Kubernetes is an open source platform for managing containerized applications developed by Google. It allows you to manage, scale, and automatically deploy your containerized applications in the clustered environment. With Kubernetes, we can orchestrate our containers across multiple hosts, scale the containerized applications with all resources on the fly, and have centralized container management environment.

In this tutorial, I will show you step-by-step how to install and configure Kubernetes on CentOS 7. We will be using 1 server 'k8s-master' as the Kubernetes Host Master, and 2 servers as Kubernetes node, 'node01' and 'node02'.

Prerequisites

- 3 CentOS 7 Servers
  - 10.0.15.10   k8s-master
  - 10.0.15.21   node01
  - 10.0.15.22   node02
- Root privileges

What we will do?

- Kubernetes Installation
- Kubernetes Cluster Initialization
- Adding node01 and node02 to the Cluster
- Testing - Create First Pod

Step 1 - Kubernetes Installation

In this first step, we will prepare those 3 servers for Kubernetes installation, so run all commands on the master and node servers.

We will prepare all servers for Kubernetes installation by changing the existing configuration on servers, and also installating some packages, including docker-ce and kubernetes itself.

- Configure Hosts

Edit hosts file on all server using the vim editor.

```
vim /etc/hosts
```

Paste the host's list below.

```
10.0.15.10      k8s-master
10.0.15.21      node01
10.0.15.22      node02
```

Save and exit.

- Disable SELinux

In this tutorial, we will not cover about SELinux configuration for Docker, so we will disable it.

Run the command below to disable SELinux.

```
setenforce 0
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

- Enable br_netfilter Kernel Module

The br_netfilter module is required for kubernetes installation. Enable this kernel module so that the packets traversing the bridge are processed by iptables for filtering and for port forwarding, and the kubernetes pods across the cluster can communicate with each other.

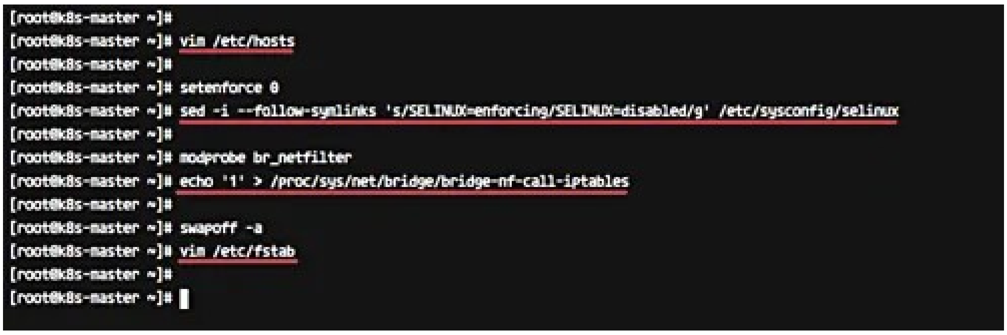
Run the command below to enable the br_netfilter kernel module.

```
modprobe br_netfilter
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

- Disable SWAP

Disable SWAP for kubernetes installation by running the following commands.

```
swapoff -a
```

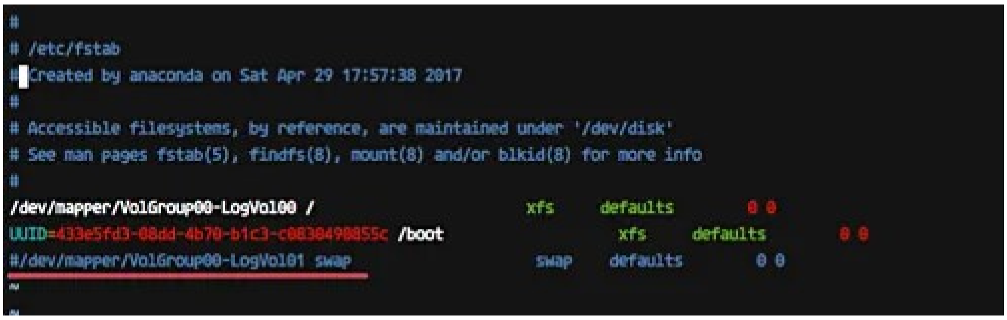


```
[root@k8s-master ~]#
[root@k8s-master ~]# vim /etc/hosts
[root@k8s-master ~]#
[root@k8s-master ~]# setenforce 0
[root@k8s-master ~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
[root@k8s-master ~]#
[root@k8s-master ~]# modprobe br_netfilter
[root@k8s-master ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@k8s-master ~]#
[root@k8s-master ~]# swapoff -a
[root@k8s-master ~]# vim /etc/fstab
[root@k8s-master ~]#
[root@k8s-master ~]# 
```

And then edit the '/etc/fstab' file.

```
vim /etc/fstab
```

Comment the swap line UUID as below.



```
#
# /etc/fstab
# Created by anaconda on Sat Apr 29 17:57:38 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/VolGroup00-LogVol00 /                       xfs     defaults        0 0
UUID=433e5fd3-08dd-4b70-b1c3-c08304908855c /boot                   xfs     defaults        0 0
#/dev/mapper/VolGroup00-LogVol01 swap                    swap    defaults        0 0
~
~
```

- Install Docker CE

Install the latest version of Docker-ce from the docker repository.

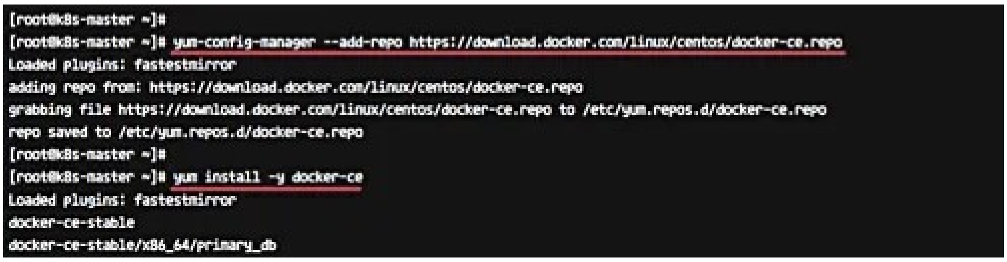
Install the package dependencies for docker-ce.

```
yum install -y yum-utils device-mapper-persistent-data lvm
2
```

Add the docker repository to the system and install docker-ce using the yum command.

```
yum-config-manager --add-repo https://download.docker.com/
linux/centos/docker-ce.repo
yum install -y docker-ce
```

Wait for the docker-ce installation.



```
[root@k8s-master ~]#
[root@k8s-master ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@k8s-master ~]#
[root@k8s-master ~]# yum install -y docker-ce
Loaded plugins: fastestmirror
docker-ce-stable
docker-ce-stable/x86_64/primary_db
```

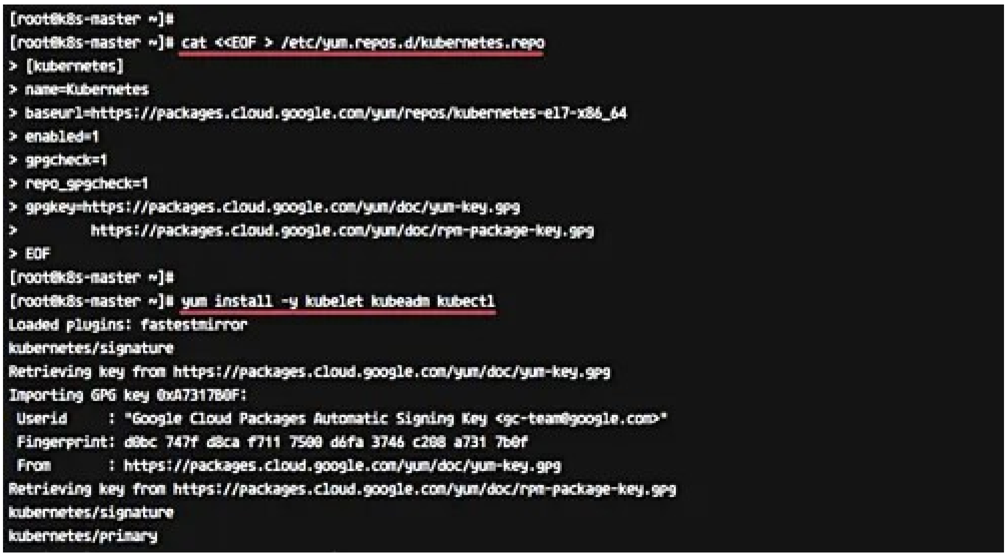
- Install Kubernetes

Add the kubernetes repository to the centos 7 system by running the following command.

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
        https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Now install the kubernetes packages kubeadm, kubelet, and kubectl using the yum command below.

```
yum install -y kubelet kubeadm kubectl
```



```
[root@k8s-master ~]#
[root@k8s-master ~]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
> [kubernetes]
> name=Kubernetes
> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
> enabled=1
> gpgcheck=1
> repo_gpgcheck=1
> gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
>         https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
> EOF
[root@k8s-master ~]#
[root@k8s-master ~]# yum install -y kubelet kubeadm kubectl
Loaded plugins: fastestmirror
kubernetes/signature
Retrieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg
Importing GPG key 0xA7317B0F:
 Userid     : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"
 Fingerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Retrieving key from https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
kubernetes/signature
kubernetes/primary
```

After the installation is complete, restart all those servers.

```
sudo reboot
```

Log in again to the server and start the services, docker and kubelet.

```
systemctl start docker && systemctl enable docker
systemctl start kubelet && systemctl enable kubelet
```

- Change the cgroup-driver

We need to make sure the docker-ce and kubernetes are using same 'cgroup'.

Check docker cgroup using the docker info command.

```
docker info | grep -i cgroup
```

And you see the docker is using '**cgroupfs**' as a cgroup-driver.

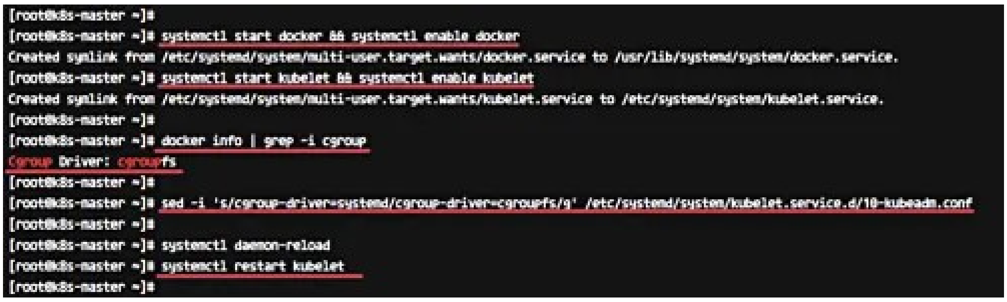
Now run the command below to change the kuberetes cgroup-driver to 'cgroupfs'.

```
sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g'
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Reload the systemd system and restart the kubelet service.

```
systemctl daemon-reload
systemctl restart kubelet
```

Now we're ready to configure the Kubernetes Cluster.



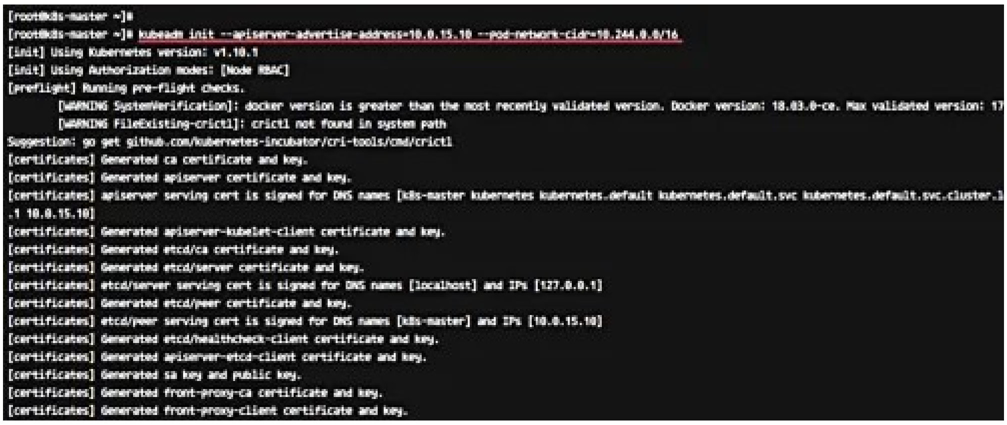
```
[root@k8s-master ~]#
[root@k8s-master ~]# systemctl start docker && systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@k8s-master ~]# systemctl start kubelet && systemctl enable kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
[root@k8s-master ~]#
[root@k8s-master ~]# docker info | grep -i cgroup
Cgroup Driver: cgroupfs
[root@k8s-master ~]#
[root@k8s-master ~]# sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g' /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
[root@k8s-master ~]#
[root@k8s-master ~]# systemctl daemon-reload
[root@k8s-master ~]# systemctl restart kubelet
[root@k8s-master ~]#
```

Step 2 - Kubernetes Cluster Initialization

In this step, we will initialize the kubernetes master cluster configuration.

Move the shell to the master server 'k8s-master' and run the command below to set up the kubernetes master.

```
kubeadm init --apiserver-advertise-address=10.0.15.10 --pod-network-cidr=10.244.0.0/16
```



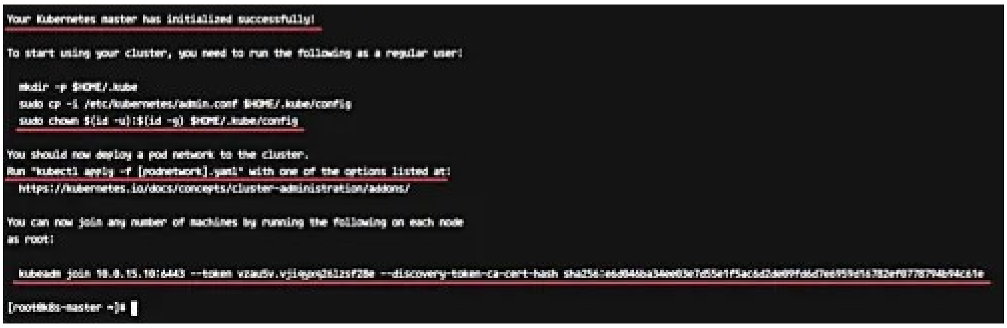
```
[root@k8s-master ~]#
[root@k8s-master ~]# kubeadm init --apiserver-advertise-address=10.0.15.10 --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.10.1
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks.
        [WARNING SystemVerification]: docker version is greater than the most recently validated version. Docker version: 18.03.0-ce. Max validated version: 17
        [WARNING FileExisting-crictl]: crictl not found in system path
Suggestion: go get github.com/kubernetes-incubator/cri-tools/cmd/crictl
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [k8s-master kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.l
.1 10.0.15.10]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [localhost] and IPs [127.0.0.1]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names [k8s-master] and IPs [10.0.15.10]
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
```

Note:

--apiserver-advertise-address = determines which IP address Kubernetes should advertise its API server on.

--pod-network-cidr = specify the range of IP addresses for the pod network. We're using the 'flannel' virtual network. If you want to use another pod network such as weave-net or calico, change the range IP address.

When the Kubernetes initialization is complete, you will get the result as below.



```
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 10.0.15.10:6443 --token vzau5v.vjiqgxq2612sf28e --discovery-token-ca-cert-hash sha256:e6d046ba34ee03e7d55e1f5ac6d2de09fd6d7e6959d16782ef0778794b94c61e

[root@k8s-master ~]#
```

Note:

Copy the '**kubeadm join ... ...**' command to your text editor. The command will be used to register new nodes to the kubernetes cluster.

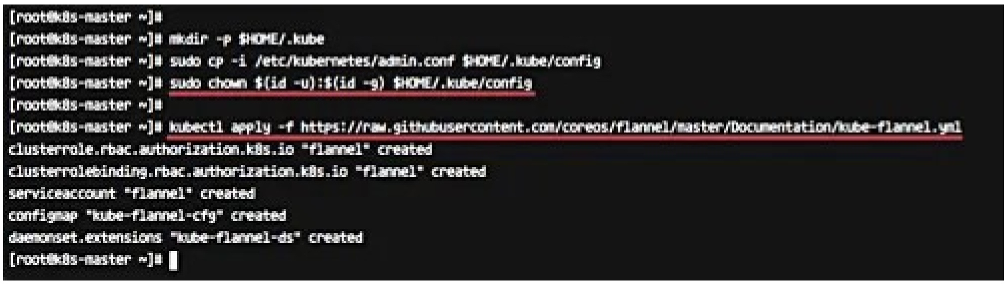
Now in order to use Kubernetes, we need to run some commands as on the result.

Create new '.kube' configuration directory and copy the configuration 'admin.conf'.

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

Next, deploy the flannel network to the kubernetes cluster using the kubectl command.

```
kubectl apply -f https://raw.githubusercontent.com/coreos/
flannel/master/Documentation/kube-flannel.yml
```



```
[root@k8s-master ~]#
[root@k8s-master ~]# mkdir -p $HOME/.kube
[root@k8s-master ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
clusterrole.rbac.authorization.k8s.io "flannel" created
clusterrolebinding.rbac.authorization.k8s.io "flannel" created
serviceaccount "flannel" created
configmap "kube-flannel-cfg" created
daemonset.extensions "kube-flannel-ds" created
[root@k8s-master ~]#
```

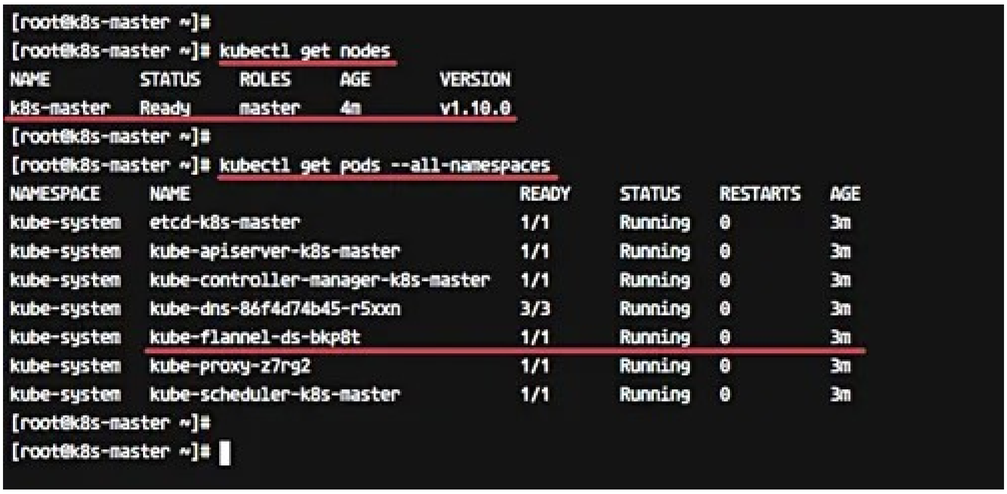
The flannel network has been deployed to the Kubernetes cluster.

Wait for a minute and then check kubernetes node and pods using commands below.

```
kubectl get nodes
kubectl get pods --all-namespaces
```

And you will get the 'k8s-master' node is running as a 'master' cluster with status 'ready', and you will get all pods that are needed for the cluster, including the 'kube-flannel-ds' for network pod configuration.

Make sure all kube-system pods status is 'running'.



```
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get nodes
NAME         STATUS    ROLES     AGE       VERSION
k8s-master   Ready     master    4m        v1.10.0
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                 READY     STATUS    RESTARTS   AGE
kube-system   etcd-k8s-master                      1/1       Running   0          3m
kube-system   kube-apiserver-k8s-master            1/1       Running   0          3m
kube-system   kube-controller-manager-k8s-master   1/1       Running   0          3m
kube-system   kube-dns-86f4d74b45-r5xxn            3/3       Running   0          3m
kube-system   kube-flannel-ds-bkp8t                1/1       Running   0          3m
kube-system   kube-proxy-z7rg2                     1/1       Running   0          3m
kube-system   kube-scheduler-k8s-master            1/1       Running   0          3m
[root@k8s-master ~]#
[root@k8s-master ~]#
```

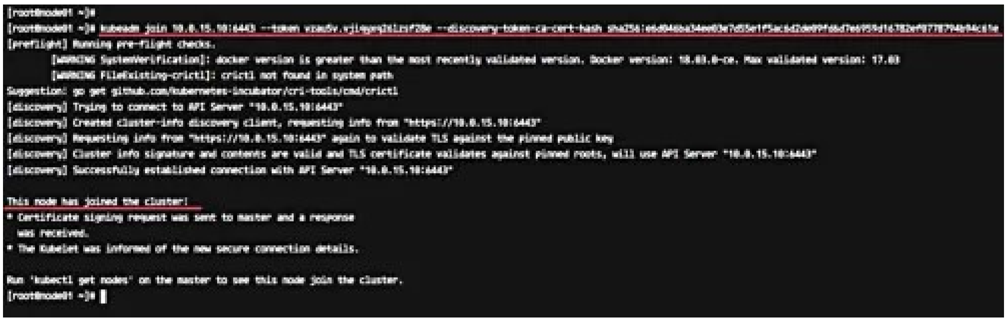
Kubernetes cluster master initialization and configuration has been completed.

Step 3 - Adding node01 and node02 to the Cluster

In this step, we will add node01 and node02 to join the 'k8s' cluster.

Connect to the node01 server and run the kubeadm join command as we copied on the top.

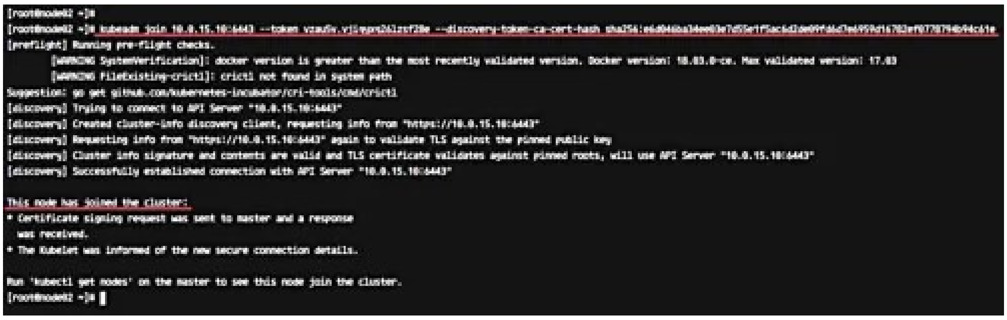
```
kubeadm join 10.0.15.10:6443 --token vzau5v.vjiqyxq26lzsf28e --discovery-token-ca-cert-hash sha256:e6d046ba34ee03e7d55e1f5ac6d2de09fd6d7e6959d16782ef0778794b94c61e
```



```
[root@node01 ~]#
[root@node01 ~]# kubeadm join 10.0.15.10:6443 --token vzau5v.vjiqyxq26lzsf28e --discovery-token-ca-cert-hash sha256:e6d046ba34ee03e7d55e1f5ac6d2de09fd6d7e6959d16782ef0778794b94c61e
[preflight] Running pre-flight checks.
        [WARNING SystemVerification]: docker version is greater than the most recently validated version. Docker version: 18.03.0-ce. Max validated version: 17.03
        [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 "10.0.15.10:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://10.0.15.10:6443"
[discovery] Requesting info from "https://10.0.15.10:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "10.0.15.10:6443"
[discovery] Successfully established connection with API Server "10.0.15.10:6443"

This node has joined the cluster:
* Certificate signing request was sent to master 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.
[root@node01 ~]#
```

Connect to the node02 server and run the kubeadm join command as we copied on the top.

```
kubeadm join 10.0.15.10:6443 --token vzau5v.vjiqyxq26lzsf28e --discovery-token-ca-cert-hash sha256:e6d046ba34ee03e7d55e1f5ac6d2de09fd6d7e6959d16782ef0778794b94c61e
```

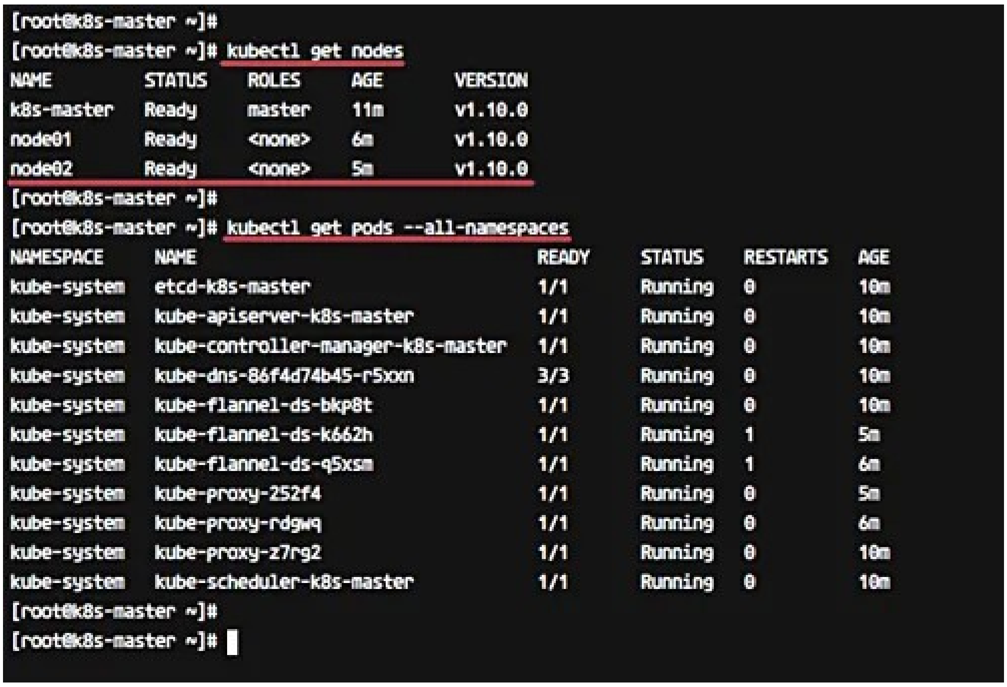


```
[root@node02 ~]#
[root@node02 ~]# kubeadm join 10.0.15.10:6443 --token vzau5v.vjiqyxq26lzsf28e --discovery-token-ca-cert-hash sha256:e6d046ba34ee03e7d55e1f5ac6d2de09fd6d7e6959d16782ef0778794b94c61e
[preflight] Running pre-flight checks.
        [WARNING SystemVerification]: docker version is greater than the most recently validated version. Docker version: 18.03.0-ce. Max validated version: 17.03
        [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 "10.0.15.10:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://10.0.15.10:6443"
[discovery] Requesting info from "https://10.0.15.10:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "10.0.15.10:6443"
[discovery] Successfully established connection with API Server "10.0.15.10:6443"

This node has joined the cluster:
* Certificate signing request was sent to master 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.
[root@node02 ~]#
```

Wait for some minutes and back to the 'k8s-master' master cluster server check the nodes and pods using the following command.

```
kubectl get nodes
kubectl get pods --all-namespaces
```

Now you will get node01 and node02 has been added to the cluster with status 'ready'.



```
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get nodes
NAME         STATUS    ROLES     AGE       VERSION
k8s-master   Ready     master    11m       v1.10.0
node01       Ready     <none>    6m        v1.10.0
node02       Ready     <none>    5m        v1.10.0
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                 READY     STATUS    RESTARTS   AGE
kube-system   etcd-k8s-master                      1/1       Running   0          10m
kube-system   kube-apiserver-k8s-master            1/1       Running   0          10m
kube-system   kube-controller-manager-k8s-master   1/1       Running   0          10m
kube-system   kube-dns-86f4d74b45-r5xxn            3/3       Running   0          10m
kube-system   kube-flannel-ds-bkp8t                1/1       Running   0          10m
kube-system   kube-flannel-ds-k662h                1/1       Running   1          5m
kube-system   kube-flannel-ds-q5xsm                1/1       Running   1          6m
kube-system   kube-proxy-252f4                     1/1       Running   0          5m
kube-system   kube-proxy-rdgwq                     1/1       Running   0          6m
kube-system   kube-proxy-z7rg2                     1/1       Running   0          10m
kube-system   kube-scheduler-k8s-master            1/1       Running   0          10m
[root@k8s-master ~]#
[root@k8s-master ~]#
```

node01 and node02 have been added to the kubernetes cluster.

Step 4 - Testing Create First Pod

In this step, we will do a test by deploying the Nginx pod to the kubernetes cluster. A pod is a group of one or more containers with shared storage and network that runs under Kubernetes. A Pod contains one or more containers, such as Docker container.

Login to the 'k8s-master' server and create new deployment named 'nginx' using the kubectl command.

```
kubectl create deployment nginx --image=nginx
```

To see details of the 'nginx' deployment sepcification, run the following command.

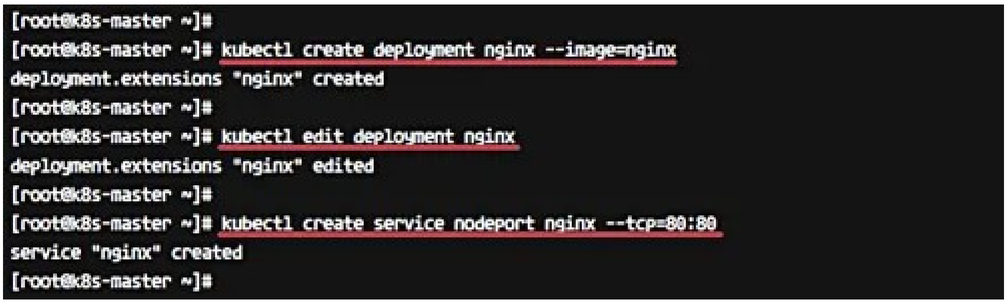
```
kubectl describe deployment nginx
```

And you will get the nginx pod deployment specification.

Next, we will expose the nginx pod accessible via the internet. And we need to create new service NodePort for this.

Run the kubectl command below.

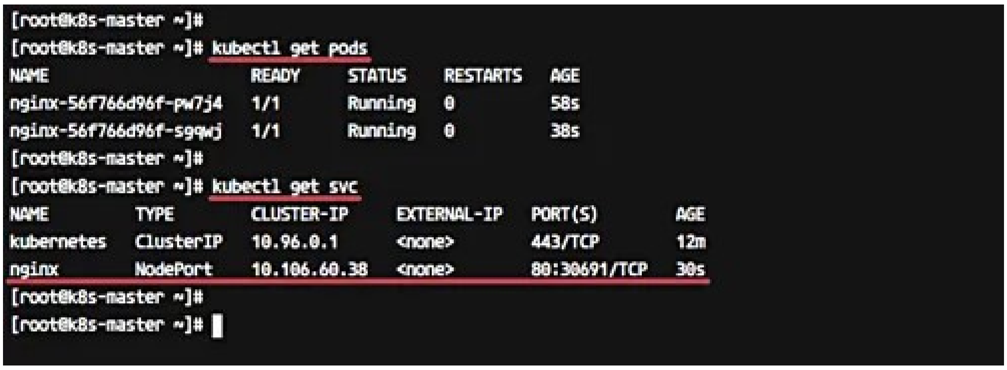
```
kubectl create service nodeport nginx --tcp=80:80
```



```
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl create deployment nginx --image=nginx
deployment.extensions "nginx" created
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl edit deployment nginx
deployment.extensions "nginx" edited
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl create service nodeport nginx --tcp=80:80
service "nginx" created
[root@k8s-master ~]#
```

Make sure there is no error. Now check the nginx service nodeport and IP using the kubectl command below.

```
kubectl get pods
kubectl get svc
```

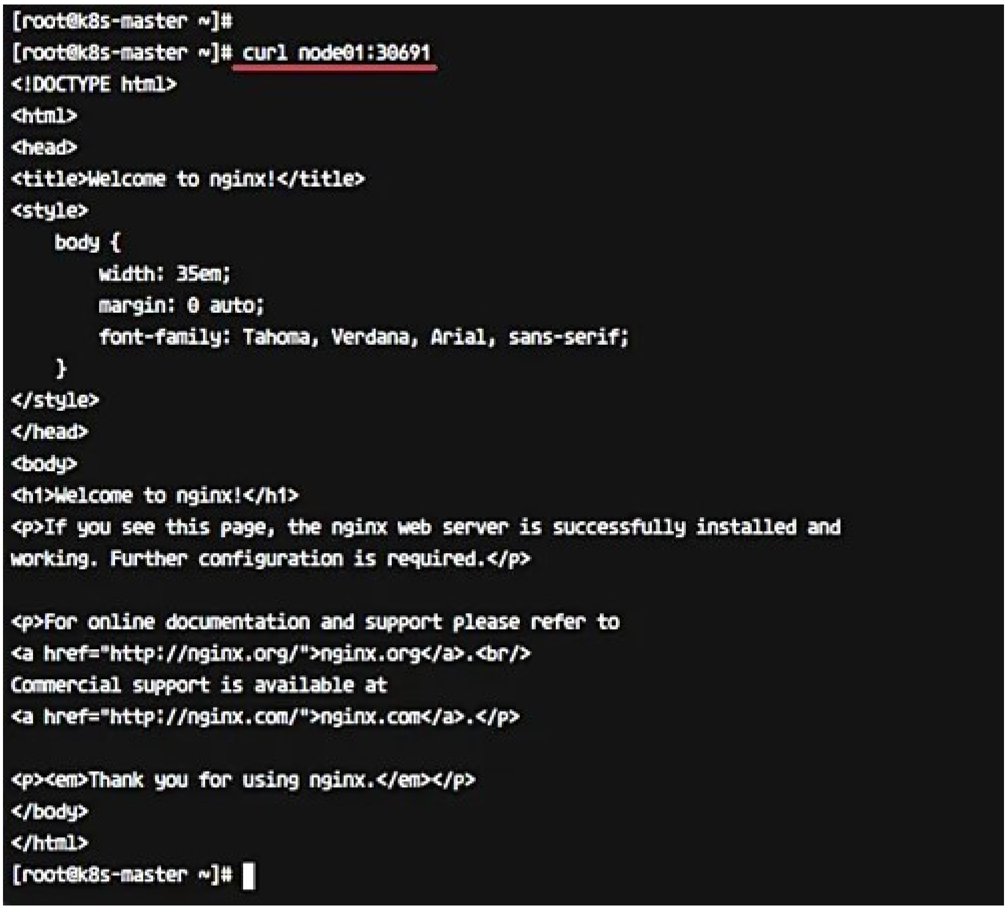


```
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-56f766d96f-pw7j4   1/1       Running   0          58s
nginx-56f766d96f-sgqwj   1/1       Running   0          38s
[root@k8s-master ~]#
[root@k8s-master ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        12m
nginx        NodePort    10.106.60.38   <none>        80:30691/TCP   30s
[root@k8s-master ~]#
[root@k8s-master ~]# 
```

Now you will get the nginx pod is now running under cluster IP address '10.160.60.38' port 80, and the node main IP address '10.0.15.x' on port '30691'.

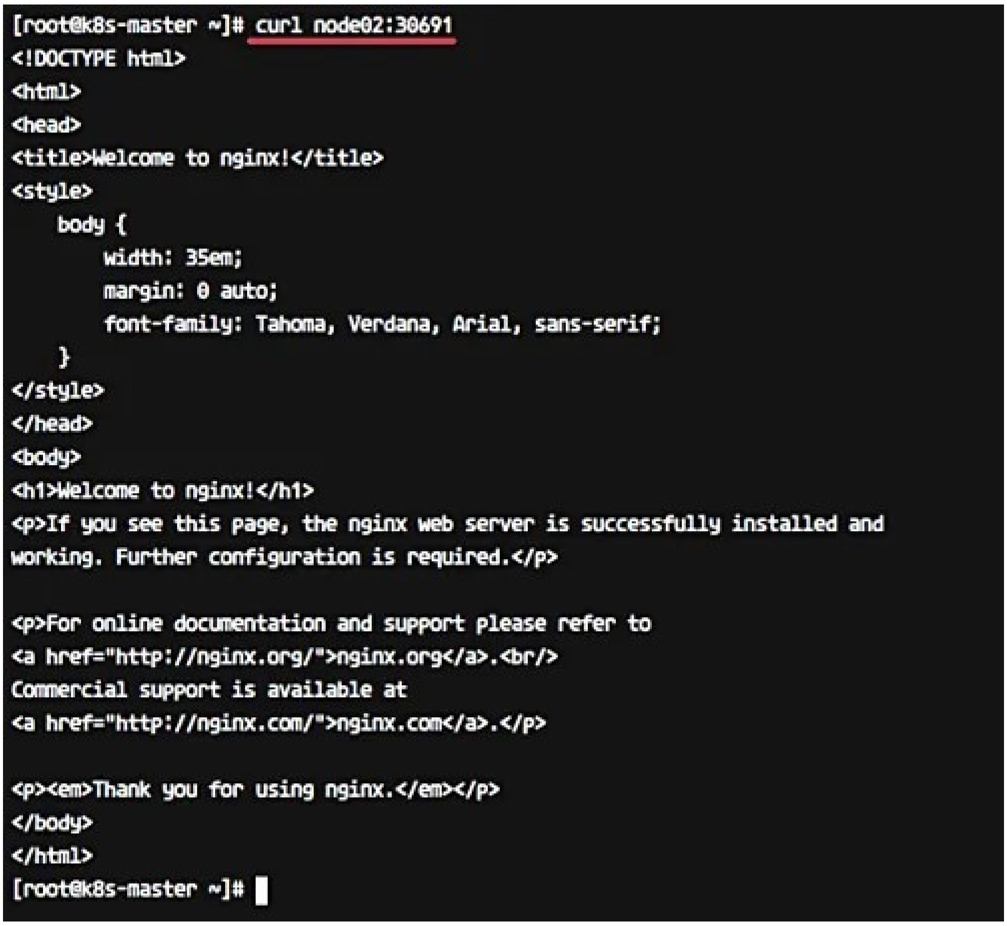
From the 'k8s-master' server run the curl command below.

```
curl node01:30691
```



```
[root@k8s-master ~]#
[root@k8s-master ~]# curl node01:30691
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@k8s-master ~]#
```

```
curl node02:30691
```



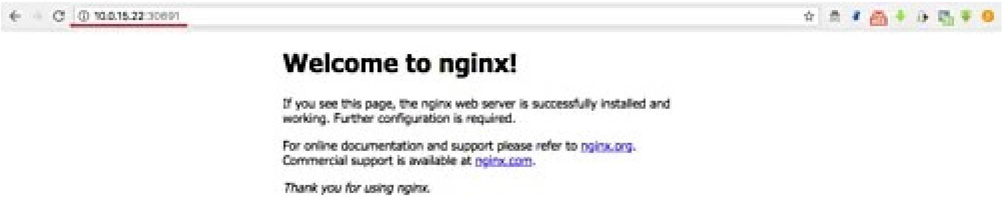
```
[root@k8s-master ~]# curl node02:30691
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@k8s-master ~]# 
```

The Nginx Pod has now been deployed under the Kubernetes cluster and it's accessible via the internet.

Now access from the web browser.

http://10.0.15.10:30691/

And you will get the Nginx default page.



10.0.15.22:30691

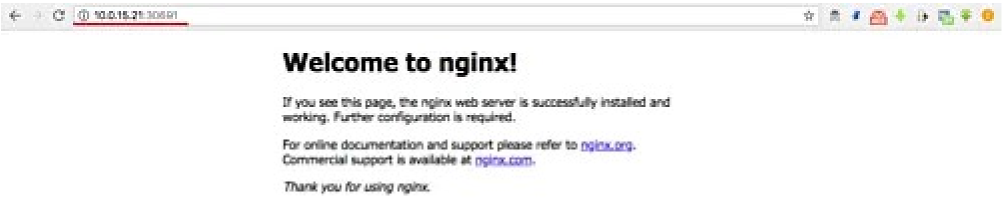
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

On the node02 server - **http://10.0.15.11:30691/**



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

The Kubernetes cluster Installation and configuration on CentOS 7 has been completed successfully.