DevOps

Ultimate Guide: Setting Up a Kubernetes Cluster with Master and Worker Nodes on Ubuntu 24.04 for Optimal Performance

mrcloudbook.com · 11 June 2024

Contents [hide]

Introduction

Kubernetes is a powerful open-source platform designed to automate the deployment, scaling, and operation of application containers. In this comprehensive guide, we'll explore how to set up a Kubernetes cluster using three nodes, leveraging the flexibility and scalability of Amazon EC2 instances. We'll ensure each step is clear and detailed, making it easy for anyone to follow along, even a fifth-grader!

What You Will Need

Before we start, let's go over the requirements and setup for creating our Kubernetes cluster.

1. **Two EC2 Instances**: We will use one instance as the master node and one instances as worker node.

2. **Instance Type**: Each instance should have at least 2GB of RAM. We recommend using the t2.medium instance type for its balance of performance and cost.

Caution: The t2.medium instance costs. Make sure to monitor your usage to avoid unexpected costs.

Step 1: Create Amazon EC2 Instances

To begin, we need to create our EC2 instances. Follow these steps to configure your instances:

Configuration for All Instances:

1. **Select the OS Version**: Choose Ubuntu 24.04 for its stability and latest features.
2. **Select an Instance Type**: Ensure each instance has at least 2GB of RAM and 2 CPUs. The t2.medium instance type is a suitable choice.

Detailed Steps to Launch EC2 Instances:

1. **Open the AWS Management Console**: Navigate to the EC2 Dashboard.
2. **Launch Instance**: Click on the “Launch Instance” button.
3. **Choose an Amazon Machine Image (AMI)**: Select “Ubuntu Server 24.04 LTS (HVM), SSD Volume Type”.
4. **Choose an Instance Type**: Select “t2.medium”.
5. **Configure Instance Details**:
    - Number of instances: 2
    - Network: Default VPC
    - Subnet: Choose an available subnet
    - Auto-assign Public IP: Enable
6. **Add Storage**: Use the default 16GB for the root volume.
7. **Add Tags**: (Optional) Add tags to identify your instances.

8. **Configure Security Group**:
    - Create a new security group or use an existing one. (Open all Ports for Learning Purpose Only)
    - Add rules to allow SSH access (port 22) from your IP address.
9. **Review and Launch**: Review your settings and click “Launch”. Select an existing key pair or create a new one for SSH access, then click “Launch Instances”.

After launching, you will have three instances ready to be configured as your Kubernetes cluster.

Step 2: Install and Configure Docker and Kubernetes

Connect to the Instances

You need to connect to each instance via SSH. Use the following command in your terminal (replace `<instance-ip>` with the actual public IP address of your instance):

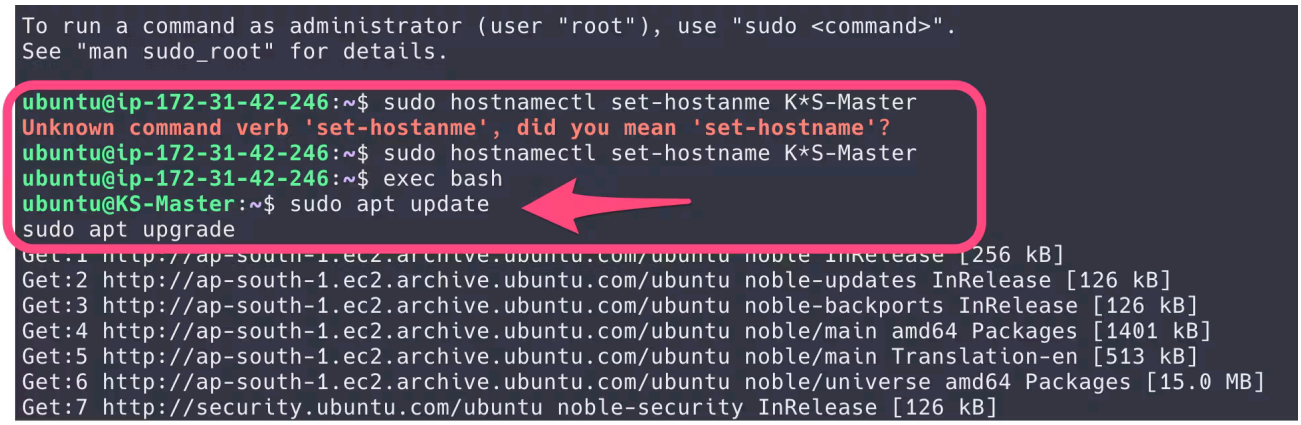
```
ssh -i <your-key-pair.pem> ubuntu@<instance-ip>
```

Install Docker and Kubernetes Components

Run the following commands on two instances (master and worker nodes):

Update and Upgrade Packages:

```
sudo apt update
sudo apt upgrade -y
```



Disable Swap:

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

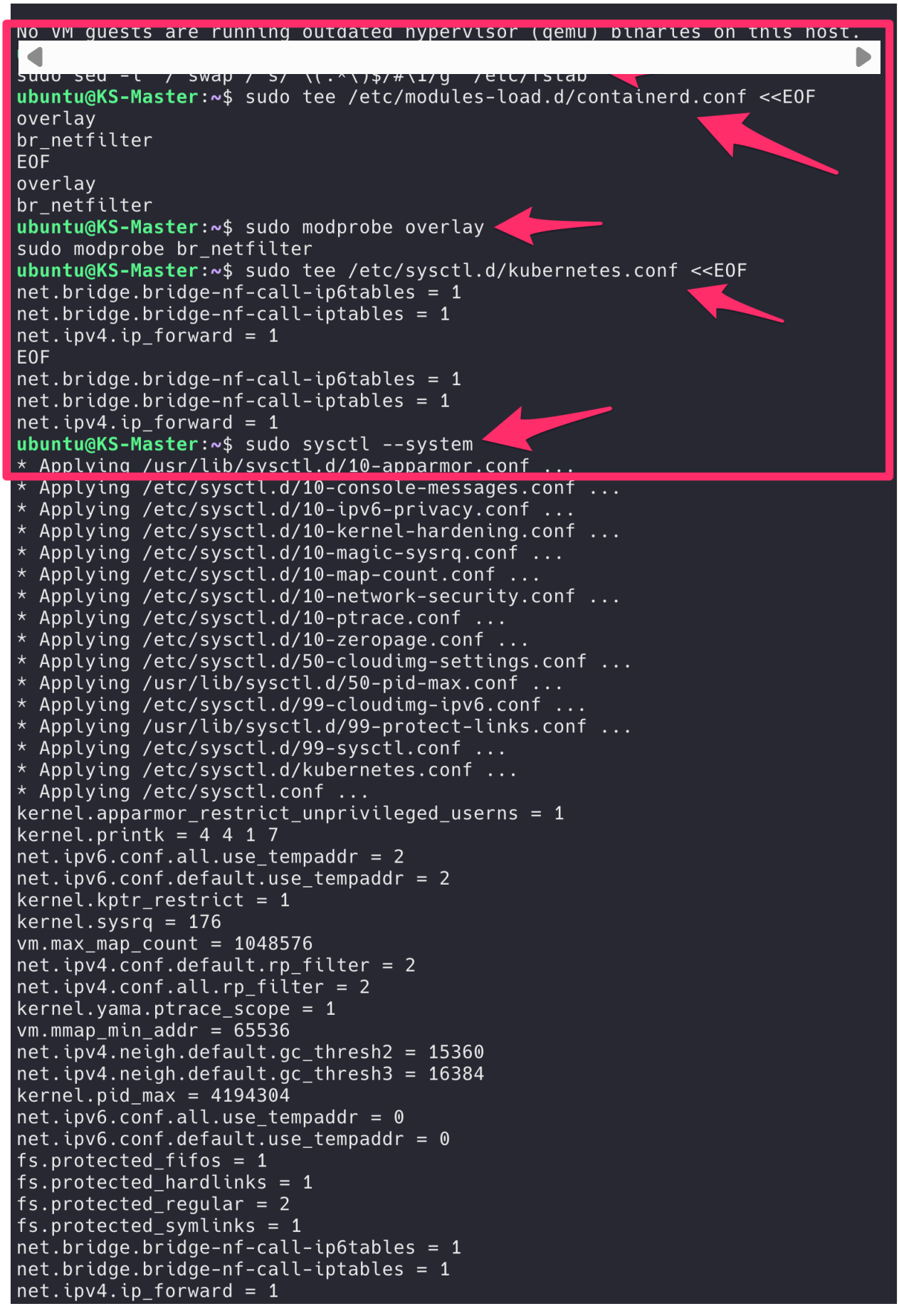
Load Kernel Modules:

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
```

Set Up Required sysctl Parameters:

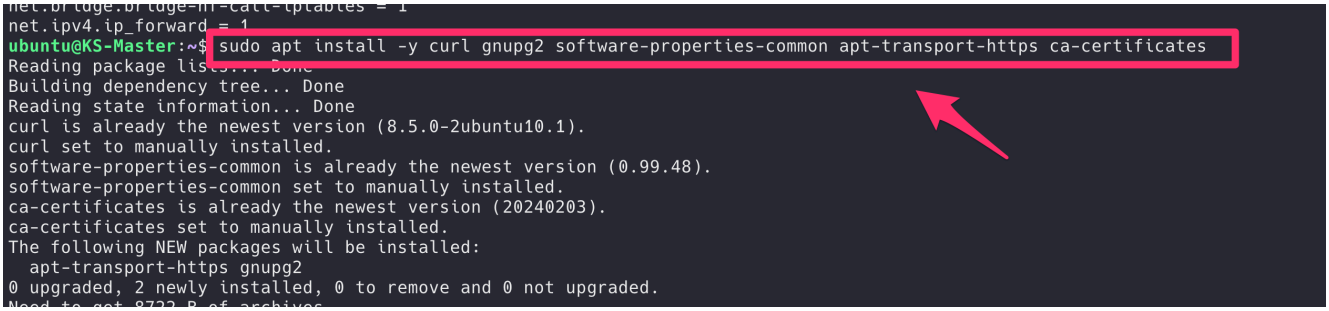
```
sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

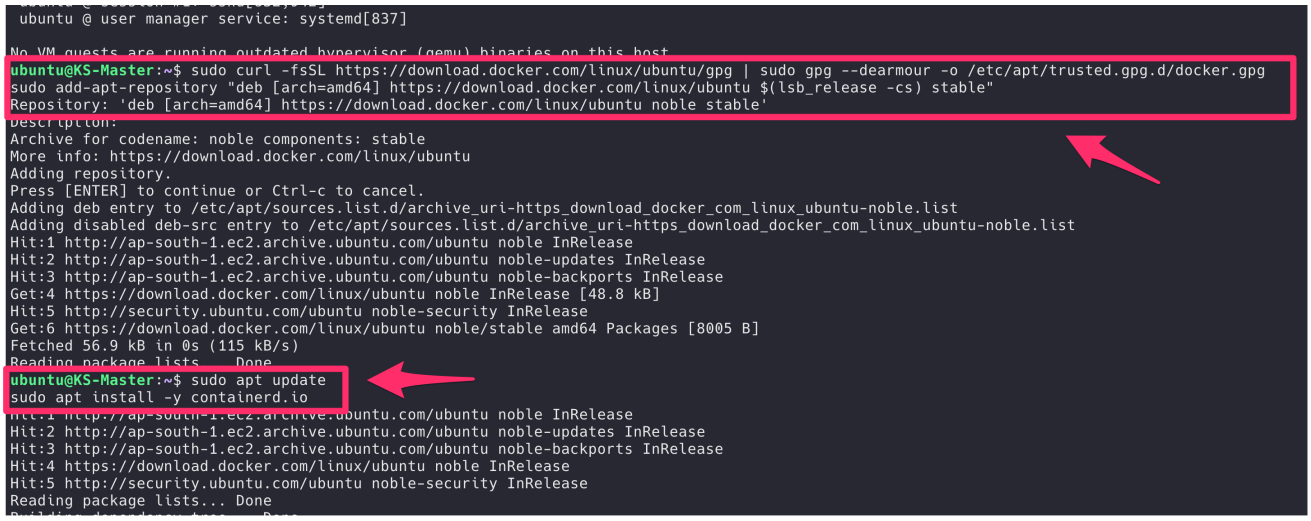
```
EOF
sudo sysctl --system
```



Install Docker

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/l
sudo apt update
sudo apt install -y containerd.io
```





Configure Containerd:

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/cont
sudo systemctl restart containerd
sudo systemctl enable containerd
```

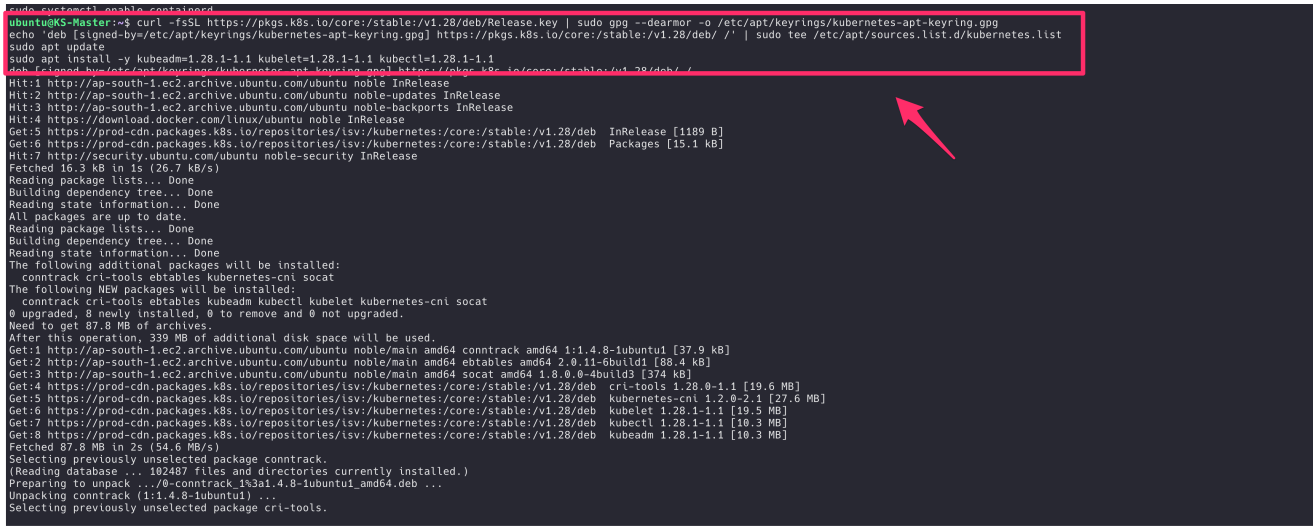
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@KS-Master:~$ containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd
sudo systemctl enable containerd
```

Install Kubernetes Packages

Add Kubernetes Repository:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | suc
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https
sudo apt update
sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1
```



if the above command doesn't work, use the Snap method:

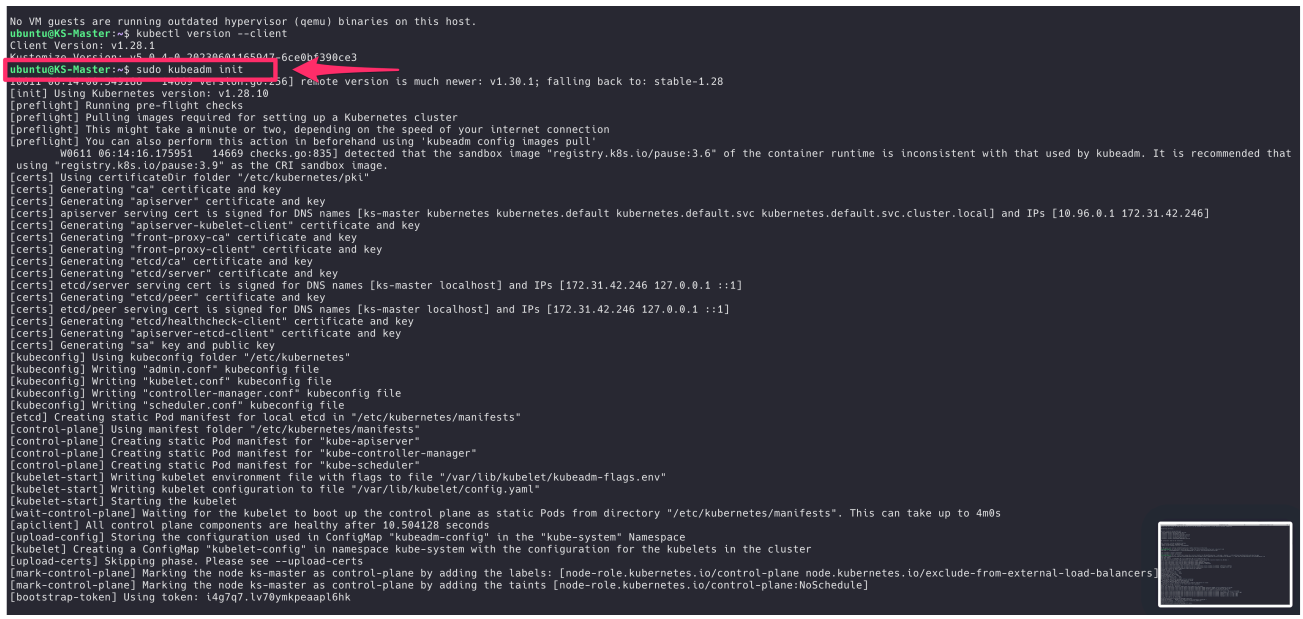
```
sudo snap install kubeadm=1.28.1-1.1 --classic
sudo snap install kubectl=1.28.1-1.1 --classic
sudo snap install kubelet=1.28.1-1.1 --classic
```

Step 3: Initialize Kubernetes Cluster on the Master Node

Initialize the Cluster

On the master node, run the following command to initialize the Kubernetes cluster:

```
sudo kubeadm init
```



Post-Initialization Setup on the Master Node

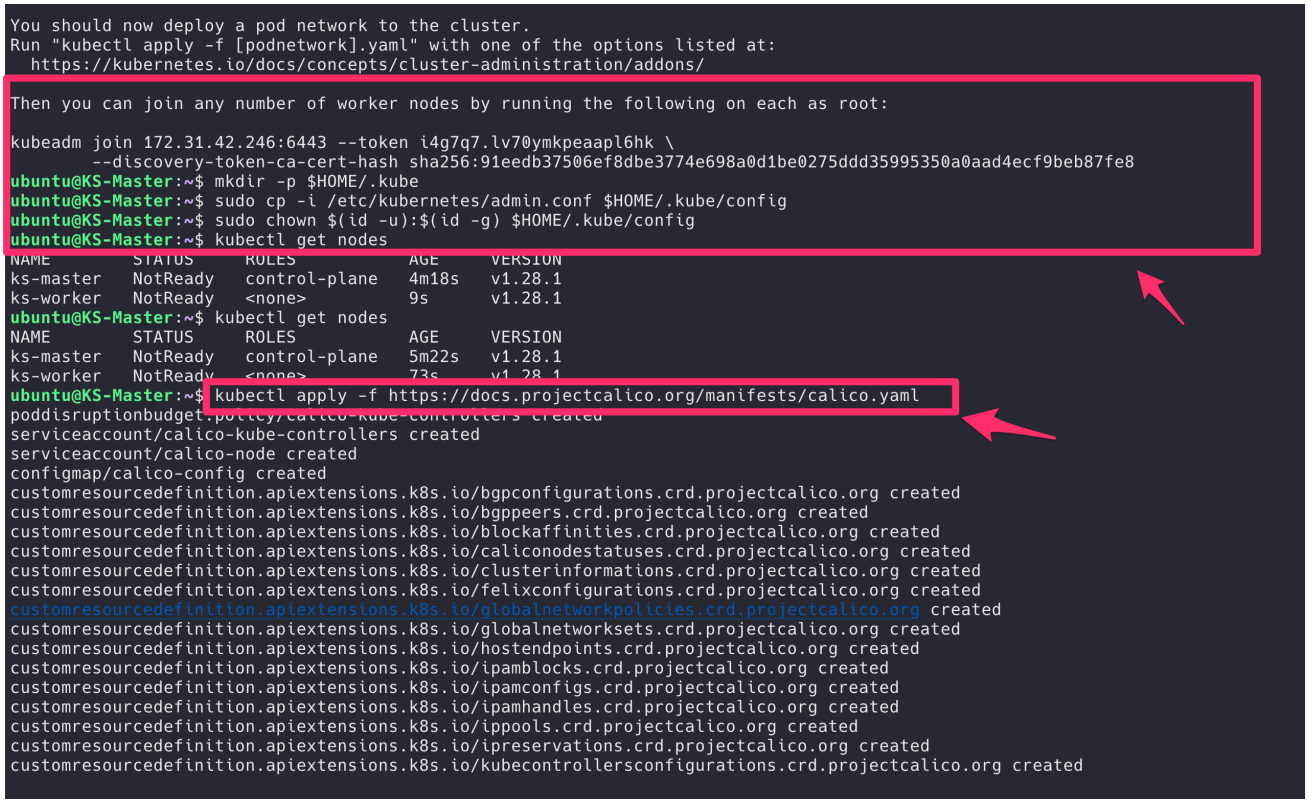
After initialization, set up the kubeconfig for the user on the master node:

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

Additional Configuration: Deploy a Pod Network

To enable communication between pods, you need to deploy a pod network. Here's how to deploy Calico, a popular networking solution for Kubernetes:

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

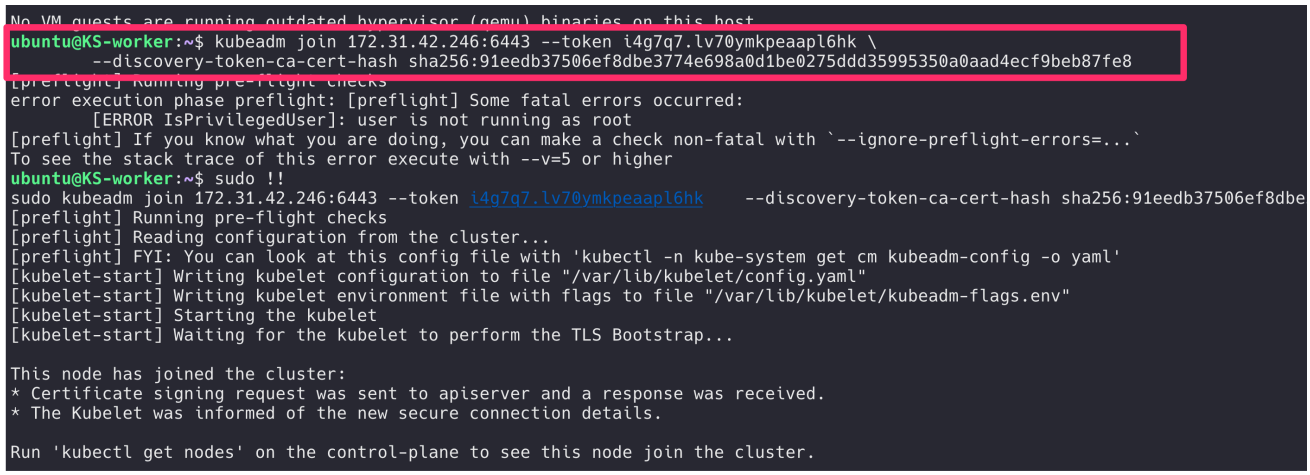


Step 4: Add Worker Nodes to the Cluster

Join Worker Nodes to the Cluster

Use the `kubeadm join` command provided during the master node initialization on each worker node. This command will look something like this (replace the placeholders with your actual values):

```
sudo kubeadm join &lt;master-node-ip&gt;:6443 --token &lt;token&gt; --d
# Dont forget to use SUDO
```



Verify the Cluster

To verify that the worker nodes have joined the cluster, run the following command on the master node:

```
kubectl get nodes
```

```
deployment.apps/calico-kube-controllers created
ubuntu@KS-Master:~$ kubectl get nodes
NAME        STATUS     ROLES           AGE    VERSION
ks-master   NotReady   control-plane   6m6s   v1.28.1
ks-worker   NotReady   <none>          117s   v1.28.1
ubuntu@KS-Master:~$ kubectl get pods -n kube-system
NAME                                    READY   STATUS     RESTARTS   AGE
calico-kube-controllers-658d97c59c-scdqn   0/1     Pending    0          8s
calico-node-272kt                       0/1     Init:0/3   0          8s
calico-node-dltv4                       0/1     Init:0/3   0          8s
coredns-5dd5756b68-5szqq                0/1     Pending    0          6m3s
coredns-5dd5756b68-h477t                0/1     Pending    0          6m3s
etcd-ks-master                          1/1     Running    0          6m8s
kube-apiserver-ks-master                1/1     Running    0          6m10s
kube-controller-manager-ks-master       1/1     Running    0          6m8s
kube-proxy-85pnw                        1/1     Running    0          6m3s
kube-proxy-k26fh                        1/1     Running    0          2m3s
kube-scheduler-ks-master                1/1     Running    0          6m8s
ubuntu@KS-Master:~$ kubectl get nodes
NAME        STATUS     ROLES           AGE     VERSION
ks-master   NotReady   control-plane   6m20s   v1.28.1
ks-worker   NotReady   <none>          2m11s   v1.28.1
ubuntu@KS-Master:~$ kubectl get pods -n kube-system
NAME                                    READY   STATUS              RESTARTS   AGE
calico-kube-controllers-658d97c59c-scdqn   0/1     ContainerCreating   0          33s
calico-node-272kt                       0/1     Running             0          33s
calico-node-dltv4                       0/1     Running             0          33s
coredns-5dd5756b68-5szqq                0/1     Running             0          6m28s
coredns-5dd5756b68-h477t                0/1     Running             0          6m28s
etcd-ks-master                          1/1     Running             0          6m33s
kube-apiserver-ks-master                1/1     Running             0          6m35s
kube-controller-manager-ks-master       1/1     Running             0          6m33s
kube-proxy-85pnw                        1/1     Running             0          6m28s
kube-proxy-k26fh                        1/1     Running             0          2m28s
kube-scheduler-ks-master                1/1     Running             0          6m33s
ubuntu@KS-Master:~$ kubectl get nodes
NAME        STATUS   ROLES           AGE     VERSION
ks-master   Ready    control-plane   6m45s   v1.28.1
ks-worker   Ready    <none>          2m36s   v1.28.1
ubuntu@KS-Master:~$ kubectl get pods -n kube-system
NAME                                    READY   STATUS    RESTARTS   AGE
calico-kube-controllers-658d97c59c-scdqn   1/1     Running   0          43s
calico-node-272kt                       1/1     Running   0          43s
calico-node-dltv4                       1/1     Running   0          43s
coredns-5dd5756b68-5szqq                1/1     Running   0          6m38s
coredns-5dd5756b68-h477t                1/1     Running   0          6m38s
etcd-ks-master                          1/1     Running   0          6m43s
kube-apiserver-ks-master                1/1     Running   0          6m45s
kube-controller-manager-ks-master       1/1     Running   0          6m43s
kube-proxy-85pnw                        1/1     Running   0          6m38s
kube-proxy-k26fh                        1/1     Running   0          2m38s
kube-scheduler-ks-master                1/1     Running   0          6m43s
ubuntu@KS-Master:~$ kubectl get nodes
NAME        STATUS   ROLES           AGE     VERSION
ks-master   Ready    control-plane   6m49s   v1.28.1
```

Conclusion

Congratulations! You have successfully set up a Kubernetes cluster on Ubuntu 24.04 using AWS EC2 instances. You can now start deploying your applications and exploring the powerful features of Kubernetes.

Summary of Key Points

1. **Created EC2 Instances**: Launched two instances for the master and worker nodes.
2. **Installed and Configured Docker**: Set up Docker and containerd on all nodes.
3. **Installed Kubernetes Components**: Installed kubeadm, kubelet, and kubectl on all nodes.
4. **Initialized the Cluster**: Initialized Kubernetes on the master node and joined worker nodes to the cluster.
5. **Deployed a Pod Network**: Set up Calico for pod communication.

Additional Tips

- **Monitoring and Logging**: Set up monitoring and logging for better cluster management.
- **Security**: Regularly update and patch your nodes to ensure security.

With these steps, setting up a Kubernetes cluster is straightforward and manageable, even for someone new to the process. Happy clustering!

K8s Kubernetes master slave ubuntu 24.04

Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA
Privacy - Terms

Post Comment

DevOps

How to Deploy website on AWS Amplify

21 August 2025

Uncategorized

Day -1: Kick Off Cloud Security with AWS Registration

18 August 2025

Uncategorized

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.

Important Links

Privacy Policy

Terms & Conditions

Contact

Resources

Blog

YouTube Channel