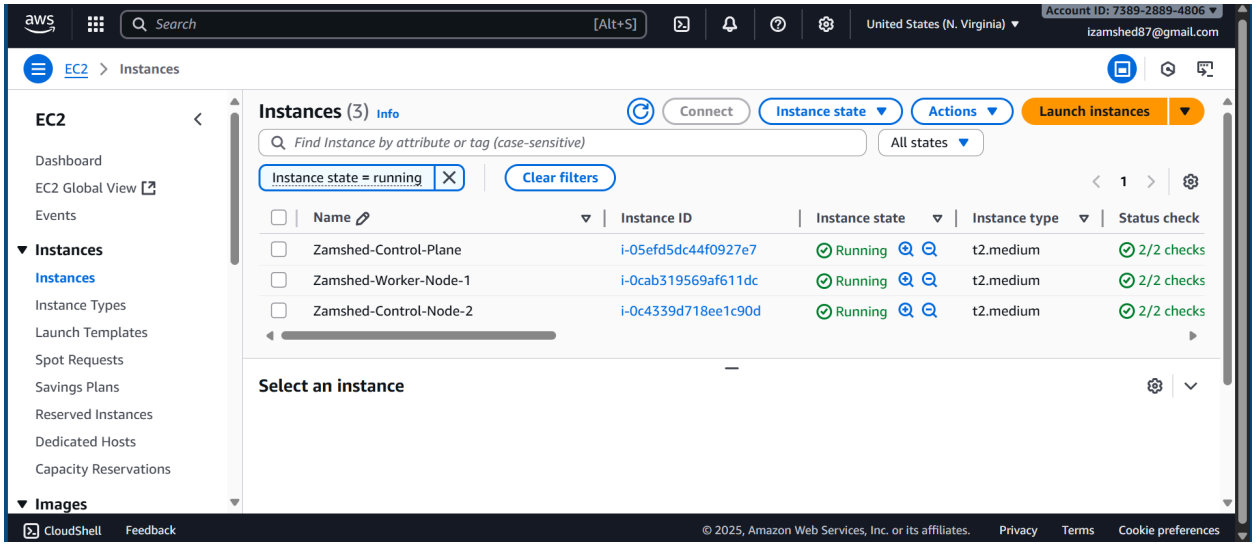
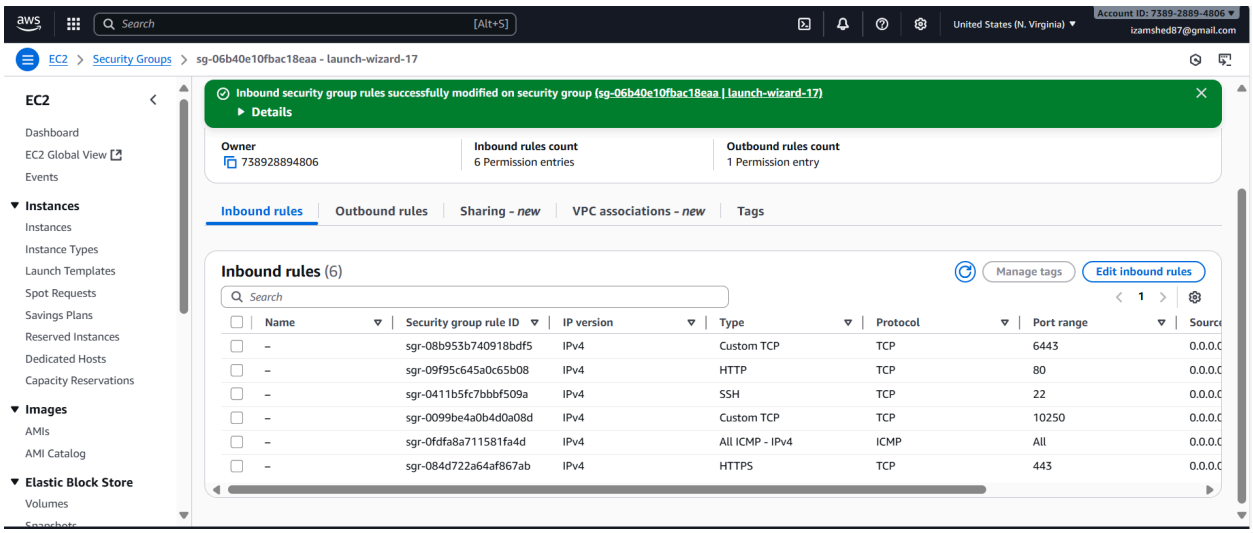


Before kubeadm join command allow these ports in the master node security group



1 Enable IP forwarding immediately

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

---

2 Make it persistent across reboots

Edit `/etc/sysctl.conf` (or create a drop-in file):

```
sudo nano /etc/sysctl.conf
```

Add (or uncomment) this line:

```
net.ipv4.ip_forward = 1
```

Then reload:

```
sudo sysctl -p
```

3️⃣ Verify

```
cat /proc/sys/net/ipv4/ip_forward
# It should print: 1
```

4️⃣ Retry kubeadm

```
sudo kubeadm init \
  --apiserver-advertise-address 172.31.28.30 \
  --pod-network-cidr 10.244.0.0/16 \
  --upload-certs
```

Tokens expire after 24h by default; regenerate if needed:

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

1. **Copy the master kubeconfig to the worker**:

On the master:

```
sudo cat /etc/kubernetes/admin.conf
```

Set up kubeconfig:

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

Copy its content to the worker and save as `$HOME/admin.conf` on the worker.

2. **Set ownership and environment variable** on the worker:

erfect — you already have the **master's `admin.conf` copied to the worker** as `$HOME/admin.conf`.

Now you just need to **point `kubectl` on the worker to that file**:

```
export KUBECONFIG=$HOME/admin.conf
echo 'export KUBECONFIG=$HOME/admin.conf' >> ~/.bashrc
```

After that, test the connection:

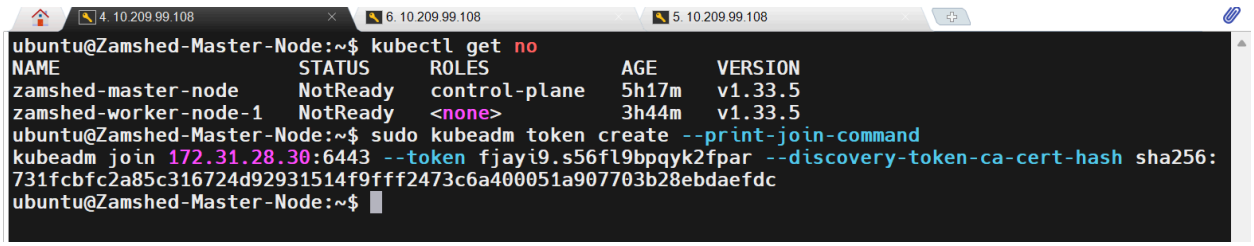
```
kubectl get nodes
kubectl get pods -A
```

✅ You should see the **master node listed**.

kubeadm join 172.31.28.30:6443 --token fjayi9.s56fl9bpqyk2fpar --discovery-token-ca-cert-hash sha256:731fcbfc2a85c316724d92931514f9fff2473c6a400051a907703b28ebdaefdc

sudo kubeadm join 172.31.28.30:6443 \

--token fjayi9.s56fl9bpqyk2fpar \
--discovery-token-ca-cert-hash sha256:731fcbfc2a85c316724d92931514f9fff2473c6a400051a907703b28ebdaefdc



Here's the **completed, polished documentation** for that section.
It walks through all the prerequisites, network settings, `kubeadm init`, copying `kubeconfig`, and finally joining the worker.

Kubernetes Cluster Setup – Master & Worker Node Join

1️⃣ Prepare the Master Node

a) Open Required Ports in Security Group

In your cloud provider (e.g., AWS), edit the **Security Group** for the master node and allow inbound:

Port	Protocol	Purpose
6443	TCP	Kubernetes API server
2379–2380	TCP	etcd server client API
10250	TCP	Kubelet API
10251	TCP	kube-scheduler
10252	TCP	kube-controller-manager
10255 (optional)	TCP	Read-only Kubelet API

Also allow ICMP (ping) for troubleshooting.

---

b) Enable IP Forwarding

Kubernetes requires IP forwarding to be enabled.

Enable immediately

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

Make it persistent

Edit `/etc/sysctl.conf`:

```
sudo nano /etc/sysctl.conf
```

Add or uncomment:

```
net.ipv4.ip_forward = 1
```

Reload settings:

```
sudo sysctl -p
```

Verify

```
cat /proc/sys/net/ipv4/ip_forward
```

```
# Expected output: 1
```

2️⃣ Initialize the Control Plane

On the master node, run:

```
sudo kubeadm init \
  --apiserver-advertise-address=172.31.28.30 \
  --pod-network-cidr=10.244.0.0/16 \
  --upload-certs
```

> Save the `kubeadm join` command from the output. Tokens expire after 24h — if needed, regenerate:

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

3️⃣ Configure `kubectl` on the Master

Set up `kubeconfig` so you can run `kubectl` as a normal user:

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

Test:

```
kubectl get nodes
```

4️⃣ Copy Kubeconfig to Worker (Optional for `kubectl`)

If you want to run `kubectl` from the worker:

On the master:

```
sudo cat /etc/kubernetes/admin.conf
```

1. Copy the content.

On the worker:

```
nano ~/admin.conf   # paste the content
export KUBECONFIG=$HOME/admin.conf
echo 'export KUBECONFIG=$HOME/admin.conf' >> ~/.bashrc
```

Verify:

```
kubectl get nodes
kubectl get pods -A
```

2.

✅ You should see the master node.

5️⃣ Join the Worker Node

Use the join command saved from the `kubeadm init` step.
Run it with **sudo**:

```
sudo kubeadm join 172.31.28.30:6443 \
  --token fjayi9.s56fl9bpqyk2fpar \
  --discovery-token-ca-cert-hash
sha256:731fcbfc2a85c316724d92931514f9fff2473c6a400051a907703b28ebdaefd
c
```

⚠️ If you forget `sudo`, you'll see:

```
[ERROR IsPrivilegedUser]: user is not running as root
```

6️⃣ Verify the Node Status

On the master node:

```
kubectl get nodes -o wide
```

Example output:

```
NAME                    STATUS     ROLES           AGE     VERSION
zamshed-master-node     Ready      control-plane   10m     v1.33.5
zamshed-worker-node-1   NotReady   <none>          1m      v1.33.5
```

> Nodes may show `NotReady` until you deploy a CNI plugin.

7️⃣ Deploy a Pod Network (CNI)

Install a network plugin, e.g., Flannel:

```
kubectl apply -f
https://raw.githubusercontent.com/flannel-io/flannel/master/Documentat
ion/kube-flannel.yml
```

Check:

```
kubectl get pods -A
kubectl get nodes
```

Once the CNI is ready, worker nodes should move to `Ready`.

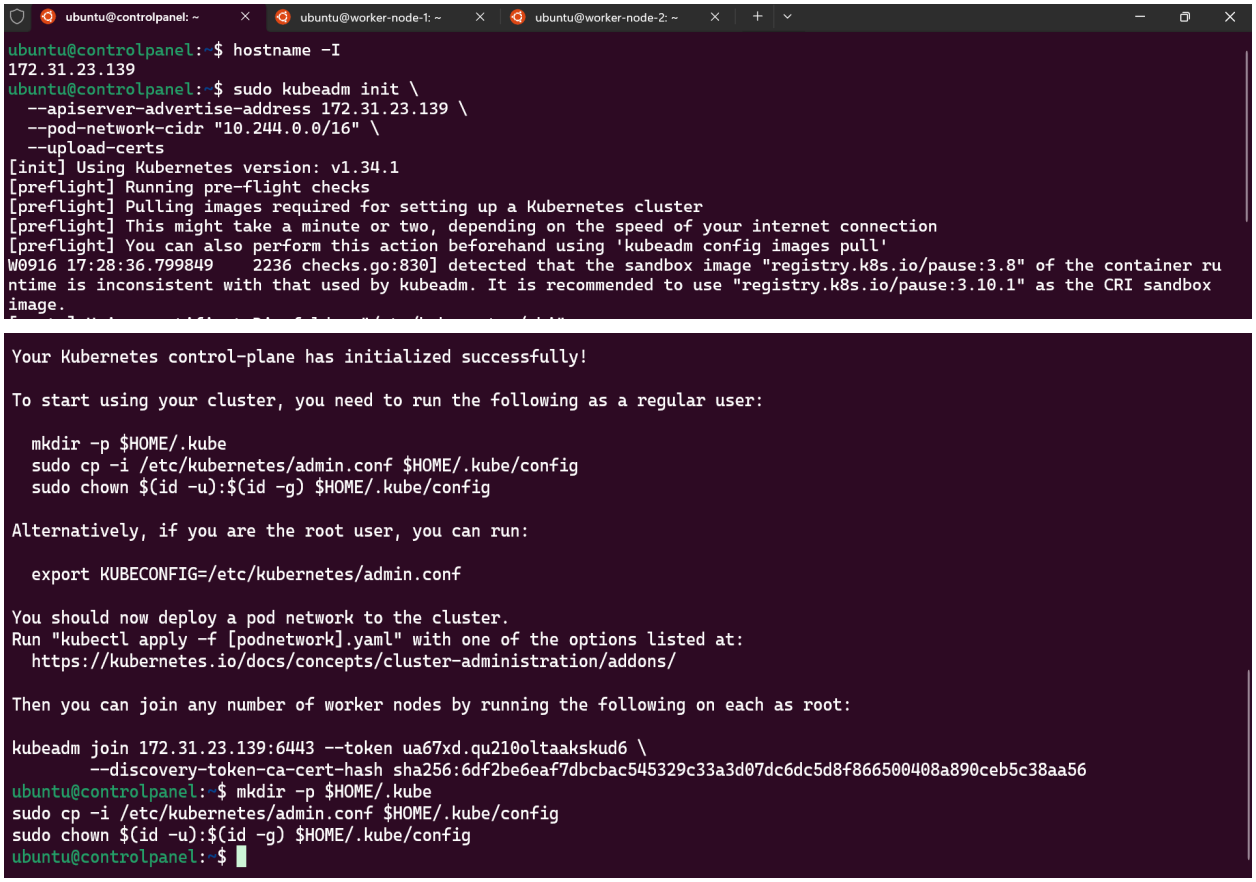
✅ Final Checklist

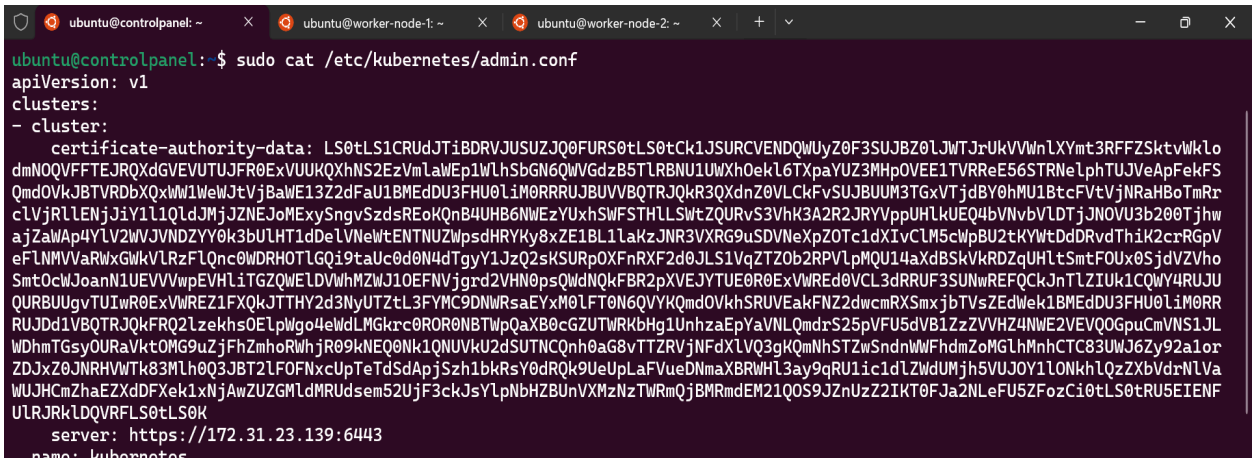
- Security group allows required ports
- IP forwarding enabled
- `kubeadm init` completed

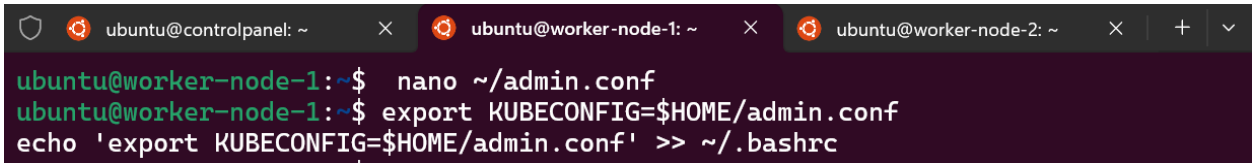
- `kubeconfig` set up
- Worker joined successfully
- CNI deployed

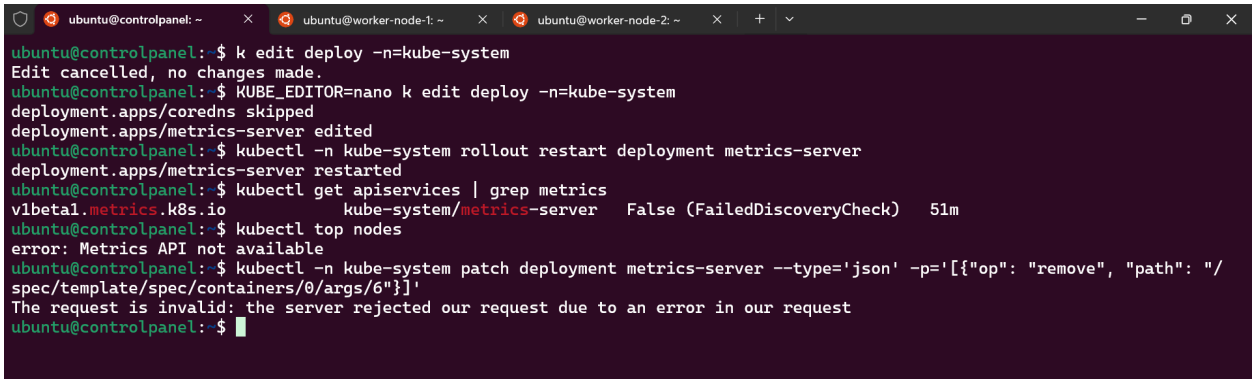
Here's a single `sed` command to do that:

```
sed -i '/- --secure-port=10250/c\        - --secure-port=4443\n
- --kubelet-i
```




ubuntu@controlpanel: ~
ubuntu@worker-node-1: ~
ubuntu@worker-node-2: ~
ubuntu@controlpanel:~$ sudo cat /etc/kubernetes/admin.conf
apiVersion: v1
clusters:
- cluster:
certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJWTJrUkVVWnlXYmt3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1WlhSbGN6QWVGdzB5TlRBNU1UWXhOekl6TXpaYUZ3MHpOVEE1TVRReE56STRNelphTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUUM3TGxVTjdBY0hMU1BtcFVtVjNRaHBoTmRrclVjRllENjJiY1l1QldJMjJZNEJoMExySngvSzdsREEoKQnB4UHB6NWEzYUxhSWFSTHlLSWtZQURvS3VhK3A2R2JRYVppUHlkUEQ4bVNvbVlDTjJNOVU3b200Tjhwajk4WAp4YlV2WVJVNDZYY0k3bUlHT1dDelVNeWtENTNUZWpsdHRYKy8xZE1BL1laKzJNR3VXRG9uSDVNeXpZOTc1dXIvClM5cWpBU2tKYWtDdDRvdThiK2crRGpVeFlNMVVaRWxGWkVlRzFlQnc0WDRHOTlGQi9taUc0d0N4dTgyY1JzQ2sKSURpOXFnRXF2d0JLS1VqZTZOb2RPVlpMQU14aXdBSkVkRDZqUHltSmtFOUx0SjdVZVho
SmtOcWJoanN1UEVVVwpEVHliTGZQWElDVWhMZWJ1OEFNVjgrd2VHN0psQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCkJnTlZIUk1CQWY4RUJU
QURBUUgvTUIwR0ExVWREZ1FXQkJTTHY2d3NyUTZtL3FYMC9DNWRsaEYxM0lFT0N6QVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RR
RUJDd1VBQTRJQkFRQ2lzekhsOElpWgo4eWdLMGkrc0ROR0NBTWpQaXB0cGZUTWRKbHg1UnhzaEpYaVNLQmdrS25pVFU5dVB1ZzZVVHZ4NWE2VEVQOGpuCmVNS1JL
WDhmTGsyOURaVktOMG9uZjFhZmhoRWhjR09kNEQ0Nk1QNUVkU2dSUTNCQnh0aG8vTTZRVjNFdXlVQ3gKQmNhSTZwSndnWWFhdmZoMGlhMnhCTC83UWJ6Zy92a1or
ZDJxZ0JNRHVWTk83Mlh0Q3JBT2lFOFNxcUpTeTdSdApjSzh1bkRsY0dRQk9UeUpLaFVueDNmaXBRWHl3ay9qRU1ic1dlZWdUMjh5VUJOY1lONkhlQzZXbVdrNlVa
WUJHCmZhaEZXdDFXek1xNjAwZUZGMldMRUdsem52UjF3ckJsYlpNbHZBUnVXMzNzTWRmQjBMRmdEM21QOS9JZnUzZ2IkT0FJa2NLeFU5ZFozCi0tLS0tRU5EIENF
UlRJRklDQVRFLS0tLS0K
server: https://172.31.23.139:6443


ubuntu@controlpanel: ~
ubuntu@worker-node-1: ~
ubuntu@worker-node-2: ~
ubuntu@worker-node-1:~$ nano ~/admin.conf
ubuntu@worker-node-1:~$ export KUBECONFIG=$HOME/admin.conf
echo 'export KUBECONFIG=$HOME/admin.conf' >> ~/.bashrc


ubuntu@controlpanel: ~
ubuntu@worker-node-1: ~
ubuntu@worker-node-2: ~
ubuntu@controlpanel:~$ k edit deploy -n=kube-system
Edit cancelled, no changes made.
ubuntu@controlpanel:~$ KUBE_EDITOR=nano k edit deploy -n=kube-system
deployment.apps/coredns skipped
deployment.apps/metrics-server edited
ubuntu@controlpanel:~$ kubectl -n kube-system rollout restart deployment metrics-server
deployment.apps/metrics-server restarted
ubuntu@controlpanel:~$ kubectl get apiservices | grep metrics
v1beta1.metrics.k8s.io kube-system/metrics-server False (FailedDiscoveryCheck) 51m
ubuntu@controlpanel:~$ kubectl top nodes
error: Metrics API not available
ubuntu@controlpanel:~$ kubectl -n kube-system patch deployment metrics-server --type='json' -p='[{"op": "remove", "path": "/
spec/template/spec/containers/0/args/6"}]'
The request is invalid: the server rejected our request due to an error in our request
ubuntu@controlpanel:~$

```
kubectl -n kube-system rollout status deploy metrics-server
kubectl get apiservices | grep metrics
kubectl top nodes
kubectl top pods
```