In this tutorial, we will show you how to install Kubernetes on Ubuntu 24.04 using kubeadm step-by-step.

Kubernetes (k8s) is a free and open-source container orchestration tool that allows automating deployment, scaling and management of container-based applications.

Prerequisites for Installing Kubernetes on Ubuntu 24.04

* Pre-Install Ubuntu 24.04 Instances
* SSH enabled on all the Instances
* Regular User with sudo rights
* Minimum of 2GB RAM, 2 CPUs and 20 GB free disk space on each instance
* Stable Internet Connectivity

For the demonstration, we are using three instances of Ubuntu 24.04 which are classified as

* Instance 1 : Master Node (k8s-master-noble 192.168.1.120)
* Instance 2 : Worker Node (k8s-worker01-noble 192.168.1.121)
* Instance 3 : Worker Node (k8s-worker02-noble 192.168.1.122)

**How to Install Kubernetes on Ubuntu 24.04**

Without any further delay, let’s jump into Installation steps of Kubernetes on Ubuntu 24.04

1) Set Host Name and Update hosts file

SSH to each Ubuntu 24.04 instance and set their respective hostname using hostnamectl command.

$ sudo hostnamectl set-hostname "k8s-master-noble"      // Master Node

$ sudo hostnamectl set-hostname "k8s-worker01-noble"  // Worker Node 1

$ sudo hostnamectl set-hostname "k8s-worker02-noble"  // Worker Node 2

Add the following lines to **/etc/hosts** file on each instance.

192.168.1.120  k8s-master-noble

192.168.1.121  k8s-worker01-noble

192.168.1.122  k8s-worker02-noble

2) Disable Swap and Load Kernel Modules

It is highly recommended to disable swap space on your Ubuntu instances so that Kubernetes cluster works smoothly. Run beneath command on each instance to disable swap space.

$ sudo swapoff -a

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

Now, load the following kernel modules using modprobe command.

$ sudo modprobe overlay

$ sudo modprobe br\_netfilter

For the permanent loading of these modules, create the file with following content.

$ sudo tee /etc/modules-load.d/k8s.conf <<EOF

overlay

br\_netfilter

EOF

Next, add the kernel parameters like IP forwarding. Create a file and load the parameters using sysctl command,

$ sudo tee /etc/sysctl.d/kubernetes.conf <<EOT

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

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

net.ipv4.ip\_forward = 1

EOT

To load the above kernel parameters, run

$ sudo sysctl --system

3) Install and Configure Containerd

Containerd provides the container run time for Kubernetes. So, Install containerd on all three instances.

First install containerd dependencies,

$ sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

Next, add containerd repository using following set of commands.

$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/containerd.gpg

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

Now, install containerd using following apt command.

$ sudo apt update && sudo apt install containerd.io -y

Next, configure containerd so that it starts using **SystemdCgroup**. Run beneath commands.

$ 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 containerd service so that above changes come into the affect.

$ sudo systemctl restart containerd

4) Add Kubernetes Package Repository

Kubernetes packages are not available in the default package repositories of Ubuntu 24.04, so for its installation first add it’s repository. Run these steps on each instance.

**Note**: At the time of writing this post, latest version of Kubernetes was **1.30**. So you can this version according your requirement.

Download the public signing key for the Kubernetes package repository using curl command.

$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/k8s.gpg

Next, add the Kubernetes repository by running following command.

$ echo 'deb [signed-by=/etc/apt/keyrings/k8s.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/k8s.list

5) Install Kubernetes Components (Kubeadm, kubelet & kubectl)

Install Kubernetes components like **Kubeadm**, **kubelet** and **kubectl**, run following apt commands on all the instances.

$ sudo apt update

$ sudo apt install kubelet kubeadm kubectl -y

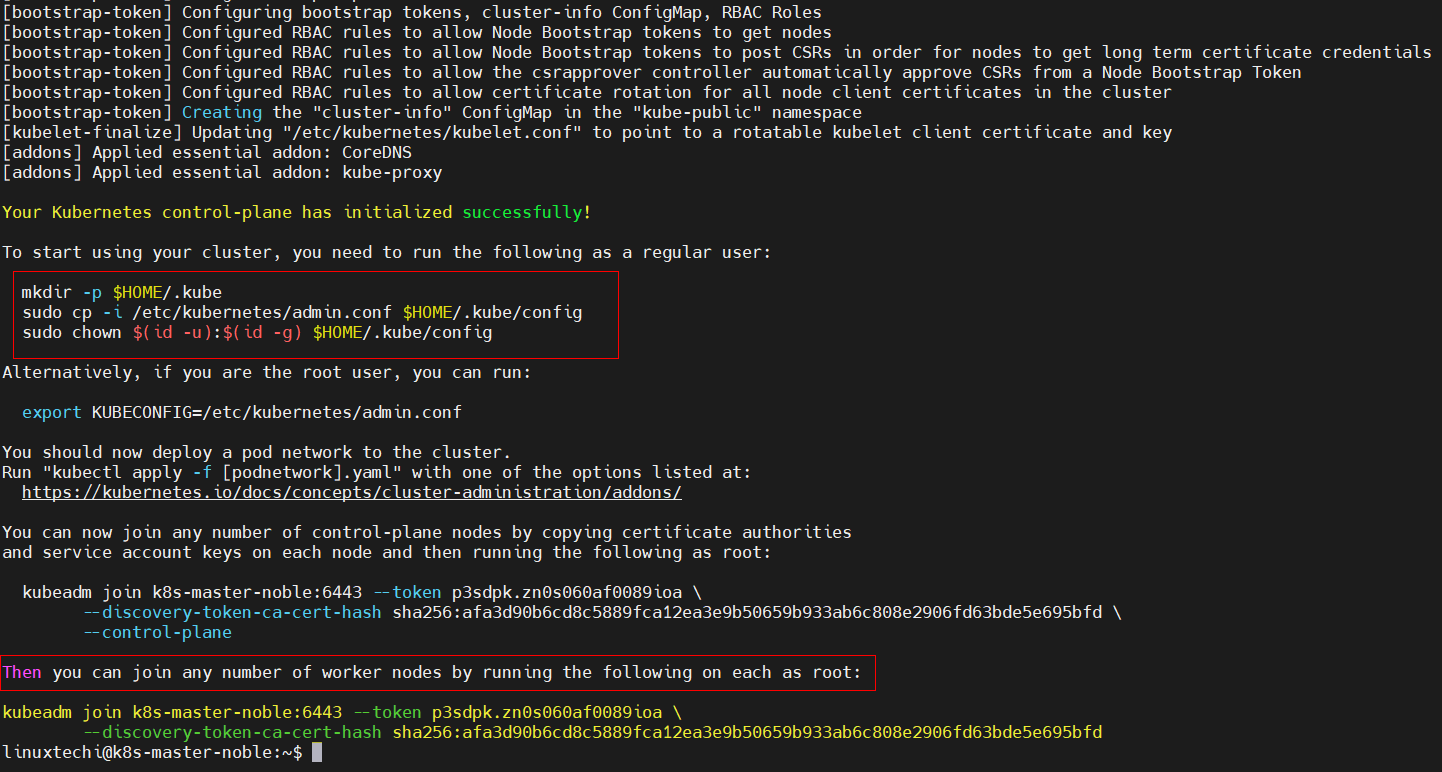
6) Initialize Kubernetes Cluster

As all the prerequisites are met, now we are good to start the installation of Kubernetes on Ubuntu 24.04.

Run following **Kubeadm** command from the master node only to initialize the Kubernetes cluster.

$ sudo kubeadm init --control-plane-endpoint=k8s-master-noble

This command will pull the required images for your Kubernetes cluster. Once this command is executed successfully, we will get the output something like below:



In the output above, we will get a series of commands like how to start interacting with your Kubernetes cluster and command to join any worker node to join this cluster.

On the master node, run following set of commands.

$ mkdir -p $HOME/.kube

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

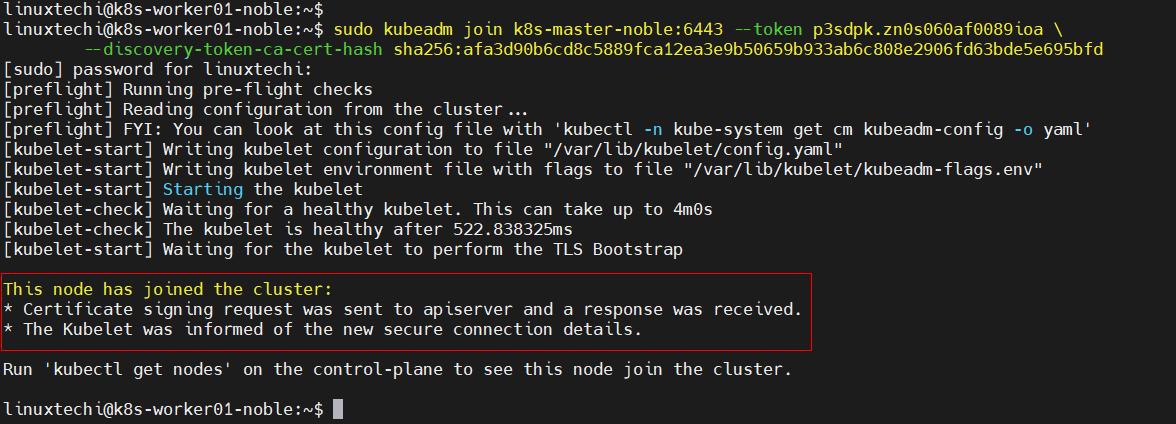
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

Next copy the command to join any worker node from the above output, run it on both the worker nodes. In my case, command would be:

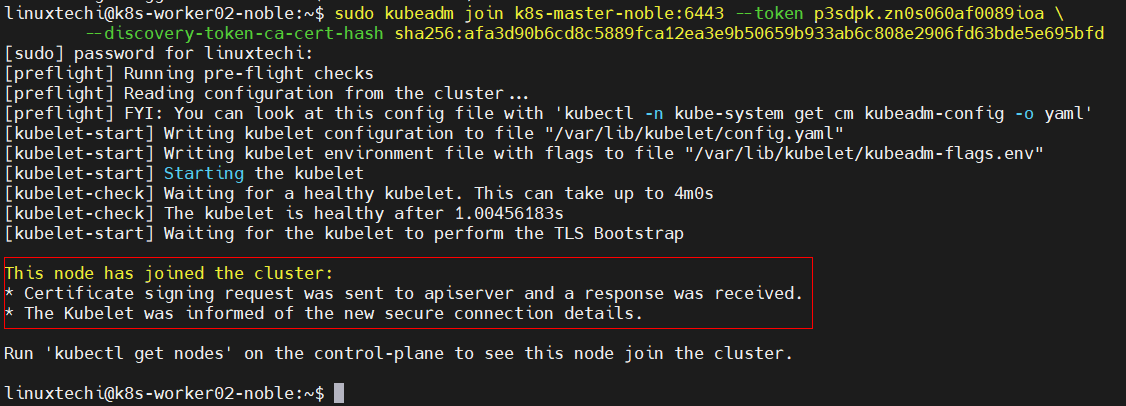
$ sudo kubeadm join k8s-master-noble:6443 --token p3sdpk.zn0s060af0089ioa \

        --discovery-token-ca-cert-hash sha256:afa3d90b6cd8c5889fca12ea3e9b50659b933ab6c808e2906fd63bde5e695bfd

Output from first worker node

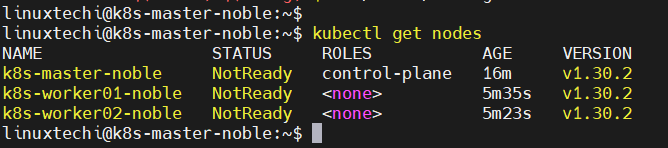


Similarly output from the second worker node



Now head back to the master node and run **kubectl get nodes** command to verify the status of worker nodes.

$ kubectl get nodes

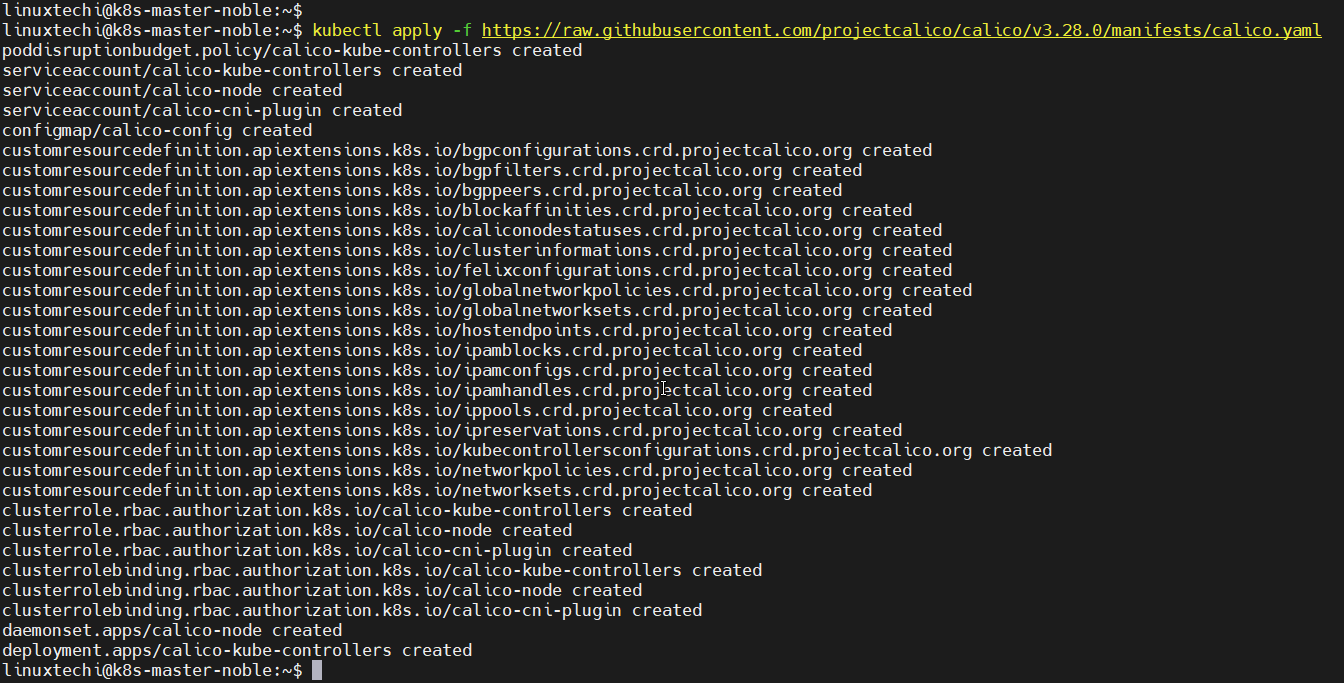


Output confirms that worker nodes have joined the cluster, but the status is **NotReady**. So,  in order to make status Ready, we need to install network add-ons plugin like **calico** on this cluster.

7) Install Calico Network Add-on Plugin

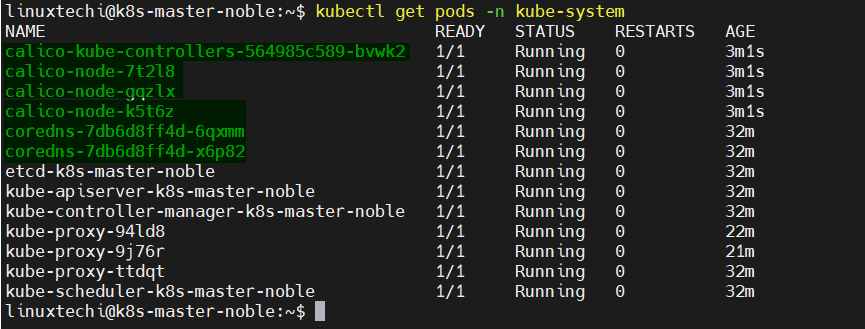
To install calico network plugin, run beneath command from the master node only.

$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.29.1/manifests/calico.yaml

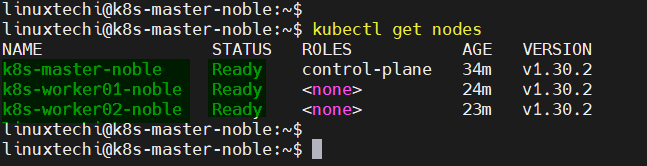


After the successful installation of calico, nodes status will change to Ready in a minute or two.

$ kubectl get pods -n kube-system



$ kubectl get nodes



Output above confirms that nodes are in Ready state.

8) Test Kubernetes Installation

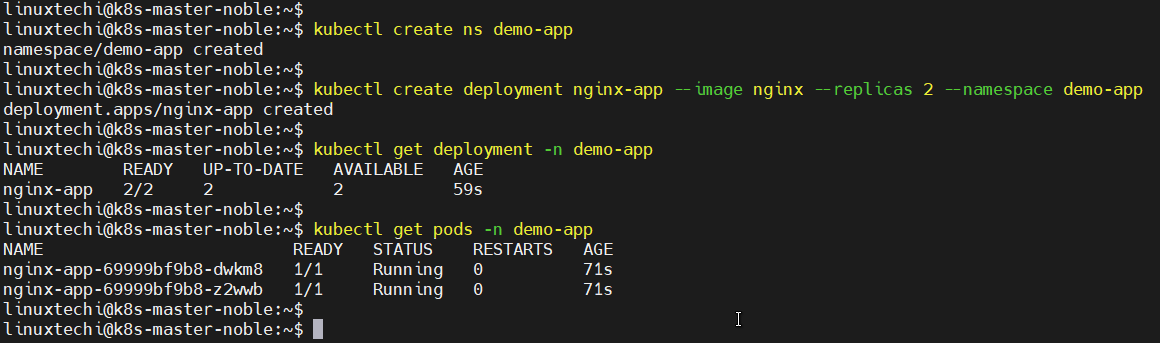
To test the Kubernetes installation, let’s create nginx based deployment with replica count 2. Execute the following **kubectl** command from the master node.

$ kubectl create ns demo-app

$ kubectl create deployment nginx-app --image nginx --replicas 2 --namespace demo-app

$ kubectl get deployment -n demo-app

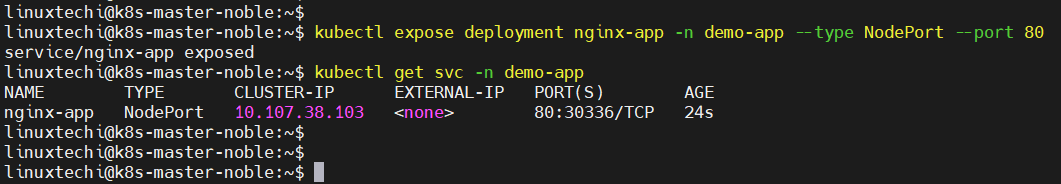
$ kubectl get pods -n demo-app



Next expose this deployment using NodePort type, run

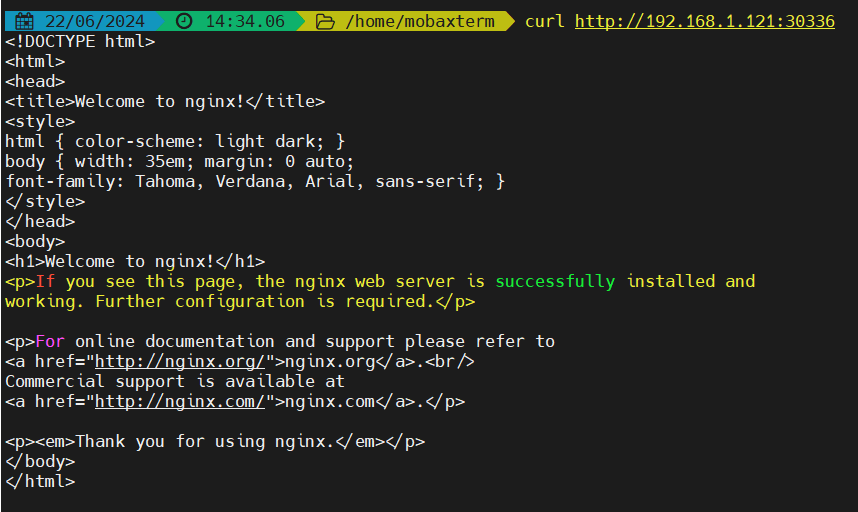
$ kubectl expose deployment nginx-app -n demo-app --type NodePort --port 80

$ kubectl get svc -n demo-app



Now try to access your application using **nodeport** as shown below

$ curl http://<Any-worker-IP>:30336



Great, output above confirms that we can access nginx based application outside of our Kubernetes cluster using the nodeport. This confirms that Kubernetes installation is successful.