KUBERNETES – MULTI MASTER – MULTI NODE WITH EXTERNAL ETCD

For create a multi-master cluster we should follow several step.
For install a minimum environment you need 6 instance:
1 load balancer, 3 master node, 2 worker node.
For this test labs, I spin up nodes with AWS, and I obtained the following configuration:

ServerName	Description	Private IP	Public IP
loadbalancer	HA Proxy and client machine	10.10.6.99/16	54.171.193.153
master-0	Kubernetes master node 0	10.10.38.131/16	34.248.196.219
master-1	Kubernetes master node 1	10.10.12.172/16	176.34.146.195
master-2	Kubernetes master node 2	10.10.24.77/16	63.35.222.59
node-0	Kubernetes worker node 0	10.10.0.79/16	34.242.32.175
node-1	Kubernetes worker node 1	10.10.28.188/16	34.242.255.210

Installing the client tools

Installing the Cloud Flare SSL

We will need two tools on the client machine:

- the Cloud Flare SSL tool to generate the different certificates
- the Kubernetes client, kubectl, to manage the Kubernetes cluster

Because of the nature of labs, we will use loadbalancer instance as a client machine. In a production environment you must use a different instance,
Download the binaries.

```
[centos@loadbalancer ~]$ pwd
/home/centos
[centos@loadbalancer ~]$ wget https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
–2019-03-15 08:30:18–  https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
Resolving pkg.cfssl.org (pkg.cfssl.org)… 104.16.235.19, 104.16.234.19, 2606:4700::6810:ea13, …
Connecting to pkg.cfssl.org (pkg.cfssl.org)|104.16.235.19|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 10376657 (9.9M) [application/octet-stream]
Saving to: ‘cfssl_linux-amd64’

100%
```

RECENT POSTS

SQL on vSAN | VMware vSAN | #StorageMinute
HPE Top 10 Reasons for vVols
Announcing vSphere 6.7 Update 3
New Release: PowerCLI 11.4.0
VMware vSAN 6.7 U3 is Generally Available

```
2019-03-15 08:30:20 (5.94 MB/s) – 'cfssl_linux-amd64' saved [10376657/10376657]
```

```
[centos@loadbalancer ~]$ wget https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
–2019-03-15 08:31:18–  https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
Resolving pkg.cfssl.org (pkg.cfssl.org)… 104.16.235.19, 104.16.234.19, 2606:4700::6810:ea13, …
Connecting to pkg.cfssl.org (pkg.cfssl.org)|104.16.235.19|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 2277873 (2.2M) [application/octet-stream]
Saving to: 'cfssljson_linux-amd64'

100%
[==========================================================================================
2,277,873   2.01MB/s in 1.1s

2019-03-15 08:31:20 (2.01 MB/s) – 'cfssljson_linux-amd64' saved [2277873/2277873]
```

Add the execution permission to the binaries.

```
[centos@loadbalancer ~]$ chmod +x cfssl*
```

Move the binaries to /usr/local/bin

```
[centos@loadbalancer ~]$ sudo mv cfssl_linux-amd64 /usr/local/bin/cfssl

[centos@loadbalancer ~]$ sudo mv cfssljson_linux-amd64 /usr/local/bin/cfssljson
```

Verify the installation

```
[centos@loadbalancer ~]$ cfssl version
Version: 1.2.0
Revision: dev
Runtime: go1.6
```

Installing kubectl

Download the binary

```
[centos@loadbalancer ~]$ wget https://storage.googleapis.com/kubernetes-release/release/v1.13.1/bin/linux/amd64/kubectl
–2019-03-15 08:36:25–  https://storage.googleapis.com/kubernetes-release/release/v1.13.1/bin/linux/amd64/kubectl
Resolving storage.googleapis.com (storage.googleapis.com)… 209.85.203.128, 2a00:1450:400b:c01::80
Connecting to storage.googleapis.com (storage.googleapis.com)|209.85.203.128|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 39222752 (37M) [application/octet-stream]
Saving to: 'kubectl'

100%
[==============================================================================================================
39,222,752  59.9MB/s in 0.6s

2019-03-15 08:36:27 (59.9 MB/s) – 'kubectl' saved [39222752/39222752]
```

Add the execution permission to the binary.

```
[centos@loadbalancer ~]$ chmod +x kubectl
```

Move the binary to /usr/local/bin.

```
[centos@loadbalancer ~]$ kubectl version
Client Version: version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.1", GitCommit:"eec55b9ba98609a46fee712359c7b5b365bdd920", GitTreeState:"clean", BuildDate:"2018-12-13T10:39:04Z", GoVersion:"go1.11.2", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused – did you specify the right host or port?
```

Installing the HAProxy load balancer

As we will deploy three Kubernetes master nodes, we need to deploy an HAPRoxy load balancer in front of them to distribute the traffic
Install HAProxy using yum.

```
[centos@loadbalancer ~]$ sudo yum install -y haproxy.x86_64
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Running transaction
  Installing : haproxy-1.5.18-8.el7.x86_64 1/1
  Verifying  : haproxy-1.5.18-8.el7.x86_64 1/1

Installed:
  haproxy.x86_64 0:1.5.18-8.el7

Complete!
```

Configure HAProxy to load balance the traffic between the three Kubernetes master nodes.

```
[centos@loadbalancer ~]$ sudo vim /etc/haproxy/haproxy.cfg
```

File: /etc/haproxy/haproxy.cfg

```
global
....
default
....

  timeout check 10s
  maxconn 3000

frontend kubernetes
bind 10.10.6.99:6443
option tcplog
mode tcp
default_backend kubernetes-master-nodes

backend kubernetes-master-nodes
mode tcp
balance roundrobin
option tcp-check
server k8s-master-0 10.10.38.131:6443 check fall 3 rise 2
```

....

Restart HAProxy.

```
[centos@loadbalancer ~]$ sudo systemctl restart haproxy
```

Check HAProxy services with systemctl. The service should be started correctly.

```
[centos@loadbalancer ~]$ sudo systemctl status haproxy
```

Generating the TLS certificates

These steps can be done on your Linux desktop if you have one or on the loadbalancer instance depending on where you installed the cfssl tool. I will use loadbalancer insance.
Create the certificate authority configuration file.

```
[centos@loadbalancer ~]$ vim ca-config.json
```

File: /home/centos/ca-config.json

```
{
 "signing": {
  "default": {
   "expiry": "8760h"
  },
  "profiles": {
   "kubernetes": {
    "usages": ["signing", "key encipherment", "server auth", "client auth"],
    "expiry": "8760h"
   }
  }
 }
}
```

Create the certificate authority signing request configuration file.

```
[centos@loadbalancer ~]$ vim ca-csr.json
```

File: /home/centos/ca-csr.json

```
{
 "CN": "Kubernetes",
 "key": {
  "algo": "rsa",
  "size": 2048
 },
 "names": [
 {
  "C": "IT",
  "L": "Altamura",
  "O": "Kubernetes",
  "OU": "CA",
  "ST": "Desotech srl"
 }
]
}
```

Generate the certificate authority certificate and private key.

```
2019/03/15 09:05:21 [INFO] generate received request
2019/03/15 09:05:21 [INFO] received CSR
2019/03/15 09:05:21 [INFO] generating key: rsa-2048
2019/03/15 09:05:21 [INFO] encoded CSR
2019/03/15 09:05:21 [INFO] signed certificate with serial number 467500906140962859175326073358531030909211319162
```

Verify that the ca-key.pem and the ca.pem were generated.

```
[centos@loadbalancer ~]$ ls
ca-config.json  ca.csr ca-csr.json  ca-key.pem ca.pem
```

Creating the certificate for the Etcd cluster

Create the certificate signing request configuration file.

```
[centos@loadbalancer ~]$ vim kubernetes-csr.json
```

File: /home/centos/kubernetes-csr.json

```
{
 "CN": "Kubernetes",
 "key": {
  "algo": "rsa",
  "size": 2048
 },
 "names": [
 {
  "C": "IT",
  "L": "Altamura",
  "O": "Kubernetes",
  "OU": "Kubernetes",
  "ST": "Desotech srl"
 }
]
}
```

Generate the certificate and private key.

```
[centos@loadbalancer ~]$ cfssl gencert \
-ca=ca.pem \
-ca-key=ca-key.pem \
-config=ca-config.json                                                    \
hostname=54.171.193.153,10.10.38.131,34.248.196.219,10.10.38.131,176.34.146.195,10.10.12.172,63.35.222.59,10.10.24.77,127.0.0.1,kuberne
\
-profile=kubernetes kubernetes-csr.json | \
cfssljson -bare kubernetes
2019/03/15 11:22:52 [INFO] generate received request
2019/03/15 11:22:52 [INFO] received CSR
2019/03/15 11:22:52 [INFO] generating key: rsa-2048
2019/03/15 11:22:53 [INFO] encoded CSR
2019/03/15 11:22:53 [INFO] signed certificate with serial number 301560059863354361452739326024537675371032634283
```

Verify that the kubernetes-key.pem and the kubernetes.pem file were generated.

```
[centos@loadbalancer ~]$ ls
ca-config.json  ca.csr ca-csr.json  ca-key.pem ca.pem kubernetes.csr  kubernetes-csr.json
```

Copy the certificate to each nodes:

Copy to master-1

```
[centos@loadbalancer ~]$ scp ca.pem kubernetes.pem kubernetes-key.pem centos@34.248.196.219:/home/centos
ca.pem                100% 1383   871.6KB/s   00:00
kubernetes.pem        100% 1545   950.6KB/s   00:00
kubernetes-key.pem    100% 1679     1.1MB/s   00:00
```

Copy to master-1

```
[centos@loadbalancer ~]$ scp ca.pem kubernetes.pem kubernetes-key.pem centos@176.34.146.195:/home/centos
ca.pem                100% 1383   969.9KB/s   00:00
kubernetes.pem        100% 1545     1.3MB/s   00:00
kubernetes-key.pem    100% 1679     1.3MB/s   00:00
```

Copy to master-2

```
[centos@loadbalancer ~]$ scp ca.pem kubernetes.pem kubernetes-key.pem centos@63.35.222.59:/home/centos
ca.pem                100% 1383   818.1KB/s   00:00
kubernetes.pem        100% 1545   947.6KB/s   00:00    kubernetes-key.pem    100% 1679   997.8KB/s   00:00
```

Copy to node-0

```
[centos@loadbalancer ~]$ scp ca.pem kubernetes.pem kubernetes-key.pem centos@34.242.32.175:/home/centos
ca.pem                100% 1383     1.0MB/s   00:00
kubernetes.pem        100% 1545     1.3MB/s   00:00
kubernetes-key.pem    100% 1679     1.3MB/s   00:00
```

Copy to node-1

```
[centos@loadbalancer ~]$ scp ca.pem kubernetes.pem kubernetes-key.pem centos@34.242.255.210:/home/centos
ca.pem                100% 1383   762.1KB/s   00:00
kubernetes.pem        100% 1545   821.0KB/s   00:00
kubernetes-key.pem    100% 1679   922.8KB/s   00:00
```

Preparing the nodes for kubeadm

Preparing the master-0 instance

Installing Docker

Remove old version of docker.

```
[centos@master-0 ~]$ sudo yum remove docker \
                  . docker-client \
                    docker-client-latest \
                    docker-common \
                    docker-latest \
                    docker-latest-logrotate \
                    docker-logrotate \
                    docker-engine
Loaded plugins: fastestmirror
No Match for argument: docker
```

No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
No Packages marked for removal

Add some utils for use docker with a persistent data and yum utils.

```
[centos@master-0 ~]$ sudo yum install -y yum-utils \device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Dependency Updated:
 device-mapper.x86_64 7:1.02.149-10.el7_6.3 device-mapper-event.x86_64
7:1.02.149-10.el7_6.3 device-mapper-event-libs.x86_64 7:1.02.149-10.el7_6.3
 device-mapper-libs.x86_64 7:1.02.149-10.el7_6.3 lvm2-libs.x86_64 7:2.02.180-
10.el7_6.3

Complete!
```

Add docker repository

```
[centos@master-0 ~]$ sudo 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
```

Setup docker daemon.

```
[centos@master-0 ~]$ sudo -i
[root@master-0 ~]# cat > /etc/docker/daemon.json <<EOF
{
 “exec-opts”: [“native.cgroupdriver=systemd”],
 “log-driver”: “json-file”,
 “log-opts”: {
  “max-size”: “100m”
 },
 “storage-driver”: “overlay2”,
 “storage-opts”: [
  “overlay2.override_kernel_check=true”
 ]
}
EOF
```

Install Docker

```
[root@master-0 ~]# exit
[centos@manager-0 ~]$ yum update -y && yum install -y docker-ce-18.06.2.ce
....
Installed:
 docker-ce.x86_64 0:18.06.2.ce-3.el7
```

libtool-ltdl.x86_64 0:2.4.2-22.el7_3

Complete!

Start docker.

```
[root@master-0 ~]# systemctl daemon-reload
[root@master-0 ~]# systemctl enable –now docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@master-0 ~]# systemctl restart docker
[root@master-0 ~]# systemctl status docker
● docker.service – Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2019-03-16 14:07:48 UTC; 6s ago
    Docs: https://docs.docker.com
```

Installing Kubeadm, Kubectl and kubelet

```
[centos@master-0 ~]$ sudo -i
[root@master-0 ~]$ 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
exclude=kube*
EOF
```

Set SELinux in permissive mode (effectively disabling it)

```
[root@master-0 ~]$ setenforce 0
[root@master-0 ~]$ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Install kubelet, kubeadm and kubectl.

```
[root@master-0 ~]# yum install -y kubelet kubeadm kubectl –disableexcludes=kubernetes
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Installed:
  kubeadm.x86_64 0:1.13.4-0      kubectl.x86_64 0:1.13.4-0      kubelet.x86_64 0:1.13.4-0

Dependency Installed:
 conntrack-tools.x86_64 0:1.4.4-4.el7              cri-tools.x86_64 0:1.12.0-0 ebtables.x86_64 0:2.0.10-16.el7              kubernetes-cni.x86_64 0:0.6.0-0
 libnetfilter_cthelper.x86_64 0:1.0.0-9.el7        libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7 libnetfilter_queue.x86_64 0:1.0.2-2.el7_2      socat.x86_64 0:1.7.3.2-2.el7
```

Start kubelet service.

```
[root@master-0 ~]# systemctl enable –now kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
```

Disable the swap.

```
[root@master-0 ~]# swapoff -a
[root@master-0 ~]# sed -i '/ swap / s/^/#/' /etc/fstab
```

Preparing the master-1 instance

Installing Docker
Remove old version of docker.

```
[centos@master-1 ~]$ sudo yum remove docker \
                  . docker-client \
                    docker-client-latest \
                    docker-common \
                    docker-latest \
                    docker-latest-logrotate \
                    docker-logrotate \
                    docker-engine
Loaded plugins: fastestmirror
No Match for argument: docker
No Match for argument: docker-client
No Match for argument: docker-client-latest
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
No Packages marked for removal
```

Add some utils for use docker with a persistent data and yum utils.

```
[centos@master-1 ~]$ sudo yum install -y yum-utils \device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Dependency Updated:
  device-mapper.x86_64  7:1.02.149-10.el7_6.3                    device-mapper-event.x86_64
7:1.02.149-10.el7_6.3 device-mapper-event-libs.x86_64 7:1.02.149-10.el7_6.3
  device-mapper-libs.x86_64  7:1.02.149-10.el7_6.3               lvm2-libs.x86_64  7:2.02.180-
10.el7_6.3

Complete!
```

Add docker repository

```
[centos@master-1 ~]$ sudo 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
```

repo saved to /etc/yum.repos.d/docker-ce.repo

Setup docker daemon.

```
[centos@master-1 ~]$ sudo -i
[root@master-1 ~]# cat > /etc/docker/daemon.json <<EOF
{
 "exec-opts": ["native.cgroupdriver=systemd"],
 "log-driver": "json-file",
 "log-opts": {
  "max-size": "100m"
 },
 "storage-driver": "overlay2",
 "storage-opts": [
  "overlay2.override_kernel_check=true"
 ]
}
EOF
```

Install Docker

```
[root@master-1 ~]# exit
[centos@manager-1 ~]$ yum update -y && yum install -y docker-ce-18.06.2.ce
....
Installed:
 docker-ce.x86_64 0:18.06.2.ce-3.el7

Dependency Installed:
 libtool-ltdl.x86_64 0:2.4.2-22.el7_3

Complete!
```

Start docker.

```
[root@master-1 ~]# systemctl daemon-reload
[root@master-1 ~]# systemctl enable –now docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@master-1 ~]# systemctl restart docker
[root@master-1 ~]# systemctl status docker
● docker.service – Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2019-03-16 14:07:48 UTC; 6s ago
    Docs: https://docs.docker.com
```

Installing Kubeadm, Kubectl and kubelet

```
[centos@master-1 ~]$ sudo -i
[root@master-1 ~]$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
```

```
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
exclude=kube*
EOF
```

Set SELinux in permissive mode (effectively disabling it)

```
[root@master-1 ~]$ setenforce 0
[root@master-1 ~]$ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Install kubelet, kubeadm and kubectl.

```
[root@master-1 ~]# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Installed:
  kubeadm.x86_64 0:1.13.4-0      kubectl.x86_64 0:1.13.4-0      kubelet.x86_64 0:1.13.4-0

Dependency Installed:
 conntrack-tools.x86_64 0:1.4.4-4.el7              cri-tools.x86_64 0:1.12.0-0 ebtables.x86_64 0:2.0.10-16.el7              kubernetes-cni.x86_64 0:0.6.0-0
 libnetfilter_cthelper.x86_64 0:1.0.0-9.el7          libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7
libnetfilter_queue.x86_64 0:1.0.2-2.el7_2      socat.x86_64 0:1.7.3.2-2.el7

Complete!
```

Start kubelet service.

```
[root@master-1 ~]# systemctl enable --now kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
```

Disable the swap.

```
[root@master-1 ~]# swapoff -a
[root@master-1 ~]# sed -i '/ swap / s/^/#/' /etc/fstab
```

Preparing the master-2 instance

Installing Docker
Remove old version of docker.

```
[centos@master-2 ~]$ sudo yum remove docker \
                  . docker-client \
                    docker-client-latest \
                    docker-common \
                    docker-latest \
                    docker-latest-logrotate \
                    docker-logrotate \
                    docker-engine
Loaded plugins: fastestmirror
```

```
No Match for argument: docker-client-latest
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
No Packages marked for removal
```

Add some utils for use docker with a persistent data and yum utils.

```
[centos@master-2 ~]$ sudo yum install -y yum-utils \device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Dependency Updated:
 device-mapper.x86_64  7:1.02.149-10.el7_6.3                device-mapper-event.x86_64
7:1.02.149-10.el7_6.3 device-mapper-event-libs.x86_64 7:1.02.149-10.el7_6.3
 device-mapper-libs.x86_64  7:1.02.149-10.el7_6.3           lvm2-libs.x86_64  7:2.02.180-
10.el7_6.3

Complete!
```

Add docker repository

```
[centos@master-2 ~]$ sudo 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
```

Setup docker daemon.

```
[centos@master-2 ~]$ sudo -i
[root@master-2 ~]# cat > /etc/docker/daemon.json <<EOF
{
 “exec-opts”: [“native.cgroupdriver=systemd”],
 “log-driver”: “json-file”,
 “log-opts”: {
   “max-size”: “100m”
 },
 “storage-driver”: “overlay2”,
 “storage-opts”: [
   “overlay2.override_kernel_check=true”
 ]
}
EOF
```

Install Docker

```
[root@master-2 ~]# exit
[centos@manager-2 ~]$ yum update -y && yum install -y docker-ce-18.06.2.ce
....
```

Dependency Installed:
libtool-ltdl.x86_64 0:2.4.2-22.el7_3

Complete!

Start docker.

[root@master-2 ~]# **systemctl daemon-reload**
[root@master-2 ~]# **systemctl enable –now docker**
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@master-2 ~]# **systemctl restart docker**
[root@master-2 ~]# **systemctl status docker**
● docker.service – Docker Application Container Engine
Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
Active: active (running) since Sat 2019-03-16 14:07:48 UTC; 6s ago
Docs: https://docs.docker.com

Installing Kubeadm, Kubectl and kubelet

[centos@master-2 ~]$ **sudo -i**
[root@master-2 ~]$ **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
exclude=kube*
EOF

Set SELinux in permissive mode (effectively disabling it)

[root@master-2 ~]$ **setenforce 0**
[root@master-2 ~]$ **sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config**

Install kubelet, kubeadm and kubectl.

[root@master-2 ~]# **yum install -y kubelet kubeadm kubectl –disableexcludes=kubernetes**
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
….
Installed:
kubeadm.x86_64 0:1.13.4-0 kubectl.x86_64 0:1.13.4-0 kubelet.x86_64 0:1.13.4-0

Dependency Installed:
conntrack-tools.x86_64 0:1.4.4-4.el7 cri-tools.x86_64 0:1.12.0-0 ebtables.x86_64 0:2.0.10-16.el7 kubernetes-cni.x86_64 0:0.6.0-0
libnetfilter_cthelper.x86_64 0:1.0.0-9.el7 libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7

Complete!

Start kubelet service.

```
[root@master-2 ~]# systemctl enable –now kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
```

Disable the swap.

```
[root@master-2 ~]# swapoff -a
[root@master-2 ~]# sed -i '/ swap / s/^/#/' /etc/fstab
```

Preparing the node-0 instance

Installing Docker
Remove old version of docker.

```
[centos@node-0 ~]$ sudo yum remove docker \
                  . docker-client \
                    docker-client-latest \
                    docker-common \
                    docker-latest \
                    docker-latest-logrotate \
                    docker-logrotate \
                    docker-engine
Loaded plugins: fastestmirror
No Match for argument: docker
No Match for argument: docker-client
No Match for argument: docker-client-latest
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
No Packages marked for removal
```

Add some utils for use docker with a persistent data and yum utils.

```
[centos@node-0 ~]$ sudo yum install -y yum-utils \device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Dependency Updated:
  device-mapper.x86_64  7:1.02.149-10.el7_6.3                    device-mapper-event.x86_64 7:1.02.149-10.el7_6.3 device-mapper-event-libs.x86_64 7:1.02.149-10.el7_6.3
  device-mapper-libs.x86_64  7:1.02.149-10.el7_6.3               lvm2-libs.x86_64  7:2.02.180-10.el7_6.3

Complete!
```

Add docker repository

```
[centos@node-0 ~]$ sudo yum-config-manager \
    –add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
```

```
/etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

Setup docker daemon.

```
[centos@node-0 ~]$ sudo -i
[root@node-0 ~]# cat > /etc/docker/daemon.json <<EOF
{
 "exec-opts": ["native.cgroupdriver=systemd"],
 "log-driver": "json-file",
 "log-opts": {
  "max-size": "100m"
 },
 "storage-driver": "overlay2",
 "storage-opts": [
  "overlay2.override_kernel_check=true"
 ]
}
EOF
```

Install Docker

```
[root@node-0 ~]# exit
[centos@manager-0 ~]$ yum update -y && yum install -y docker-ce-18.06.2.ce
....
Installed:
 docker-ce.x86_64 0:18.06.2.ce-3.el7

Dependency Installed:
 libtool-ltdl.x86_64 0:2.4.2-22.el7_3

Complete!
```

Start docker.

```
[root@node-0 ~]# systemctl daemon-reload
[root@node-0 ~]# systemctl enable –now docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@node-0 ~]# systemctl restart docker
[root@node-0 ~]# systemctl status docker
● docker.service – Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2019-03-16 14:07:48 UTC; 6s ago
    Docs: https://docs.docker.com
```

Installing Kubeadm, Kubectl and kubelet

```
[centos@node-0 ~]$ sudo -i
[root@node-0 ~]$ cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
```

```
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
exclude=kube*
EOF
```

Set SELinux in permissive mode (effectively disabling it)

```
[root@node-0 ~]$ setenforce 0
[root@node-0 ~]$ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Install kubelet, kubeadm and kubectl.

```
[root@node-0 ~]# yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Installed:
  kubeadm.x86_64 0:1.13.4-0      kubectl.x86_64 0:1.13.4-0      kubelet.x86_64 0:1.13.4-0

Dependency Installed:
 conntrack-tools.x86_64 0:1.4.4-4.el7          cri-tools.x86_64 0:1.12.0-0 ebtables.x86_64 0:2.0.10-16.el7          kubernetes-cni.x86_64 0:0.6.0-0
  libnetfilter_cthelper.x86_64 0:1.0.0-9.el7          libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7
libnetfilter_queue.x86_64 0:1.0.2-2.el7_2      socat.x86_64 0:1.7.3.2-2.el7

Complete!
```

Start kubelet service.

```
[root@node-0 ~]# systemctl enable --now kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
```

Disable the swap.

```
[root@node-0 ~]# swapoff -a
[root@node-0 ~]# sed -i '/ swap / s/^/#/' /etc/fstab
```

Preparing the node-1 instance

Installing Docker

Remove old version of docker.

```
[centos@node-1 ~]$ sudo yum remove docker \
                  . docker-client \
                    docker-client-latest \
                    docker-common \
                    docker-latest \
                    docker-latest-logrotate \
                    docker-logrotate \
                    docker-engine
Loaded plugins: fastestmirror
No Match for argument: docker
```

```
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-engine
No Packages marked for removal
```

Add some utils for use docker with a persistent data and yum utils.

```
[centos@node-1 ~]$ sudo yum install -y yum-utils \device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Dependency Updated:
 device-mapper.x86_64  7:1.02.149-10.el7_6.3                device-mapper-event.x86_64
7:1.02.149-10.el7_6.3 device-mapper-event-libs.x86_64 7:1.02.149-10.el7_6.3
 device-mapper-libs.x86_64  7:1.02.149-10.el7_6.3           lvm2-libs.x86_64  7:2.02.180-
10.el7_6.3

Complete!
```

Add docker repository

```
[centos@node-1 ~]$ sudo 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
```

Setup docker daemon.

```
[centos@node-1 ~]$ sudo -i
[root@node-1 ~]# cat > /etc/docker/daemon.json <<EOF
{
 “exec-opts”: [“native.cgroupdriver=systemd”],
 “log-driver”: “json-file”,
 “log-opts”: {
   “max-size”: “100m”
 },
 “storage-driver”: “overlay2”,
 “storage-opts”: [
   “overlay2.override_kernel_check=true”
 ]
}
EOF
```

Install Docker

```
[root@node-1 ~]# exit
[centos@manager-1 ~]$ yum update -y && yum install -y docker-ce-18.06.2.ce
....
Installed:
```

libtool-ltdl.x86_64 0:2.4.2-22.el7_3

Complete!

Start docker.

```
[root@node-1 ~]# systemctl daemon-reload
[root@node-1 ~]# systemctl enable –now docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@node-1 ~]# systemctl restart docker
[root@node-1 ~]# systemctl status docker
● docker.service – Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2019-03-16 14:07:48 UTC; 6s ago
    Docs: https://docs.docker.com
```

Installing Kubeadm, Kubectl and kubelet

```
[centos@node-1 ~]$ sudo -i
[root@node-1 ~]$ 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
exclude=kube*
EOF
```

Set SELinux in permissive mode (effectively disabling it)

```
[root@node-1 ~]$ setenforce 0
[root@node-1 ~]$ sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Install kubelet, kubeadm and kubectl.

```
[root@node-1 ~]# yum install -y kubelet kubeadm kubectl –disableexcludes=kubernetes
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
....
Installed:
  kubeadm.x86_64 0:1.13.4-0      kubectl.x86_64 0:1.13.4-0      kubelet.x86_64 0:1.13.4-0


Dependency Installed:
 conntrack-tools.x86_64 0:1.4.4-4.el7              cri-tools.x86_64 0:1.12.0-0 ebtables.x86_64 0:2.0.10-16.el7              kubernetes-cni.x86_64 0:0.6.0-0
 libnetfilter_cthelper.x86_64 0:1.0.0-9.el7          libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7
 libnetfilter_queue.x86_64 0:1.0.2-2.el7_2          socat.x86_64 0:1.7.3.2-2.el7
```

Start kubelet service.

```
[root@node-1 ~]# systemctl enable --now kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /etc/systemd/system/kubelet.service.
```

Disable the swap.

```
[root@node-1 ~]# swapoff -a
[root@node-1 ~]# sed -i '/ swap / s/^/#/' /etc/fstab
```

Installing and configuring Etcd

Installing and configuring Etcd on master-0 instance.

Create etcd folders

```
[root@master-0 ~]# exit
logout
[centos@master-0 ~]$ sudo mkdir /etc/etcd /var/lib/etcd
```

Move certificates on /etc/etcd folder.

```
[centos@master-0 ~]$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd
```

Check the last release of etcd on https://github.com/etcd-io/etcd/releases
Download the etcd binaries.
Set version of ETCD

```
[centos@master-0 ~]$ ETCD_VER=v3.3.12
```

You can choice to download from Google or Github. I will download it from Google.

```
[centos@master-0 ~]$ GOOGLE_URL=https://storage.googleapis.com/etcd

[centos@master-0 ~]$ GITHUB_URL=https://github.com/etcd-io/etcd/releases/download

[centos@master-0 ~]$ DOWNLOAD_URL=${GOOGLE_URL}
```

Clean old downloaded etcd file in /tmp

```
[centos@master-0 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
[centos@master-0 ~]$ rm -rf /tmp/etcd-download-test && mkdir -p /tmp/etcd-download-test
```

Download ETCD

```
[centos@master-0 ~]$ curl -L ${DOWNLOAD_URL}/${ETCD_VER}/etcd-${ETCD_VER}-linux-amd64.tar.gz -o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
 % Total    % Received % Xferd  Average Speed Time    Time Time Current
                                 Dload Upload Total Spent Left  Speed
 0     0 0      0 0 0       0 0 --:--:-- --:--:-- --:--:--      0tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp/etcd-download-test --strip-components=1
100 10.8M  100 10.8M  0 0 9019k      0 0:00:01 0:00:01 --:--:-- 9026k
```

Decompress downloaded ETCD in /tmp/etcd-download-test

```
etcd-v3.3.12-linux-amd64/README.md
....
etcd-v3.3.12-linux-amd64/etcd
```

Delete downloaded ETCD file.

```
[centos@master-0 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
```

List ETCD files.

```
[centos@master-0 ~]$ ls /tmp/etcd-download-test/
Documentation  etcd etcdctl README-etcdctl.md  README.md README-etcdctl.md
```

Move binary to /usr/local/bin folder.

```
[centos@master-0 ~]$ sudo mv /tmp/etcd-download-test/etcd* /usr/local/bin/
```

Check the binary version, should match the ETCD_VER variables set few step ago.

```
[centos@master-0 ~]$ etcdctl –version
etcdctl version: 3.3.12
API version: 2
```

Create a service file for systemd

```
[centos@master-0 ~]$ sudo vim /etc/systemd/system/etcd.service
```

File: /etc/systemd/system/etcd.service

```
[Unit]
Description=etcd
Documentation=https://github.com/coreos

[Service]
ExecStart=/usr/local/bin/etcd \
 –name 10.10.38.131 \
 –cert-file=/etc/etcd/kubernetes.pem \
 –key-file=/etc/etcd/kubernetes-key.pem \
 –peer-cert-file=/etc/etcd/kubernetes.pem \
 –peer-key-file=/etc/etcd/kubernetes-key.pem \
 –trusted-ca-file=/etc/etcd/ca.pem \
 –peer-trusted-ca-file=/etc/etcd/ca.pem \
 –peer-client-cert-auth \
 –client-cert-auth \
 –initial-advertise-peer-urls https://10.10.38.131:2380 \
 –listen-peer-urls https://10.10.38.131:2380 \
 –listen-client-urls https://10.10.38.131:2379,http://127.0.0.1:2379 \
 –advertise-client-urls https://10.10.38.131:2379 \
 –initial-cluster-token etcd-cluster-0 \
                                                    –initial-cluster
10.10.38.131=https://10.10.38.131:2380,10.10.12.172=https://10.10.12.172:2380,10.10.24.77=https://10.10.24.77:2380
\
 –initial-cluster-state new \
 –data-dir=/var/lib/etcd
Restart=on-failure
RestartSec=5
```

Reload the daemon configuration.

```
[centos@master-0 ~]$ sudo systemctl daemon-reload
```

Enable etcd to start at boot time.

```
[centos@master-0 ~]$ sudo systemctl enable etcd
Created symlink from /etc/systemd/system/multi-user.target.wants/etcd.service to /etc/systemd/system/etcd.service.
```

Start etcd.

```
[centos@master-0 ~]$ sudo systemctl status etcd
● etcd.service – etcd
 Loaded: loaded (/etc/systemd/system/etcd.service; enabled; vendor preset: disabled)
 Active: active (running) since Fri 2019-03-15 14:35:21 UTC; 2min 6s ago
  Docs: https://github.com/coreos
```

Installing and configuring Etcd on master-1 instance.

Create etcd folders

```
[root@master-1 ~]# exit
logout
[centos@master-1 ~]$ sudo mkdir /etc/etcd /var/lib/etcd
```

Move certificates on /etc/etcd folder.

```
[centos@master-1 ~]$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd
```

Check the last release of etcd on https://github.com/etcd-io/etcd/releases
Download the etcd binaries.
Set version of ETCD

```
[centos@master-1 ~]$ ETCD_VER=v3.3.12
```

You can choice to download from Google or Github. I will download it from Google.

```
[centos@master-1 ~]$ GOOGLE_URL=https://storage.googleapis.com/etcd

[centos@master-1 ~]$ GITHUB_URL=https://github.com/etcd-io/etcd/releases/download

[centos@master-1 ~]$ DOWNLOAD_URL=${GOOGLE_URL}
```

Clean old downloaded etcd file in /tmp

```
[centos@master-1 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
[centos@master-1 ~]$ rm -rf /tmp/etcd-download-test && mkdir -p /tmp/etcd-download-test
```

Download ETCD

```
[centos@master-1 ~]$ curl -L ${DOWNLOAD_URL}/${ETCD_VER}/etcd-${ETCD_VER}-linux-amd64.tar.gz -o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
 % Total    % Received % Xferd  Average Speed Time    Time Time Current
                                 Dload Upload Total Spent Left  Speed
  0     0 0      0 0 0       0 0 –:–:– –:–:– –:–:–      0tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp/etcd-download-test –strip-components=1
```

Decompress downloaded ETCD in /tmp/etcd-download-test

```
[centos@master-1 ~]$ tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp/etcd-download-test --strip-components=1
etcd-v3.3.12-linux-amd64/README.md
....
etcd-v3.3.12-linux-amd64/etcd
```

Delete downloaded ETCD file.

```
[centos@master-1 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
```

List ETCD files.

```
[centos@master-1 ~]$ ls /tmp/etcd-download-test/
Documentation  etcd etcdctl README-etcdctl.md  README.md README-v2-etcdctl.md
```

Move binary to /usr/local/bin folder.

```
[centos@master-1 ~]$ sudo mv /tmp/etcd-download-test/etcd* /usr/local/bin/
```

Check the binary version, should match the ETCD_VER variables set few step ago.

```
[centos@master-1 ~]$ etcdctl --version
etcdctl version: 3.3.12
API version: 2
```

Create a service file for systemd

```
[centos@master-1 ~]$ sudo vim /etc/systemd/system/etcd.service
```

File: /etc/systemd/system/etcd.service

```
[Unit]
Description=etcd
Documentation=https://github.com/coreos

[Service]
ExecStart=/usr/local/bin/etcd \
 --name 10.10.12.172 \
 --cert-file=/etc/etcd/kubernetes.pem \
 --key-file=/etc/etcd/kubernetes-key.pem \
 --peer-cert-file=/etc/etcd/kubernetes.pem \
 --peer-key-file=/etc/etcd/kubernetes-key.pem \
 --trusted-ca-file=/etc/etcd/ca.pem \
 --peer-trusted-ca-file=/etc/etcd/ca.pem \
 --peer-client-cert-auth \
 --client-cert-auth \
 --initial-advertise-peer-urls https://10.10.12.172:2380 \
 --listen-peer-urls https://10.10.12.172:2380 \
 --listen-client-urls https://10.10.12.172:2379,http://127.0.0.1:2379 \
 --advertise-client-urls https://10.10.12.172:2379 \
 --initial-cluster-token etcd-cluster-0 \
                                                    --initial-cluster
10.10.38.131=https://10.10.38.131:2380,10.10.12.172=https://10.10.12.172:2380,10.10.24.77=https://10.10.24.77:2380
\
 --initial-cluster-state new \
 --data-dir=/var/lib/etcd
```

```
[Install]
WantedBy=multi-user.target
```

Reload the daemon configuration.

```
[centos@master-1 ~]$ sudo systemctl daemon-reload
```

Enable etcd to start at boot time.

```
[centos@master-1 ~]$ sudo systemctl enable etcd
Created symlink from /etc/systemd/system/multi-user.target.wants/etcd.service to /etc/systemd/system/etcd.service.
```

Start etcd.

```
[centos@master-1 ~]$ sudo systemctl status etcd
● etcd.service – etcd
  Loaded: loaded (/etc/systemd/system/etcd.service; enabled; vendor preset: disabled)
  Active: active (running) since Fri 2019-03-15 16:50:12 UTC; 12s ago
    Docs: https://github.com/coreos
```

Installing and configuring Etcd on master-2 instance.

Create etcd folders

```
[root@master-2 ~]# exit
logout
[centos@master-2 ~]$ sudo mkdir /etc/etcd /var/lib/etcd
```

Move certificates on /etc/etcd folder.

```
[centos@master-2 ~]$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd
```

Check the last release of etcd on https://github.com/etcd-io/etcd/releases
Download the etcd binaries.
Set version of ETCD

```
[centos@master-2 ~]$ ETCD_VER=v3.3.12
```

You can choice to download from Google or Github. I will download it from Google.

```
[centos@master-2 ~]$ GOOGLE_URL=https://storage.googleapis.com/etcd

[centos@master-2 ~]$ GITHUB_URL=https://github.com/etcd-io/etcd/releases/download

[centos@master-2 ~]$ DOWNLOAD_URL=${GOOGLE_URL}
```

Clean old downloaded etcd file in /tmp

```
[centos@master-2 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
[centos@master-2 ~]$ rm -rf /tmp/etcd-download-test && mkdir -p /tmp/etcd-download-test
```

Download ETCD

```
[centos@master-2 ~]$ curl -L ${DOWNLOAD_URL}/${ETCD_VER}/etcd-${ETCD_VER}-linux-amd64.tar.gz -o /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

```
amd64.tar.gz -C /tmp/etcd-download-test –strip-components=1
100 10.8M  100 10.8M  0 0 9019k      0 0:00:01 0:00:01 –:–:– 9026k
```

Decompress downloaded ETCD in /tmp/etcd-download-test

```
[centos@master-2 ~]$ tar xzvf /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz -C /tmp/etcd-download-test –strip-components=1
etcd-v3.3.12-linux-amd64/README.md
….
etcd-v3.3.12-linux-amd64/etcd
```

Delete downloaded ETCD file.

```
[centos@master-2 ~]$ rm -f /tmp/etcd-${ETCD_VER}-linux-amd64.tar.gz
```

List ETCD files.

```
[centos@master-2 ~]$ ls /tmp/etcd-download-test/
Documentation  etcd etcdctl README-etcdctl.md  README.md README-v2-etcdctl.md
```

Move binary to /usr/local/bin folder.

```
[centos@master-2 ~]$ sudo mv /tmp/etcd-download-test/etcd* /usr/local/bin/
```

Check the binary version, should match the ETCD_VER variables set few step ago.

```
[centos@master-2 ~]$ etcdctl –version
etcdctl version: 3.3.12
API version: 2
```

Create a service file for systemd

```
[centos@master-2 ~]$ sudo vim /etc/systemd/system/etcd.service
```

File: /etc/systemd/system/etcd.service

```
[Unit]
Description=etcdDocumentation=https://github.com/coreos
[Service]
ExecStart=/usr/local/bin/etcd \
–name 10.10.24.77 \
–cert-file=/etc/etcd/kubernetes.pem \
–key-file=/etc/etcd/kubernetes-key.pem \
 –peer-cert-file=/etc/etcd/kubernetes.pem \
 –peer-key-file=/etc/etcd/kubernetes-key.pem \
 –trusted-ca-file=/etc/etcd/ca.pem \
 –peer-trusted-ca-file=/etc/etcd/ca.pem \
 –peer-client-cert-auth \
 –client-cert-auth \
 –initial-advertise-peer-urls https://10.10.24.77:2380 \
 –listen-peer-urls https://10.10.24.77:2380 \
 –listen-client-urls https://10.10.24.77:2379,http://127.0.0.1:2379 \
 –advertise-client-urls https://10.10.24.77:2379 \
 –initial-cluster-token etcd-cluster-0 \
                                                    –initial-cluster
10.10.38.131=https://10.10.38.131:2380,10.10.12.172=https://10.10.12.172:2380,10.10.24.77=https://10.10.24.77:2380
\
```

```
[Install]
WantedBy=multi-user.target
```

Reload the daemon configuration.

```
[centos@master-0 ~]$ sudo systemctl daemon-reload
```

Enable etcd to start at boot time.

```
[centos@master-0 ~]$ sudo systemctl enable etcd
Created symlink from /etc/systemd/system/multi-user.target.wants/etcd.service to /etc/systemd/system/etcd.service.
```

Start etcd.

```
[centos@master-0 ~]$ sudo systemctl status etcd
● etcd.service – etcd Loaded: loaded (/etc/systemd/system/etcd.service; enabled; vendor preset: disabled) Active: active (running) since Fri 2019-03-15 16:56:57 UTC; 2s ago Docs: https://github.com/coreos
```

Verify if the cluster is up and running

```
[centos@master-2 ~]$ ETCDCTL_API=3 etcdctl member list
697203cad16ef50c, started, 10.10.12.172, https://10.10.12.172:2380, https://10.10.12.172:2379
7c8847fed53db543, started, 10.10.24.77, https://10.10.24.77:2380, https://10.10.24.77:2379
d34af8f4bc6b02e0, started, 10.10.38.131, https://10.10.38.131:2380, https://10.10.38.131:2379
```

Initializing the master nodes

Initialize master-0 instance as a master node

Create the configuration file for kubeadm.

```
[centos@master-0 ~]$ vim config.yaml
```

File: /home/centos/config.yaml

```
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: stable
apiServerCertSANs:
– 10.10.6.99
controlPlaneEndpoint: “10.10.6.99:6443”
etcd:
 external:
  endpoints:
  – https://10.10.38.131:2379
  – https://10.10.12.172:2379
  – https://10.10.24.77:2379
  caFile: /etc/etcd/ca.pem
  certFile: /etc/etcd/kubernetes.pem
  keyFile: /etc/etcd/kubernetes-key.pem
 networking:
 podSubnet: 10.30.0.0/24
apiServerExtraArgs:
```

Change system parameter for pass bridged IPv4 traffic to iptables chains.

```
[centos@master-0 ~]$ sudo -i
[root@master-0 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@master-0 ~]# exit
logout
```

Initialize the master-0 instance as a master node.

```
[centos@master-0 ~]$ sudo kubeadm init –config=config.yaml
[init] Using Kubernetes version: v1.13.4
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] External etcd mode: Skipping etcd/ca certificate authority generation
[certs] External etcd mode: Skipping etcd/peer certificate authority generation
[certs] External etcd mode: Skipping etcd/healthcheck-client certificate authority generation
[certs] External etcd mode: Skipping etcd/server certificate authority generation
[certs] External etcd mode: Skipping apiserver-etcd-client certificate authority generation
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [master-0 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.10.38.131 10.10.6.99 54.171.193.153]
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 18.004697 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the configuration for the kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "master-0" as an annotation
```

[mark control-plane] Marking the node master-0 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: 9le6nf.btptxk0b2auud87t
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[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 10.10.6.99:6443 –token 9le6nf.btptxk0b2auud87t –discovery-token-ca-cert-hash
sha256:9eb8d9b1957f984ad6d921642229f81a061d793ff56eeffab17fdabcc3a6a440

Copy the certificates to the master-1 instance.

```
[centos@master-0 ~]$ sudo scp -r /etc/kubernetes/pki centos@10.10.12.172:~
front-proxy-ca.key                    100% 1679   637.7KB/s   00:00
front-proxy-ca.crt                    100% 1038   737.3KB/s   00:00
front-proxy-client.key                100% 1679     1.1MB/s   00:00
front-proxy-client.crt                100% 1058   680.8KB/s   00:00
ca.key                                100% 1679     1.2MB/s   00:00
ca.crt                                100% 1025   764.4KB/s   00:00
apiserver-kubelet-client.key          100% 1675     1.3MB/s   00:00
apiserver-kubelet-client.crt          100% 1099   840.7KB/s   00:00
apiserver.key                         100% 1675     1.2MB/s   00:00
apiserver.crt                         100% 1241   866.7KB/s   00:00
sa.key                                100% 1675     1.2MB/s   00:00
sa.pub                                100%  451   359.4KB/s   00:00
```

Copy the certificates to the master-2 instance.

```
[centos@master-0 ~]$ sudo scp -r /etc/kubernetes/pki centos@10.10.24.77:~
front-proxy-ca.key                    100% 1679   637.7KB/s   00:00
front-proxy-ca.crt                    100% 1038   737.3KB/s   00:00
```

```
ca.key                        100% 1679 1.2MB/s 00:00
ca.crt                        100% 1025 764.4KB/s 00:00
apiserver-kubelet-client.key  100% 1675 1.3MB/s 00:00
apiserver-kubelet-client.crt  100% 1099 840.7KB/s 00:00
apiserver.key                 100% 1675 1.2MB/s 00:00
apiserver.crt                 100% 1241 866.7KB/s 00:00
sa.key                        100% 1675 1.2MB/s 00:00
sa.pub                        100% 451 359.4KB/s 00:00
```

Initialize the master-1 instance as a master node.

Come back to user centos

```
[root@master-1 ~]# exit
logout
```

Remove the apiserver.crt and apiserver.key.

```
[centos@master-1 ~]$ rm ~/pki/apiserver.*
```

Move the certificates to the /etc/kubernetes directory.

```
[centos@master-1 ~]$ sudo mv ~/pki /etc/kubernetes/
```

Create the configuration file for kubeadm.

```
[centos@master-1 ~]$ vim config.yaml
```

File: /home/centos/config.yaml

```
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: stable
apiServerCertSANs:
– 10.10.6.99
controlPlaneEndpoint: “10.10.6.99:6443”
etcd:
 external:
  endpoints:
  – https://10.10.38.131:2379
  – https://10.10.12.172:2379
  – https://10.10.24.77:2379
  caFile: /etc/etcd/ca.pem
  certFile: /etc/etcd/kubernetes.pem
  keyFile: /etc/etcd/kubernetes-key.pem
networking:
 podSubnet: 10.30.0.0/24
apiServerExtraArgs:
 apiserver-count: “3”
```

Change system parameter for pass bridged IPv4 traffic to iptables chains.

```
[centos@master-1 ~]$ sudo -i
[root@master-1 ~]# echo ‘1’ > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@master-1 ~]# exit
logout
```

Initialize the machine as a master node.

[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Using existing ca certificate authority
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [master-1 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.10.12.172 10.10.6.99 10.10.6.99]
[certs] Using existing apiserver-kubelet-client certificate and key on disk
[certs] Using existing front-proxy-ca certificate authority
[certs] Using existing front-proxy-client certificate and key on disk
[certs] External etcd mode: Skipping etcd/ca certificate authority generation
[certs] External etcd mode: Skipping etcd/peer certificate authority generation
[certs] External etcd mode: Skipping etcd/healthcheck-client certificate authority generation
[certs] External etcd mode: Skipping apiserver-etcd-client certificate authority generation
[certs] External etcd mode: Skipping etcd/server certificate authority generation
[certs] Using the existing "sa" key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 0.016893 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the configuration for the kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "master-1" as an annotation
[mark-control-plane] Marking the node master-1 as control-plane by adding the label "node-role.kubernetes.io/master=""
[mark-control-plane] Marking the node master-1 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: bidfhz.tb700xydj8rz1xjk
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

```
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[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 10.10.6.99:6443 –token bidfhz.tb700xydj8rz1xjk –discovery-token-ca-cert-hash
sha256:9eb8d9b1957f984ad6d921642229f81a061d793ff56eeffab17fdabcc3a6a440
```

Initialize the master-2 instance as a master node.

Come back to user centos

```
[root@master-2 ~]# exit
logout
```

Remove the apiserver.crt and apiserver.key.

```
[centos@master-2 ~]$ rm ~/pki/apiserver.*
```

Move the certificates to the /etc/kubernetes directory.

```
[centos@master-2 ~]$ sudo mv ~/pki /etc/kubernetes/
```

Create the configuration file for kubeadm.

```
[centos@master-2 ~]$ vim config.yaml
```

File: /home/centos/config.yaml

```
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: stable
apiServerCertSANs:
– 10.10.6.99
controlPlaneEndpoint: "10.10.6.99:6443"
etcd:
 external:
  endpoints:
  – https://10.10.38.131:2379
  – https://10.10.12.172:2379
  – https://10.10.24.77:2379
  caFile: /etc/etcd/ca.pem
```

```
 podSubnet: 10.30.0.0/24
apiServerExtraArgs:
 apiserver-count: "3"
```

Change system parameter for pass bridged IPv4 traffic to iptables chains.

```
[centos@master-2 ~]$ sudo -i
[root@master-2 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@master-2 ~]# exit
logout
```

Initialize the machine as a master node.

```
[centos@master-2 ~]$ sudo kubeadm init –config=config.yaml
[init] Using Kubernetes version: v1.13.4
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Using existing ca certificate authority
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [master-2 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.10.24.77 10.10.6.99 10.10.6.99]
[certs] Using existing apiserver-kubelet-client certificate and key on disk
[certs] Using existing front-proxy-ca certificate authority
[certs] Using existing front-proxy-client certificate and key on disk
[certs] External etcd mode: Skipping etcd/ca certificate authority generation
[certs] External etcd mode: Skipping etcd/server certificate authority generation
[certs] External etcd mode: Skipping etcd/peer certificate authority generation
[certs] External etcd mode: Skipping etcd/healthcheck-client certificate authority generation
[certs] External etcd mode: Skipping apiserver-etcd-client certificate authority generation
[certs] Using the existing "sa" key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 0.014372 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
```

[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "master-2" as an annotation
[mark-control-plane] Marking the node master-2 as control-plane by adding the label "node-role.kubernetes.io/master=""
[mark-control-plane] Marking the node master-2 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: jtei7y.yq956cpqbru34pqh
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[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 10.10.6.99:6443 –token jtei7y.yq956cpqbru34pqh –discovery-token-ca-cert-hash
sha256:9eb8d9b1957f984ad6d921642229f81a061d793ff56eeffab17fdabcc3a6a440

Copy the "kubeadm join" command line printed as the result of the previous command.

Initializing the worker nodes

Initializing the worker node node-0

Change system parameter for pass bridged IPv4 traffic to iptables chains.

```
[root@node-0 ~]# sudo -i
[root@node-0 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

[root@node-0 ~]# exit
logout
```

Execute the "kubeadm join" command that you copied from the last step of the initialization of the masters.

```
[root@node-0 ~]# kubeadm join 10.10.6.99:6443 –token jtei7y.yq956cpqbru34pqh –discovery-token-ca-cert-hash
```

```
[discovery] Created cluster-info discovery client, requesting info from "https://10.10.6.99:6443"
[discovery] Requesting info from "https://10.10.6.99:6443" 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 "10.10.6.99:6443"
[discovery] Successfully established connection with API Server "10.10.6.99:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.13" ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] 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 the Node API object "node-0" 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.
```

Initializing the worker node node-1

Change system parameter for pass bridged IPv4 traffic to iptables chains.

```
[root@node-0 ~]# sudo -i
[root@node-0 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

[root@node-0 ~]# exit
logout
```

Execute the "kubeadm join" command that you copied from the last step of the initialization of the masters.

```
[centos@node-1 ~]$ sudo kubeadm join 10.10.6.99:6443 –token jtei7y.yq956cpqbru34pqh –discovery-token-ca-cert-hash sha256:9eb8d9b1957f984ad6d921642229f81a061d793ff56eeffab17fdabcc3a6a440
[preflight] Running pre-flight checks
[discovery] Trying to connect to API Server "10.10.6.99:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://10.10.6.99:6443"
[discovery] Requesting info from "https://10.10.6.99:6443" 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 "10.10.6.99:6443"
[discovery] Successfully established connection with API Server "10.10.6.99:6443"
[join] Reading configuration from the cluster...
[join] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.13"
```

```
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] 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 the Node API object "node-1" 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.
```

Verifying that the workers joined the cluster

Go on master-0 node

```
[centos@master-0 ~]$ sudo kubectl –kubeconfig /etc/kubernetes/admin.conf get nodes
NAME       STATUS ROLES    AGE VERSION
master-0   NotReady master   68m v1.13.4
master-1   NotReady master   15m v1.13.4
master-2   NotReady master   9m50s v1.13.4
node-0     NotReady <none>   6m59s v1.13.4
node-1     NotReady <none>   89s v1.13.4
```

The status of the nodes is NotReady as we haven't configured the networking overlay yet.

Configuring kubectl on the client machine

In a master-0 instance change permission to the admin.conf file.

```
[centos@master-0 ~]$ sudo chmod +r /etc/kubernetes/admin.conf
```

From loadbalancer instance copy the file admin.conf

```
[centos@loadbalancer ~]$ scp centos@10.10.38.131:/etc/kubernetes/admin.conf .
admin.conf          5446 3.8MB/s 00:00
```

Create the kubectl configuration directory.

```
[centos@loadbalancer ~]$ mkdir ~/.kube
```

Move the configuration file to the configuration directory.

```
[centos@loadbalancer ~]$ mv admin.conf ~/.kube/config
```

Modify the permissions of the configuration file.

```
[centos@loadbalancer ~]$ chmod 600 ~/.kube/config
```

Go back to master-0 instance and change back the permissions of the configuration file.

```
[centos@master-0 ~]$ sudo chmod 600 /etc/kubernetes/admin.conf
```

Check that you can access the Kubernetes API from the client machine.

```
[centos@loadbalancer ~]$ kubectl get nodes
NAME       STATUS ROLES    AGE VERSION
master-0   NotReady master   84m v1.13.4
master-1   NotReady master   32m v1.13.4
master-2   NotReady master   26m v1.13.4
```

Deploying the overlay network

We are going to use Weavenet as the overlay network. You can also use static route or another overlay network tool like Calico or Flannel. Please see https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#pod-network
Deploy the overlay network pods from the client machine.

```
[centos@loadbalancer ~]$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.extensions/weave-net created
```

Check that the pods are deployed properly.

```
[centos@loadbalancer ~]$ kubectl get pods -n kube-system
NAME                          READY STATUS RESTARTS AGE
coredns-86c58d9df4-vp2b5         1/1 Running 0 87m
coredns-86c58d9df4-x2vdd         1/1 Running 0 87m
kube-apiserver-master-0          1/1 Running 0 88m
kube-apiserver-master-1          1/1 Running 0 37m
kube-apiserver-master-2          1/1 Running 0 30m
kube-controller-manager-master-0   1/1 Running 0 86m
kube-controller-manager-master-1   1/1 Running 0 37m
kube-controller-manager-master-2   1/1 Running 0 31m
kube-proxy-2d7p2                 1/1 Running 0 37m
kube-proxy-8k5hs                 1/1 Running 0 87m
kube-proxy-9rmfm                 1/1 Running 0 31m
kube-proxy-jt49g                 1/1 Running 0 22m
kube-proxy-s74tj                 1/1 Running 0 28m
kube-scheduler-master-0          1/1 Running 0 88m
kube-scheduler-master-1          1/1 Running 0 37m
kube-scheduler-master-2          1/1 Running 0 30m
weave-net-876js                  2/2 Running 0 21s
weave-net-bgnvp                  2/2 Running 0 21s
weave-net-c649n                  2/2 Running 0 21s
weave-net-svspk                  2/2 Running 0 21s
weave-net-vr4qn                  2/2 Running 0 21s
```

Check that the nodes are in Ready state.

```
[centos@loadbalancer ~]$ kubectl get nodes
NAME      STATUS ROLES    AGE VERSION
master-0   Ready master   90m v1.13.4
master-1   Ready master   37m v1.13.4
master-2   Ready master   31m v1.13.4
node-0     Ready <none>   28m v1.13.4
node-1     Ready <none>   23m v1.13.4
```

Docker Kubernetes Linux 1.13 centos containers docker kubeadm kubernetes multimaster multinode