**Set up Docker**

Kubernetes requires a CRI-compliant container engine runtime such as [Docker, containerd, or CRI-O](https://phoenixnap.com/kb/docker-vs-containerd-vs-cri-o). This article shows you how to deploy Kubernetes using [Docker](https://phoenixnap.com/kb/what-is-docker).

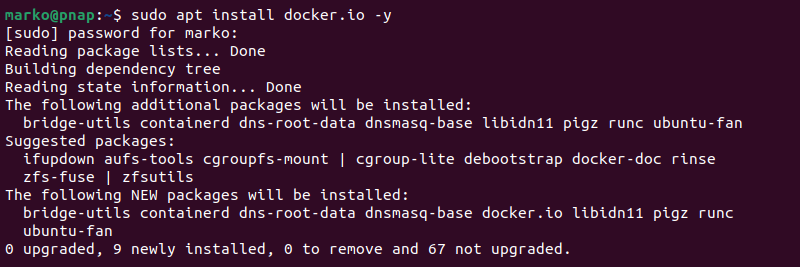
[Install Docker](https://phoenixnap.com/kb/install-docker-on-ubuntu-20-04) on each server node by executing the steps below:

1. Update the package list:

sudo apt update

2. Install Docker with the following command:

sudo apt install docker.io -y

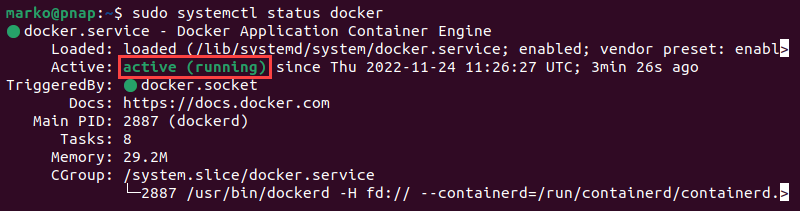


3. Set Docker to launch on [boot](https://phoenixnap.com/glossary/boot-definition) by entering:

sudo systemctl enable docker

4. Verify Docker is running:

sudo systemctl status docker



5. If Docker is not running, start it with the following command:

sudo systemctl start docker

**Install Kubernetes**

Setting up Kubernetes on an Ubuntu system involves adding the Kubernetes [repository](https://phoenixnap.com/glossary/what-is-a-repository) to the [APT](https://phoenixnap.com/kb/apt-linux) sources list and installing the relevant tools. Follow the steps below to install Kubernetes on all the nodes in your cluster.

**Step 1: Add Kubernetes Signing Key**

Since Kubernetes comes from a non-standard repository, download the signing key to ensure the software is authentic.

On each node, use the [curl command](https://phoenixnap.com/kb/curl-command) to download the key and store it in a safe place (default is */etc/apt/keyrings/*:

curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes.gpg

**Step 2: Add Software Repositories**

Kubernetes is not included in the default Ubuntu repositories. To add the Kubernetes repository to your list, enter this command on each node:

echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/kubernetes.gpg] http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list

Ensure all packages are up to date:

sudo apt update

**Step 3: Install Kubernetes Tools**

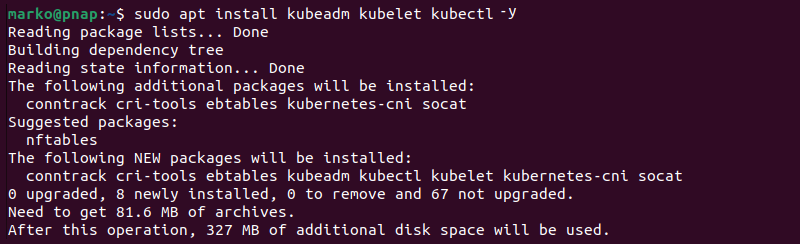
Each Kubernetes deployment consists of three separate tools:

* **Kubeadm**. A tool that initializes a Kubernetes cluster by fast-tracking the setup using community-sourced [best practices](https://phoenixnap.com/kb/kubernetes-best-practices).
* **Kubelet**. The work package that runs on every node and starts containers. The tool gives you command-line access to clusters.
* [Kubectl](https://phoenixnap.com/kb/kubectl-commands-cheat-sheet). The [command-line interface](https://phoenixnap.com/glossary/command-line-interface) for interacting with clusters.

Execute the following commands on each server node to install the [Kubernetes tools](https://phoenixnap.com/blog/kubernetes-tools):

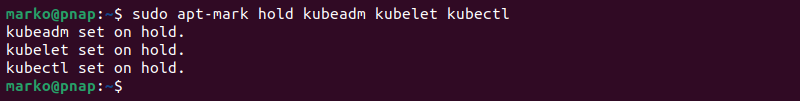
1. Run the **install**command:

sudo apt install kubeadm kubelet kubectl



2. Mark the packages as held back to prevent automatic installation, upgrade, or removal:

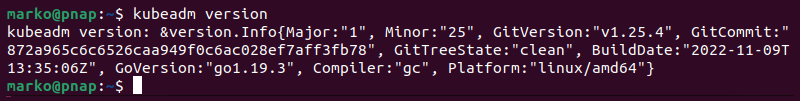
sudo apt-mark hold kubeadm kubelet kubectl



**Note:** The process presented in this tutorial prevents APT from automatically updating Kubernetes. For instructions on how to update, please see the official [developers' instructions](https://kubernetes.io/docs/tasks/).

3. Verify the installation with:

kubeadm version



The output of the **version** command shows basic deployment information.

**Note**: BMC offers balanced and affordable [server instances](https://phoenixnap.com/bare-metal-cloud/instances) well suited for containerized services deployment. To simplify and streamline the process, deploy Kubernetes clusters on BMC using our [Rancher solution](https://phoenixnap.com/kb/rancher-bmc-integration-kubernetes).

**Deploy Kubernetes**

With the necessary tools installed, proceed to deploy the cluster. Follow the steps below to make the necessary system adjustments, initialize the cluster, and join worker nodes.

**Step 1: Prepare for Kubernetes Deployment**

This section shows you how to prepare the servers for a Kubernetes deployment. Execute the steps below on each server node:

1. Disable all [swap spaces](https://phoenixnap.com/kb/swap-space) with the **swapoff** command:

sudo swapoff -a

Then use the [sed command](https://phoenixnap.com/kb/linux-sed" \t "_blank) below to make the necessary adjustments to the */etc/fstab* file:

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

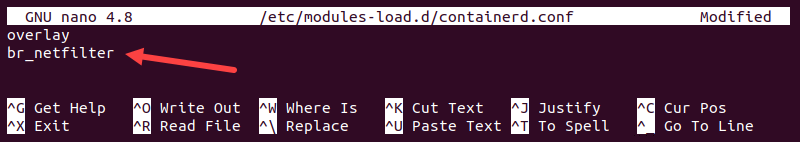
2. Load the required **containerd** modules. Start by opening the containerd configuration file in a [text editor](https://phoenixnap.com/kb/best-linux-text-editors-for-coding), such as [nano](https://phoenixnap.com/kb/use-nano-text-editor-commands-linux" \t "_blank):

sudo nano /etc/modules-load.d/containerd.conf

3. Add the following two lines to the file:

overlay

br\_netfilter



Save the file and exit.

4. Next, use the [modprobe command](https://phoenixnap.com/kb/modprobe-command" \t "_blank) to add the modules:

sudo modprobe overlay

sudo modprobe br\_netfilter

5. Open the **kubernetes.conf** file to configure Kubernetes networking:

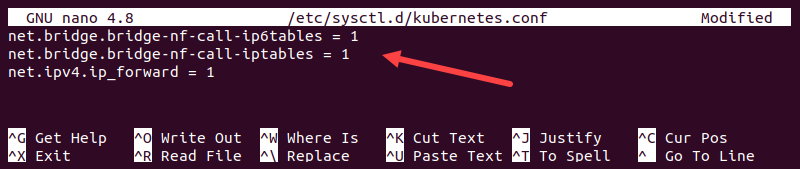
sudo nano /etc/sysctl.d/kubernetes.conf

6. Add the following lines to the file:

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

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

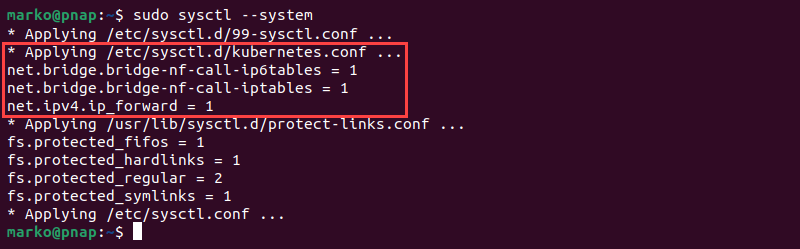
net.ipv4.ip\_forward = 1



Save the file and exit.

7. Reload the configuration by typing:

sudo sysctl --system



**Step 2: Assign Unique Hostname for Each Server Node**

1. Decide which server will be the master node. Then, enter the command on that node to name it accordingly:

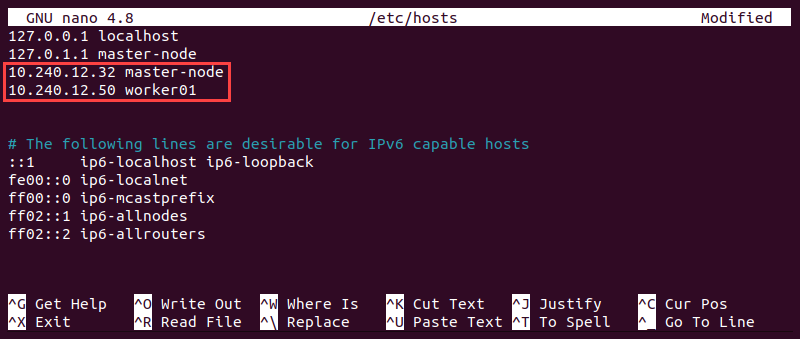
sudo hostnamectl set-hostname master-node

2. Next, [set the hostname](https://phoenixnap.com/kb/ubuntu-20-04-change-hostname) on the first worker node by entering the following command:

sudo hostnamectl set-hostname worker01

If you have additional worker nodes, use this process to set a unique hostname on each.

3. [Edit the hosts file](https://phoenixnap.com/kb/linux-hosts-file) on each node by adding the [IP addresses](https://phoenixnap.com/kb/how-to-find-ip-address-linux) and hostnames of the servers that will be part of the cluster.



4. Restart the terminal application to apply the hostname change.

**Step 3: Initialize Kubernetes on Master Node**

Once you finish setting up hostnames on cluster nodes, switch to the master node and follow the steps to initialize Kubernetes on it:

1. Open the **kubelet** file in a text editor.

sudo nano /etc/default/kubelet

2. Add the following line to the file:

KUBELET\_EXTRA\_ARGS="--cgroup-driver=cgroupfs"

Save and exit.

3. Reload the configuration and restart the kubelet:

sudo systemctl daemon-reload && sudo systemctl restart kubelet

4. Open the Docker daemon configuration file:

sudo nano /etc/docker/daemon.json

5. Append the following configuration block:

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

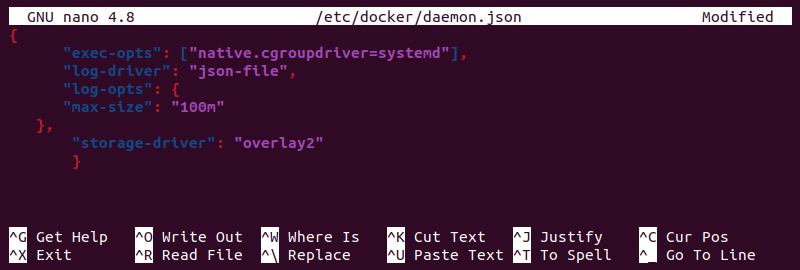
"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}



Save the file and exit.

6. Reload the configuration and restart Docker:

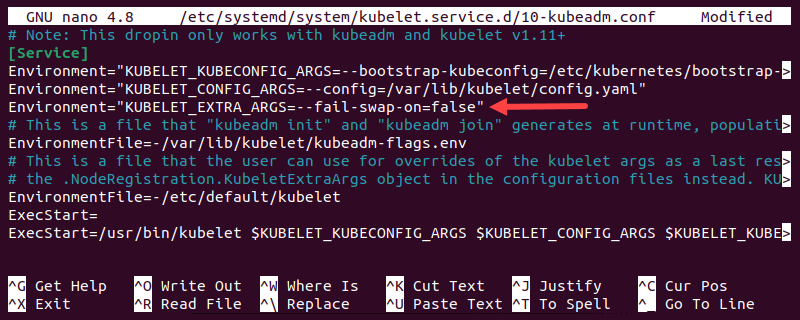
sudo systemctl daemon-reload && sudo systemctl restart docker

7. Open the **kubeadm** configuration file:

sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

8. Add the following line to the file:

Environment="KUBELET\_EXTRA\_ARGS=--fail-swap-on=false"



Save the file and exit.

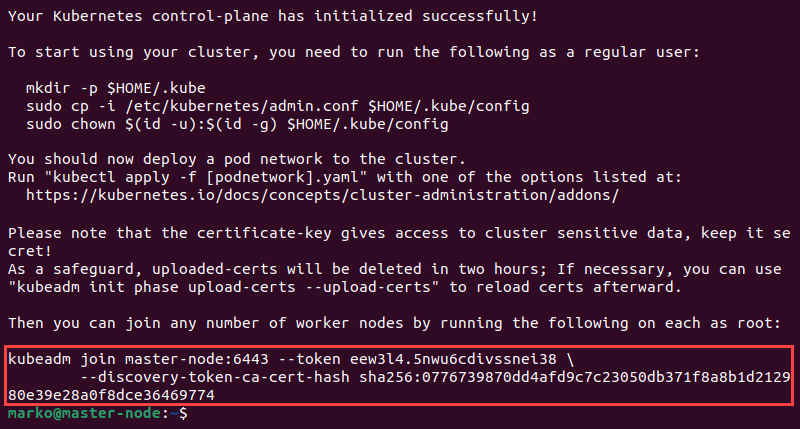
9. Reload the configuration and restart the kubelet:

sudo systemctl daemon-reload && sudo systemctl restart kubelet

10. Finally, initialize the cluster by typing:

sudo kubeadm init --control-plane-endpoint=master-node --upload-certs

Once the operation finishes, the output displays a **kubeadm join** command at the bottom. Make a note of this command, as you will use it to join the worker nodes to the cluster.



11. Create a [directory](https://phoenixnap.com/glossary/what-is-a-directory) for the Kubernetes cluster:

mkdir -p $HOME/.kube

12. Copy the configuration file to the directory:

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

13. Change the ownership of the directory to the current user and group using the [chown command](https://phoenixnap.com/kb/linux-chown-command-with-examples" \t "_blank):

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

**Step 4: Deploy Pod Network to Cluster**

A pod network is a way to allow communication between different nodes in the cluster. This tutorial uses the **Flannel** node network manager to create a pod network.

Apply the Flannel manager to the master node by executing the steps below:

1. Use **kubectl** to install Flannel:

kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

2. Untaint the node:

kubectl taint nodes --all node-role.kubernetes.io/control-plane-

**Step 5: Join Worker Node to Cluster**

Repeat the following steps on each worker node to create a cluster:

1. Stop and disable **AppArmor**:

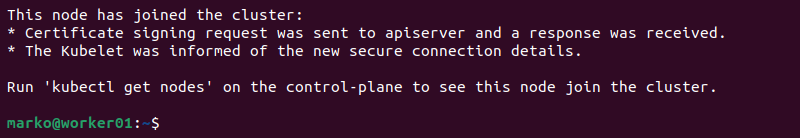
sudo systemctl stop apparmor && sudo systemctl disable apparmor

2. Restart **containerd**:

sudo systemctl restart containerd.service

3. Apply the **kubeadm join** command from **Step 3** on worker nodes to connect them to the master node. Prefix the command with **sudo**:

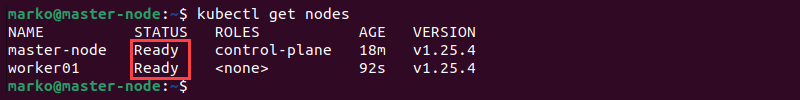
sudo kubeadm join [master-node-ip]:6443 --token [token] --discovery-token-ca-cert-hash sha256:[hash]



Replace **[master-node-ip]**, **[token]**, and **[hash]** with the values from the **kubeadm join** command output.

4. After a few minutes, switch to the master server and enter the following command to check the status of the nodes:

kubectl get nodes



The system displays the master node and the worker nodes in the cluster.