# **K8s installation on Ubuntu 18.04**

## **Os prerequisites**

1. IP address
2. Disk
3. Set hostfile
4. swapoff -a
5. Disable IPv6 - <https://www.thegeekdiary.com/how-to-disable-ipv6-on-ubuntu-18-04-bionic-beaver-linux/>
   1. Edit /etc/default/grub and append ipv6.disable=1 to GRUB\_CMDLINE\_LINUX and GRUB\_CMDLINE\_LINUX\_DEFAULT like the following sample:  
      FROM:

GRUB\_CMDLINE\_LINUX\_DEFAULT=""

GRUB\_CMDLINE\_LINUX=""

TO:

GRUB\_CMDLINE\_LINUX\_DEFAULT="ipv6.disable=1"

GRUB\_CMDLINE\_LINUX="ipv6.disable=1"

* 1. 2. Run the **update-grub** command to regenerate the grub.cfg file:

# update-grub

* 1. 3. Reboot the system to disable IPv6 support.

## **K8S Installation** - <https://loves.cloud/setting-up-a-kubernetes-cluster-on-ubuntu-18-04/>

**Step 1 :** All nodes

$ sudo su

# apt-get update

**Step 2 :** Install Docker - All nodes

# apt-get install -y docker.io

**Step 3:** Update cgroupo on docker- All nodes

<https://kubernetes.io/docs/setup/production-environment/container-runtimes/#docker>

cat <<EOF | sudo tee /etc/docker/daemon.json

{

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

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

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker

**Step 5:** Install kubeadm, Kubelet And Kubectl - All nodes

* kubeadm: the command to bootstrap the cluster.
* kubelet: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
* kubectl: the command line utility to communicate with your cluster.

# apt-get update && apt-get install -y apt-transport-https curl

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

# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

# apt-get update

# apt-get install -y kubelet kubeadm kubectl

# apt-mark hold kubelet kubeadm kubectl

**Step 6 :**SSH to the Master Node and run the below commands.

$ sudo su

# kubeadm init --apiserver-advertise-address=<internal-ip-address-of-master-node> --pod-network-cidr=192.168.0.0/16

In our cluster kubeadm init --pod-network-cidr=192.168.0.0/16

**–apiserver-advertise-address** is used to set the advertise address for this particular control-plane node’s API server.

By default, Calico uses 192.168.0.0/16 as the Pod network CIDR, and Calico to work correctly, we need to pass this same CIDR to the kubeadm init command above.

**Example**

# kubeadm init --apiserver-advertise-address=172.31.41.115 --pod-network-cidr=192.168.0.0/16

You will get the below output if the master initialization is successful :-

Text

Description automatically generated

Copy and save the kubeadm join –token output on a notepad as it would be required later as shown in the above image. This token will be used to add worker nodes to the Cluster.

ubeadm join 10.201.55.70:6443 --token qdb18z.lkas81xflggs7vrp \

--discovery-token-ca-cert-hash sha256:4c72a7b61350764b224be7d93a06a64f51a8a607dad966db3dd76e436c54dcba

**Step 7 :** To make kubectl work for your non-root user, run these commands.

# exit ; from sudo

$ 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 :** Let us now verify if kubectl is as working as expected, run the below command.

$ kubectl get pods -o wide --all-namespaces

You will notice that all the pods are in running state except two: coredns pods. To resolve this, we will install a Pod network add-on in a later stage.

You can install a Pod network add-on on the control-plane node or a node that has the kubeconfig credentials.

## **Adding Worker Nodes to the Cluster**

**Step 9 :** Now let us add worker nodes to our Cluster. SSH to the Worker Node-1 and run the join token command generated during master initialization in step-6.

$ sudo su



Text

Description automatically generated

Repeat step 9 on worker Node-2.

**Step 10 :** Check if the Node-1 and Node-2 is successfully added to the cluster, run the following command.

$ kubectl get nodes

The master as well as the worker nodes will be in the <not ready> Status.

Do not panic, this is just because we have not configured the CNI (Container network interface) yet.

## **Deploying the CNI**

For this tutorial, we will be installing Calico as our CNI.

Calico is an open source networking and network security solution for containers, virtual machines, and bare-metal workloads. Calico uses standard Linux networking tools to provide two major services for Cloud Native applications:

* Network connectivity between workloads.
* Network security policy enforcement between workloads.

**Step 11 :** Let us run the following command to deploy the CNI.

\*Verify version is the latest

$ kubectl apply -f https://docs.projectcalico.org/v3.19/manifests/calico.yaml

**Step 12 :** Now, let us check the status of the coredns Pods again in the kube-system namespace, run these commands.

$ kubectl get po -n kube-system

Text

Description automatically generated

**Step 13 :** Let us check if the state of our nodes (Master and worker) have changed to Ready.

As you can observe that after the installation of CNI the state of our nodes have changed to Ready.

$ kubectl get nodes

Calendar

Description automatically generated

## **Summary**

That’s all, folks! We have created a working single node Kubernetes cluster. In the next post we will show you how to deploy applications to the kubernetes clusters and use a service to expose your application.

The connection to the server 10.201.55.70:6443 was refused - did you specify the right host or port?

/var/lib/kubelet/config.yaml

# **OpenEBS installation** <https://docs.openebs.io/docs/next/uglocalpv-hostpath.html>

## **Os prerequisites**

1. Make sure you have the same storage path on all nodes

## **Installation**

1. Download the operation wget <https://openebs.github.io/charts/openebs-operator.yaml>
2. Vi openebs-operator.yaml
3. Navigate to the storage class section
   1. Update the storage path   
      - name: BasePath

value: "/data/openebs/"