Steps:

1. Pre-requisites to install Kubernetes
2. Setting up Kubernetes environment
3. Installing Kubeadm, Kubelet, Kubectl
4. Starting the Kubernetes cluster from master
5. Getting the nodes to join the cluster

Pre-requisites To Install Kubernetes

Master:

- 2 GB RAM
- 2 Cores of CPU

Slave/ Node:

- 1 GB RAM
- 1 Core of CPU

Pre-Installation Steps On Both Master & Slave (To Install Kubernetes)

The following steps have to be executed on both the master and node machines. Let's call the the master as '*kmaster*' and node as '*knode*'.

First, login as 'sudo' user because the following set of commands need to be executed with 'sudo' permissions. Then, update your 'apt-get' repository.

```
$ sudo su
# apt-get update
```

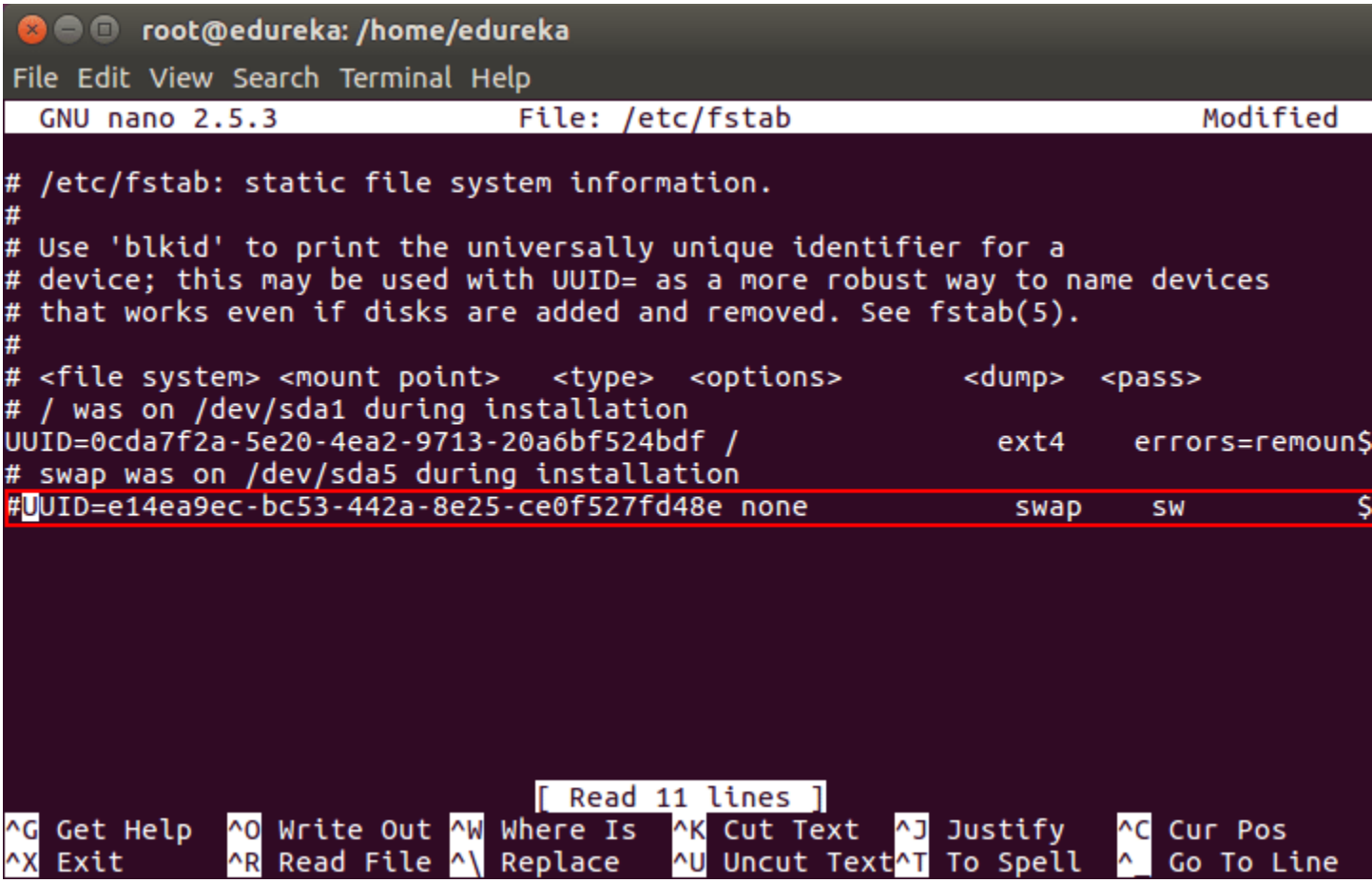
Note: After logging-in as 'sudo' user, note that your shell symbol will change to '#' from '$'.

Turn Off Swap Space

Next, we have to turn off the swap space because Kubernetes will start throwing random errors otherwise. After that you need to open the 'fstab' file and comment out the line which has mention of swap partition.

```
# swapoff -a
```

```
# nano /etc/fstab
```



Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

Update The Hostnames

To change the hostname of both machines, run the below command to open the file and subsequently rename the master machine to ‘kmaster’ and your node machine to ‘knode’.

```
# nano /etc/hostname
```

```
root@edureka: /home/edureka
File  Edit  View  Search  Terminal  Help
  GNU nano 2.5.3            File: /etc/hostname

kmaster

                              [ Read 1 line ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^T To Spell   ^_ Go To Line
```

Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

Update The Hosts File with IPs of Master & Node

Run the following command on both machines to note the IP addresses of each.

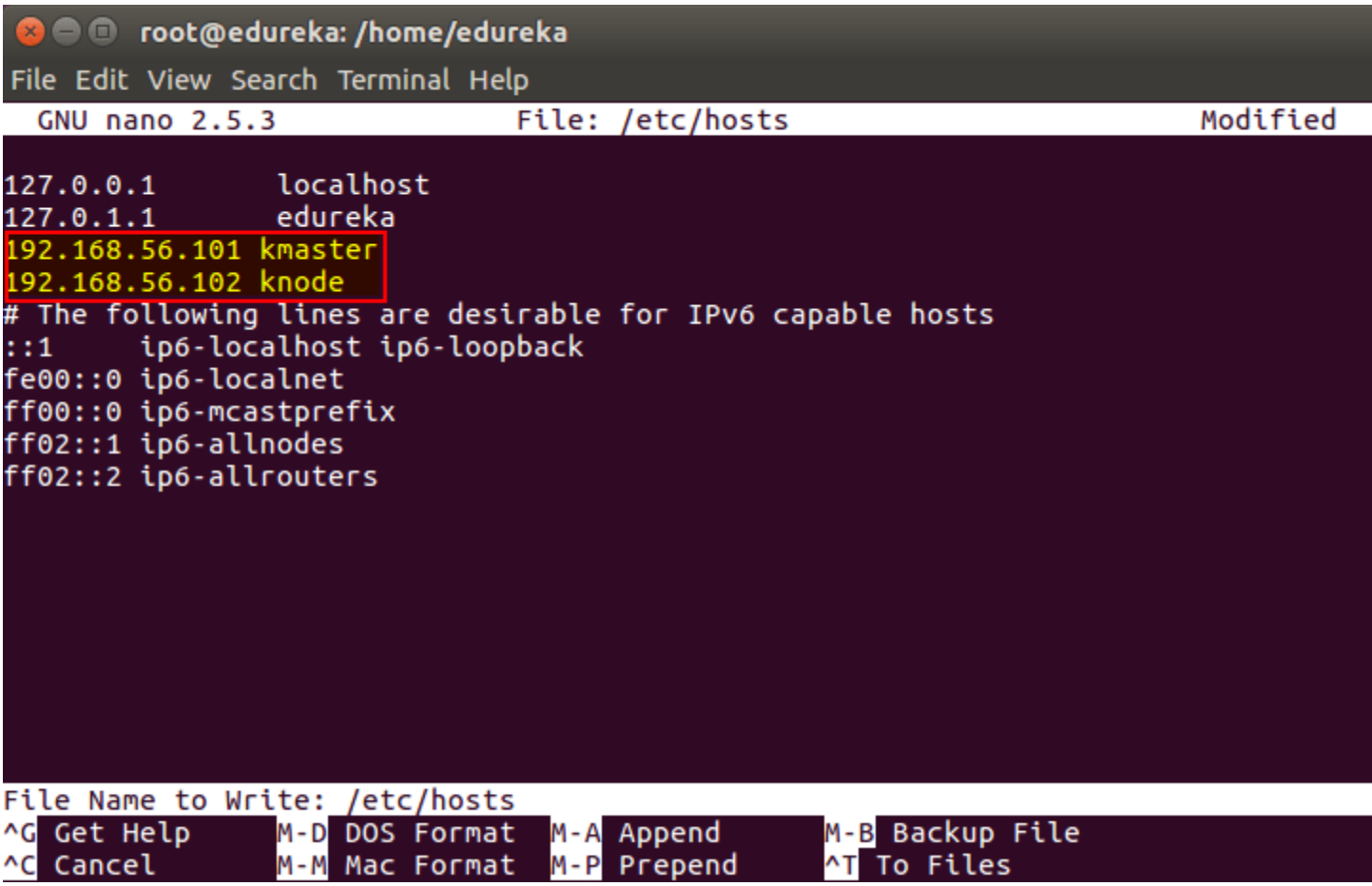
```
# ifconfig
```

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.

```
root@edureka: /home/edureka
File  Edit  View  Search  Terminal  Help
          RX packets:2056 errors:0 dropped:0 overruns:0 frame:0
          TX packets:883 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1967910 (1.9 MB)  TX bytes:62801 (62.8 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:54:bf:cd
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::ab4b:e95e:9dd1:5d49/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:90 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2761 (2.7 KB)  TX bytes:11699 (11.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4904 (4.9 KB)  TX bytes:4904 (4.9 KB)

root@edureka:/home/edureka#
```

Now go to the ‘hosts’ file on both the master and node and add an entry specifying their respective IP addresses along with their names ‘kmaster’ and ‘knode’. This is used for referencing them in the cluster. It should look like the below screenshot on both the machines.

```
# nano /etc/hosts
```



Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

Setting Static IP Addresses

Next, we will make the IP addresses used above, static for the VMs. We can do that by modifying the network interfaces file. Run the following command to open the file:

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

Now enter the following lines in the file.

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

It will look something like the below screenshot.

```
root@edureka: /home/edureka
File  Edit  View  Search  Terminal  Help
  GNU nano 2.5.3          File: /etc/network/interfaces                Modified

# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s8
iface enp0s8 inet static
address 192.168.56.101

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

After this, restart your machine(s).

Install OpenSSH-Server

Now we have to install openshh-server. Run the following command:

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

Install Docker

Now we have to install Docker because Docker images will be used for managing the containers in the cluster. Run the following commands:

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

Next we have to install these 3 essential components for setting up Kubernetes environment: kubeadm, kubectl, and kubelet.

Run the following commands before installing the Kubernetes environment.

```
# 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
```

Install kubeadm, Kubelet And Kubectl

Now its time to install the 3 essential components. ***Kubelet*** is the lowest level component in Kubernetes. It's responsible for what's running on an individual machine. ***Kuebadm*** is used for administrating the Kubernetes cluster. ***Kubectl*** is used for controlling the configurations on various nodes inside the cluster.

```
# apt-get install -y kubelet kubeadm kubectl
```

Updating Kubernetes Configuration

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

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

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

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

```
root@kmaster: /home/edureka
  GNU nano 2.5.3 File: ...md/system/kubelet.service.d/10-kubeadm.conf Modified

[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/boo$
Environment="KUBELET_SYSTEM_PODS_ARGS=--pod-manifest-path=/etc/kubernetes/manif$
Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/$
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster$
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook --client-ca-file=/$
Environment="KUBELET_CADVISOR_ARGS=--cadvisor-port=0"
Environment="KUBELET_CERTIFICATE_ARGS=--rotate-certificates=true --cert-dir=/va$
Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_SYSTEM_PODS_ARGS $$

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

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

Voila! You have successfully installed Kubernetes on both the machines now!

As of now, only the Kubernetes environment has been setup. But now, it is time to install Kubernetes completely, by moving onto the next 2 phases, where we will individually set the configurations in both machines.

Steps Only For Kubernetes Master VM (kmaster)

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

Step 1: We will now start our Kubernetes cluster from the master's machine. Run 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

```
root@kmaster: /home/edureka
space
[addons] Applied essential addon: kube-dns
[addons] Applied essential addon: kube-proxy

Your Kubernetes master 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/

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

  kubeadm join 192.168.56.101:6443 --token k44k0v.u2s9q6gjoykpoxk0 --discovery-t
oken-ca-cert-hash sha256:d210bd373c0c9d628260496c90b23f62c3c8e89f0a41f26f223fed6
3a30e31ba

root@kmaster:/home/edureka#
```

Step 2: As 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
```

It should look like this:

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

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

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

```
edureka@kmaster: ~
edureka@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                READY     STATUS    RESTARTS   AG
kube-system   etcd-kmaster                        1/1       Running   0          4s
kube-system   kube-apiserver-kmaster              1/1       Running   0          4s
kube-system   kube-controller-manager-kmaster     1/1       Running   0          4s
kube-system   kube-dns-86f4d74b45-ggg8z           0/3       Pending   0          12
kube-system   kube-proxy-85tp2                    1/1       Running   0          12
kube-system   kube-scheduler-kmaster              1/1       Running   0          4s
edureka@kmaster:~$
```

Next

Step 3: You will notice from the previous command, that all the pods are running except one: ‘kube-dns’. For resolving this we will install a pod network. To install the CALICO pod network, run the following command:

```
$ kubectl apply -f https://docs.projectcalico.org/v3.0/getting-started/kubernetes/installation/hosted/kubeadm/1.7/calico.yaml
```

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    RES
kube-system   calico-etcd-b46dk                          1/1       Running   0
kube-system   calico-kube-controllers-5d74847676-lrhvc   1/1       Running   0
kube-system   calico-node-n9v8k                          2/2       Running   0
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
```

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/master/src/depl
oy/recommended/kubernetes-dashboard.yaml
```

It will look something like this:

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

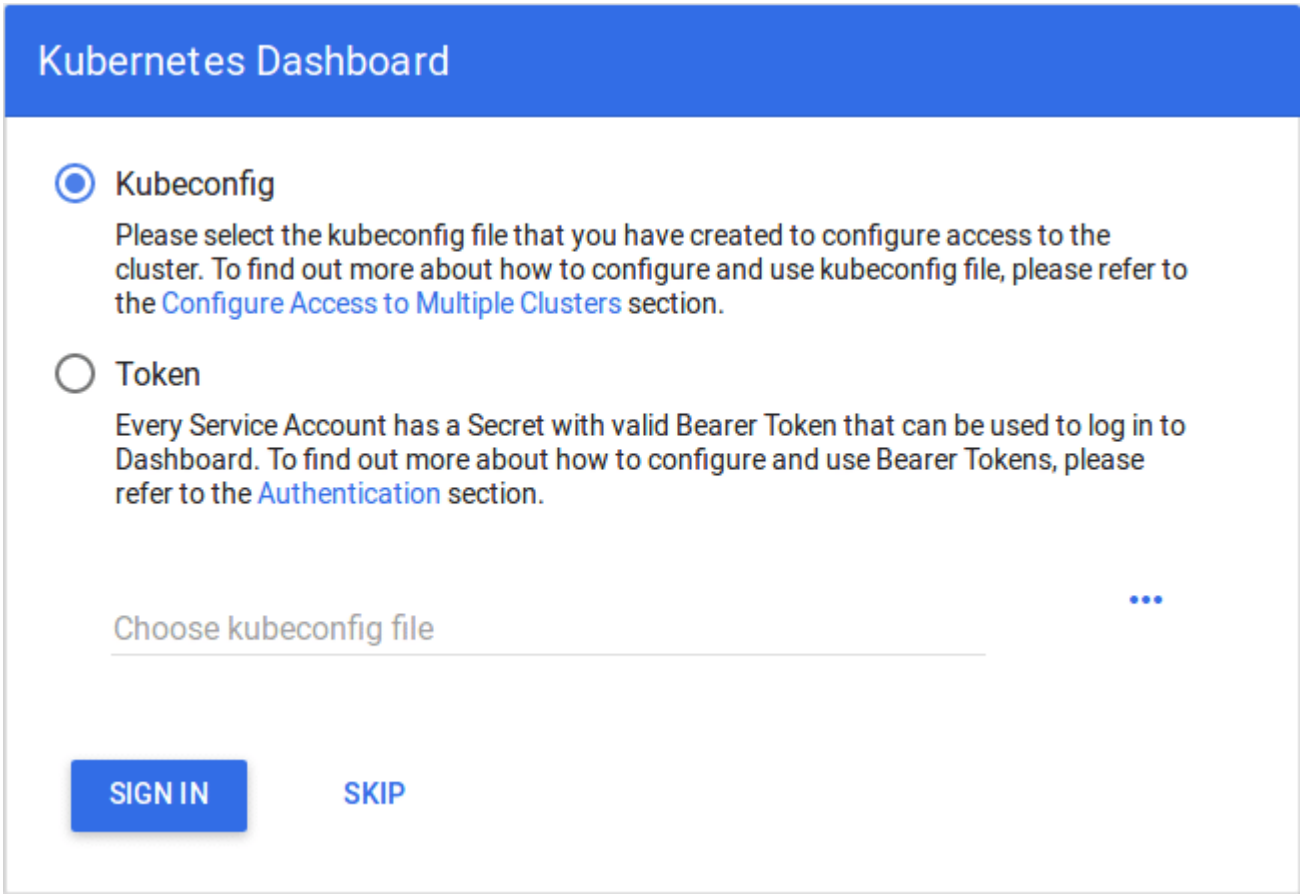
```
$ kubectl proxy
```

Then you will get something like this:

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

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/

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

Run the following commands:

1. This command will create a 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
```

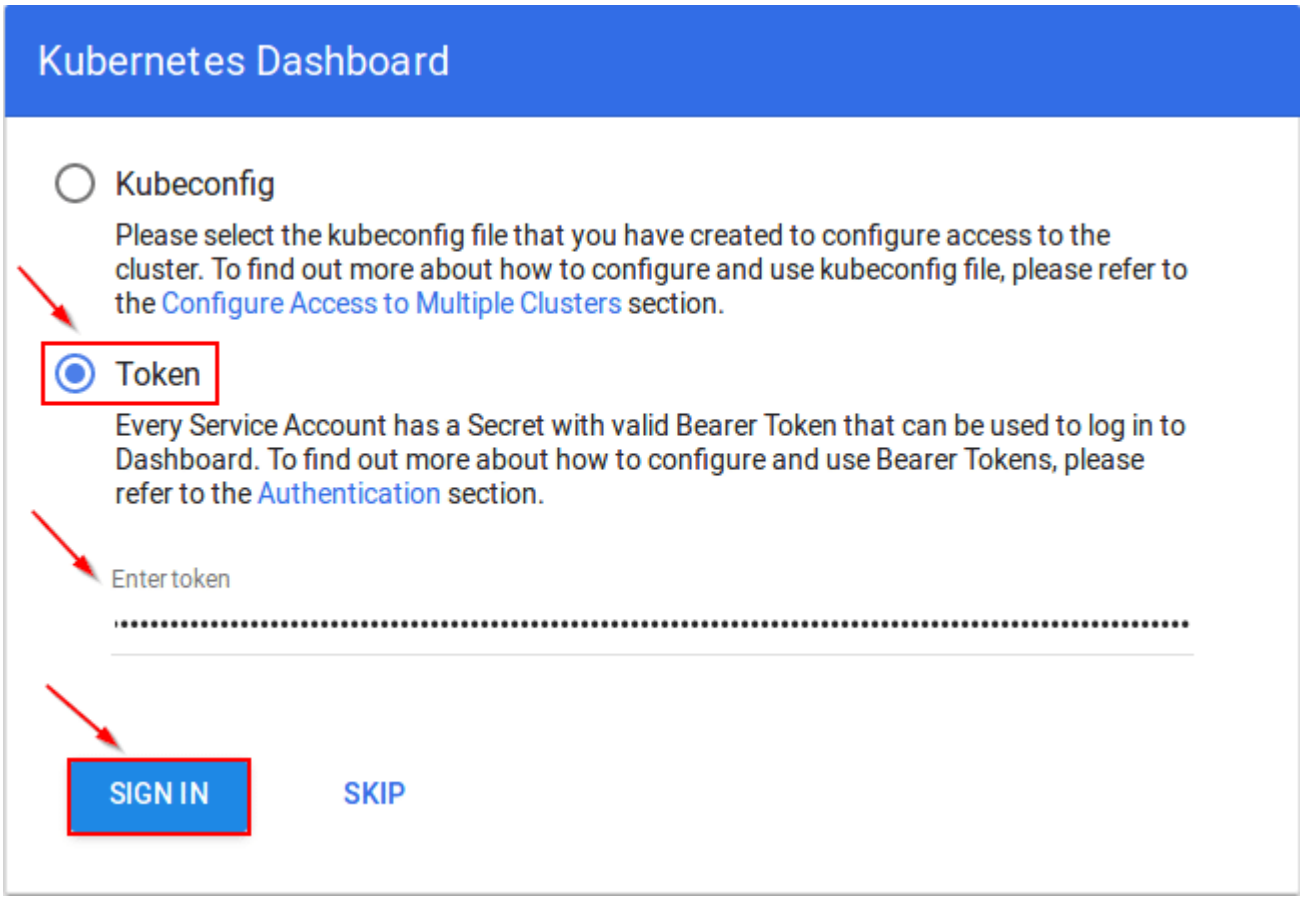
3. This command will give you the token required for your dashboard login

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

You should get the token like this:

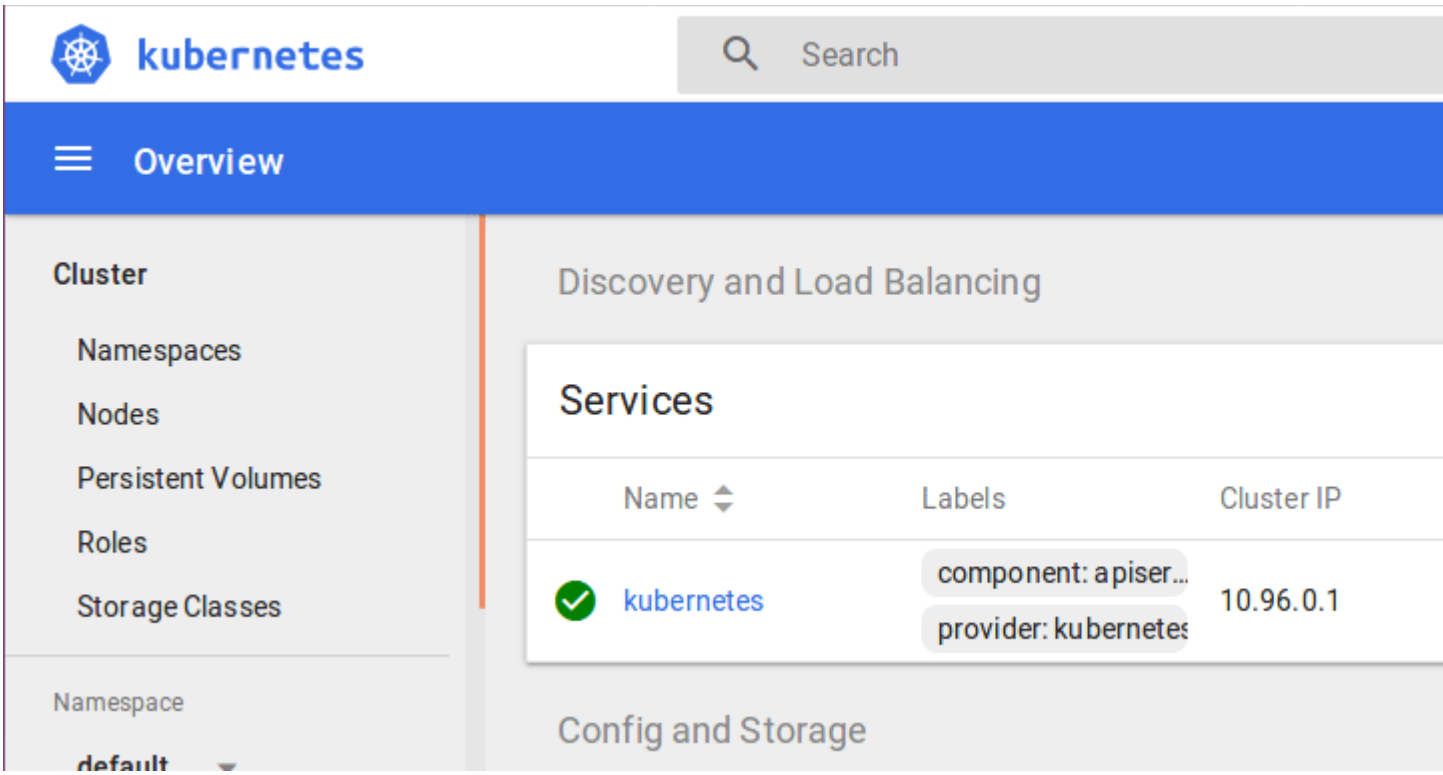
```
edureka@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard -o
jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiw
ia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXR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

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



5. You have successfully logged into your dashboard!

See Batch Details



Steps For Only Kubernetes Node VM (knode)

It is time to get your node, to join 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 if you get something similar to the above screenshot.