

Kubernetes Installation & Setup

Set up Docker

1. Update the package list:

```
sudo apt update
```

2. Install Docker with the following command:

```
sudo apt install docker.io -y
```

```
marko@pnap:~$ sudo apt install docker.io -y
[sudo] password for marko:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base libidn11 pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io libidn11 pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 67 not upgraded.
```

3. Set Docker to launch on boot by entering:

```
sudo systemctl enable docker
```

4. Verify Docker is running:

```
sudo systemctl status docker
```

```
marko@pnap:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabl>
   Active: active (running) since Thu 2022-11-24 11:26:27 UTC; 3min 26s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 2887 (dockerd)
       Tasks: 8
      Memory: 29.2M
     CGroup: /system.slice/docker.service
             └─2887 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.>
```

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](#) to the [APT](#) 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](#) to download the key and store it in a safe place (default is `/etc/apt/keyrings/`):

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.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 [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo
```

```
tee /etc/apt/sources.list.d/kubernetes.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](#).
- **Kubelet.** The work package that runs on every node and starts containers. The tool gives you command-line access to clusters.
- **Kubect.** The [command-line interface](#) for interacting with clusters.

Execute the following commands on each server node to install the [Kubernetes tools](#):

1. Run the **install** command:

```
sudo apt install kubeadm kubelet kubectl
```

```
marko@pnap:~$ sudo apt install kubeadm kubelet kubectl -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 67 not upgraded.
Need to get 81.6 MB of archives.
After this operation, 327 MB of additional disk space will be used.
```

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

```
sudo apt-mark hold kubeadm kubelet kubect1
```

```
marko@pnap:~$ sudo apt-mark hold kubeadm kubelet kubect1
kubeadm set on hold.
kubelet set on hold.
kubect1 set on hold.
marko@pnap:~$
```

3. Verify the installation with:

```
kubeadm version
```

```
marko@pnap:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.4", GitCommit:"
872a965c6c6526caa949f0c6ac028ef7aff3fb78", GitTreeState:"clean", BuildDate:"2022-11-09T
13:35:06Z", GoVersion:"go1.19.3", Compiler:"gc", Platform:"linux/amd64"}
marko@pnap:~$
```

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 with the **swapoff** command:

```
sudo swapoff -a
```

Then use the sed command below to make the necessary adjustments to the */etc/fstab* file:

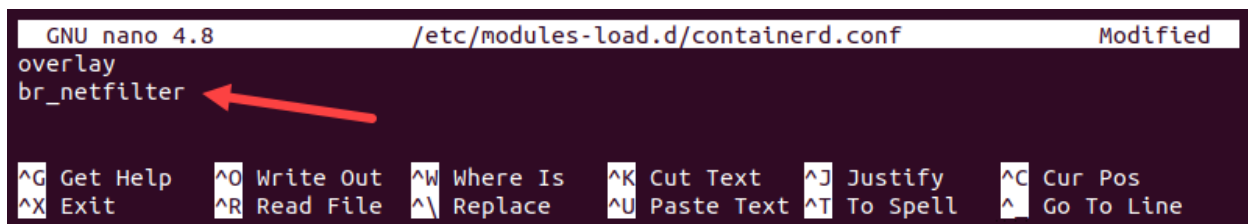
```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

2. Load the required **containerd** modules. Start by opening the containerd configuration file in a text editor, such as nano:

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

3. Add the following two lines to the file:

```
overlay  
br_netfilter
```



```
GNU nano 4.8 /etc/modules-load.d/containerd.conf Modified  
overlay  
br_netfilter  
  
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos  
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell   ^_ Go To Line
```

Save the file and exit.

4. Next, use the modprobe command to add the modules:

```
sudo modprobe overlay
```

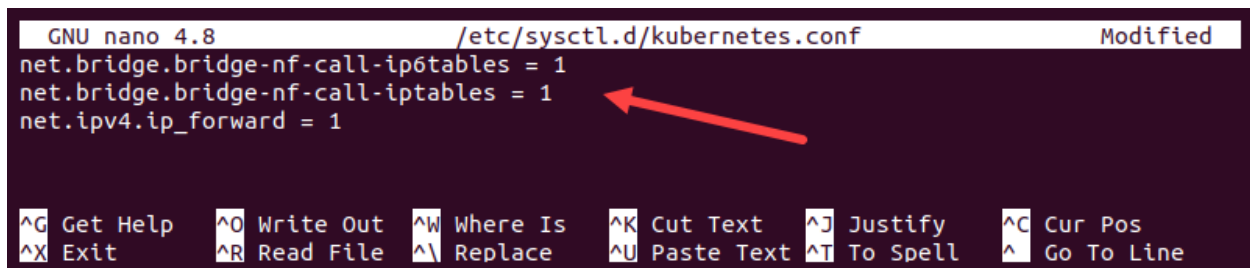
```
sudo modprobe br_netfilter
```

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
```



```
GNU nano 4.8 /etc/sysctl.d/kubernetes.conf Modified
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Save the file and exit.

7. Reload the configuration by typing:

```
sudo sysctl --system
```



```
marko@pnap:~$ sudo sysctl --system
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
* Applying /usr/lib/sysctl.d/protect-links.conf ...
fs.protected_fifos = 1
fs.protected_hardlinks = 1
fs.protected_regular = 2
fs.protected_symlinks = 1
* Applying /etc/sysctl.conf ...
marko@pnap:~$
```

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
```

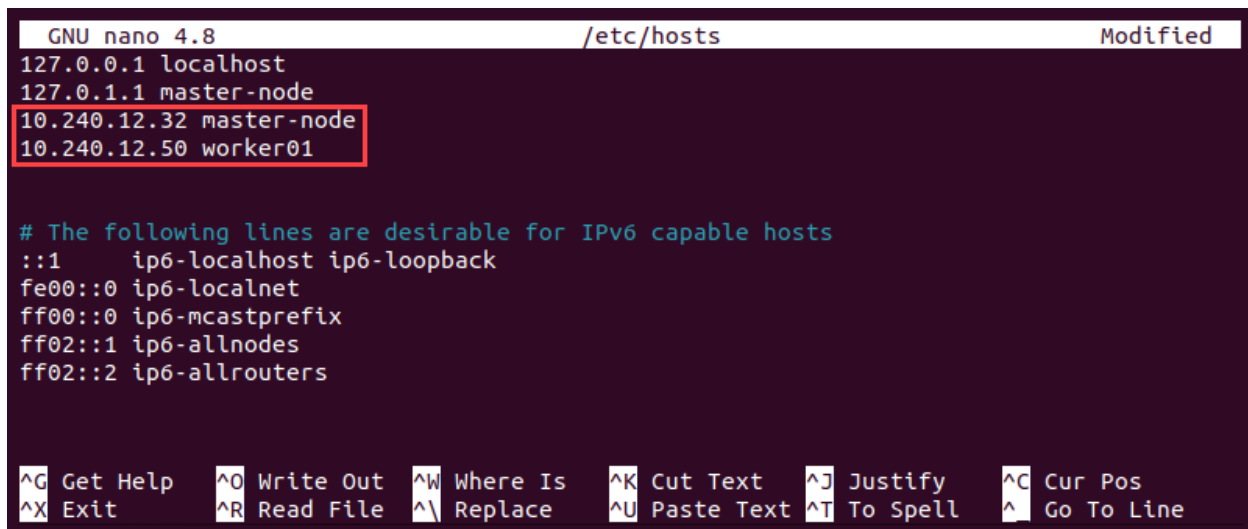
2. Next, set the hostname on the first worker node by entering the following command:

```
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 on each node by adding the IP addresses and hostnames of the servers that will be part of the cluster.

```
sudo nano /etc/hosts
```



```
GNU nano 4.8 /etc/hosts Modified
127.0.0.1 localhost
127.0.1.1 master-node
10.240.12.32 master-node
10.240.12.50 worker01

# The following lines are desirable for IPv6 capable hosts
::1    ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace     ^U Paste Text  ^T To Spell   ^_ Go To Line
```

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"
}
```

A screenshot of the GNU nano 4.8 text editor editing the file /etc/docker/daemon.json. The editor's status bar at the top shows 'GNU nano 4.8', the file path '/etc/docker/daemon.json', and the word 'Modified'. The main editing area contains a JSON configuration block with the following content: {, "exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file", "log-opts": {, "max-size": "100m", }, "storage-driver": "overlay2", }. The bottom status bar displays various keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^R Read File, ^\ Replace, ^U Paste Text, ^T To Spell, and ^_ Go To Line.

Save the file and exit.

6. Reload the configuration and restart Docker:

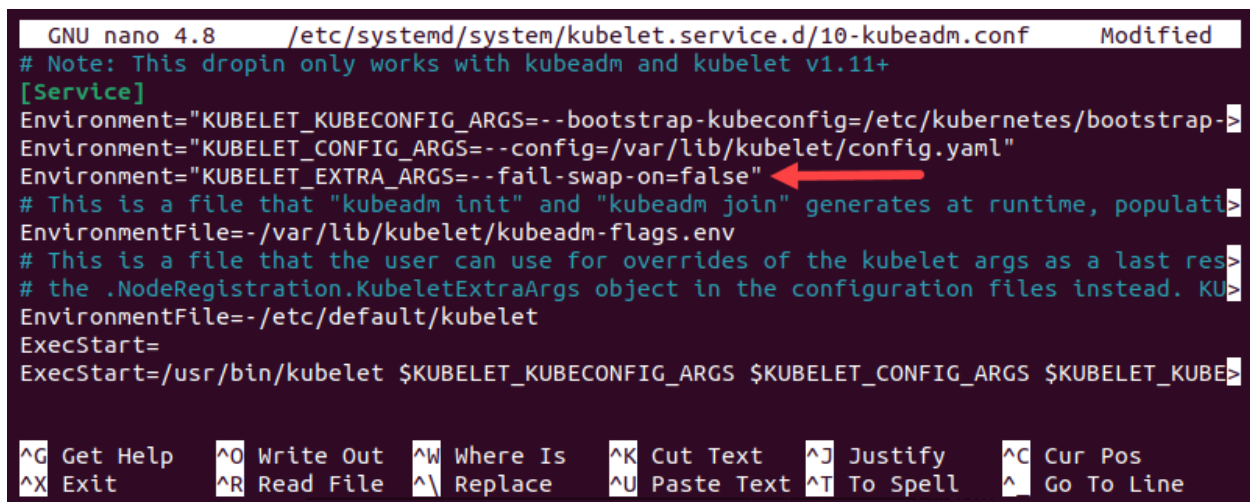
```
sudo systemctl daemon-reload && sudo systemctl restart docker
```

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"
```



```
GNU nano 4.8 /etc/systemd/system/kubelet.service.d/10-kubeadm.conf Modified
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap->
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populati>
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last res>
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KU>
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBE>

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell   ^_ Go To Line
```

Save the file and exit.

9. Reload the configuration and restart the kubelet:

```
sudo systemctl daemon-reload && sudo systemctl restart kubele
t
```

10. Finally, initialize the cluster by typing:

```
sudo kubeadm init --control-plane-endpoint=master-node --uplo
```

ad-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.

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

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/

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join master-node:6443 --token eew3l4.5nwu6cdlvssnei38 \
--discovery-token-ca-cert-hash sha256:0776739870dd4afd9c7c23050db371f8a8b1d212980e39e28a0f8dce36469774
marko@master-node:~$
```

Note: You might need to reinitialize the cluster with another command with the network falannel plugin or ip address of the node (if you face network plugin issue latter)

11. Create 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:

```
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 (Only if you have worker nodes)

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] --dis
```

```
coverly-token-ca-cert-hash sha256:[hash]
```

```
This node has joined the cluster:
* Certificate signing request was sent to apiservert 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.

marko@worker01:~$
```

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
```

```
marko@master-node:~$ kubectl get nodes
NAME           STATUS    ROLES    AGE   VERSION
master-node    Ready     control-plane  18m   v1.25.4
worker01       Ready     <none>      92s   v1.25.4
marko@master-node:~$
```

Conclusion

After following the steps presented in this article, you should have **Kubernetes installed on Ubuntu**. The article included instructions on installing the necessary packages and deploying Kubernetes on all your nodes.

If you are a beginner with no experience in Kubernetes deployment, [Minikube](#) is a great place to start.

[Automatic Installation/Uninstallation Elasticsearch, Kibana, logstash, Nmap with Sheel Script](#)