<https://blog.inkubate.io/install-and-configure-a-multi-master-kubernetes-cluster-with-kubeadm/>

# Prerequisites

For this lab, we will use a standard Ubuntu 16.04 installation as a base image for the seven machines needed. The machines will all be configured on the same network, 10.10.40.0/24, and this network needs to have access to the Internet.

The first machine needed is the machine on which the HAProxy load balancer will be installed. We will assign the IP 10.10.40.93 to this machine.

We also need three Kubernetes master nodes. These machines will have the IPs 10.10.40.90, 10.10.40.91, and 10.10.40.92.

Finally, we will also have three Kubernetes worker nodes with the IPs 10.10.40.100, 10.10.40.101, and 10.10.40.102.

We also need an IP range for the pods. This range will be 10.30.0.0/16, but it is only internal to Kubernetes.

I will use my Linux desktop as a client machine to generate all the necessary certificates, but also to manage the Kubernetes cluster. If you don't have a Linux desktop, you can use the HAProxy machine to do the same thing.

# Installing the client tools

We will need two tools on the client machine: the Cloud Flare SSL tool to generate the different certificates, and the Kubernetes client, kubectl, to manage the Kubernetes cluster.

## Installing cfssl

1- Download the binaries.

$ wget https://pkg.cfssl.org/R1.2/cfssl\_linux-amd64

$ wget https://pkg.cfssl.org/R1.2/cfssljson\_linux-amd64

2- Add the execution permission to the binaries.

$ chmod +x cfssl\*

3- Move the binaries to /usr/local/bin.

$ sudo mv cfssl\_linux-amd64 /usr/local/bin/cfssl

$ sudo mv cfssljson\_linux-amd64 /usr/local/bin/cfssljson

4- Verify the installation.

$ cfssl version

## Installing kubectl

1- Download the binary.

$ wget <https://storage.googleapis.com/kubernetes->release/release/v1.12.1/bin/linux/amd64/kubectl

2- Add the execution permission to the binary.

$ chmod +x kubectl

3- Move the binary to /usr/local/bin.

$ sudo mv kubectl /usr/local/bin

4- Verify the installation.

$ kubectl version

# 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.

1- SSH to the 10.10.40.93 Ubuntu machine.

2- Update the machine.

$ sudo apt-get update

$ sudo apt-get upgrade

3- Install HAProxy.

$ sudo apt-get install haproxy

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

$ sudo vim /etc/haproxy/haproxy.cfg

global

...

default

...

frontend kubernetes

bind 10.10.40.93:6443

option tcplog

mode tcp

default\_backend kubernetes-master-nodes

backend kubernetes-master-nodes

mode tcp

balance roundrobin

option tcp-check

server k8s-master-0 10.10.40.90:6443 check fall 3 rise 2

server k8s-master-1 10.10.40.91:6443 check fall 3 rise 2

server k8s-master-2 10.10.40.92:6443 check fall 3 rise 2

5- Restart HAProxy.

$ sudo systemctl restart haproxy

# Generating the TLS certificates

These steps can be done on your Linux desktop if you have one or on the HAProxy machine depending on where you installed the cfssl tool.

## Creating a certificate authority

1- Create the certificate authority configuration file.

$ vim ca-config.json

{

"signing": {

"default": {

"expiry": "8760h"

},

"profiles": {

"kubernetes": {

"usages": ["signing", "key encipherment", "server auth", "client auth"],

"expiry": "8760h"

}

}

}

}

2- Create the certificate authority signing request configuration file.

$ vim ca-csr.json

{

"CN": "Kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "IE",

"L": "Cork",

"O": "Kubernetes",

"OU": "CA",

"ST": "Cork Co."

}

]

}

3- Generate the certificate authority certificate and private key.

$ cfssl gencert -initca ca-csr.json | cfssljson -bare ca

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

$ ls -la

## Creating the certificate for the Etcd cluster

1- Create the certificate signing request configuration file.

$ vim kubernetes-csr.json

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "IE",

"L": "Cork",

"O": "Kubernetes",

"OU": "Kubernetes",

"ST": "Cork Co."

}

]

}

2- Generate the certificate and private key.

$ cfssl gencert \

-ca=ca.pem \

-ca-key=ca-key.pem \

-config=ca-config.json \

-hostname=10.10.40.90,10.10.40.91,10.10.40.92,10.10.40.93,127.0.0.1,kubernetes.default \

-profile=kubernetes kubernetes-csr.json | \

cfssljson -bare kubernetes

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

$ ls -la

4- Copy the certificate to each nodes.

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.90:~

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.91:~

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.92:~

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.100:~

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.101:~

$ scp ca.pem kubernetes.pem kubernetes-key.pem sguyennet@10.10.40.102:~

# Preparing the nodes for kubeadm

## Preparing the 10.10.40.90 machine

### Installing Docker

1- SSH to the 10.10.40.90 machine.

$ ssh sguyennet@10.10.40.90

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## Preparing the 10.10.40.91 machine

### Installing Docker

1- SSH to the 10.10.40.91 machine.

$ ssh sguyennet@10.10.40.91

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## Preparing the 10.10.40.92 machine

### Installing Docker

1- SSH to the 10.10.40.92 machine.

$ ssh sguyennet@10.10.40.92

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## Preparing the 10.10.40.100 machine

### Installing Docker

1- SSH to the 10.10.40.100 machine.

$ ssh sguyennet@10.10.40.100

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## Preparing the 10.10.40.101 machine

### Installing Docker

1- SSH to the 10.10.40.101 machine.

$ ssh sguyennet@10.10.40.101

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

## Preparing the 10.10.40.102 machine

### Installing Docker

1- SSH to the 10.10.40.102 machine.

$ ssh sguyennet@10.10.40.102

2- Get administrator privileges.

$ sudo -s

3- Add the Docker repository key.

# curl -fsSL https://[download.docker.com/linux/ubuntu/gpg](http://download.docker.com/linux/ubuntu/gpg) | apt-key add -

4- Add the Docker repository.

# add-apt-repository \

"deb <https://download.docker.com/linux/>$(. /etc/os-release; echo "$ID") \

$(lsb\_release -cs) \

stable"

5- Update the list of packages.

# apt-get update

6- Install Docker 17.03.

# apt-get install -y docker-ce=$(apt-cache madison docker-ce | grep 17.03 | head -1 | awk '{print $3}')

### Installing kubeadm, kublet, and kubectl

1- Add the Google repository key.

# curl -s https://[packages.cloud.google.com/apt/doc/apt-key.gpg](http://packages.cloud.google.com/apt/doc/apt-key.gpg) | apt-key add -

2- Add the Google repository.

# vim /etc/apt/sources.list.d/kubernetes.list

deb <http://apt.kubernetes.io> kubernetes-xenial main

3- Update the list of packages.

# apt-get update

4- Install kubelet, kubeadm and kubectl.

# apt-get install kubelet kubeadm kubectl

5- Disable the swap.

# swapoff -a

# sed -i '/ swap / s/^/#/' /etc/fstab

# Installing and configuring Etcd

## Installing and configuring Etcd on the 10.10.40.90 machine

1- SSH to the 10.10.40.90 machine.

$ ssh sguyennet@10.10.40.90

2- Create a configuration directory for Etcd.

$ sudo mkdir /etc/etcd /var/lib/etcd

3- Move the certificates to the configuration directory.

$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd

4- Download the etcd binaries.

$ wget https://[github.com/coreos](http://github.com/coreos)/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz

5- Extract the etcd archive.

$ tar xvzf etcd-v3.3.9-linux-amd64.tar.gz

6- Move the etcd binaries to /usr/local/bin.

$ sudo mv etcd-v3.3.9-linux-amd64/etcd\* /usr/local/bin/

7- Create an etcd systemd unit file.

$ sudo vim /etc/systemd/system/etcd.service

[Unit]

Description=etcd

Documentation=https://[github.com/coreos](http://github.com/coreos)

[Service]

ExecStart=/usr/local/bin/etcd \

--name 10.10.40.90 \

--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://10.10.40.90:2380 \

--listen-peer-urls https://10.10.40.90:2380 \

--listen-client-urls https://10.10.40.90:2379,http://127.0.0.1:2379 \

--advertise-client-urls https://10.10.40.90:2379 \

--initial-cluster-token etcd-cluster-0 \

--initial-cluster 10.10.40.90=https://10.10.40.90:2380,10.10.40.91=https://10.10.40.91:2380,10.10.40.92=https://10.10.40.92:2380 \

--initial-cluster-state new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

8- Reload the daemon configuration.

$ sudo systemctl daemon-reload

9- Enable etcd to start at boot time.

$ sudo systemctl enable etcd

10- Start etcd.

$ sudo systemctl start etcd

## Installing and configuring Etcd on the 10.10.40.91 machine

1- SSH to the 10.10.40.91 machine.

$ ssh sguyennet@10.10.40.91

2- Create a configuration directory for Etcd.

$ sudo mkdir /etc/etcd /var/lib/etcd

3- Move the certificates to the configuration directory.

$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd

4- Download the etcd binaries.

$ wget https://[github.com/coreos](http://github.com/coreos)/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz

5- Extract the etcd archive.

$ tar xvzf etcd-v3.3.9-linux-amd64.tar.gz

6- Move the etcd binaries to /usr/local/bin.

$ sudo mv etcd-v3.3.9-linux-amd64/etcd\* /usr/local/bin/

7- Create an etcd systemd unit file.

$ sudo vim /etc/systemd/system/etcd.service

[Unit]

Description=etcd

Documentation=https://[github.com/coreos](http://github.com/coreos)

[Service]

ExecStart=/usr/local/bin/etcd \

--name 10.10.40.91 \

--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://10.10.40.91:2380 \

--listen-peer-urls https://10.10.40.91:2380 \

--listen-client-urls https://10.10.40.91:2379,http://127.0.0.1:2379 \

--advertise-client-urls https://10.10.40.91:2379 \

--initial-cluster-token etcd-cluster-0 \

--initial-cluster 10.10.40.90=https://10.10.40.90:2380,10.10.40.91=https://10.10.40.91:2380,10.10.40.92=https://10.10.40.92:2380 \

--initial-cluster-state new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

8- Reload the daemon configuration.

$ sudo systemctl daemon-reload

9- Enable etcd to start at boot time.

$ sudo systemctl enable etcd

10- Start etcd.

$ sudo systemctl start etcd

## Installing and configuring Etcd on the 10.10.40.92 machine

1- SSH to the 10.10.40.92 machine.

$ ssh sguyennet@10.10.40.92

2- Create a configuration directory for Etcd.

$ sudo mkdir /etc/etcd /var/lib/etcd

3- Move the certificates to the configuration directory.

$ sudo mv ~/ca.pem ~/kubernetes.pem ~/kubernetes-key.pem /etc/etcd

4- Download the etcd binaries.

$ wget https://[github.com/coreos](http://github.com/coreos)/etcd/releases/download/v3.3.9/etcd-v3.3.9-linux-amd64.tar.gz

5- Extract the etcd archive.

$ tar xvzf etcd-v3.3.9-linux-amd64.tar.gz

6- Move the etcd binaries to /usr/local/bin.

$ sudo mv etcd-v3.3.9-linux-amd64/etcd\* /usr/local/bin/

7- Create an etcd systemd unit file.

$ sudo vim /etc/systemd/system/etcd.service

[Unit]

Description=etcd

Documentation=https://[github.com/coreos](http://github.com/coreos)

[Service]

ExecStart=/usr/local/bin/etcd \

--name 10.10.40.92 \

--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://10.10.40.92:2380 \

--listen-peer-urls https://10.10.40.92:2380 \

--listen-client-urls https://10.10.40.92:2379,http://127.0.0.1:2379 \

--advertise-client-urls https://10.10.40.92:2379 \

--initial-cluster-token etcd-cluster-0 \

--initial-cluster 10.10.40.90=https://10.10.40.90:2380,10.10.40.91=https://10.10.40.91:2380,10.10.40.92=https://10.10.40.92:2380 \

--initial-cluster-state new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

8- Reload the daemon configuration.

$ sudo systemctl daemon-reload

9- Enable etcd to start at boot time.

$ sudo systemctl enable etcd

10- Start etcd.

$ sudo systemctl start etcd

11- Verify that the cluster is up and running.

$ ETCDCTL\_API=3 etcdctl member list

# Initializing the master nodes

## Initializing the 10.10.40.90 master node

1- SSH to the 10.10.40.90 machine.

$ ssh sguyennet@10.10.40.90

2- Create the configuration file for kubeadm.

$ vim config.yaml

apiVersion: [kubeadm.k8s.io/v1alpha3](http://kubeadm.k8s.io/v1alpha3)

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.40.93

controlPlaneEndpoint: "10.10.40.93:6443"

etcd:

external:

endpoints:

- https://10.10.40.90:2379

- https://10.10.40.91:2379

- https://10.10.40.92:2379

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"

3- Initialize the machine as a master node.

$ sudo kubeadm init --config=config.yaml

4- Copy the certificates to the two other masters.

$ sudo scp -r /etc/kubernetes/pki sguyennet@10.10.40.91:~

$ sudo scp -r /etc/kubernetes/pki sguyennet@10.10.40.92:~

## Initializing the 10.10.40.91 master node

1- SSH to the 10.10.40.91 machine.

$ ssh sguyennet@10.10.40.91

2- Remove the apiserver.crt and apiserver.key.

$ rm ~/pki/apiserver.\*

3- Move the certificates to the /etc/kubernetes directory.

$ sudo mv ~/pki /etc/kubernetes/

4 - Create the configuration file for kubeadm.

$ vim config.yaml

apiVersion: [kubeadm.k8s.io/v1alpha3](http://kubeadm.k8s.io/v1alpha3)

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.40.93

controlPlaneEndpoint: "10.10.40.93:6443"

etcd:

external:

endpoints:

- https://10.10.40.90:2379

- https://10.10.40.91:2379

- https://10.10.40.92:2379

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"

5- Initialize the machine as a master node.

$ sudo kubeadm init --config=config.yaml

## Initializing the 10.10.40.92 master node

1- SSH to the 10.10.40.92 machine.

$ ssh sguyennet@10.10.40.92

2- Remove the apiserver.crt and apiserver.key.

$ rm ~/pki/apiserver.\*

3- Move the certificates to the /etc/kubernetes directory.

$ sudo mv ~/pki /etc/kubernetes/

4 - Create the configuration file for kubeadm.

$ vim config.yaml

apiVersion: [kubeadm.k8s.io/v1alpha3](http://kubeadm.k8s.io/v1alpha3)

kind: ClusterConfiguration

kubernetesVersion: stable

apiServerCertSANs:

- 10.10.40.93

controlPlaneEndpoint: "10.10.40.93:6443"

etcd:

external:

endpoints:

- https://10.10.40.90:2379

- https://10.10.40.91:2379

- https://10.10.40.92:2379

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"

5- Initialize the machine as a master node.

$ sudo kubeadm init --config=config.yaml

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

# Initializing the worker nodes

## Initializing the 10.10.40.100 worker node

1- SSH to the 10.10.40.100 machine.

$ ssh sguyennet@10.10.40.100

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

$ sudo kubeadm join 10.10.40.93:6443 --token [your\_token] --discovery-token-ca-cert-hash sha256:[your\_token\_ca\_cert\_hash]

## Initializing the 10.10.40.101 worker node

1- SSH to the 10.10.40.101 machine.

$ ssh sguyennet@10.10.40.101

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

$ sudo kubeadm join 10.10.40.93:6443 --token [your\_token] --discovery-token-ca-cert-hash sha256:[your\_token\_ca\_cert\_hash]

## Initializing the 10.10.40.102 worker node

1- SSH to the 10.10.40.102 machine.

$ ssh sguyennet@10.10.40.102

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

$ sudo kubeadm join 10.10.40.93:6443 --token [your\_token] --discovery-token-ca-cert-hash sha256:[your\_token\_ca\_cert\_hash]

# Verifying that the workers joined the cluster

1- SSH to one of the master node.

$ ssh sguyennet@10.10.40.90

2- Get the list of the nodes.

$ sudo kubectl --kubeconfig /etc/kubernetes/admin.conf get nodes

NAME STATUS ROLES AGE VERSION

k8s-kubeadm-master-0 NotReady master 1h v1.12.1

k8s-kubeadm-master-1 NotReady master 1h v1.12.1

k8s-kubeadm-master-2 NotReady master 1h v1.12.1

k8s-kubeadm-worker-0 NotReady 2m v1.12.1

k8s-kubeadm-worker-1 NotReady 1m v1.12.1

k8s-kubeadm-worker-2 NotReady 1m v1.12.1

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

# Configuring kubectl on the client machine

1- SSH to one of the master node.

$ ssh sguyennet@10.10.40.90

2- Add permissions to the admin.conf file.

$ sudo chmod +r /etc/kubernetes/admin.conf

3- From the client machine, copy the configuration file.

$ scp sguyennet@10.10.40.90:/etc/kubernetes/admin.conf .

4- Create the kubectl configuration directory.

$ mkdir ~/.kube

5- Move the configuration file to the configuration directory.

$ mv admin.conf ~/.kube/config

6- Modify the permissions of the configuration file.

$ chmod 600 ~/.kube/config

7- Go back to the SSH session on the master and change back the permissions of the configuration file.

$ sudo chmod 600 /etc/kubernetes/admin.conf

8- check that you can access the Kubernetes API from the client machine.

$ kubectl get nodes

# 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.

1- Deploy the overlay network pods from the client machine.

$ kubectl apply -f https://[git.io/weave](http://git.io/weave)-kube-1.6

2- Check that the pods are deployed properly.

$ kubectl get pods -n kube-system

3- Check that the nodes are in Ready state.

$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-kubeadm-master-0 Ready master 18h v1.12.1

k8s-kubeadm-master-1 Ready master 18h v1.12.1

k8s-kubeadm-master-2 Ready master 18h v1.12.1

k8s-kubeadm-worker-0 Ready 16h v1.12.1

k8s-kubeadm-worker-1 Ready 16h v1.12.1

k8s-kubeadm-worker-2 Ready 16h v1.12.1

# Installing Kubernetes add-ons

We will deploy two Kubernetes add-ons on our new cluster: the dashboard add-on to have a graphical view of the cluster, and the Heapster add-on to monitor our workload.

## Installing the Kubernetes dashboard

1- Create the Kubernetes dashboard manifest.

$ vim kubernetes-dashboard.yaml

# Copyright 2017 The Kubernetes Authors.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

# Configuration to deploy release version of the Dashboard UI compatible with

# Kubernetes 1.8.

#

# Example usage: kubectl create -f

# ------------------- Dashboard Secret ------------------- #

apiVersion: v1

kind: Secret

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard-certs

namespace: kube-system

type: Opaque

---

# ------------------- Dashboard Service Account ------------------- #

apiVersion: v1

kind: ServiceAccount

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

---

# ------------------- Dashboard Role & Role Binding ------------------- #

kind: Role

apiVersion: rbac.authorization.[k8s.io/v1](http://k8s.io/v1)

metadata:

name: kubernetes-dashboard-minimal

namespace: kube-system

rules:

# Allow Dashboard to create 'kubernetes-dashboard-key-holder' secret.

- apiGroups: [""]

resources: ["secrets"]

verbs: ["create"]

# Allow Dashboard to create 'kubernetes-dashboard-settings' config map.

- apiGroups: [""]

resources: ["configmaps"]

verbs: ["create"]

# Allow Dashboard to get, update and delete Dashboard exclusive secrets.

- apiGroups: [""]

resources: ["secrets"]

resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-dashboard-certs"]

verbs: ["get", "update", "delete"]

# Allow Dashboard to get and update 'kubernetes-dashboard-settings' config map.

- apiGroups: [""]

resources: ["configmaps"]

resourceNames: ["kubernetes-dashboard-settings"]

verbs: ["get", "update"]

# Allow Dashboard to get metrics from heapster.

- apiGroups: [""]

resources: ["services"]

resourceNames: ["heapster"]

verbs: ["proxy"]

- apiGroups: [""]

resources: ["services/proxy"]

resourceNames: ["heapster", "http:heapster:", "https:heapster:"]

verbs: ["get"]

---

apiVersion: rbac.authorization.[k8s.io/v1](http://k8s.io/v1)

kind: RoleBinding

metadata:

name: kubernetes-dashboard-minimal

namespace: kube-system

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: kubernetes-dashboard-minimal

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kube-system

---

# ------------------- Dashboard Deployment ------------------- #

kind: Deployment

apiVersion: apps/v1beta2

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

replicas: 1

revisionHistoryLimit: 10

selector:

matchLabels:

k8s-app: kubernetes-dashboard

template:

metadata:

labels:

k8s-app: kubernetes-dashboard

spec:

containers:

- name: kubernetes-dashboard

image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.8.3

ports:

- containerPort: 8443

protocol: TCP

args:

- --auto-generate-certificates

# Uncomment the following line to manually specify Kubernetes API server Host

# If not specified, Dashboard will attempt to auto discover the API server and connect

# to it. Uncomment only if the default does not work.

# - --apiserver-host=<http://my-address:port>

volumeMounts:

- name: kubernetes-dashboard-certs

mountPath: /certs

# Create on-disk volume to store exec logs

- mountPath: /tmp

name: tmp-volume

livenessProbe:

httpGet:

scheme: HTTPS

path: /

port: 8443

initialDelaySeconds: 30

timeoutSeconds: 30

volumes:

- name: kubernetes-dashboard-certs

secret:

secretName: kubernetes-dashboard-certs

- name: tmp-volume

emptyDir: {}

serviceAccountName: kubernetes-dashboard

# Comment the following tolerations if Dashboard must not be deployed on master

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

---

# ------------------- Dashboard Service ------------------- #

kind: Service

apiVersion: v1

metadata:

labels:

k8s-app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

ports:

- port: 443

targetPort: 8443

selector:

k8s-app: kubernetes-dashboard

2- Deploy the dashboard.

$ kubectl create -f kubernetes-dashboard.yaml

## Installing Heapster

1- Create a manifest for Heapster.

$ vim heapster.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: heapster

namespace: kube-system

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: heapster

namespace: kube-system

spec:

replicas: 1

template:

metadata:

labels:

task: monitoring

k8s-app: heapster

spec:

serviceAccountName: heapster

containers:

- name: heapster

image: gcr.io/google\_containers/heapster-amd64:v1.4.2

imagePullPolicy: IfNotPresent

command:

- /heapster

- --source=kubernetes.summary\_api:''?useServiceAccount=true&kubeletHttps=true&kubeletPort=10250&insecure=true

---

apiVersion: v1

kind: Service

metadata:

labels:

task: monitoring

# For use as a Cluster add-on (https://github.com/kubernetes/kubernetes/tree/master/cluster/addons)

# If you are NOT using this as an addon, you should comment out this line.

kubernetes.io/cluster-service: 'true'

kubernetes.io/name: Heapster

name: heapster

namespace: kube-system

spec:

ports:

- port: 80

targetPort: 8082

selector:

k8s-app: heapster

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.[k8s.io/v1beta1](http://k8s.io/v1beta1)

metadata:

name: heapster

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: system:heapster

subjects:

- kind: ServiceAccount

name: heapster

namespace: kube-system

2- Deploy Heapster.

$ kubectl create -f heapster.yaml

3- Edit the Heapster RBAC role and add the get permission on the nodes statistic at the end.

$ kubectl edit clusterrole system:heapster

...

- apiGroups:

- ""

resources:

- nodes/stats

verbs:

- get

# Accessing the Kubernetes dashboard

1- Create an admin user manifest.

$ vim kubernetes-dashboard-admin.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kube-system

2- Create the admin user.

$ kubectl create -f kubernetes-dashboard-admin.yaml

3- Get the admin user token.

$ kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

4- Copy the token.

5- Start the proxy to access the dashboard.

$ kubectl proxy

6- Browse to <http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy>.

7- Select Token and paste the token from step 4.



