How to Install Kubernetes Cluster (kubeadm) on Ubuntu 22.04

Kubernetes is a powerful container orchestration platform used for automating the deployment, scaling, and management of containerized applications. The step-by-step process of installing Kubernetes on Ubuntu 22.04. This cluster configuration includes a master node and worker nodes.

Kubernetes Nodes

In a Kubernetes cluster, we have two distinct categories of nodes:

Master Nodes: These nodes play a crucial role in managing the control API calls for various components within the Kubernetes cluster. This includes overseeing pods, replication controllers, services, nodes, and more.

Worker Nodes: Worker nodes are responsible for providing runtime environments for containers. It’s worth noting that a group of

container pods can extend across multiple worker nodes, ensuring optimal resource allocation and management.

Prerequisites

Before diving into the installation, ensure that your environment meets the following prerequisites:

- An Ubuntu 22.04 system.
- Privileged access to the system (root or sudo user).
- Minimum 2GB RAM or more.
- Minimum 2 CPU cores (or 2 vCPUs).
- 20 GB of free disk space (or more).

Step 1: Update and Upgrade Ubuntu (all nodes)
Open a terminal and execute the following commands:

- sudo apt-get update

Step 2: Disable Swap (all nodes)
To enhance Kubernetes performance, disable swap and set essential kernel parameters.

- sudo swapoff -a
- sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab (or)
- sudo vi /etc/fstab (Comment the swapfile line & save)

Step 3: Add Kernel Parameters (all nodes)

Load the required kernel modules on all nodes:

- cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
  overlay
  br_netfilter
  EOF
- sudo modprobe overlay
- sudo modprobe br_netfilter

Verify that the br_netfilter, overlay modules are loaded by running the following commands:

- lsmod | grep br_netfilter
- lsmod | grep overlay

Configure the critical kernel parameters for Kubernetes using the

following:

- 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

Then, reload the changes:

- sudo sysctl –system

This command is configuring some sysctl kernel parameters that are required for Kubernetes networking to function properly. Here's what it's doing:

sudo tee /etc/sysctl.d/kubernetes.conf<<EOF — Creates a new sysctl config file called /etc/sysctl.d/kubernetes.conf and writes the following parameters into it:

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

The first two parameters enable bridged IPv4 and IPv6 traffic to be passed to iptables chains. This is required for Kubernetes networking

policies and traffic routing to work.

net.ipv4.ip_forward = 1 enables IP forwarding in the kernel, which is required for packet routing between pods in Kubernetes.

sudo sysctl — system — Applies the sysctl parameters from the new /etc/sysctl.d/kubernetes.conf file to the running kernel. This enables the settings without requiring a reboot.

In summary, this command configures three key sysctl parameters needed for Kubernetes networking and traffic policies and loads them into the running kernel so they are active immediately. The /etc/sysctl.d/kubernetes.conf file will persist these settings across reboots as well.

Step 4: Install Containerd Runtime (all nodes)

- Wget https://download.docker.com/linux/ubuntu/dists/jammy/pool/stable/amd64/containerd.io_1.6.28-1_amd64.deb
- sudo dpkg -i containerd.io_1.6.28-1_amd64.deb

Configure containerd to start using systemd as cgroup:

- 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

Restart and enable the containerd service:

- sudo systemctl restart containerd
- sudo systemctl enable containerd

Step 5: Add Apt Repository for Kubernetes (all nodes)

Kubernetes packages are not available in the default Ubuntu 22.04 repositories. Add the Kubernetes repositories with the following commands:

- 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

Step 6: Install Kubectl, Kubeadm, and Kubelet (all nodes)

After adding the repositories, install essential Kubernetes components, including kubectl, kubelet, and kubeadm, on all nodes with the following commands:

- sudo apt-get update
- sudo apt-get install -y kubelet kubeadm kubectl
- sudo apt-mark hold kubelet kubeadm kubectl

Step 7: Initialize Kubernetes Cluster with Kubeadm (master node)

- sudo kubeadm init

After the initialization is complete make a note of the **kubeadm join** command for future reference.

Run the following commands 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

Step 8: Add Worker Nodes to the Cluster (Master nodes)

- kubeadm token create --print-join-command

Token will get Generated Copy and run the command on worker Node.

Step :9 Install Kubernetes Network Plugin (master node)

To enable communication between pods in the cluster, you need a network plugin. Install the Calico network plugin with the following command from the master node:

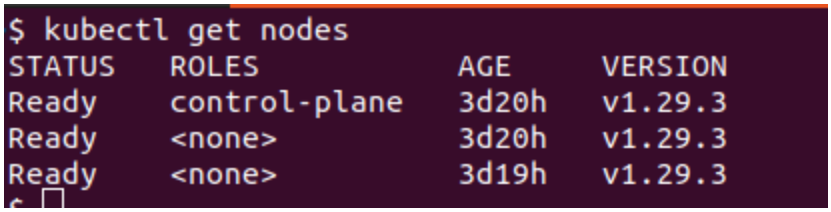
- **curl https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml -O**
- **kubectl apply -f calico.yaml**

```
NAME                                       READY   STATUS    RESTARTS       AGE
calico-kube-controllers-658d97c59c-qqhsz   1/1     Running   0              3d19h
calico-node-92cwd                          1/1     Running   0              3d19h
calico-node-lz6gz                          1/1     Running   1 (3d1h ago)   3d19h
calico-node-zz2t2                          1/1     Running   0              3d19h
```

Step 10: Verify the cluster and test (master node)

Finally, we want to verify whether our cluster is successfully created.

- kubectl get nodes
- kubectl get pods -n kube-system


$ kubectl get nodes
STATUS ROLES AGE VERSION
Ready control-plane 3d20h v1.29.3
Ready <none> 3d20h v1.29.3
Ready <none> 3d19h v1.29.3