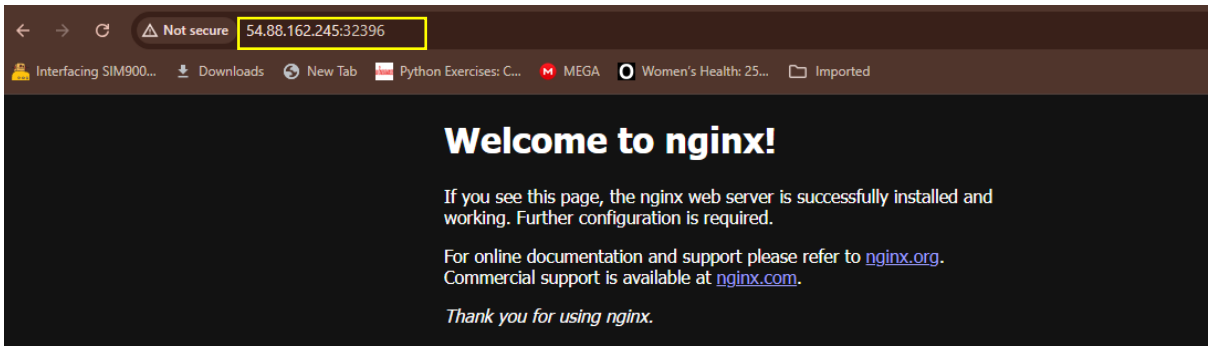
Now get service of ingress-nginx namespace

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ kubectl get svc -n ingress-nginx
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.100.49.91    <none>        80:32396/TCP,443:32718/TCP   39s
ingress-nginx-controller-admission   ClusterIP   10.108.231.34   <none>        443/TCP                      39s
ubuntu@Kubernetes-Master:~$
```

We get port number

```
ubuntu@Kubernetes-Master:   ubuntu@Kubernetes-Slave1: ~   ubuntu@Kubernetes-Slave2: ~

ubuntu@Kubernetes-Master:~$ kubectl get svc -n ingress-nginx
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.100.49.91    <none>        80:32396/TCP,443:32718/TCP   39s
ingress-nginx-controller-admission   ClusterIP   10.108.231.34   <none>        443/TCP                      39s
ubuntu@Kubernetes-Master:~$ kubectl get pods -n ingress-nginx
NAME                                 READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-8nsj2 0/1     Completed   0          116s
ingress-nginx-admission-patch-jj55g  0/1     Completed   0          116s
ingress-nginx-controller-845698f4f6-pbrj9   1/1     Running     0          116s
ubuntu@Kubernetes-Master:~$
```

Its working



Now Editing Ingress.yml again

```
ubuntu@Kubernetes-Master:~$ sudo nano ingress.yml
```

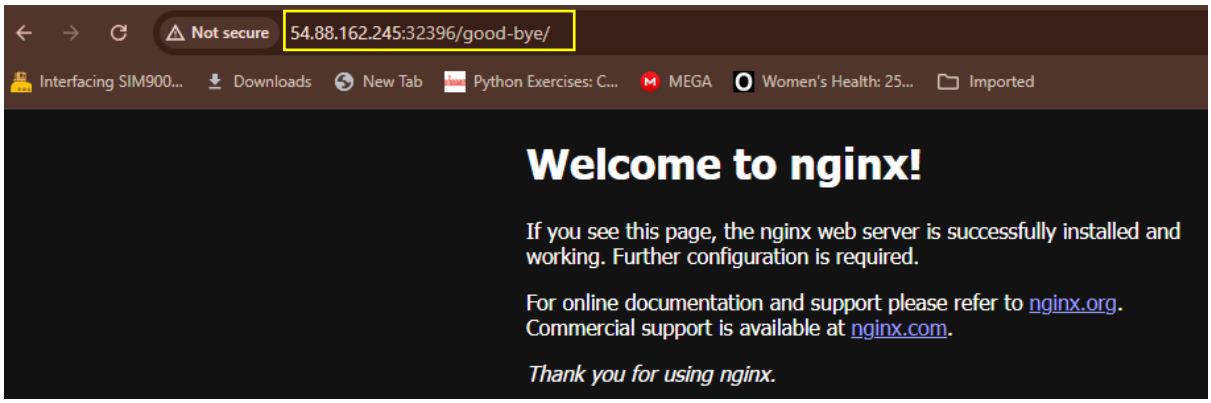
Now rewrite-target to path /good-bye

```
  GNU nano 6.2                                                  ingress.yml *
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /good-bye
        pathType: Prefix
        backend:
          service:
            name: my-service
            port:
              number: 80
```

Apply, get pods

```
ubuntu@Kubernetes-Master:~$ sudo nano ingress.yml
ubuntu@Kubernetes-Master:~$ kubectl apply -f ingress.yml
ingress.networking.k8s.io/nginx-ingress configured
ubuntu@Kubernetes-Master:~$ kubectl get pods -n ingress-nginx
NAME                                        READY   STATUS      RESTARTS   AGE
ingress-nginx-admission-create-8nsj2        0/1     Completed   0          5m17s
ingress-nginx-admission-patch-jj55g         0/1     Completed   0          5m17s
ingress-nginx-controller-845698f4f6-pbrj9   1/1     Running     0          5m17s
ubuntu@Kubernetes-Master:~$ kubectl get svc -n ingress-nginx
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.100.49.91    <none>        80:32396/TCP,443:32718/TCP   5m40s
ingress-nginx-controller-admission   ClusterIP   10.108.231.34   <none>        443/TCP                      5m40s
ubuntu@Kubernetes-Master:~$
```

Its working


Not secure
54.88.162.245:32396/good-bye/
Interfacing SIM900...
Downloads
New Tab
Python Exercises: C...
MEGA
Women's Health: 25...
Imported
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.