

Mr Cloud Book

Home Blog DevSecOps Contact About Me Testimonials

Search Blogs

DevOps

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



mrcloobook.com · 11 June 2024



Contents [hide]

Introduction
What You Will Need
Step 1: Create Amazon EC2 Instances
Configuration for All Instances:
Detailed Steps to Launch EC2 Instances:
Step 2: Install and Configure Docker and Kubernetes
Connect to the Instances
Install Docker and Kubernetes Components
Install Kubernetes Packages
Step 3: Initialize Kubernetes Cluster on the Master Node
Initialize the Cluster
Post-Initialization Setup on the Master Node
Additional Configuration: Deploy a Pod Network
Step 4: Add Worker Nodes to the Cluster
Join Worker Nodes to the Cluster
Verify the Cluster
Conclusion
Summary of Key Points
Additional Tips

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
```



```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-42-246:~$ sudo hostnamectl set-hostname K*S-Master
Unknown command verb 'set-hostname', did you mean 'set-hostname'?
ubuntu@ip-172-31-42-246:~$ sudo hostnamectl set-hostname K*S-Master
ubuntu@ip-172-31-42-246:~$ exec bash
ubuntu@KS-Master:~$ sudo apt update
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 Packages [1401 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main Translation-en [513 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
```

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
```



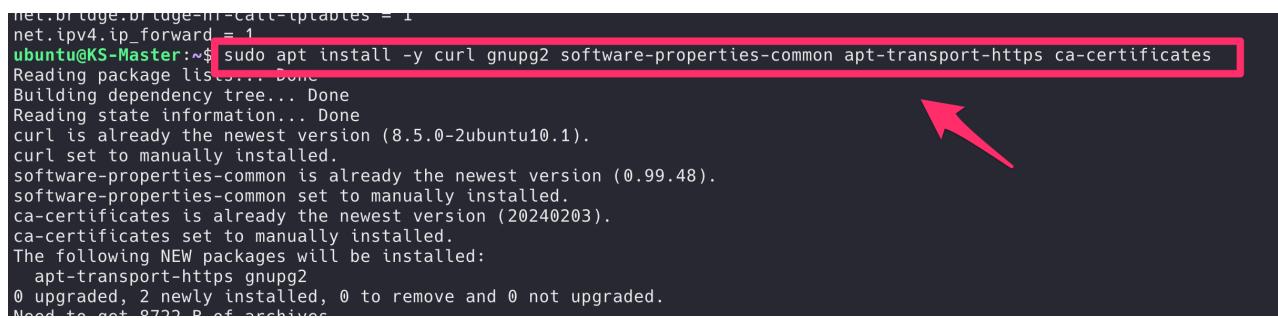
EOF

```
sudo sysctl --system
```

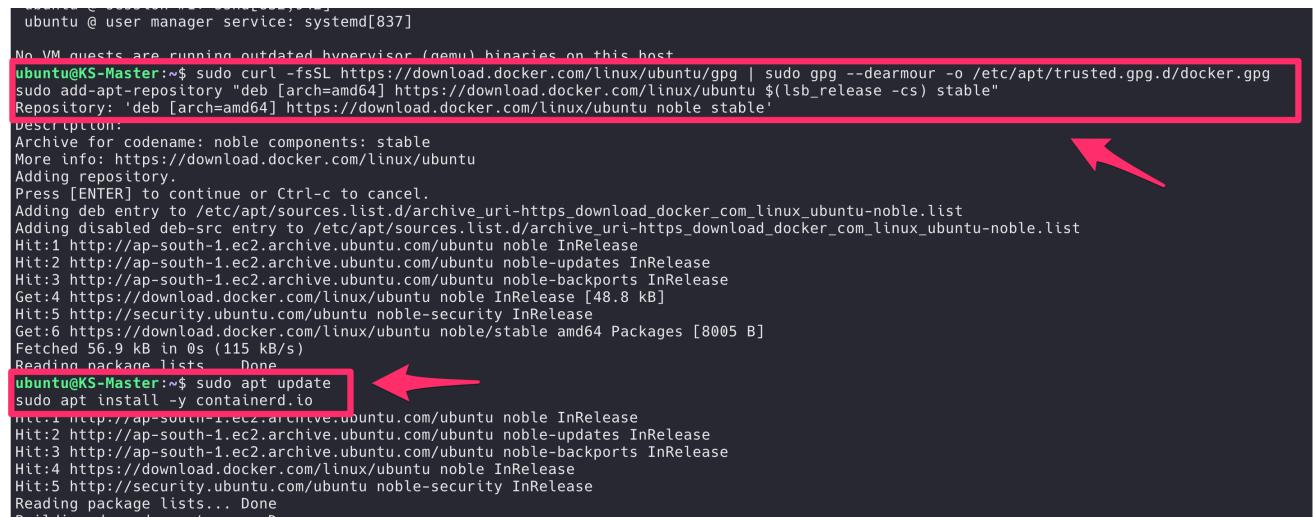
```
NO VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@KS-Master:~$ sudo tee /etc/modules-load.d/containerd.conf <<EOF  
overlay  
br_netfilter  
EOF  
overlay  
br_netfilter  
ubuntu@KS-Master:~$ sudo modprobe overlay ←  
sudo modprobe br_netfilter  
ubuntu@KS-Master:~$ 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  
net.bridge.bridge-nf-call-ip6tables = 1  
net.bridge.bridge-nf-call-iptables = 1  
net.ipv4.ip_forward = 1  
ubuntu@KS-Master:~$ sudo sysctl --system ←  
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...  
* Applying /etc/sysctl.d/10-console-messages.conf ...  
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...  
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...  
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...  
* Applying /etc/sysctl.d/10-map-count.conf ...  
* Applying /etc/sysctl.d/10-network-security.conf ...  
* Applying /etc/sysctl.d/10-ptrace.conf ...  
* Applying /etc/sysctl.d/10-zero-page.conf ...  
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...  
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...  
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...  
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...  
* Applying /etc/sysctl.d/99-sysctl.conf ...  
* Applying /etc/sysctl.d/kubernetes.conf ...  
* Applying /etc/sysctl.conf ...  
kernel.apparmor_restrict_unprivileged_userns = 1  
kernel.printk = 4 4 1 7  
net.ipv6.conf.all.use_tempaddr = 2  
net.ipv6.conf.default.use_tempaddr = 2  
kernel.kptr_restrict = 1  
kernel.sysrq = 176  
vm.max_map_count = 1048576  
net.ipv4.conf.default.rp_filter = 2  
net.ipv4.conf.all.rp_filter = 2  
kernel.yama.ptrace_scope = 1  
vm.mmap_min_addr = 65536  
net.ipv4.neigh.default.gc_thresh2 = 15360  
net.ipv4.neigh.default.gc_thresh3 = 16384  
kernel.pid_max = 4194304  
net.ipv6.conf.all.use_tempaddr = 0  
net.ipv6.conf.default.use_tempaddr = 0  
fs.protected_fifos = 1  
fs.protected_hardlinks = 1  
fs.protected_regular = 2  
fs.protected_symlinks = 1  
net.bridge.bridge-nf-call-ip6tables = 1  
net.bridge.bridge-nf-call-iptables = 1  
net.ipv4.ip_forward = 1
```

Install Docker

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update
sudo apt install -y containerd.io
```



```
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
ubuntu@KS-Master:~$ sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.1).
curl set to manually installed.
software-properties-common is already the newest version (0.99.48).
software-properties-common set to manually installed.
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following NEW packages will be installed:
  apt-transport-https gnupg2
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 872 B of archives.
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@KS-Master:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/docker.gpg
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository...
Press [ENTER] to continue or Ctrl-C to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:6 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [8005 B]
Fetched 56.9 kB in 0s (115 kB/s)
Reading package lists... Done
ubuntu@KS-Master:~$ sudo apt update
sudo apt install -y containerd.io
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu/noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
```

Configure Containerd:

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/r
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/con
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 | sudo apt-key add -
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt update
sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1
```



```
ubuntu@KS-Master:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo apt-key add -
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt update
sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1
Hit:1 http://ap-south-1.ec2.archive/ubuntu/noble InRelease
Hit:2 http://ap-south-1.ec2.archive/ubuntu/noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive/ubuntu/noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu/noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb InRelease [1189 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb Packages [15.1 kB]
Hit:7 http://security.ubuntu.com/ubuntu/noble-security InRelease
Fetched 16 374 in 1s (20.6 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools etables kubernetes-cni socat
The following NEW packages will be installed:
  conntrack cri-tools etables kubeadm kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 87.8 MB of archives.
After this operation, 339 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive/ubuntu/noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 http://ap-south-1.ec2.archive/ubuntu/noble/main amd64 etables amd64 2.0.11-6build1 [88.4 kB]
Get:3 http://ap-south-1.ec2.archive/ubuntu/noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb cri-tools 1.28.0-1.1 [19.6 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb kubernetes-cni 1.2.0-2.1 [27.6 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb kubelet 1.28.1-1.1 [19.5 MB]
Get:7 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb kubetl 1.28.1-1.1 [16.3 MB]
Get:8 https://prod-cdn.packages.k8s.io/repositories/lsv:/kubernetes/:core:/stable:/v1.28/deb kubeadm 1.28.1-1.1 [16.3 MB]
Fetched 87.8 MB in 7s (54.6 kB/s)
Selecting previously unselected package conntrack.
(Reading database ... 102407 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Selecting previously unselected package cri-tools.
```



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
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@KS-Master:~$ kubectl version --client
Client Version: v1.28.1
Kubernetes Version: v5.0.4-0~20230601165047~6ce0bf390ce3
ubuntu@KS-Master:$ sudo kubeadm init
[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] Generating "apiserver" certificate and key
[certs] Generating "apiserver-serving" certificate and key
[certs] apiserver serving cert is signed for DNS names [ks-master kubernetes.kubernetes.default kubernetes.default.svc.kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.42.246]
[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/peer" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ks-master localhost] and IPs [172.31.42.246 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ks-master localhost] and IPs [172.31.42.246 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa-key" certificate and key
[kubeconfig] Using Kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "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] 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 Master API server to boot up
[client] All control plane components are healthy after 10.504128 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 ks-master as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node ks-master as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: t4g7q7.lv70ymkpaa16hk
```



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
```



```
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 172.31.42.246:6443 --token i4g7q7.lv70ymkpeapl6hk \  
--discovery-token-ca-cert-hash sha256:91eedb37506ef8dbe3774e698a0d1be0275ddd35995350a0aad4ecf9beb87fe8  
ubuntu@KS-Master:~$ mkdir -p $HOME/.kube  
ubuntu@KS-Master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
ubuntu@KS-Master:~$ sudo chown $1id -u:$1id -g $HOME/.kube/config  
ubuntu@KS-Master:~$ kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
ks-master NotReady control-plane 4m18s v1.28.1  
ks-worker NotReady <none> 9s v1.28.1  
ubuntu@KS-Master:~$ kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
ks-master NotReady control-plane 5m22s v1.28.1  
ks-worker NotReady <none> 73s v1.28.1  
ubuntu@KS-Master:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml  
poddisruptionbudget.potccy/calico-kube-controllers created  
serviceaccount/calico-kube-controllers created  
serviceaccount/calico-node created  
configmap/calico-config created  
customresourcedefinition.apirextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/bgppeers.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/blockaffinities.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/clusterinformations.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/felixconfigurations.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/globalnetworksets.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/hostendpoints.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/ipamblocks.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/ipamconfigs.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/ipamhandles.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/ippools.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/ippreservations.crd.projectcalico.org created  
customresourcedefinition.apirextensions.k8s.io/kubecontrollerconfigurations.crd.projectcalico.org created
```

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 <master-node-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:91eedb37506ef8dbe3774e698a0d1be0275ddd35995350a0aad4ecf9beb87fe8  
# Dont forget to use SUDO
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@KS-worker:~$ kubeadm join 172.31.42.246:6443 --token i4g7q7.lv70ymkpeapl6hk \  
--discovery-token-ca-cert-hash sha256:91eedb37506ef8dbe3774e698a0d1be0275ddd35995350a0aad4ecf9beb87fe8  
[preflight] Running pre-flight checks  
error execution phase preflight: [preflight] Some fatal errors occurred:  
    [ERROR IsPrivilegedUser]: user is not running as root  
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`  
To see the stack trace of this error execute with --v=5 or higher  
ubuntu@KS-worker:~$ sudo !!  
sudo kubeadm join 172.31.42.246:6443 --token i4g7q7.lv70ymkpeapl6hk --discovery-token-ca-cert-hash sha256:91eedb37506ef8dbe  
[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:  
* 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.
```

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](#)

