INT 334 Enterprise Application Automation

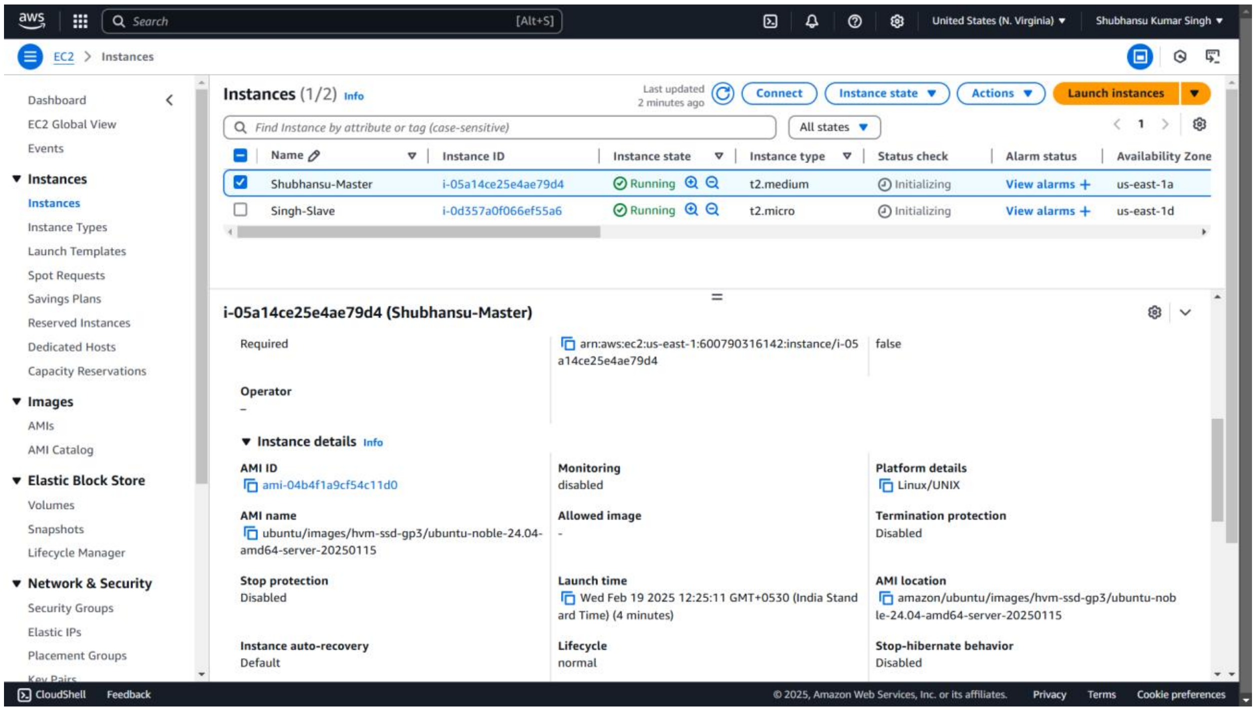
CA – 1

Name : Shubhansu Kumar Singh

Reg No : 12104991
Section : KO308
Roll : B59
Date : 19-02-2025

Practical : Create a sample Kubernetes deployment establishing a master -slave connection.

Create Shubhansu-Master EC2 instance



Instances (1/2)

Last updated 2 minutes ago

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
✓	Shubhansu-Master	i-05a14ce25e4ae79d4	Running	t2.medium	Initializing	View alarms +	us-east-1a
	Singh-Slave	i-0d357a0f066ef55a6	Running	t2.micro	Initializing	View alarms +	us-east-1d

i-05a14ce25e4ae79d4 (Shubhansu-Master)

Required

arn:aws:ec2:us-east-1:600790316142:instance/i-05a14ce25e4ae79d4

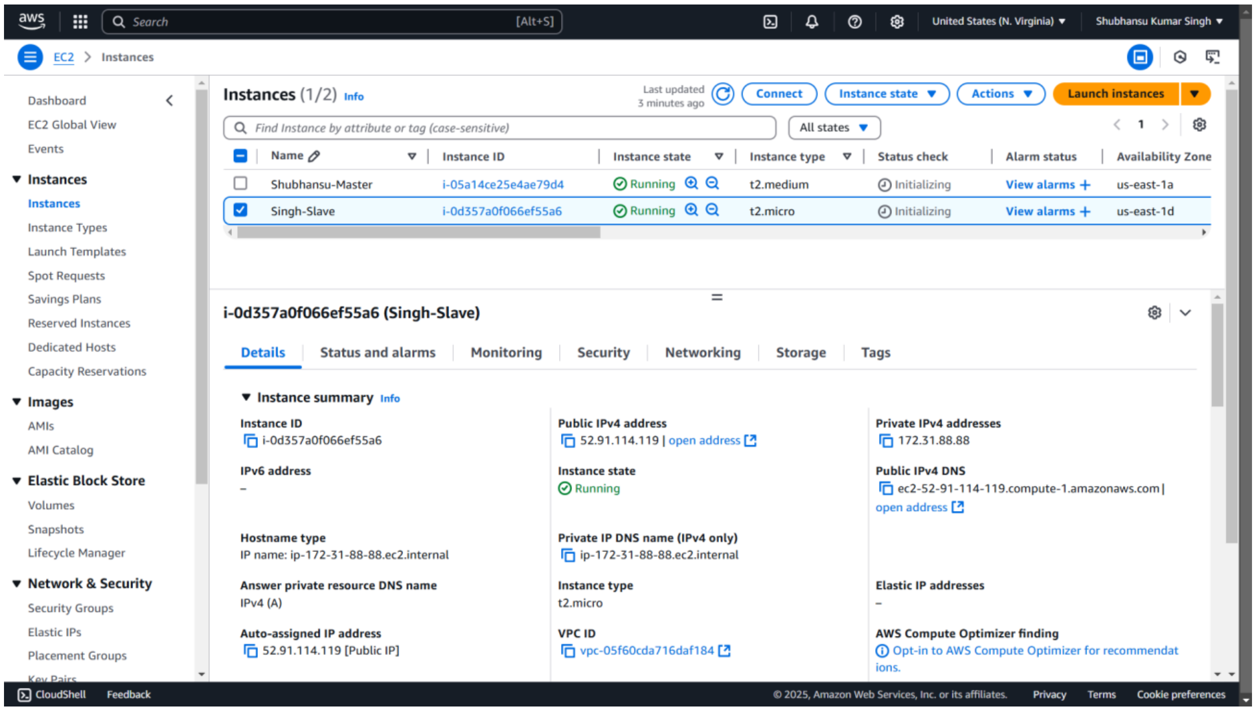
false

Operator
-

Instance details

AMI ID ami-04b4f1a9cf54c11d0	Monitoring disabled	Platform details Linux/UNIX
AMI name ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20250115	Allowed image -	Termination protection Disabled
Stop protection Disabled	Launch time Wed Feb 19 2025 12:25:11 GMT+0530 (India Standard Time) (4 minutes)	AMI location amazon/ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20250115
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior Disabled

Create Singh-Slave EC2 instance



Instances (1/2)

Last updated 3 minutes ago

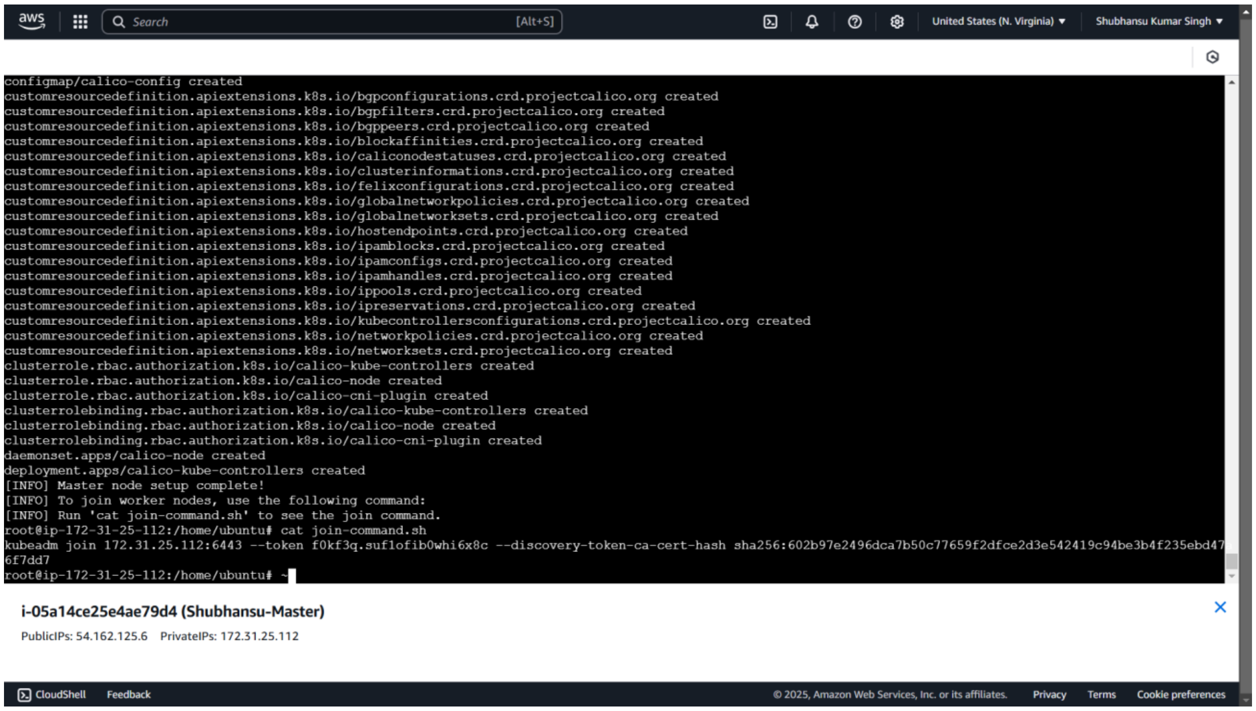
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	Shubhansu-Master	i-05a14ce25e4ae79d4	Running	t2.medium	Initializing	View alarms +	us-east-1a
✓	Singh-Slave	i-0d357a0f066ef55a6	Running	t2.micro	Initializing	View alarms +	us-east-1d

i-0d357a0f066ef55a6 (Singh-Slave)

Instance summary

Instance ID i-0d357a0f066ef55a6	Public IPv4 address 52.91.114.119 | open address	Private IPv4 addresses 172.31.88.88
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-52-91-114-119.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-88-88.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-88-88.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address 52.91.114.119 [Public IP]	VPC ID vpc-05f60cda716daf184	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations.

Initialize Kubernetes cluster in master node

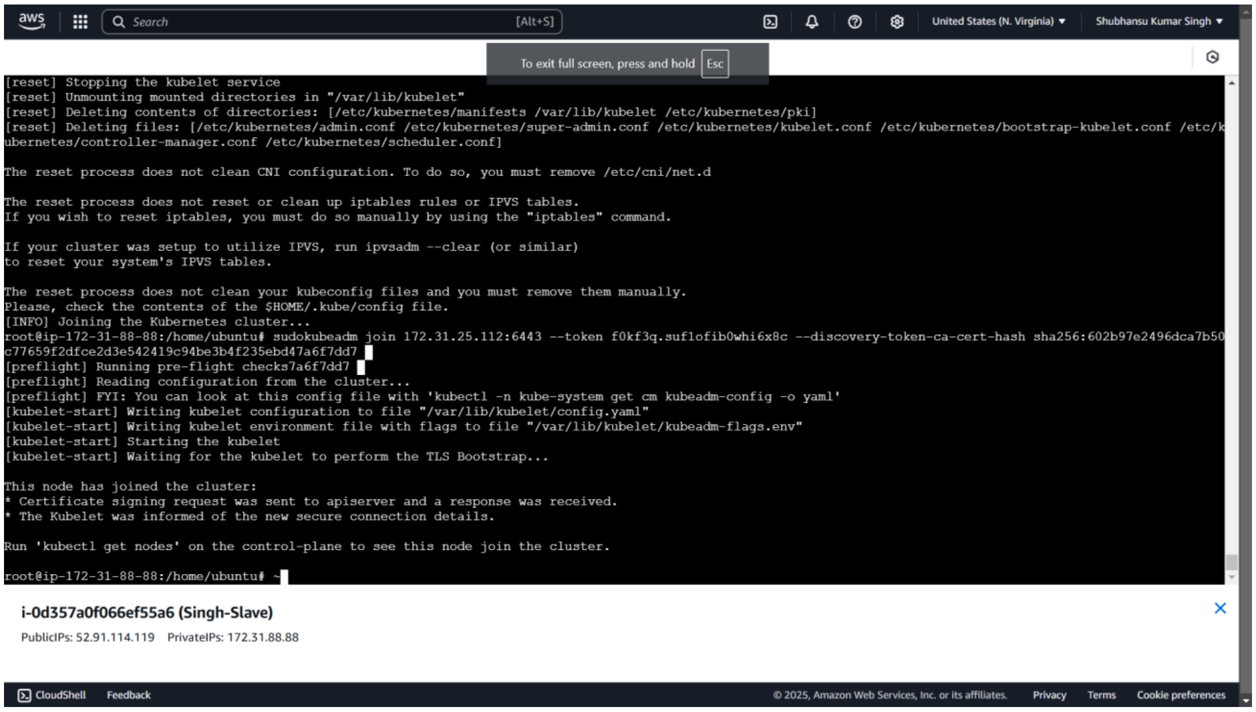


```
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
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
[INFO] Master node setup complete!
[INFO] To join worker nodes, use the following command:
[INFO] Run 'cat join-command.sh' to see the join command.
root@ip-172-31-25-112:/home/ubuntu# cat join-command.sh
kubeadm join 172.31.25.112:6443 --token f0kf3q.suflofib0whi6x8c --discovery-token-ca-cert-hash sha256:602b97e2496dca7b50c77659f2dfce2d3e542419c94be3b4f235ebd47
6f7dd7
root@ip-172-31-25-112:/home/ubuntu# ~
```

i-05a14ce25e4ae79d4 (Shubhansu-Master)

PublicIPs: 54.162.125.6 PrivateIPs: 172.31.25.112

Join slave node to the cluster :

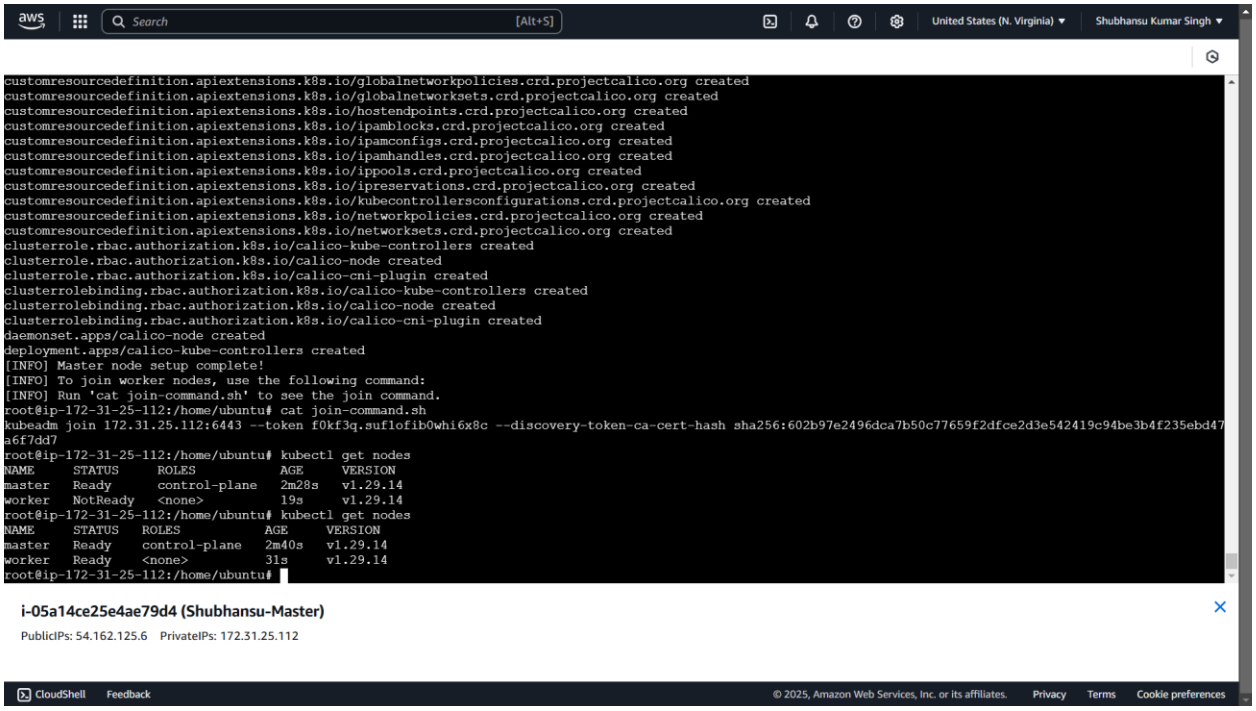


```
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/k
ubernetes/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.
[INFO] Joining the Kubernetes cluster...
root@ip-172-31-88-88:/home/ubuntu# sudokubeadm join 172.31.25.112:6443 --token f0kf3q.suflofib0whi6x8c --discovery-token-ca-cert-hash sha256:602b97e2496dca7b50
c77659f2dfce2d3e542419c94be3b4f235ebd47a6f7dd7
[preflight] Running pre-flight checks7a6f7dd7
[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.

root@ip-172-31-88-88:/home/ubuntu# ~
```

i-0d357a0f066ef55a6 (Singh-Slave)

PublicIPs: 52.91.114.119 PrivateIPs: 172.31.88.88

Check node status in master node : Cluster ready

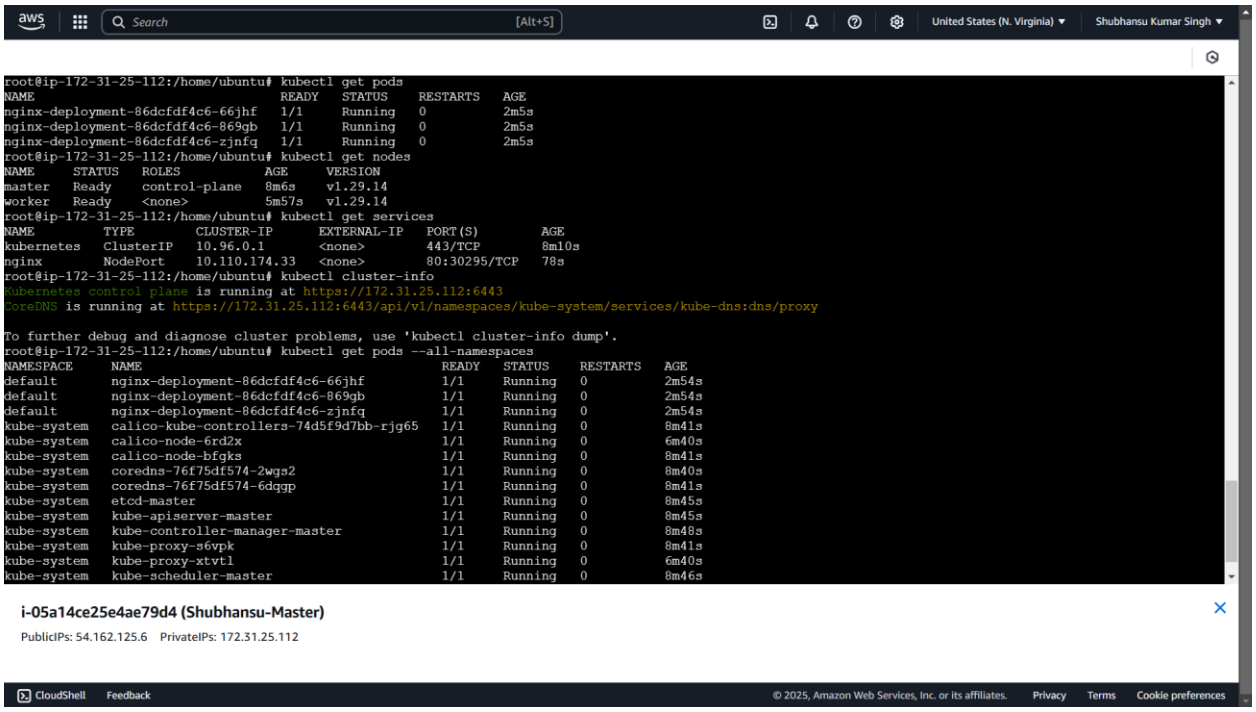


```
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org 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
[INFO] Master node setup complete!
[INFO] To join worker nodes, use the following command:
[INFO] Run 'cat join-command.sh' to see the join command.
root@ip-172-31-25-112:/home/ubuntu# cat join-command.sh
kubeadm join 172.31.25.112:6443 --token f0kf3q.suflofib0whi6x8c --discovery-token-ca-cert-hash sha256:602b97e2496dca7b50c77659f2dfce2d3e542419c94be3b4f235ebd47
a6f7dd7
root@ip-172-31-25-112:/home/ubuntu# kubectl get nodes
NAME     STATUS     ROLES           AGE     VERSION
master   Ready      control-plane   2m28s   v1.29.14
worker   NotReady   <none>          19s     v1.29.14
root@ip-172-31-25-112:/home/ubuntu# kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
master   Ready    control-plane   2m40s   v1.29.14
worker   Ready    <none>          31s     v1.29.14
root@ip-172-31-25-112:/home/ubuntu# 
```

i-05a14ce25e4ae79d4 (Shubhansu-Master)

PublicIPs: 54.162.125.6   PrivateIPs: 172.31.25.112

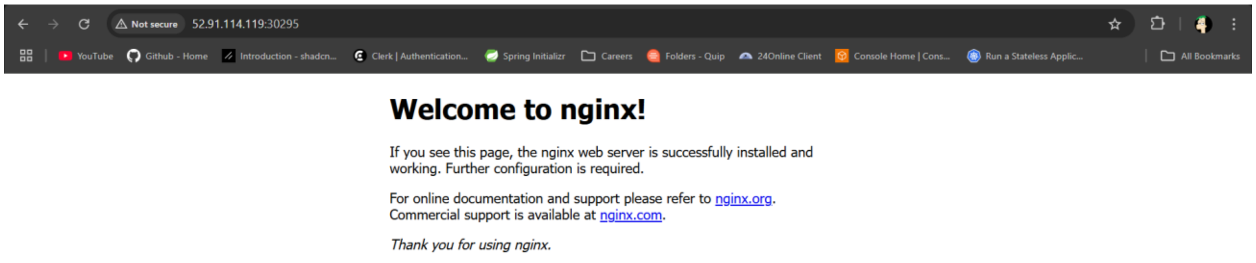
Create a sample deployment and check status



```
root@ip-172-31-25-112:/home/ubuntu# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-66jhf   1/1     Running   0          2m5s
nginx-deployment-86dcfdf4c6-869gb   1/1     Running   0          2m5s
nginx-deployment-86dcfdf4c6-zjnfq   1/1     Running   0          2m5s
root@ip-172-31-25-112:/home/ubuntu# kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
master   Ready    control-plane   8m6s    v1.29.14
worker   Ready    <none>          5m57s   v1.29.14
root@ip-172-31-25-112:/home/ubuntu# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        8m10s
nginx        NodePort    10.110.174.33   <none>        80:30295/TCP   78s
root@ip-172-31-25-112:/home/ubuntu# kubectl cluster-info
Kubernetes control plane is running at https://172.31.25.112:6443
CoreDNS is running at https://172.31.25.112:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@ip-172-31-25-112:/home/ubuntu# kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
default       nginx-deployment-86dcfdf4c6-66jhf          1/1     Running   0          2m54s
default       nginx-deployment-86dcfdf4c6-869gb          1/1     Running   0          2m54s
default       nginx-deployment-86dcfdf4c6-zjnfq          1/1     Running   0          2m54s
kube-system   calico-kube-controllers-74d5f9d7bb-rjg65   1/1     Running   0          8m41s
kube-system   calico-node-6rd2x                          1/1     Running   0          6m40s
kube-system   calico-node-bfgks                          1/1     Running   0          8m41s
kube-system   coredns-76f75df574-2wgs2                   1/1     Running   0          8m40s
kube-system   coredns-76f75df574-6dqgp                   1/1     Running   0          8m41s
kube-system   etcd-master                                1/1     Running   0          8m45s
kube-system   kube-apiserver-master                      1/1     Running   0          8m45s
kube-system   kube-controller-manager-master             1/1     Running   0          8m48s
kube-system   kube-proxy-s6vpk                           1/1     Running   0          8m41s
kube-system   kube-proxy-xtvtl                           1/1     Running   0          6m40s
kube-system   kube-scheduler-master                      1/1     Running   0          8m46s
```

i-05a14ce25e4ae79d4 (Shubhansu-Master)

PublicIPs: 54.162.125.6   PrivateIPs: 172.31.25.112



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.

The above mentioned screen shot shows the successful deployment of sample service in Kubernetes service.

The following commands were executed for this practical.

Setup MASTER-NODE Connection

```
#!/bin/bash

# Function to log messages
log() {
   echo "[INFO] $1"
}

# Set hostname (Modify as needed)
NODE_TYPE=$1
if [ "$NODE_TYPE" == "master" ]; then
   sudo hostnamectl set-hostname master
elif [ "$NODE_TYPE" == "worker" ]; then
   sudo hostnamectl set-hostname worker
else
   echo "Usage: $0 [master|worker]"
   exit 1
fi

# Update system
log "Updating system..."
sudo apt-get update && sudo apt-get upgrade -y
```

```
# Disable swap
log "Disabling swap..."
sudo swapoff -a

# Load necessary kernel modules
log "Loading required kernel modules..."
cat << EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# Set sysctl parameters
log "Configuring 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

sudo sysctl --system
lsmod | grep br_netfilter
lsmod | grep overlay
```

```
# Install container runtime (containerd)

log "Installing container runtime..."

sudo apt-get update

sudo apt-get install -y ca-certificates curl


# Add Docker GPG key and repository

log "Adding Docker repository..."

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


# Configure containerd

log "Configuring containerd..."

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


sudo systemctl restart containerd

sudo systemctl status containerd --no-pager
```

```
# Install Kubernetes packages

log "Installing Kubernetes components..."

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 kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl


# Initialize the Kubernetes cluster (Only on master node)

if [ "$NODE_TYPE" == "master" ]; then

  log "Initializing Kubernetes master node..."

  sudo kubeadm init


  mkdir -p "$HOME/.kube"

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

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


  # Apply Calico network plugin

  log "Applying Calico networking..."
```

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


    log "Master node setup complete!"
    log "To join worker nodes, use the following command:"
    kubeadm token create --print-join-command > join-command.sh
    chmod +x join-command.sh
    log "Run 'cat join-command.sh' to see the join command."
elif [ "$NODE_TYPE" == "worker" ]; then
    log "Resetting Kubernetes on worker node..."
    sudo kubeadm reset --force


    log "Joining the Kubernetes cluster..."
    # Run the join command (manual step)
    log "Run the join command from the master node."
fi


# For Master


# chmod +x setup_k8s.sh
# ./setup_k8s.sh master
# cat join-command.sh


# For Worker


# chmod +x setup_k8s.sh
# ./setup_k8s.sh worker

# Copy the join command from the master node and run it manually on the worker node.
```

Create Deployment

```
#!/bin/bash


# Function to log messages with spacing
log() {
  echo -e "\n[INFO] $1\n"
}


# Cleanup option - Placed at the beginning to avoid unnecessary execution
if [ "$1" == "cleanup" ]; then
  log "Deleting the Nginx deployment..."
  kubectl delete deployment nginx-deployment
  log "Deleting the Nginx service..."
  kubectl delete service nginx
  log "Cleanup complete!"
  exit 0
fi


# Define YAML file name and URL
YAML_FILE="nginx-deployment.yaml"
YAML_URL="https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/controllers/nginx-deployment.yaml"


# Download YAML file
log "Downloading Nginx deployment YAML..."
```

```
curl -s -o "$YAML_FILE" "$YAML_URL"

if [ ! -f "$YAML_FILE" ]; then
  echo -e "\n[ERROR] Failed to download $YAML_FILE. Exiting.\n"
  exit 1
fi

# Apply the YAML configuration
log "Applying the Nginx deployment..."
kubectl apply -f "$YAML_FILE"

# Wait for deployment to be ready
log "Waiting for the deployment to be ready..."
kubectl rollout status deployment/nginx-deployment

# Create a NodePort service for Nginx
log "Creating a NodePort service for Nginx..."
kubectl create service nodeport nginx --tcp=80:80

# Display service information
log "Fetching details of the created Nginx service..."
kubectl get svc nginx

# Display deployment and cluster info
log "Listing all running pods..."
kubectl get pods
```

```
sleep 1

log "Listing all deployments..."
kubectl get deployment
sleep 1

log "Listing all services..."
kubectl get service
sleep 1

log "Fetching cluster information..."
kubectl cluster-info
sleep 1

log "Listing all pods across all namespaces..."
kubectl get pods --all-namespaces

log "Nginx deployment setup complete!"
```