Lab Manual – Kubernetes Multinode Cluster on Ubuntu

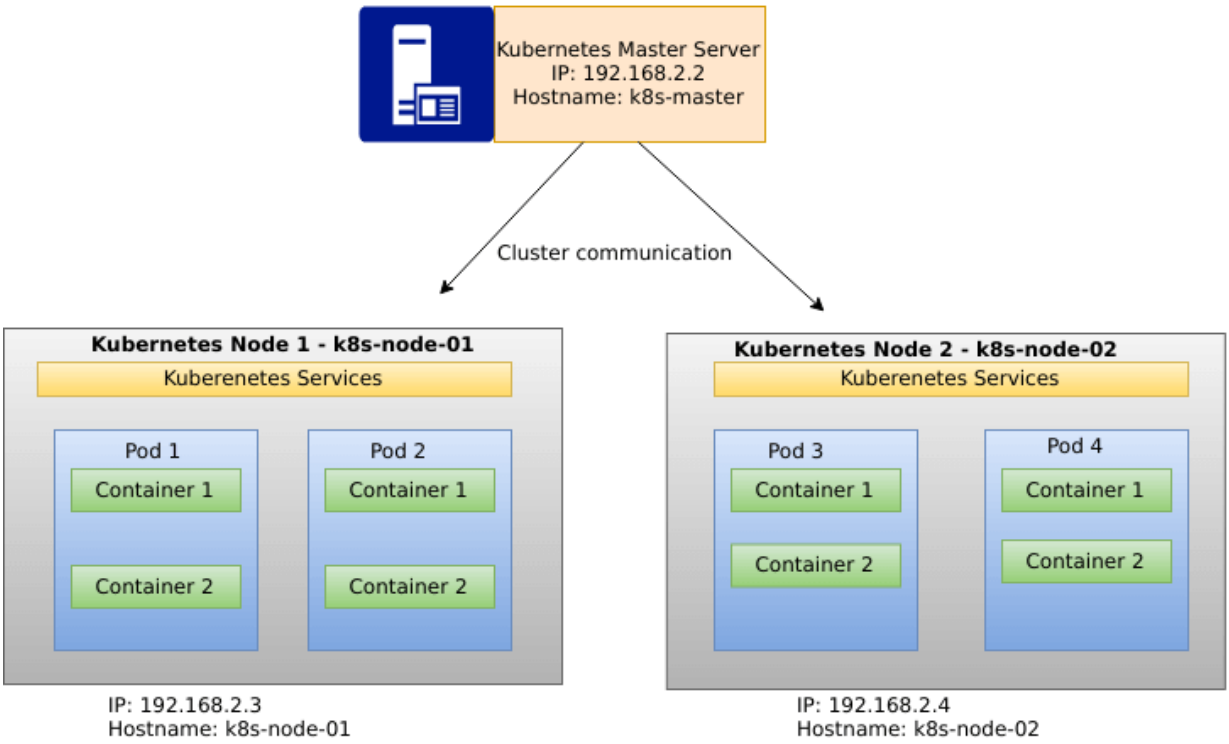
Steps:

1. Diagram
2. Configure system hostnames and the /etc/hosts file
3. Prerequisites (Run on all nodes)
4. Install Docker Engine (Run on all Nodes)
5. Install and Configure Kubernetes Master
6.

How to setup 3 node Kubernetes Cluster on Ubuntu 18.04

Step1: Diagram

This setup is based on the following diagram:



Name	Instance ID	Instance Type	Availability Zone	Instance State
Master	i-0466f9e6a26087ef7	t2.small	us-west-2c	running
Node-1	i-08085c5958da72323	t2.small	us-west-2c	running
Node-2	i-0c0b7be78a875ab35	t2.small	us-west-2c	running

1. **Step2: Configure system hostnames and the /etc/hosts file**

On Master Node:

Set hostname like below:

```
$ sudo hostnamectl set-hostname k8s-master
```

On Worker Node 01:

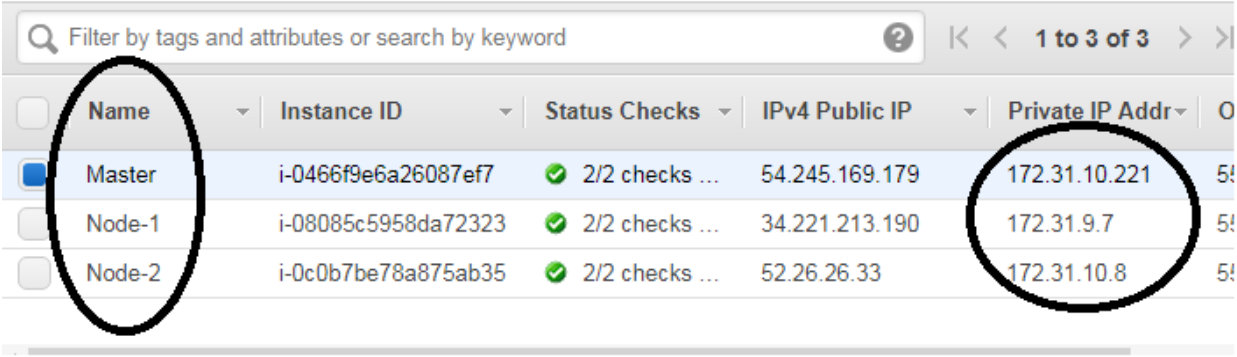
Set the hostname using `hostamectl` command line tool.

```
$ sudo hostnamectl set-hostname k8s-node-01
```

On Worker Node 02:

Also set hostname for Kubernetes worker node 02.

```
$ sudo hostnamectl set-hostname k8s-node-02
```



Once correct hostname has been configured on each host, update the /etc/hosts file with the same as below.

```
$ sudo vi /etc/hosts
```

Master Node

```
ubuntu@k8s-master:~$ sudo vi /etc/hosts

127.0.0.1 localhost

172.31.10.221 k8s-master
172.31.9.7 k8s-node-01
172.31.10.8 k8s-node-02
```

Make sure you use the Private IP's of the EC2 instance

And NOT the Public IP.

Node-01

```
ubuntu@k8s-node-01:~$ sudo vi /etc/hosts

127.0.0.1 localhost

172.31.10.221 k8s-master
172.31.9.7 k8s-node-01
172.31.10.8 k8s-node-02
```

Node-02

```
ubuntu@k8s-node-02:~$ vi /etc/hosts

127.0.0.1 localhost

172.31.10.221 k8s-master
172.31.9.7 k8s-node-01
172.31.10.8 k8s-node-02
```

Step3: Prerequisites (Run on all nodes)

Update system packages to the latest release on all nodes:

```
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install linux-image-extra-virtual

sudo reboot
```

Add user to manage Kubernetes cluster:

```
sudo useradd -s /bin/bash -m k8s-admin

sudo passwd k8s-admin

sudo usermod -aG sudo k8s-admin

echo "k8s-admin ALL=(ALL) NOPASSWD:ALL" | sudo tee
/etc/sudoers.d/k8s-admin
```

If you prefer entering sudo password when running sudo commands as `k8s-admin` user, then you can ignore the last line. You can test if no password prompt for sudo:

```
$ su - k8s-admin

k8s-admin@k8s-master:~$ sudo su -

root@k8s-master:~#
```

All looks good, let's proceed to install Docker engine.

Step4: Install Docker Engine

Ensure any old version of Docker engine is uninstalled on your system:

```
sudo apt-get remove docker docker-engine docker.i
```

Install dependencies:

```
$ sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
software-properties-common
```

Import Docker repository GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

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

Install docker:

```
sudo apt-get update

sudo apt-get install docker-ce -y

sudo usermod -aG docker k8s-admin
```

When docker has been installed, you can continue to configure the Kubernetes master node.

Step5: Install and Configure Kubernetes Master

we will add a repository for Ubuntu 16.04

$ sudo vi `/etc/apt/sources.list.d/kubernetes.list`

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

```
ubuntu@k8s-master:~$ vi /etc/apt/sources.list.d/kubernetes.list

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

Then import GPG key:

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

Update apt package index:

```
sudo apt update
```

Install Kubernetes Master Components

Install kubectl, `kubelet, kubernetes-cni and kubeadm`Kubernetes master components:

```
sudo apt install kubectl kubelet kubeadm kubernetes-cni
```

Confirm that all package binaries are present on the file system.

```
$ which kubelet

/usr/bin/kubelet

$ which kubeadm

/usr/bin/kubeadm
```

If swap is on, turn it off. (Run this on all the nodes)

```
sudo swapoff -a
```

Initialize Kubernetes Cluster

When all Kubernetes packages have been installed, you're ready to initialize the cluster using `kubeadm` command line tool.

Export required variables (**Optional)**

```
export API_ADDR=`ifconfig eth0 | grep 'inet'| cut -d':' -f2 | awk
'{print $2}'`
export DNS_DOMAIN="k8s.local"
export POD_NET="10.4.0.0/16"
export SRV_NET="10.5.0.0/16"
```

Then initialize the Kubernetes cluster using variables defined above:

```
kubeadm init --pod-network-cidr ${POD_NET} --service-cidr
${SRV_NET} \
--service-dns-domain "${DNS_DOMAIN}" --apiserver-advertise-
address ${API_ADDR}
```

Note: -- If you have selected a VM's less than 2 CPU, you might get an warning.

Then run the below

```
kubeadm init --pod-network-cidr ${POD_NET} --service-cidr
${SRV_NET} --service-dns-domain "${DNS_DOMAIN}" --apiserver-
advertise-address ${API_ADDR} --ignore-preflight-errors=all
```

If all goes well , here is the output

OUTPUT:

```
ubuntu@k8s-master:~$ sudo kubeadm init --pod-network-cidr ${POD_NET} --service-cidr ${SRV_NET
} --service-dns-domain "${DNS_DOMAIN}" --apiserver-advertise-address ${API_ADDR} --ignore-pre
flight-errors=all
[init] Using Kubernetes version: v1.14.2
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The
recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/
cri/
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connecti
on
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm
-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "etcd/ca" certificate and key
```

```
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/

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

kubeadm join 172.31.10.221:6443 --token cvnimi.vyisg97tg859xggx \
    --discovery-token-ca-cert-hash sha256:8776127cfb819f4c9056a6cc44fdb8d9805c2c6db844952c9e4
e56a2272ee5e2
ubuntu@k8s-master:~$
```

In the above the kubeadm join is visible, which shows the Cluster is enabled successfully on Master node.

Configure Access for `k8s-admin` user on the Master server

Switch to `k8s-admin`and copy Kubernetes configuration file with cluster information.

```
su - k8s-admin
mkdir -p $HOME/.k8s
sudo cp -i /etc/kubernetes/admin.conf $HOME/.k8s/config
sudo chown $(id -u):$(id -g) $HOME/.k8s/config
export KUBECONFIG=$HOME/.k8s/config
echo "export KUBECONFIG=$HOME/.k8s/config" | tee -a
~/.bashrc
```