Kubernetes Installation Guide

Installation Notes

Connect to an EC2 instance(Ubuntu).

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

```
root@ip-172-31-42-111:/home/ubuntu# sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease [1
09 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
[107 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/main Sources [868 kB
]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/restricted Sources [
4,808 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/universe Sources [7,
728 kB]
Get:7 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/multiverse Sources [
179 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 Packa
ges [7,532 kB]
```

```
root@ip-172-31-42-111:/home/ubuntu# swapoff -a
root@ip-172-31-42-111:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Reading package lists... Done
root@ip-172-31-42-111:/home/ubuntu# apt-get install -y docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount ubuntu-fan
```

Step 2:

1. Run the following command:
   nano /etc/hostname

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

```
root@ip-172-31-42-111:/home/ubuntu# nano /etc/hostname
```

```
 GNU nano 2.5.3                File: /etc/hostname

kmaster
```

Step 3:

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”.

```
 GNU nano 2.5.3                File: /etc/hosts

127.0.0.1 localhost
172.31.42.111 kmaster
172.31.41.145 knode
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Step 4: Now we will ssh from one machine to another. Run the below command:

1. su - ubuntu
2. sudo service ufw stop
3. sudo service ssh start
4. eval `ssh-agent -s`

Note: Where ` in the above command refers to grave(`) and it is not a single quote(')

5. chmod 400 file_name.pem

6. ssh-add file_name.pem

Note: The file_name.pem is the file which you have transferred using Filezilla software in the first step.

7. ssh ubuntu@<private_IP_of_the_other_instance>

```
ubuntu@ip-172-31-42-111:~$ sudo su
root@ip-172-31-42-111:/home/ubuntu# su - ubuntu
ubuntu@ip-172-31-42-111:~$ sudo service ufw stop
ubuntu@ip-172-31-42-111:~$ sudo service ssh start
ubuntu@ip-172-31-42-111:~$ ls
test.pem
ubuntu@ip-172-31-42-111:~$ eval  `ssh-agent -s`
Agent pid 1990
ubuntu@ip-172-31-42-111:~$ chmod 400 test.pem
ubuntu@ip-172-31-42-111:~$ ssh-add test.pem
Identity added: test.pem (test.pem)
ubuntu@ip-172-31-42-111:~$ ssh ubuntu@172.31.41.145
The authenticity of host '172.31.41.145 (172.31.41.145)' can't be established.
ECDSA key fingerprint is SHA256:9jENsEy/7juxGpDbyj0WWxYcwa9jPSC5zs33ptSK298.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.41.145' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1067-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Fri Oct 12 09:02:32 2018 from 106.201.105.203
ubuntu@ip-172-31-41-145:~$ exit
logout
Connection to 172.31.41.145 closed.
```

Step 5: Next, we will install Docker. Run the following commands:

sudo su

apt-get update

apt-get install -y docker.io

```
root@ip-172-31-42-111:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Reading package lists... Done
root@ip-172-31-42-111:/home/ubuntu# apt-get install -y docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount ubuntu-fan
Suggested packages:
  mountall aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount docker.io ubuntu-fan
0 upgraded, 4 newly installed, 0 to remove and 32 not upgraded.
Need to get 17.1 MB of archives.
After this operation, 90.5 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/main amd64 bridge-ut
ils amd64 1.5-9ubuntu1 [28.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 cgrou
pfs-mount all 1.2 [4,970 B]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/universe amd
64 docker.io amd64 17.03.2-0ubuntu2~16.04.1 [17.1 MB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 u
buntu-fan all 0.12.8~16.04.2 [35.6 kB]
Fetched 17.1 MB in 0s (51.9 MB/s)
Preconfiguring packages ...
Selecting previously unselected package bridge-utils.
(Reading database ... 51284 files and directories currently installed.)
Preparing to unpack .../bridge-utils_1.5-9ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.5-9ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../cgroupfs-mount_1.2_all.deb ...
Unpacking cgroupfs-mount (1.2) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../docker.io_17.03.2-0ubuntu2~16.04.1_amd64.deb ...
Unpacking docker.io (17.03.2-0ubuntu2~16.04.1) ...
Selecting previously unselected package ubuntu-fan.
```

Step 6: 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

```
root@ip-172-31-42-111:/home/ubuntu# apt-get update && apt-get install -y apt-tra
nsport-https curl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.2.27).
The following packages will be upgraded:
  curl libcurl3-gnutls
2 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
Need to get 323 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 c
url amd64 7.47.0-1ubuntu2.9 [138 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 l
ibcurl3-gnutls amd64 7.47.0-1ubuntu2.9 [184 kB]
Fetched 323 kB in 0s (22.5 MB/s)
(Reading database ... 51443 files and directories currently installed.)
Preparing to unpack .../curl_7.47.0-1ubuntu2.9_amd64.deb ...
Unpacking curl (7.47.0-1ubuntu2.9) over (7.47.0-1ubuntu2.8) ...
Preparing to unpack .../libcurl3-gnutls_7.47.0-1ubuntu2.9_amd64.deb ...
Unpacking libcurl3-gnutls:amd64 (7.47.0-1ubuntu2.9) over (7.47.0-1ubuntu2.8) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libc-bin (2.23-0ubuntu10) ...
Setting up libcurl3-gnutls:amd64 (7.47.0-1ubuntu2.9) ...
Setting up curl (7.47.0-1ubuntu2.9) ...
Processing triggers for libc-bin (2.23-0ubuntu10) ...
root@ip-172-31-42-111:/home/ubuntu# curl -s https://packages.cloud.google.com/ap
t/doc/apt-key.gpg | apt-key add -
OK
root@ip-172-31-42-111:/home/ubuntu# cat <<EOF >/etc/apt/sources.list.d/kubernete
s.list
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
root@ip-172-31-42-111:/home/ubuntu# apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8,993 B
```

```
root@ip-172-31-42-111:/home/ubuntu# apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cri-tools ebtables kubernetes-cni socat
The following NEW packages will be installed:
  cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 7 newly installed, 0 to remove and 30 not upgraded.
Need to get 54.9 MB of archives.
After this operation, 364 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 e
btables amd64 2.0.10.4-3.4ubuntu2.16.04.2 [79.9 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 socat
 amd64 1.7.3.1-1 [321 kB]
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-too
ls amd64 1.12.0-00 [5,343 kB]
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kuberne
tes-cni amd64 0.6.0-00 [5,910 kB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet
 amd64 1.12.1-00 [24.7 MB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl
 amd64 1.12.1-00 [9,594 kB]
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm
 amd64 1.12.1-00 [8,987 kB]
Fetched 54.9 MB in 3s (15.9 MB/s)
Selecting previously unselected package cri-tools.
```

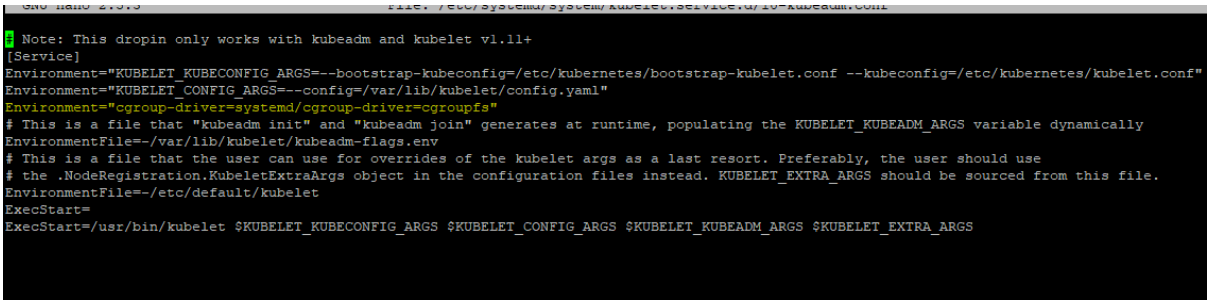
Step 7:

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"



Step 8: 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>

```
ubuntu@kmaster:~$ sudo su
root@kmaster:/home/ubuntu# kubeadm init --apiserver-advertise-address=172.31.42.
111
[init] using Kubernetes version: v1.12.1
[preflight] running pre-flight checks
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of yo
ur internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm
 config images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/
kubeadm-flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
```

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.

```
[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 172.31.42.111:6443 --token f3yk4i.7em8g9x051y48wjm --discovery-to
ken-ca-cert-hash sha256:cecd0ea050a1c35782ae010c3e30a2eecc27df681a5d8de1cb688992
c85e59ee

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

Step 2:

1. Like mentioned before, run the commands from the above output as a nonroot user.

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

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

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

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

   kubectl get pods -o wide --all-namespaces

```
ubuntu@kmaster:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
ubuntu@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                              READY   STATUS     RESTARTS   AG
E     IP               NODE      NOMINATED NODE
kube-system   coredns-576cbf47c7-bbtj6          0/1     Pending    0          5m
36s   <none>           <none>    <none>
kube-system   coredns-576cbf47c7-wfprl          0/1     Pending    0          5m
36s   <none>           <none>    <none>
kube-system   etcd-kmaster                      1/1     Running    0          1s
      172.31.42.111    kmaster   <none>
kube-system   kube-apiserver-kmaster            1/1     Running    0          1s
      172.31.42.111    kmaster   <none>
kube-system   kube-controller-manager-kmaster   1/1     Running    0          1s
      172.31.42.111    kmaster   <none>
kube-system   kube-proxy-w8wvm                  1/1     NodeLost   0          5m
36s   172.31.42.111    kmaster   <none>
kube-system   kube-scheduler-kmaster            0/1     Pending    0          1s
      <none>           kmaster   <none>
ubuntu@kmaster:~$
```

Step 3:

1. You will notice from the previous command, all the pods are not running. 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/v3.1/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml
```

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml
```

```
ubuntu@kmaster:~$ kubectl apply -f https://docs.projectcalico.org/v3.1/getting-s
tarted/kubernetes/installation/hosted/rbac-kdd.yaml
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
ubuntu@kmaster:~$ kubectl apply -f https://docs.projectcalico.org/v3.1/getting-s
tarted/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7
/calico.yaml
configmap/calico-config created
service/calico-typha created
deployment.apps/calico-typha created
daemonset.extensions/calico-node created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcal
ico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org cre
ated
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalic
o.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org crea
ted
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.or
g created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcal
ico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectc
alico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalic
o.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.
org created
serviceaccount/calico-node created
```

```
ubuntu@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
    IP              NODE      NOMINATED NODE
kube-system   calico-node-qjcmc                   2/2     Running   0          79s
    172.31.47.231   knode     <none>
kube-system   calico-node-w55tp                   2/2     Running   0          79s
    172.31.33.134   kmaster   <none>
kube-system   coredns-576cbf47c7-q2c2g            1/1     Running   0          9m5
s   192.168.1.2     knode     <none>
kube-system   coredns-576cbf47c7-xmfmx            1/1     Running   0          9m5
s   192.168.1.3     knode     <none>
kube-system   etcd-kmaster                        1/1     Running   0          50s
    172.31.33.134   kmaster   <none>
kube-system   kube-apiserver-kmaster              1/1     Running   0          50s
    172.31.33.134   kmaster   <none>
kube-system   kube-controller-manager-kmaster     1/1     Running   0          50s
    172.31.33.134   kmaster   <none>
kube-system   kube-proxy-jrjx5                    1/1     Running   0          9m5
s   172.31.33.134   kmaster   <none>
kube-system   kube-proxy-pqsh6                    1/1     Running   0          8m4
6s  172.31.47.231   knode     <none>
kube-system   kube-scheduler-kmaster              1/1     Running   0          50s
    172.31.33.134   kmaster   <none>
ubuntu@kmaster:~$
```

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/deploy/recommended/kubernetes-dashboard.yaml
```

```
ubuntu@kmaster:~$ kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-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.

```
ubuntu@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                    READY   STATUS    RESTARTS   AGE     IP              NODE      NOMINATED NODE
kube-system   calico-node-qjcmc                       2/2     Running   0          9m19s   172.31.47.231   knode     <none>
kube-system   calico-node-w55tp                       2/2     Running   0          9m19s   172.31.33.134   kmaster   <none>
kube-system   coredns-576cbf47c7-q2c2g                1/1     Running   0          17m     192.168.1.2     knode     <none>
kube-system   coredns-576cbf47c7-xmfmx                1/1     Running   0          17m     192.168.1.3     knode     <none>
kube-system   etcd-kmaster                            1/1     Running   0          8m50s   172.31.33.134   kmaster   <none>
kube-system   kube-apiserver-kmaster                  1/1     Running   0          8m50s   172.31.33.134   kmaster   <none>
kube-system   kube-controller-manager-kmaster         1/1     Running   0          8m50s   172.31.33.134   kmaster   <none>
kube-system   kube-proxy-jrjx5                        1/1     Running   0          17m     172.31.33.134   kmaster   <none>
kube-system   kube-proxy-pqsh6                        1/1     Running   0          16m     172.31.47.231   knode     <none>
kube-system   kube-scheduler-kmaster                  1/1     Running   0          8m50s   172.31.33.134   kmaster   <none>
kube-system   kubernetes-dashboard-77fd78f978-jpd9c   1/1     Running   0          46s     192.168.1.4     knode     <none>
```

Step 6: Now list all the services

```
ubuntu@kmaster:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   18m
ubuntu@kmaster:~$ kubectl get svc --all-namespaces
NAMESPACE     NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
default       kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP          18m
kube-system   calico-typha           ClusterIP   10.102.198.6    <none>        5473/TCP         10m
kube-system   kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP    18m
kube-system   kubernetes-dashboard   ClusterIP   10.106.33.55    <none>        443/TCP          2m10s
```

Step 7:

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

```
ubuntu@kmaster:~$ kubectl proxy &
[1] 8429
ubuntu@kmaster:~$ Starting to serve on 127.0.0.1:8001
^C
```

2. Curl the site

   curl 127.0.0.1:8001

```
ubuntu@kmaster:~$ curl 127.0.0.1:8001
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io",
    "/apis/apiregistration.k8s.io/v1",
    "/apis/apiregistration.k8s.io/v1beta1",
    "/apis/apps",
    "/apis/apps/v1",
    "/apis/apps/v1beta1",
    "/apis/apps/v1beta2",
    "/apis/authentication.k8s.io",
    "/apis/authentication.k8s.io/v1",
    "/apis/authentication.k8s.io/v1beta1",
    "/apis/authorization.k8s.io",
    "/apis/authorization.k8s.io/v1",
    "/apis/authorization.k8s.io/v1beta1",
    "/apis/autoscaling",
    "/apis/autoscaling/v1",
    "/apis/autoscaling/v2beta1",
    "/apis/autoscaling/v2beta2",
    "/apis/batch",
    "/apis/batch/v1",
    "/apis/batch/v1beta1",
    "/apis/certificates.k8s.io",
    "/apis/certificates.k8s.io/v1beta1",
    "/apis/coordination.k8s.io",
    "/apis/coordination.k8s.io/v1beta1",
    "/apis/crd.projectcalico.org",
    "/apis/crd.projectcalico.org/v1",
    "/apis/events.k8s.io",
```

3. Run the below command:

   kubectl get svc --all-namespaces

```
}ubuntu@kmaster:~$kubectl get svc --all-namespaces
NAMESPACE     NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
default       kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP          25m
kube-system   calico-typha           ClusterIP   10.102.198.6    <none>        5473/TCP         17m
kube-system   kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP    25m
kube-system   kubernetes-dashboard   ClusterIP   10.106.33.55    <none>        443/TCP          8m35s
```

4. Edit the file svc file for kubernetes dashboard using the below command and change the type to NodePort.

kubectl edit svc kubernetes-dashboard –namespaces=kube-system

```
ubuntu@kmaster:~$ kubectl edit svc kubernetes-dashboard --namespace=kube-system
service/kubernetes-dashboard edited
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: 2018-10-12T14:03:55Z
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "10117"
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: a80f4aa1-ce27-11e8-b88e-0e4d5ea30750
spec:
  clusterIP: 10.106.33.55
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 30466
    port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
~
~
```

5. Run the below command again:
   Kubectl get svc –all-namespaces

   You will get the port on which kubernetes dashboard is running as shown in the below screenshot.

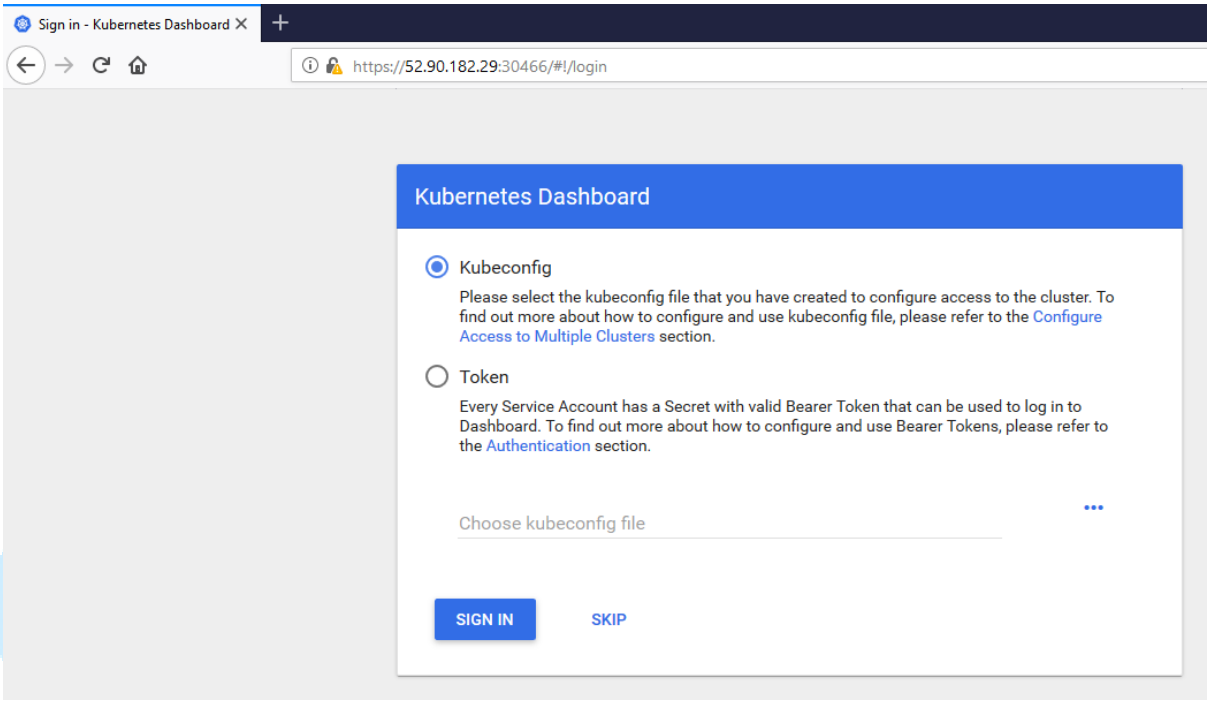
```
ubuntu@kmaster:~$ kubectl get svc --all-namespaces
NAMESPACE     NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
default       kubernetes             ClusterIP   10.96.0.1       <none>        443/TCP          151m
kube-system   calico-typha           ClusterIP   10.102.198.6    <none>        5473/TCP         143m
kube-system   kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP    151m
kube-system   kubernetes-dashboard   NodePort    10.106.33.55    <none>        443:30466/TCP    135m
```

```
ubuntu@kmaster:~$ kubectl proxy &
[1] 415
ubuntu@kmaster:~$ Starting to serve on 127.0.0.1:8001
^C
```

6. To view the dashboard in the browser, navigate to the following address in the browser of your Master VM and use mozilla firefox and any other browser other than chrome.

   https://External_IP_of_Instance:Port_Number

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



Step 8: 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

```
ubuntu@kmaster:~$ kubectl create serviceaccount dashboard -n default
serviceaccount/dashboard created
```

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

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

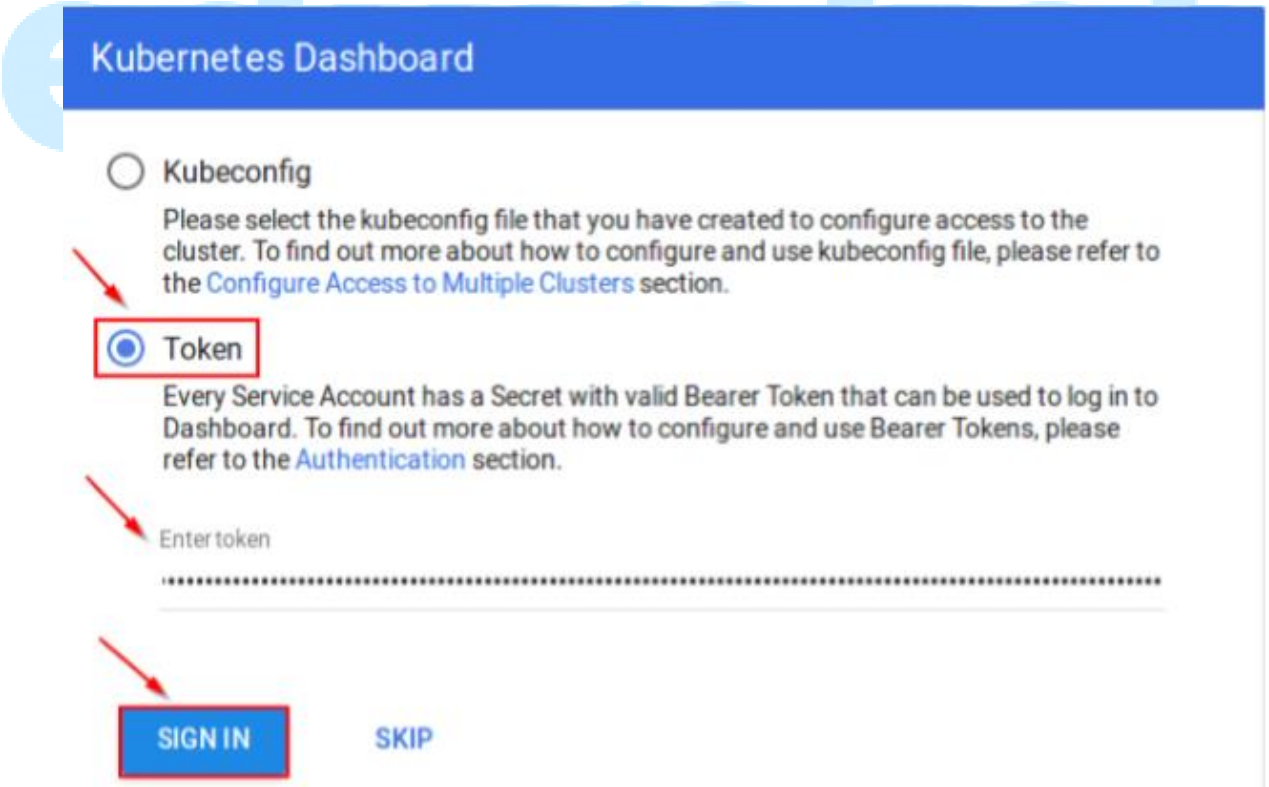
```
ubuntu@kmaster:~$ kubectl create clusterrolebinding dashboard-admin -n default \
> --clusterrole=cluster-admin \
> --serviceaccount=default:dashboard
clusterrolebinding.rbac.authorization.k8s.io/dashboard-admin created
```

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

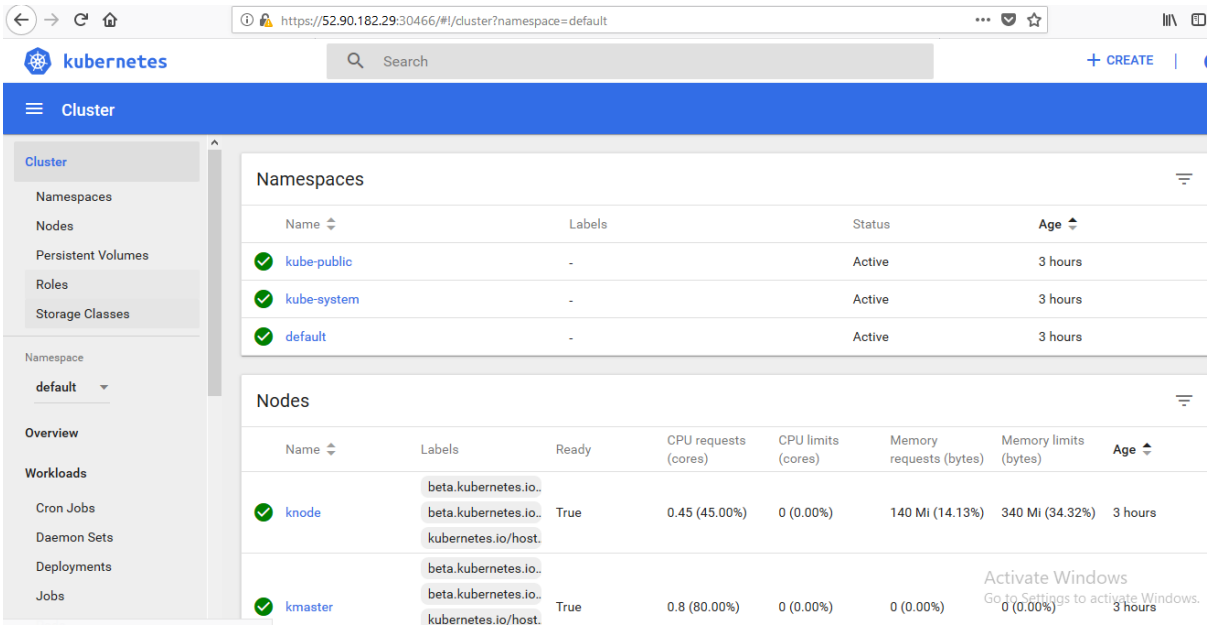
   kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode

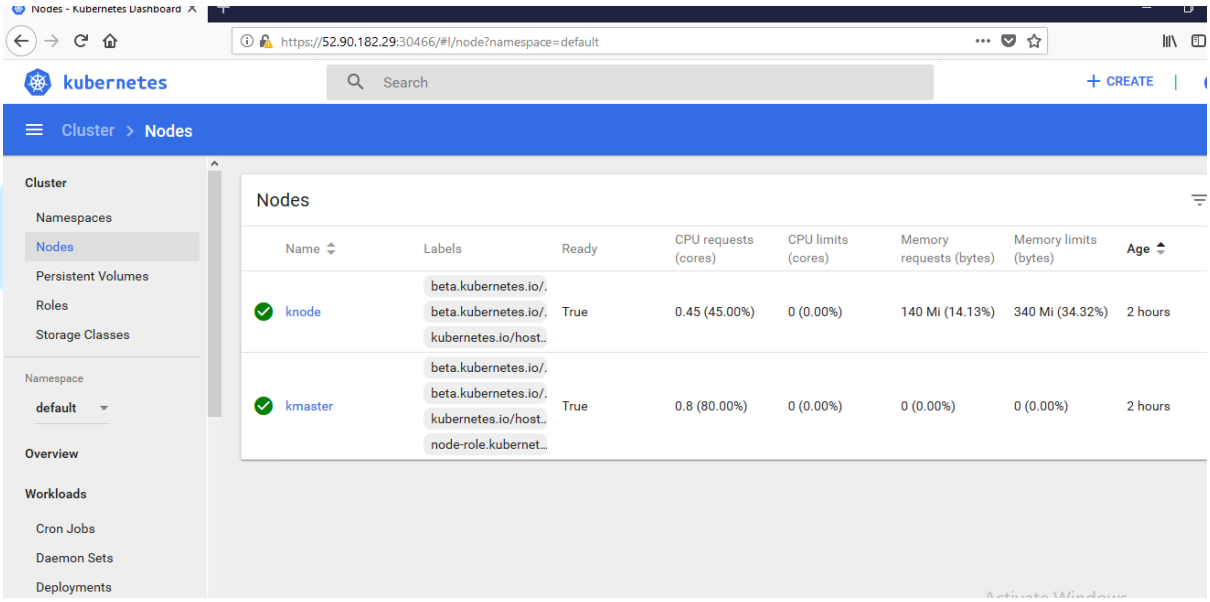
```
ubuntu@kmaster:~$ kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9
zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6ImRhc2hib2FyZC10b2tlbi16dmNxeiIsImtlYmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJkYXNoYm9hcmQiLCJrdWJlcm5ldGV
zLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5laWQiOiI3NjY0Yzg4ZiljZTQ3LTExZTgtYjg4ZS0wZTRkNWVhMzA3NTAiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDpkYXNoYm9hcmQ
ifQ.o3fbV9C92U1AxJrSooSUyVdDojMR8XDlF5cUFGGWto8GBl-_f15CG8zgD8M3Tbt75e5yuH2kAKGHX_LDBve_mgceP4V0tXYNmlbAdzMs4G4xRMpVLp6ZaSpIW7zfKZ3_T_palYiqldHiYsyUPQxv6OganN-mEHknlVjc
nsl6lwTFnvHjHjXER9fqgm64_uDEoUtfYFur0aSAXcUjUyg8TTlWfWfteAgTXwpXmRTGYd_VoPiXETQ5GBaw46DXv3hPKGGf9n0WagOQxGbO-KTDRSnhnu4s0u4PdF7oi9TrrHZINvOE2vJx8GJNjlPDl7Hoqx0rooQ0QgUM
zIes2vBZTAubuntu@k^C
```

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”.

```
root@knode:/home/ubuntu# kubeadm join 172.31.33.134:6443 --token jwlxwo.d96hqnng
4w0v2iyu --discovery-token-ca-cert-hash sha256:0d81d584cdce4596209237d30e980ef88
86ff8bcf5b549872bf25f69e996be82
[preflight] running pre-flight checks
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not
be used, because the following required kernel modules are not loaded: [ip_vs_sh
 ip_vs ip_vs_rr ip_vs_wrr] or no builtin kernel ipvs support: map[ip_vs_wrr:{} i
p_vs_sh:{} nf_conntrack_ipv4:{} ip_vs:{} ip_vs_rr:{}]
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

[discovery] Trying to connect to API Server "172.31.33.134:6443"
[discovery] Created cluster-info discovery client, requesting info from "https:/
/172.31.33.134:6443"
[discovery] Requesting info from "https://172.31.33.134:6443" 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 "172.31.33.134:6443"
[discovery] Successfully established connection with API Server "172.31.33.134:6
443"
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.1
2" ConfigMap in the kube-system namespace
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/
kubeadm-flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to t
he Node API object "knode" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.
```

Your Kubernetes Cluster is ready!