Install and Deploy Kubernetes on Ubuntu 18.04 LTS

What is Kubernetes?

Kubernetes is a free and open-source container management system that provides a platform for deployment automation, scaling, and operations of application containers across clusters of host computers. With Kubernetes, you can freely make use of the hybrid,on-premise, and public cloud infrastructure in order to run deployment tasks of your organization.

In this tutorial, we will explain how to install Kubernetes on an Ubuntu system and also deploy Kubernetes on a two-node Ubuntu cluster.

The commands and procedures mentioned in this article have been run on an Ubuntu 18.04 LTS system. Since we will be using the Ubuntu command line, the Terminal, for running all the commands, you can open it either through the system Dash or the Ctrl+Alt+T shortcut.

Kubernetes Installation

The two-node cluster that we will be forming in this article will consist of a Master node and a Slave node. Both these nodes need to have Kubernetes installed on them. Therefore, follow the steps described below to install Kubernetes on both the Ubuntu nodes.

Step 1: Install Docker on both the nodes

Install the Docker utility on both the nodes by running the following command as sudo in the Terminal of each node:

```
$ sudo apt install docker.io
```

```
File  Edit  View  Search  Terminal  Help
sana@Linux:~$ sudo apt install docker.io
[sudo] password for sana:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
  chromium-codecs-ffmpeg-extra gsfonts-x11 libid3tag0 libimlib2 lynx-
  pepperflashplugin-nonfree
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  cgroupfs-mount git git-man liberror-perl ubuntu-fan
Suggested packages:
  aufs-tools btrfs-tools debootstrap docker-doc rinse zfs-fuse | zfsu
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-g
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  cgroupfs-mount docker.io git git-man liberror-perl ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 171 not upgraded.
Need to get 34.8 MB of archives.
After this operation, 171 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

You will be prompted with a Y/n option in order to proceed with the installation. Please enter Y and then hit enter to continue. Docker will then be installed on your system. You can verify the installation and also check the version number of Docker through the following command:

```
$ docker --version
```

```
sana@Linux:~$ docker --version
Docker version 17.12.1-ce, build 7390fc6
```

Step 2: Enable Docker on both the nodes

Enable the Docker utility on both the nodes by running the following command on each:

```
$ sudo systemctl enable docker
```

```
sana@Linux:~$  sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
```

Step 3: Add the Kubernetes signing key on both the nodes

Run the following command in order to get the Kubernetes signing key:

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key
add
```

```
sana@Linux:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | su
do apt-key add
OK
```

If Curl is not installed on your system, you can install it through the following command as root:

```
$ sudo apt install curl
```

```
sana@Linux:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no lor
  chromium-codecs-ffmpeg-extra gsfonts-x11 libid3tag0 libimlib2 ly
  pepperflashplugin-nonfree
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libcurl4
The following packages will be REMOVED:
  libcurl3
The following NEW packages will be installed:
  curl libcurl4
0 upgraded, 2 newly installed, 1 to remove and 171 not upgraded.
Need to get 373 kB of archives.
After this operation, 392 kB of additional disk space will be used
Do you want to continue? [Y/n]
```

You will be prompted with a Y/n option in order to proceed with the installation. Please enter Y and then hit enter to continue. The Curl utility will then be installed on your system.

Step 4: Add Xenial Kubernetes Repository on both the nodes

Run the following command on both the nodes in order to add the Xenial Kubernetes repository:

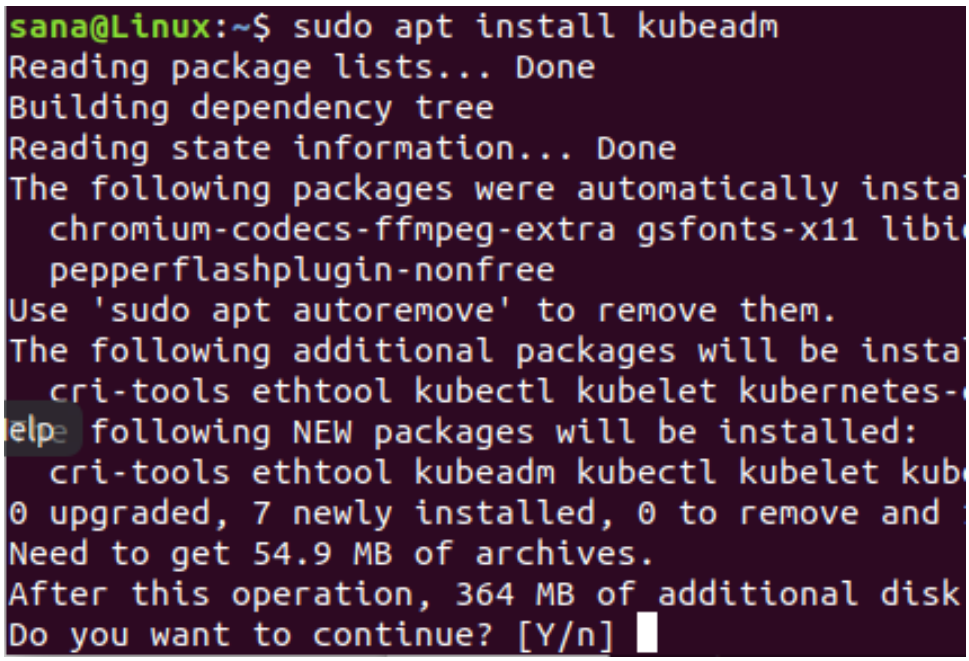
```
$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

```
sana@Linux:~$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-x
enial main"
Hit:1 http://ppa.launchpad.net/apt-fast/stable/ubuntu bionic InRelease
Hit:2 http://archive.canonical.com bionic InRelease
Ign:3 http://ppa.launchpad.net/jd-team/jdownloader/ubuntu bionic InRelease
Hit:4 http://ppa.launchpad.net/linrunner/tlp/ubuntu bionic InRelease
Hit:5 http://pk.archive.ubuntu.com/ubuntu bionic InRelease
Hit:6 http://pk.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:7 http://pk.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:8 http://ppa.launchpad.net/webupd8team/java/ubuntu bionic InRelease
Err:9 http://ppa.launchpad.net/jd-team/jdownloader/ubuntu bionic Release
  404  Not Found [IP: 91.189.95.83 80]
Hit:10 http://security.ubuntu.com/ubuntu bionic-security InRelease
```

Step 5: Install Kubeadm

The final step in the installation process is to install Kubeadm on both the nodes through the following command:

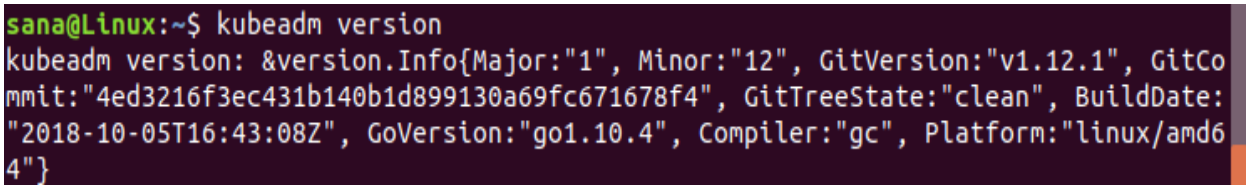
```
$ sudo apt install kubeadm
```



You will be prompted with a Y/n option in order to proceed with the installation. Please enter Y and then hit enter to continue. Kubeadm will then be installed on your system.

You can check the version number of Kubeadm and also verify the installation through the following command:

```
$ kubeadm version
```

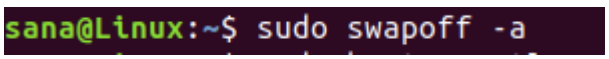


Kubernetes Deployment

Step 1: Disable swap memory (if running) on both the nodes

You need to disable swap memory on both the nodes as Kubernetes does not perform properly on a system that is using swap memory. Run the following command on both the nodes in order to disable swap memory

```
$ sudo swapoff -a
```



Step 2: Give Unique hostnames to each node

Run the following command in the master node in order to give it a unique hostname:

```
$ sudo hostnamectl set-hostname master-node
```

Run the following command in the slave node in order to give it a unique hostname:

```
$ hostnamectl set-hostname slave-node
```

Step3: Initialize Kubernetes on the master node

Run the following command as sudo on the master node:

```
$ sudo kubeadm init --pod-network-cidr=192.168.1.0/24
```

The process might take a minute or more depending on your internet connection. The output of this command is very important:

Please note down the following information from the output:

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

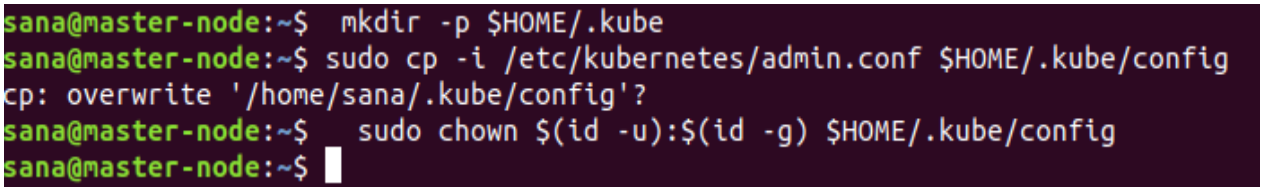
```
mkdir -p $HOME/.kube

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

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

You can now join any number of machines by running the following on each node

as root:

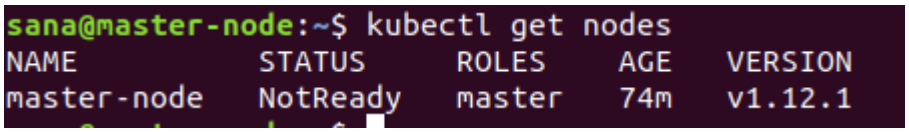
```
kubeadm join 192.168.100.6:6443 --token 06tl4c.oqn35jzecidg0r0m --discovery-
token-ca-cert-hash
sha256:c40f5fa0aba6ba311efcdb0e8cb637ae0eb8ce27b7a03d47be6d966142f2204c
```

Now run the commands suggested in the output in order to start using the cluster:



You can check the status of the master node by running the following command:

```
$ kubectl get nodes
```



You will see that the status of the master node is “not ready” yet. It is because no pod has yet been deployed on the master node and thus the Container Networking Interface is empty.

Step 4: Deploy a Pod Network through the master node

A pod network is a medium of communication between the nodes of a network. In this tutorial, we are deploying a Flannel pod network on our cluster through the following command:

```
$ sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

```
$ sudo kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

```
sana@master-node:~$ sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
```

Use the following command in order to view the status of the network:

```
$ kubectl get pods --all-namespaces
```

```
sana@master-node:~$ sudo kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   coredns-576cbf47c7-hb4qr              1/1     Running   1          6m2s
kube-system   coredns-576cbf47c7-ld6dg              1/1     Running   1          6m2s
kube-system   etcd-master-node                      1/1     Running   0          6m6s
kube-system   kube-apiserver-master-node            1/1     Running   0          6m3s
kube-system   kube-controller-manager-master-node   1/1     Running   0          5m53s
kube-system   kube-flannel-ds-amd64-ccjld           1/1     Running   0          4m4s
kube-system   kube-proxy-5vvrc                      1/1     Running   0          6m2s
kube-system   kube-scheduler-master-node            1/1     Running   0          5m56s
```

Now when you see the status of the nodes, you will see that the master-node is ready:

```
$ sudo kubectl get nodes
```

```
sana@master-node:~$ sudo kubectl get nodes
NAME          STATUS   ROLES    AGE     VERSION
master-node   Ready    master   8m13s   v1.12.1
```

Step 5: Add the slave node to the network in order to form a cluster

On the slave node, run the following command you generated while initializing Kubernetes on the master-node:

```
$ sudo kubeadm join 192.168.100.6:6443 --token 06tl4c.oqn35jzecidg0r0m --discovery-token-ca-cert-hash sha256:c40f5fa0aba6ba311efcdb0e8cb637ae0eb8ce27b7a03d47be6d966142f2204c
```

```
sana@slave-node:~$ sudo kubeadm join 192.168.100.6:6443 --token 06tl4c.oqn35jzec
idg0r0m --discovery-token-ca-cert-hash sha256:c40f5fa0aba6ba311efcdb0e8cb637ae0e
b8ce27b7a03d47be6d966142f2204c
[preflight] running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of
validated versions: 17.12.1-ce. Latest validated version: 18.06
[discovery] Trying to connect to API Server "192.168.100.6:6443"
[discovery] Created cluster-info discovery client, requesting info from "https:/
/192.168.100.6:6443"
```

Now when you run the following command on the master node, it will confirm that two nodes, the master node, and the server nodes are running on your system.

```
$ sudo kubectl get nodes
```

This shows that the two-node cluster is now up and running through the Kubernetes container management system.

In this document, we have explained the installation of the Kubernetes container management system on two Ubuntu nodes.

We have then formed a simple two-node cluster and deployed Kubernetes on it.

You can now deploy and use any service such as Nginx server or the Apache container to make use of this clustered network.