Kubernetes Cluster on Ubuntu VMs

Installation Guide

Installation Guide

Note: For this installation we recommend a fresh ubuntu image since Kubernetes can take up a lot of resources.

Following are the preferable VM settings:

Master:

- 2GB RAM
- 2 Cores of CPU

Slave Node:

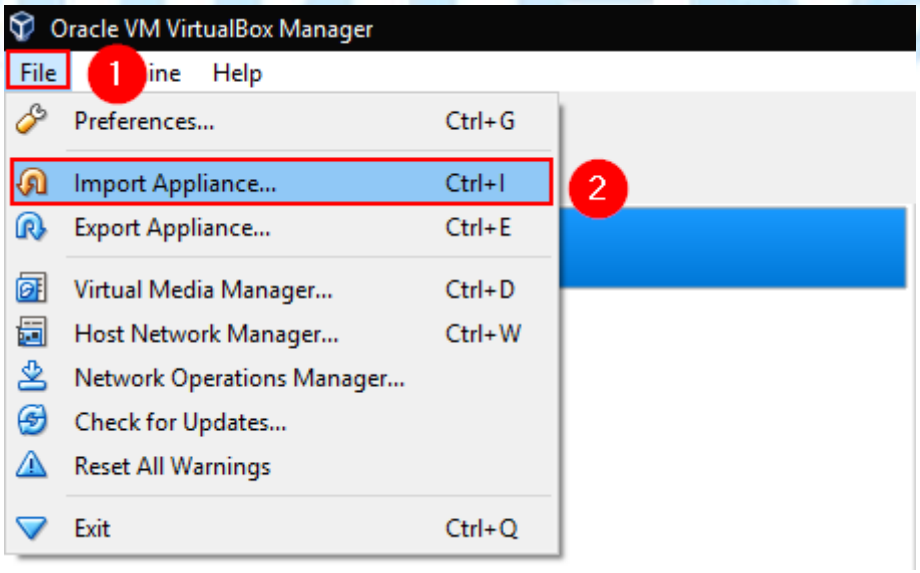
- 1 GB RAM
- 1 Core of CPU

Importing the Clean Ubuntu VM

Note: You will find the download link to this VM in the Pre-Installed VMs Guide

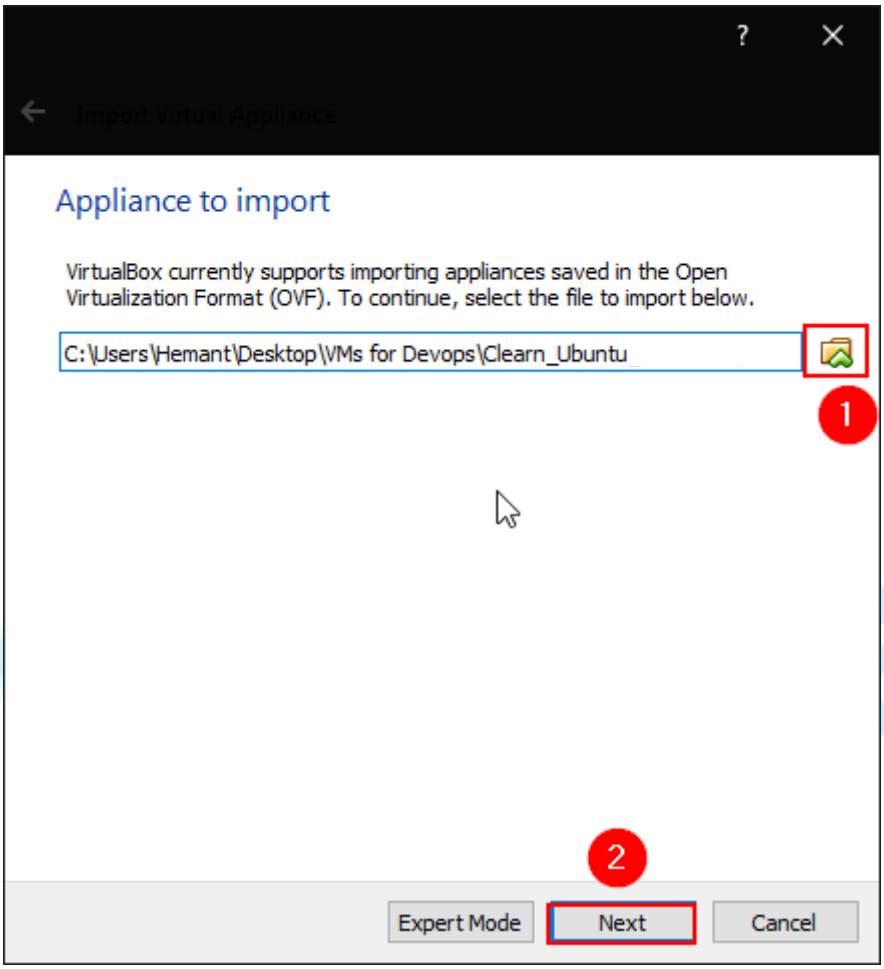
Step 1:

1. Open your Virtual Box Manager, and click on File.
2. Click on Import Appliance.



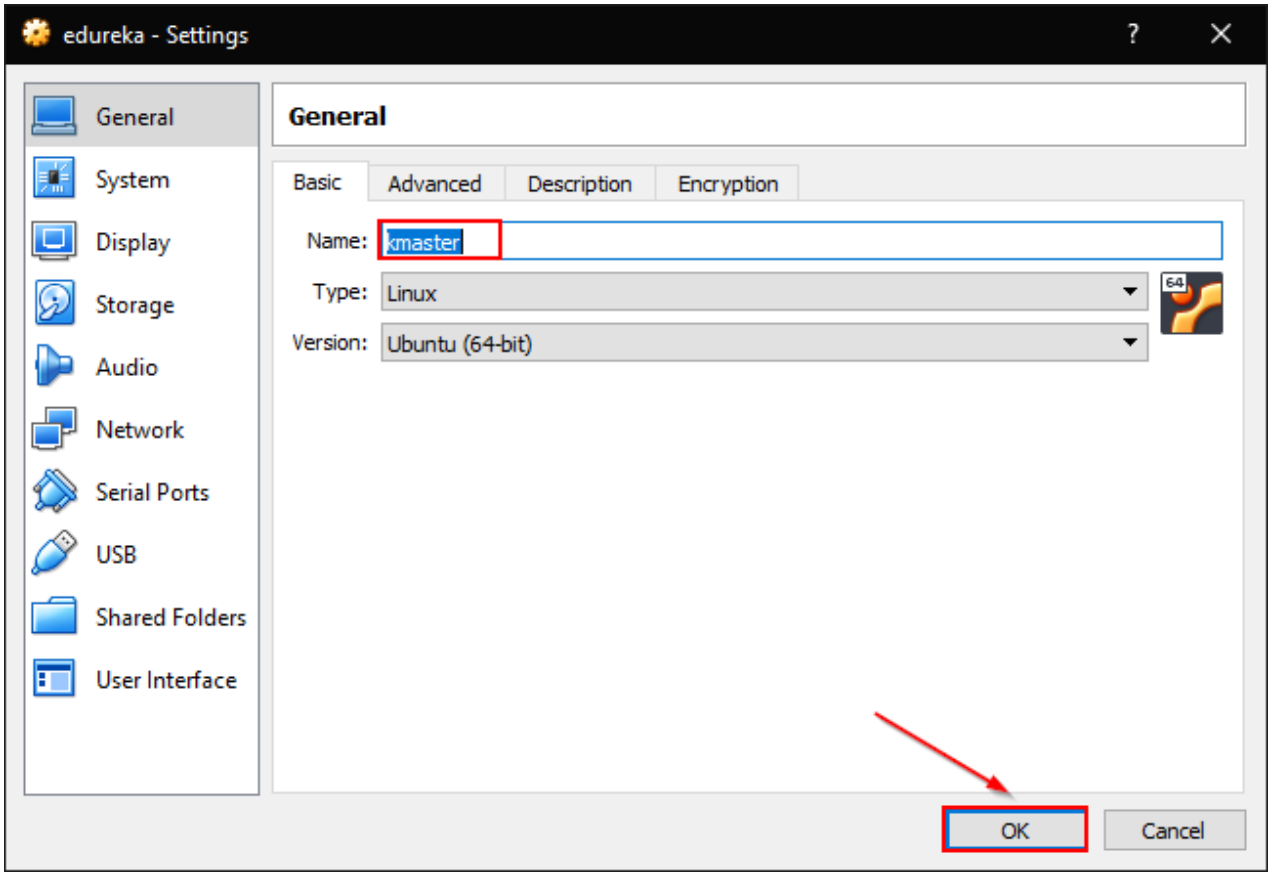
Step 2:

1. Click on Browse, and navigate to the place where you have downloaded the VM and select it.
2. Click on Next, and on the next page leave everything at default and Click on Import.



Step 3: Right Click on your VM, and click on Settings.

Step 4: Edit the name as kmaster, and click on OK



Step 5: Repeat the same steps to get a Slave Node, and name it as “knode”.

Steps for Master and Slave VMs

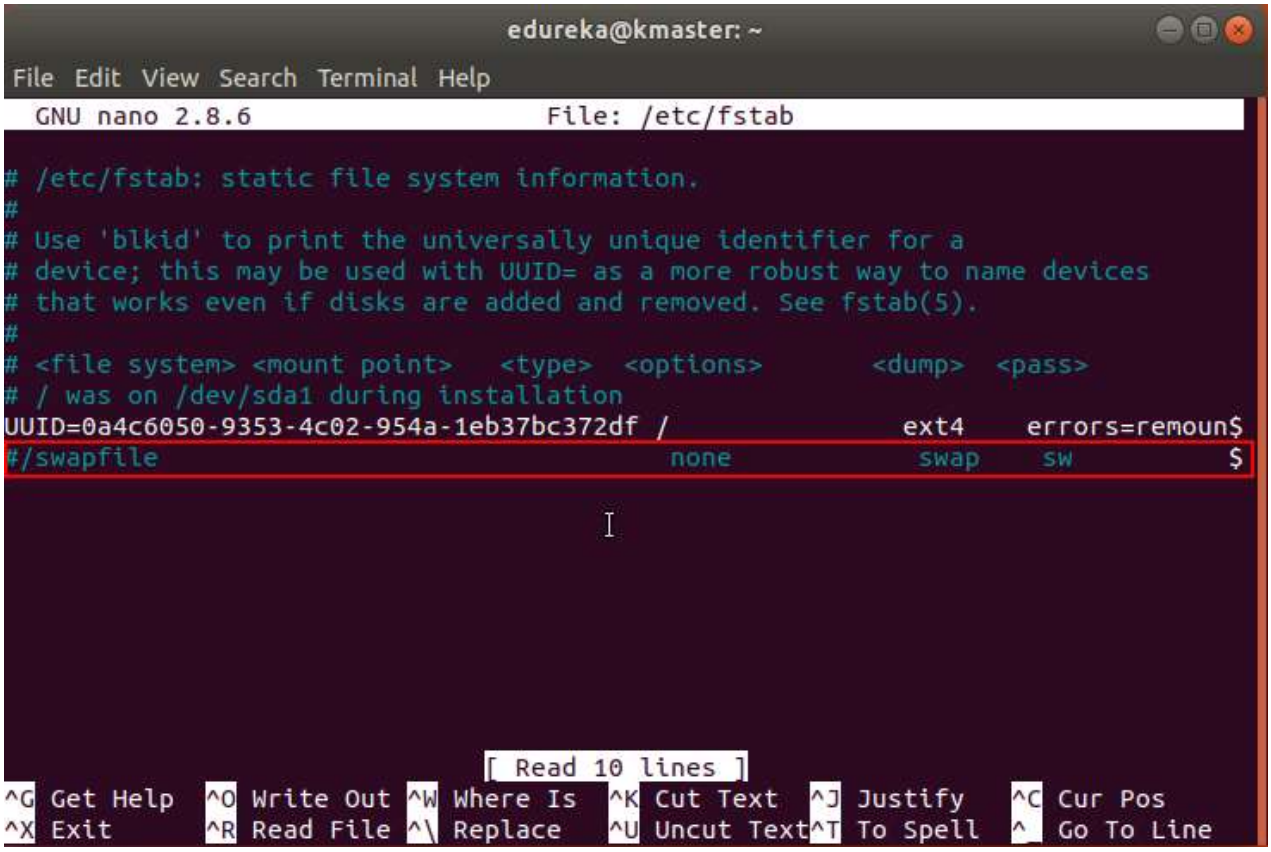
Note: These steps are common to both kmaster and knode VMs

Step 1:

1. Run the following commands:

```
sudo su
apt-get update
swapoff -a
nano /etc/fstab
```

2. Add a "#" in front of the last line, to comment it.



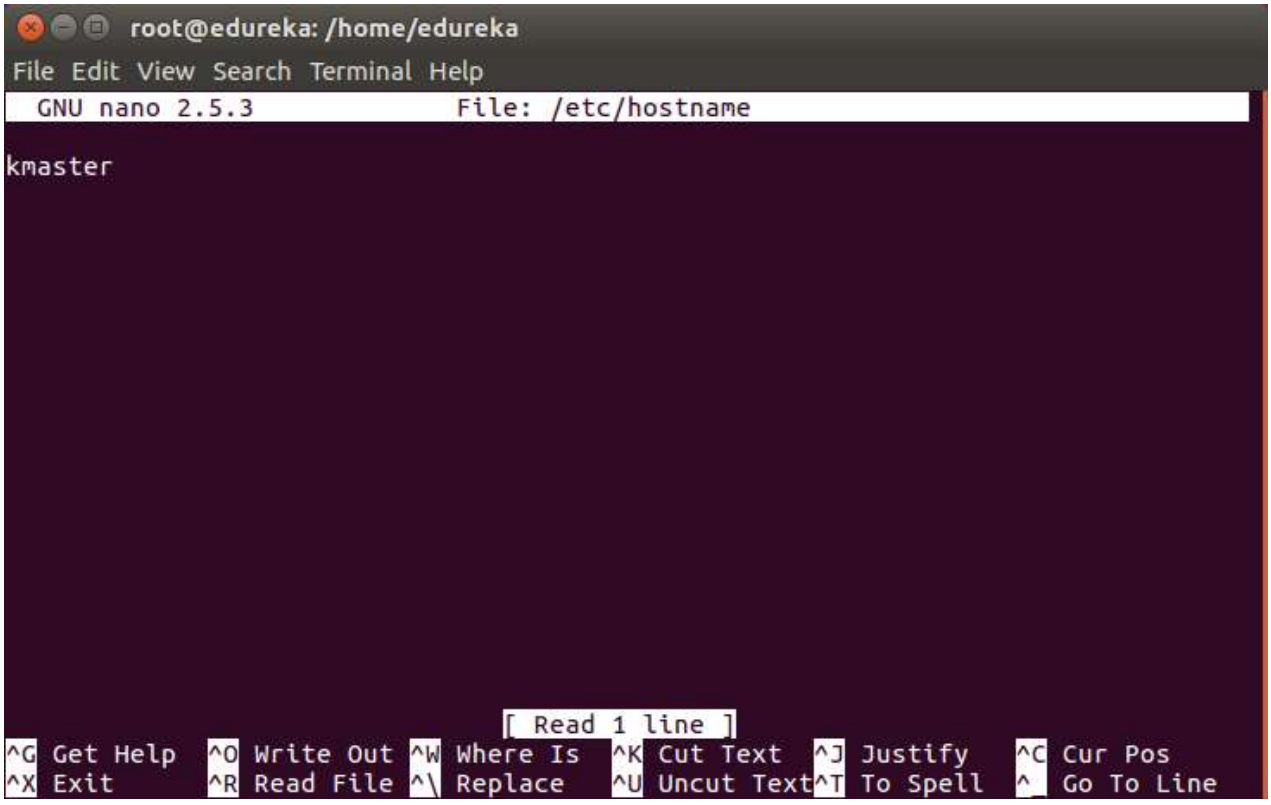
3. Press Ctrl+X, then press Y, and then press Enter to Save.

Step 2:

1. Run the following command:

```
nano /etc/hostname
```

2. Edit the name to “kmaster” for kmaster VM, and “knode” for knode VM.



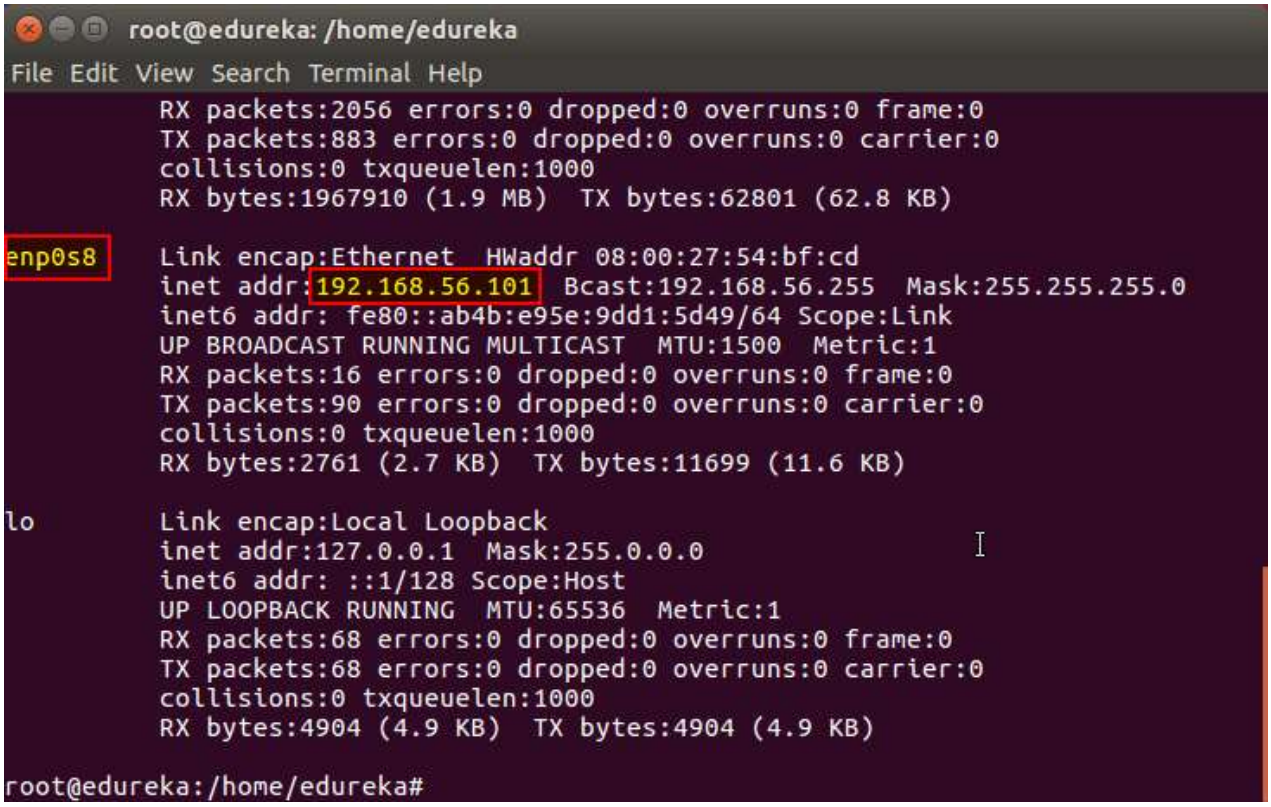
3. Press Ctrl+X, then press Y, and then press Enter to Save.

Step 3:

1. Run the following command:

```
ifconfig
```

2. Make a note of the IP address from the output of the above command. The IP address which has to be copied should be under "enp0s8", as shown in the screenshot below.



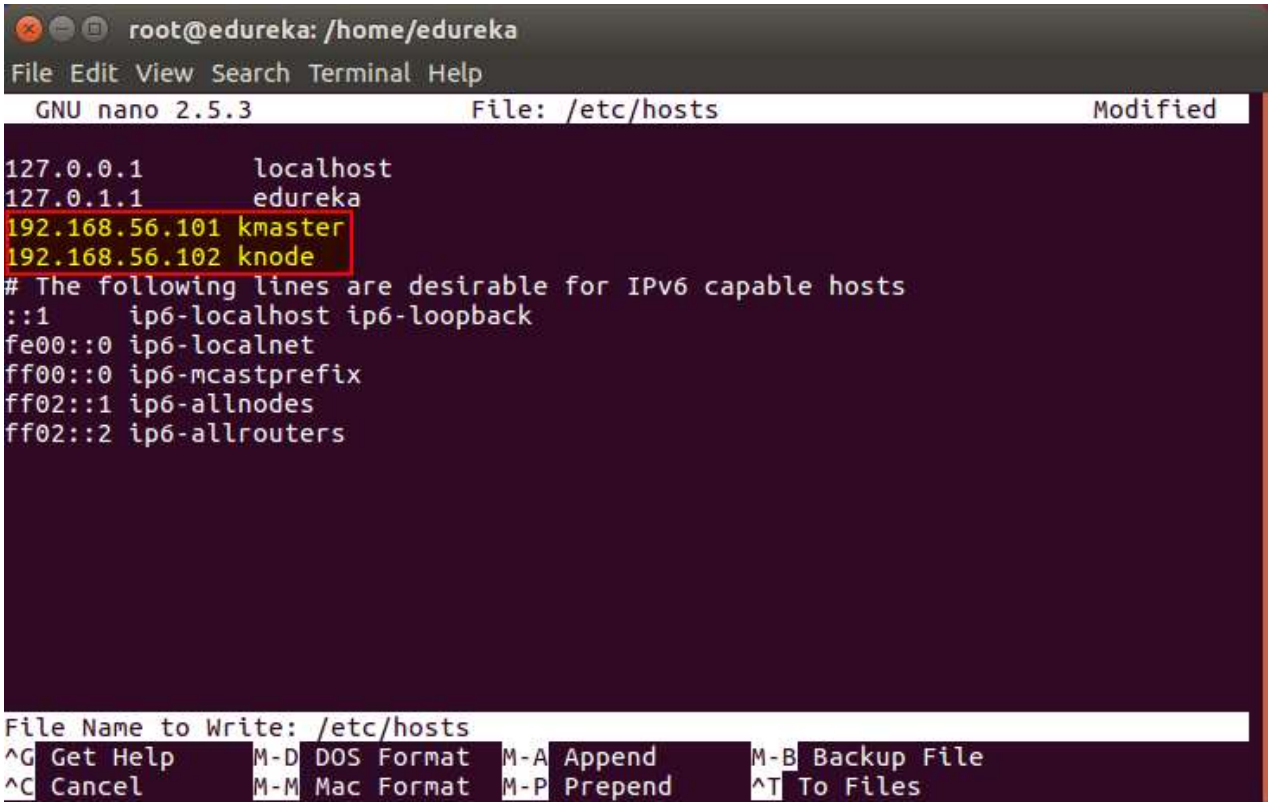
3. Make a note of the IP address of both the VMs using the above commands.

Step 4:

1. Run the following command:

```
nano /etc/hosts
```

2. Enter the IP address of the kmaster VM and the knode VM both in this file. (This has to be done in both the VMs). In front of the IP address of master write, “kmaster”. Similarly, in front of the Node IP address write “knode”.



3. Press Ctrl+X, then press Y, and then press Enter to Save.

Step 5: Next, we will make the IP addresses used above, static for the VMs.

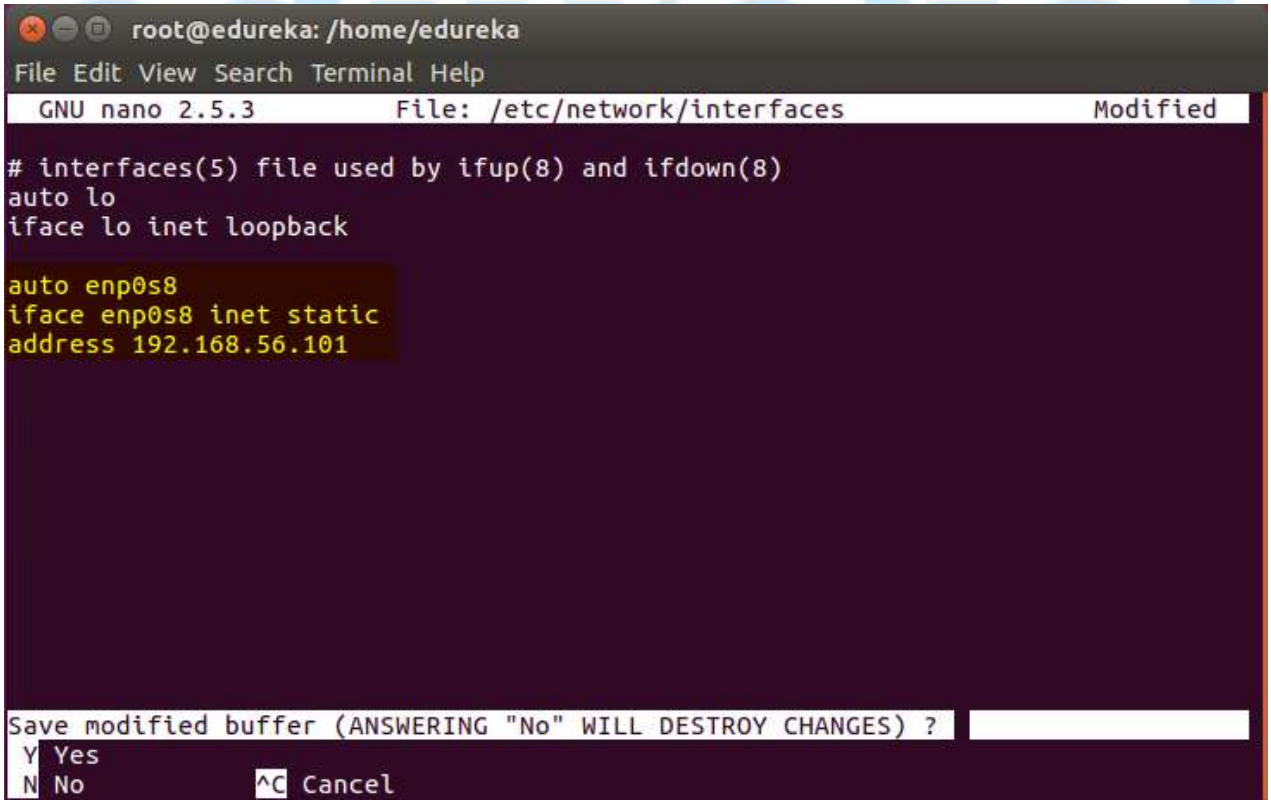
1. Run the following command:

```
nano /etc/network/interfaces
```

2. Enter the following in this document:

```
auto enp0s8
iface enp0s8 inet static
address <IP-Address-Of-VM>
```

3. It will look like the following:



4. Press Ctrl+X, then press Y, and then press Enter to Save.
5. After this, restart your machine.

Step 6: Now we will install openssh-server. Run the following command:

```
sudo apt-get install openssh-server
```

Step 7: Next, we will install Docker. Run the following commands:

```
sudo su
apt-get update
apt-get install -y docker.io
```

Step 8: Next, we will install kubeadm, kubelet and kubectl. Run the following commands:

```
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
```

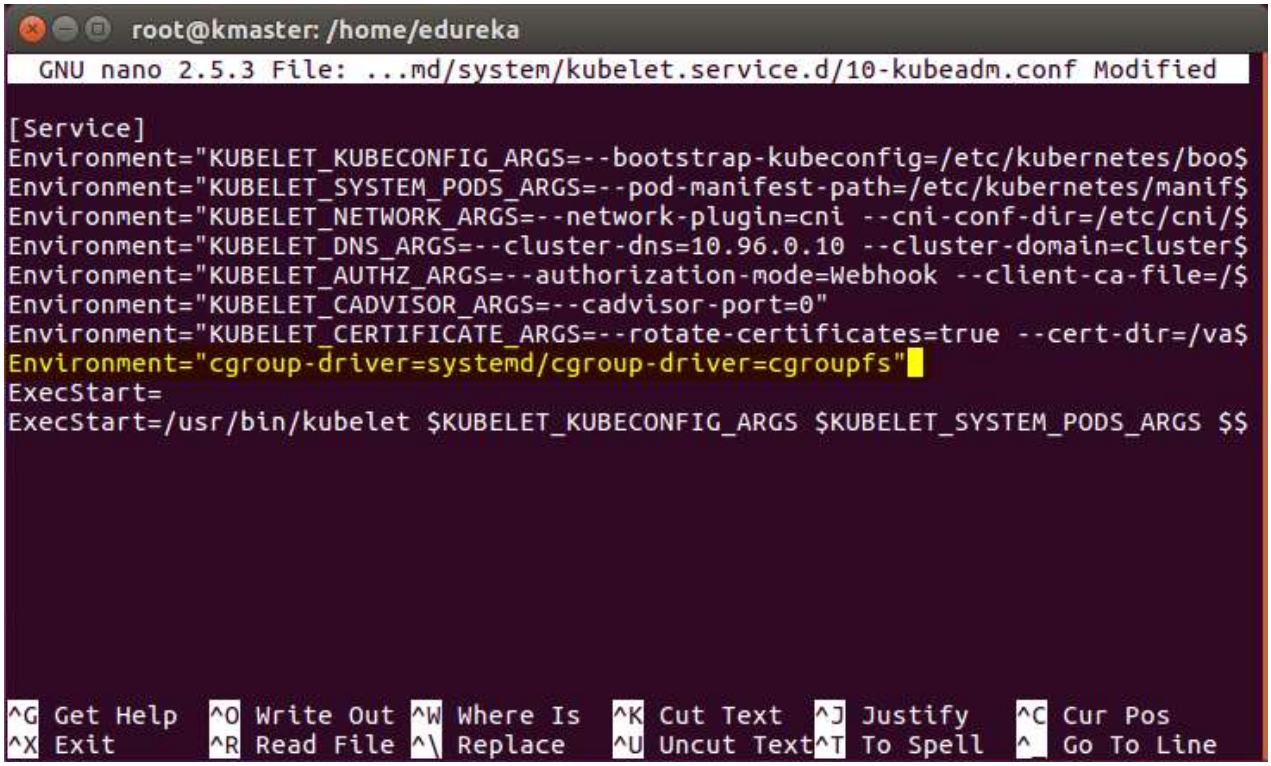
Step 9:

1. Next, we will change the configuration file of Kubernetes. Run the following command:

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

2. This will open a text editor, enter the following line after the last "Environment" Variable.

```
Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"
```



3. Press Ctrl+X, then press Y, and then press Enter to Save.

Step 10: Restart your VMs for the changes to take effect.

You have successfully installed Kubernetes on both the machines now!

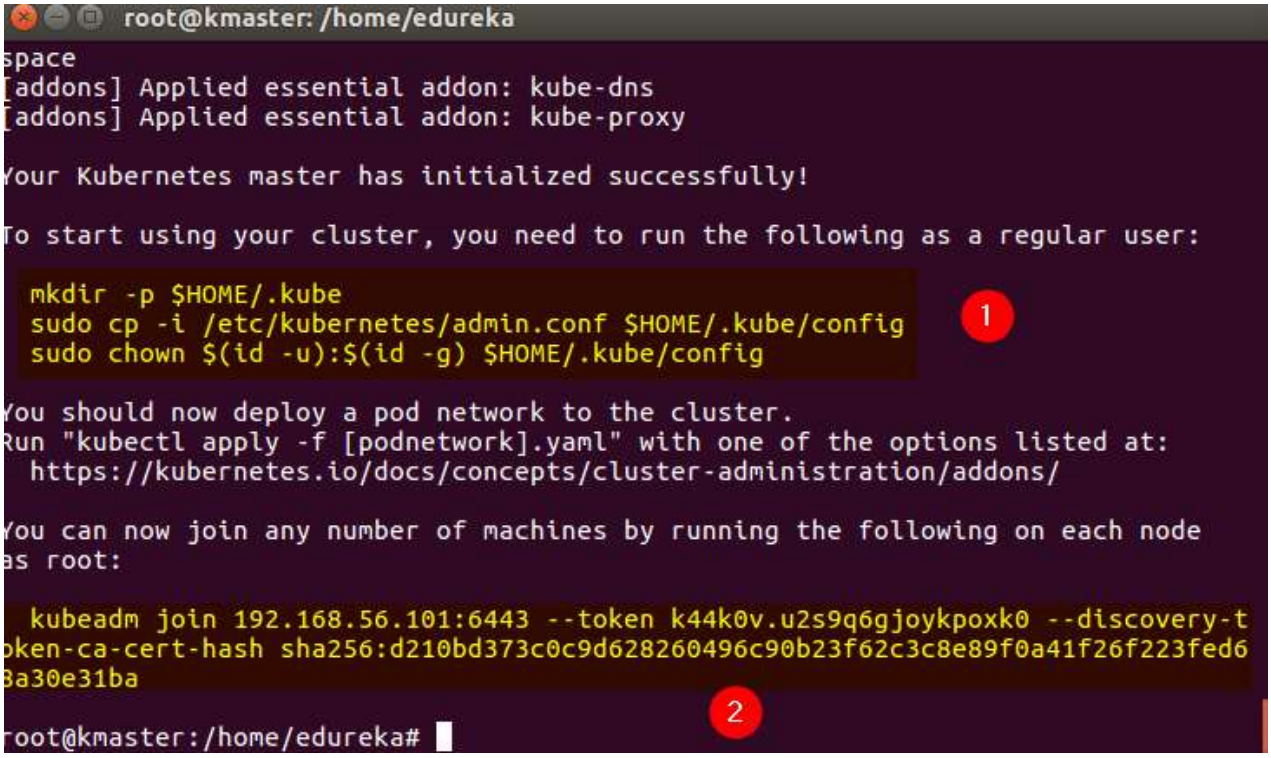
Steps for only Master VM

Note: These steps will only be executed on the master node (kmaster VM).

Step 1: We will now Initialize our Master VM. For that execute the following command:

```
kubeadm init --apiserver-advertise-address=<ip-address-of-kmaster-vm> --pod-network-cidr=192.168.0.0/16
```

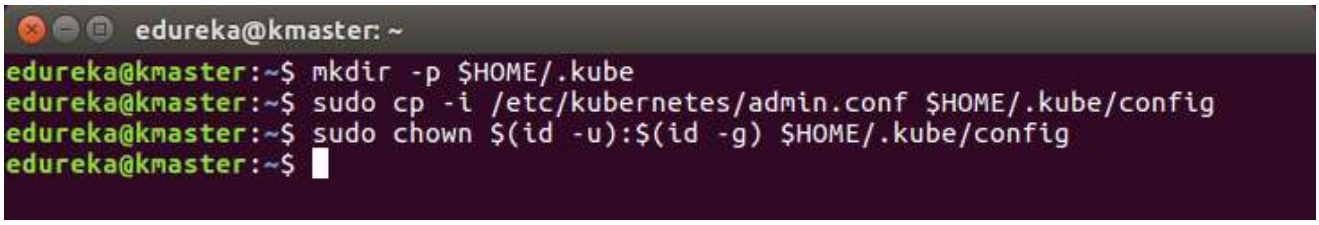
1. You will get the below output. The commands marked as (1), execute them as a non-root user. This will enable you to use kubectl from the CLI
2. The command marked as (2) should also be saved for future. This will be used to join nodes to your cluster.



Step 2:

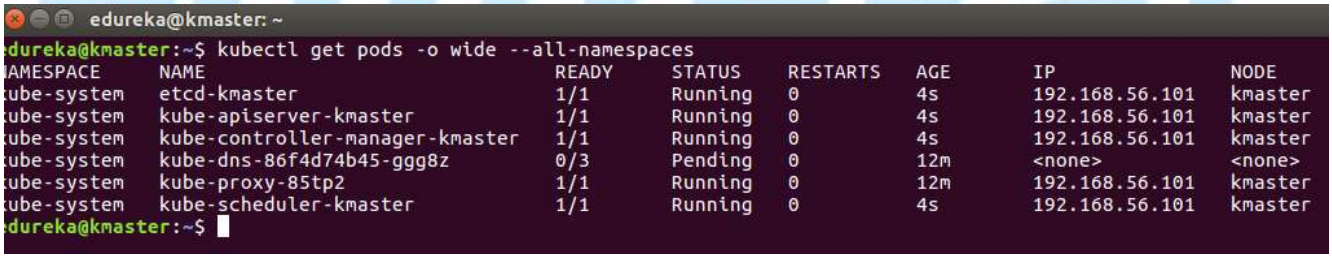
1. Like mentioned before, run the commands from the above output as a non-root user.

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



2. To verify, if kubectl is working or not, run the following command:

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



Step 3:

1. You will notice from the previous command, all the pods are running except one, kube-dns. For resolving this we will install a pod network. To install the pod network, run the following command:

```
kubectl apply -f
https://docs.projectcalico.org/v3.11/manifests/calico.yaml
```

2. After some time, you will notice that all pods shift to the running state.

```
edureka@kmaster: ~
edureka@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE     IP               NODE
kube-system   calico-etcd-b46dk                          1/1     Running   0          2m      192.168.56.101   kmaster
kube-system   calico-kube-controllers-5d74847676-lrhvc   1/1     Running   0          2m      192.168.56.101   kmaster
kube-system   calico-node-n9v8k                          2/2     Running   0          2m      192.168.56.101   kmaster
kube-system   etcd-kmaster                               1/1     Running   0          1m      192.168.56.101   kmaster
kube-system   kube-apiserver-kmaster                     1/1     Running   0          1m      192.168.56.101   kmaster
kube-system   kube-controller-manager-kmaster            1/1     Running   0          1m      192.168.56.101   kmaster
kube-system   kube-dns-86f4d74b45-ggg8z                  3/3     Running   0          23m     192.168.189.1    kmaster
kube-system   kube-proxy-85tp2                           1/1     Running   0          23m     192.168.56.101   kmaster
kube-system   kube-scheduler-kmaster                     1/1     Running   0          1m      192.168.56.101   kmaster
```

Troubleshooting:

1) If the internet stops working after installing the pod network or you get an image pull back error in pods, edit the **resolv.conf** file inside the **/etc** directory and change the nameserver to **8.8.8.8**

Syntax: sudo vi /etc/reslov.conf

```
edureka@kmaster:/etc$ sudo vi resolv.conf
```

```
# This file is managed by man:sy
#
# 127.0.0.53 is the systemd-resc
# run "systemd-resolve --status"
nameserver 8.8.8.8
```

The internet should work fine now.

2) In case calico doesn't start, use flannel instead as your pod network. To do that you'll need to restart the server with the flannel pod-network-cidr

```
sudo kubeadm reset

sudo kubeadm init -apiserver-advertise-address=<kmaster-IP>
--pod-network-cidr=10.244.0.0/16
```

And use the following command to start the flannel pod network

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/2140ac876ef
134e0ed5af15c65e414cf26827915/Documentation/kube-flannel.yml
```

Step 4: Next, we will install the dashboard. To install the Dashboard, run the following command:

```
kubectl create -f
https://raw.githubusercontent.com/kubernetes/dashboard/mas
ter/src/deploy/recommended/kubernetes-dashboard.yaml
```

```
edureka@kmaster:~$ kubectl create -f https://raw.githubusercontent.com/kuberne
-dashboard.yaml
secret "kubernetes-dashboard-certs" created
serviceaccount "kubernetes-dashboard" created
role.rbac.authorization.k8s.io "kubernetes-dashboard-minimal" created
rolebinding.rbac.authorization.k8s.io "kubernetes-dashboard-minimal" created
deployment.apps "kubernetes-dashboard" created
service "kubernetes-dashboard" created
```

Step 5: Your dashboard is now ready with it's the pod in the running state.

```
kube-system   etcd-kmaster                          1/1     Running   0
kube-system   kube-apiserver-kmaster                1/1     Running   0
kube-system   kube-controller-manager-kmaster       1/1     Running   0
kube-system   kube-dns-86f4d74b45-ggg8z             3/3     Running   0
kube-system   kube-proxy-85tp2                      1/1     Running   0
kube-system   kube-scheduler-kmaster                1/1     Running   0
kube-system   kubernetes-dashboard-7d5dcdb6d9-bbmmr 1/1     Running   0
```

Step 6:

1. By default dashboard will not be visible on the Master VM. Run the following command in the command line:

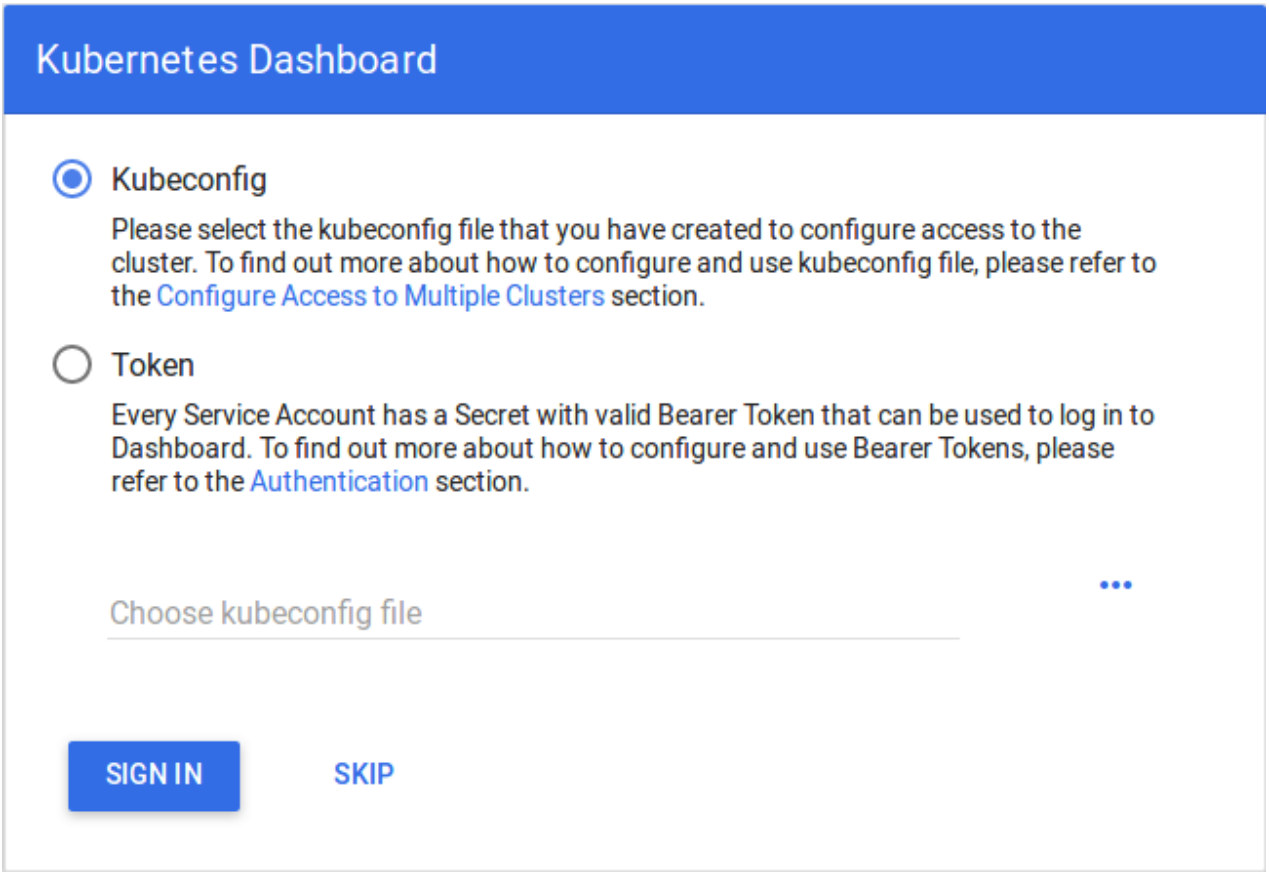
```
kubectl proxy
```

```
edureka@kmaster:~$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

2. To view the dashboard in the browser, navigate to the following address in the browser of your Master VM.

```
http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
```

3. You will prompted with this page, to enter the credentials.



Step 7: In this step, we will create the service account for the dashboard and get it's credentials. Run the following commands:

Note: Run all these commands in a new terminal, or your kubectl proxy command will stop.

1. This command will create service account for dashboard in the default namespace.

```
kubectl create serviceaccount dashboard -n default
```

2. This command will add the cluster binding rules to your dashboard account

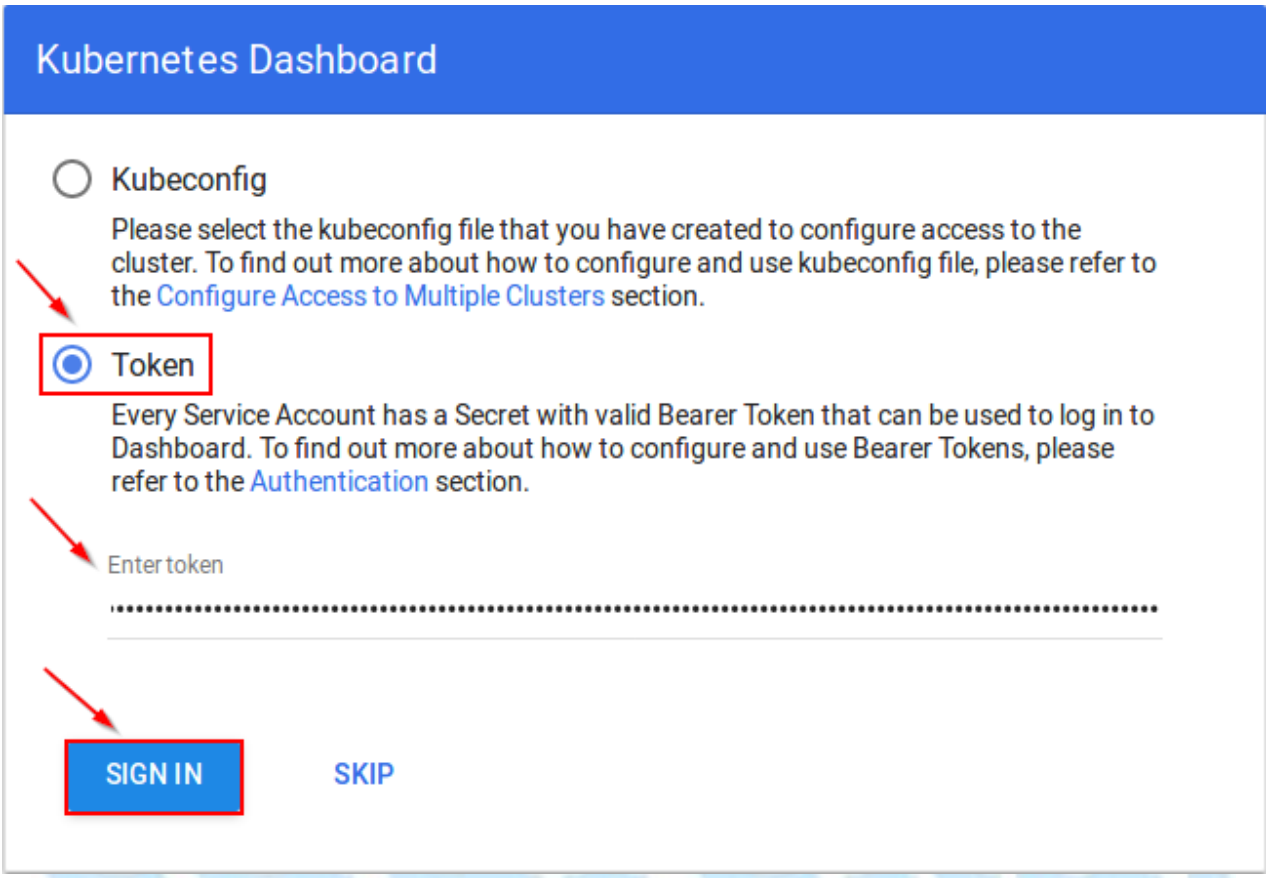
```
kubectl create clusterrolebinding dashboard-admin -n
default \
  --clusterrole=cluster-admin \
  --serviceaccount=default:dashboard
```

3. This command will give you the token required for your dashboard login.

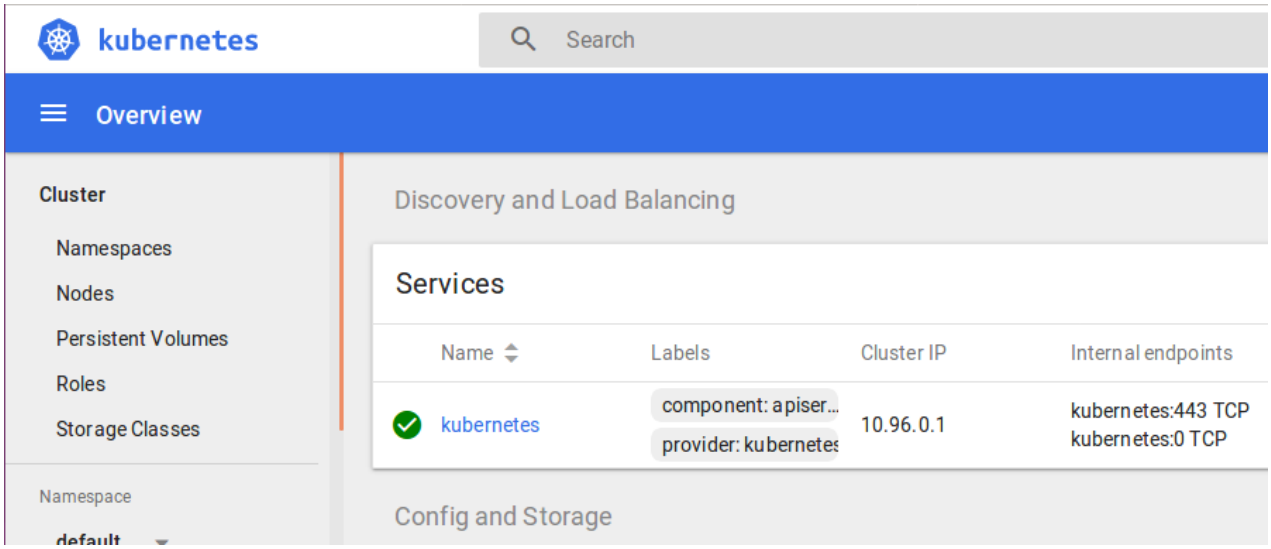
```
kubectl get secret $(kubectl get serviceaccount dashboard -
o jsonpath="{.secrets[0].name}") -o
jsonpath="{.data.token}" | base64 --decode
```

```
edureka@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard -o
jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiw
ia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXR
lcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRhc2hib2FyZC10b2tlbi1iOTRocSIsImt
1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkYXNoYm9hcmQ
iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJhYWY1Yzl
iMS01YWE0LTExZTgtOGY3YS0wODAwMjdmODlkZWQiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ
6ZGVmYXVsdDpkYXNoYm9hcmQifQ.wKPklOjENDmJ4l74LhQNCHIQ2Gs2jUlo0vYdk4pkU4vN8iB54x7I
9BqOYUIujW_zEZqjnWyQdjnDu2DAMtXwC_5uILo4SaTTl_bVaRVrbOoVCxxElaUyHQfppzEL8-EJNXXG
UuIqzvzYr8zkYRtAqTIcjb3tXBlIcRg5Ru-moN7IdPxXwaeRWjdJWiH96h_VRmO5myiCoX_gTBHztWQ0
0sdgOWuFf2fTodCO-e516vxBzNOThKdzGKBE2m7FenwXcCLTkZwHUhUK6yZuJq_vDpON1P7ARqQYnwXj
h6eHzKgJ9b8rf41D6m6DmlSOvgd0SCPfwjkZ_ppv_tl-XVPdTQedureka@kmaster:~$
```

4. Copy this token and paste it in Dashboard Login Page, by selecting token option.



5. You have successfully logged in your dashboard!



Steps for only Node VM

Step 1: It is time to join your node to the cluster! This is probably the only step that you will be doing on the node, after installing kubernetes on it. Run the join command that you saved, when you ran kubeadm init command on the master.
Note: Run this command with "sudo".

```
sudo kubeadm join --apiserver-advertise-address=<ip-address-of-the master> --pod-network-cidr=192.168.0.0/16
```

```
edureka@knode:~$ sudo kubeadm join 192.168.56.101:6443 --token n6qrh0.opyhe2c655
ay3j04 --discovery-token-ca-cert-hash sha256:84dd965586c1b2d82b345706382ec43bc62
aa8e460b54dfc02b367f85f218b84
[sudo] password for edureka:
[preflight] Running pre-flight checks.
        [WARNING Service-Docker]: docker service is not enabled, please run 'sys
temctl enable docker.service'
        [WARNING FileExisting-crictl]: crictl not found in system path
Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
[discovery] Trying to connect to API Server "192.168.56.101:6443"
[discovery] Created cluster-info discovery client, requesting info from "https:/
/192.168.56.101:6443"
[discovery] Requesting info from "https://192.168.56.101:6443" again to validate
 TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate va
lidates against pinned roots, will use API Server "192.168.56.101:6443"
[discovery] Successfully established connection with API Server "192.168.56.101:
6443"

This node has joined the cluster:
```

Your Kubernetes Cluster is ready! :-)