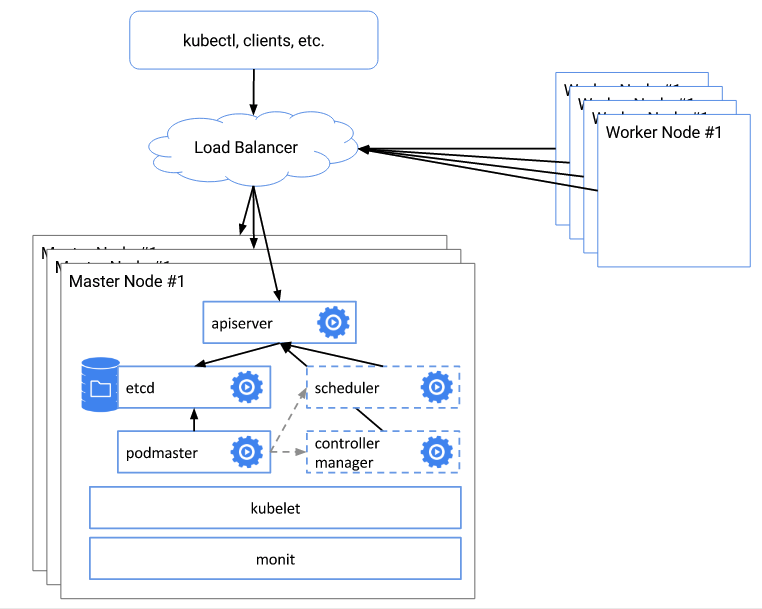
Kubernetes Installation

HA Kubenetes cluster



Referenec: <https://kubernetes.io/docs/admin/high-availability/>

System requirement

|  |  |  |
| --- | --- | --- |
| Master1: 192.168.1.100  Master2: 192.168.1.101 | Node1: 192.168.1.102  Node2: 192.168.1.103 | HAProxy: 192.168.1.105 |
| etcd  kube-apiserver  kube-controller-manager  kube-scheduler  flannel (option)  docker (option) | kubelet  Kube-proxy  flannel  docker | docker  flannel (option) |

Config

SERVICE\_CLUSTER\_IP\_RANGE = 192.168.100.0/24

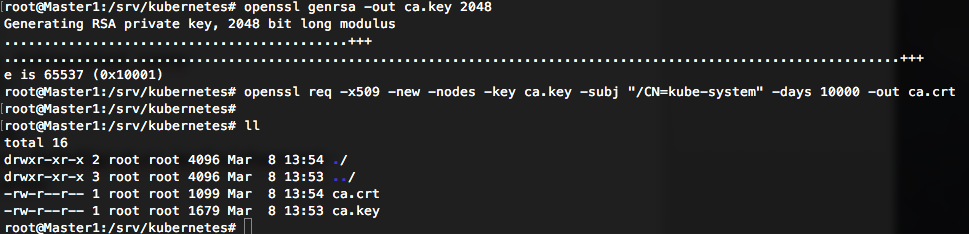
Flannel\_NET = 172.17.0.0/16

1. Create CA Certification

$ mkdir /srv/kubernetes; cd -

$ openssl genrsa -out ca.key 2048

$ openssl req -x509 -new -nodes -key ca.key -subj "/CN=kube-system" -days 10000 -out ca.crt



1. Install Master Node
   1. Create Master’s Key

$ cat <<EOF | sudo tee server-openssl.cnf

[req]

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 127.0.0.1

IP.2 = 192.168.1.105

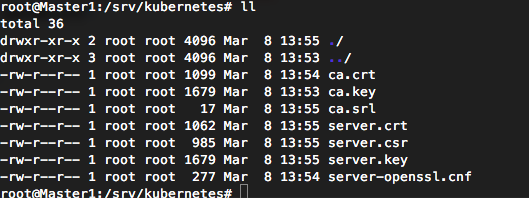
EOF

$ openssl genrsa -out server.key 2048

$ openssl req -new -key server.key -subj "/CN=192.168.1.105" -out server.csr -config server-openssl.cnf

$ openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 10000 -extensions v3\_req -extfile server-openssl.cnf

openssl x509 -noout -text -in server.crt



$ scp -r /srv/kubernetes 192.168.1.101:/srv/ #copy to Master1

$ scp -r /srv/kubernetes 192.168.1.101:/srv/ #copy to Master2

$ scp -r /srv/kubernetes 192.168.1.102:/srv/ #copy to Node1

$ scp -r /srv/kubernetes 192.168.1.103:/srv/ #copy to Node2

* 1. install etcd service on Master1

$ curl -L https://github.com/coreos/etcd/releases/download/v3.0.7/etcd-v3.0.7-linux-amd64.tar.gz -o etcd-v3.0.7-linux-amd64.tar.gz

$ tar xzvf etcd-v3.0.7-linux-amd64.tar.gz && cd etcd-v3.0.7-l inux-amd64

$ mkdir -p /opt/etcd/bin

$ mkdir -p /opt/etcd/config/

$ cp etcd\* /opt/etcd/bin/

$ mkdir -p /var/lib/etcd/

$ cat <<EOF | sudo tee /opt/etcd/config/etcd.conf

ETCD\_DATA\_DIR=/var/lib/etcd

ETCD\_NAME=Master1

ETCD\_LISTEN\_PEER\_URLS=http://0.0.0.0:2380

ETCD\_LISTEN\_CLIENT\_URLS=http://0.0.0.0:2379

ETCD\_INITIAL\_CLUSTER\_STATE=new

ETCD\_INITIAL\_CLUSTER=Master1=http://192.168.1.100:2380,Master2=http://192.168.1.101:2380

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS=http://192.168.1.100:2380

ETCD\_ADVERTISE\_CLIENT\_URLS=http://192.168.1.100:2379

ETCD\_HEARTBEAT\_INTERVAL=6000

ETCD\_ELECTION\_TIMEOUT=30000

GOMAXPROCS=$(nproc)

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/etcd.service

[Unit]

Description=Etcd Server

Documentation=https://github.com/coreos/etcd

After=network.target

[Service]

User=root

Type=simple

EnvironmentFile=-/opt/etcd/config/etcd.conf

ExecStart=/opt/etcd/bin/etcd

Restart=on-failure

RestartSec=10s

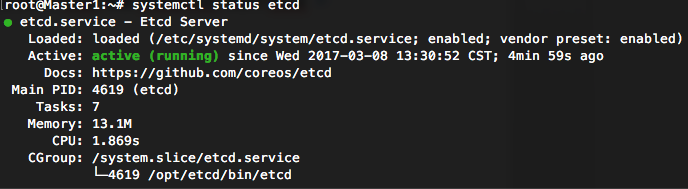
LimitNOFILE=40000

[Install]

WantedBy=multi-user.target

EOF

$ systemctl daemon-reload && systemctl enable etcd && systemctl start etcd



* 1. Install etcd service on Master2

$ curl -L https://github.com/coreos/etcd/releases/download/v3.0.7/etcd-v3.0.7-linux-amd64.tar.gz -o etcd-v3.0.7-linux-amd64.tar.gz

$ tar xzvf etcd-v3.0.7-linux-amd64.tar.gz && cd etcd-v3.0.7-l inux-amd64

$ mkdir -p /opt/etcd/bin

$ mkdir -p /opt/etcd/config/

$ cp etcd\* /opt/etcd/bin/

$ mkdir -p /var/lib/etcd/

$ cat <<EOF | sudo tee /opt/etcd/config/etcd.conf

ETCD\_DATA\_DIR=/var/lib/etcd

ETCD\_NAME=Master2

ETCD\_LISTEN\_PEER\_URLS=http://0.0.0.0:2380

ETCD\_LISTEN\_CLIENT\_URLS=http://0.0.0.0:2379

ETCD\_INITIAL\_CLUSTER\_STATE=new

ETCD\_INITIAL\_CLUSTER=Master1=http://192.168.1.100:2380,Master2=http://192.168.1.101:2380

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS=http://192.168.1.101:2380

ETCD\_ADVERTISE\_CLIENT\_URLS=http://192.168.1.101:2379

ETCD\_HEARTBEAT\_INTERVAL=6000

ETCD\_ELECTION\_TIMEOUT=30000

GOMAXPROCS=$(nproc)

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/etcd.service

[Unit]

Description=Etcd Server

Documentation=https://github.com/coreos/etcd

After=network.target

[Service]

User=root

Type=simple

EnvironmentFile=-/opt/etcd/config/etcd.conf

ExecStart=/opt/etcd/bin/etcd

Restart=on-failure

RestartSec=10s

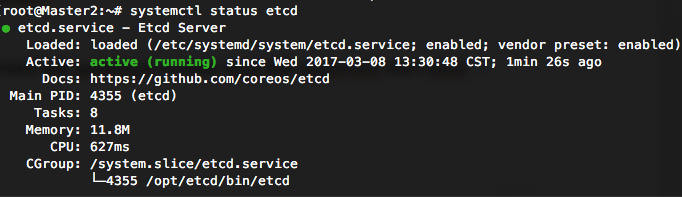
LimitNOFILE=40000

[Install]

WantedBy=multi-user.target

EOF

$ systemctl daemon-reload && systemctl enable etcd && systemctl start etcd

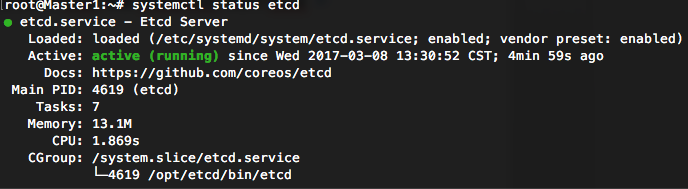


* 1. Test etcd cluster

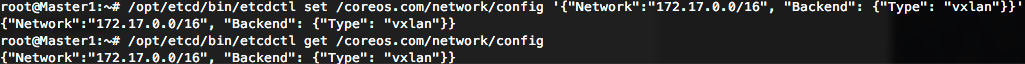
$ curl -L http://192.168.1.100:2379/v2/keys/test -XPUT -d value="awesome"

$ curl –L http://192.168.1.100:2379/v2/keys/test

$ curl –L http://192.168.1.101:2379/v2/keys/test



* 1. Set FLANNEL\_NET to etcd

$ /opt/etcd/bin/etcdctl set /coreos.com/network/config '{"Network":"172.17.0.0/16", "Backend": {"Type": "vxlan"}}'

OpenStack: config your security group to allow 2379 and 2380 ports

* 1. Install kube-apiserver, kube-controller-manager and kube-scheduler on Master1

$ curl -L 'https://github.com/kubernetes/kubernetes/releases/download/v1.4.9/kubernetes.tar.gz' -O kubernetes.tar.gz

$ tar xvf  kubernetes.tar.gz && cd kubernetes

$ tar xf ./server/kubernetes-server-linux-amd64.tar.gz -C /opt/

$ cat <<EOF | sudo tee /etc/systemd/system/kube-apiserver.service

[Unit]

Description=Kubernetes API Server

Documentation=https://github.com/kubernetes/kubernetes

After=network.target

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-apiserver \

--insecure-bind-address=127.0.0.1 \

--insecure-port=8080 \

--etcd-servers=http://192.168.1.100:2379,http://192.168.1.101:2379 \

--logtostderr=true \

--allow-privileged=false \

--service-cluster-ip-range=192.168.100.0/24 \

--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,SecurityContextDeny,ResourceQuota \

--service-node-port-range=30000-32767 \

--advertise-address=192.168.1.105 \

--client-ca-file=/srv/kubernetes/ca.crt \

--tls-cert-file=/srv/kubernetes/server.crt \

--tls-private-key-file=/srv/kubernetes/server.key

Restart=on-failure

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/kube-controller-manager.service

[Unit]

Description=Kubernetes Controller Manager

Documentation=https://github.com/kubernetes/kubernetes

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-controller-manager \

--master=127.0.0.1:8080 \

--root-ca-file=/srv/kubernetes/ca.crt \

--service-account-private-key-file=/srv/kubernetes/server.key \

--logtostderr=true

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/kube-scheduler.service

[Unit]

Description=Kubernetes Scheduler

Documentation=https://github.com/kubernetes/kubernetes

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-scheduler \

--logtostderr=true \

--master=127.0.0.1:8080

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

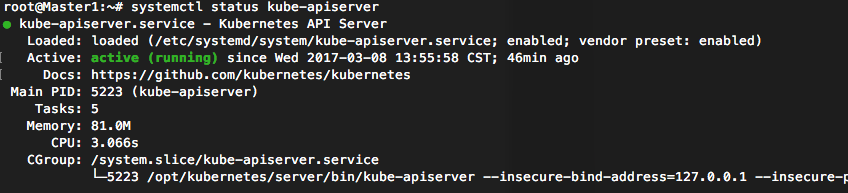
$ systemctl daemon-reload

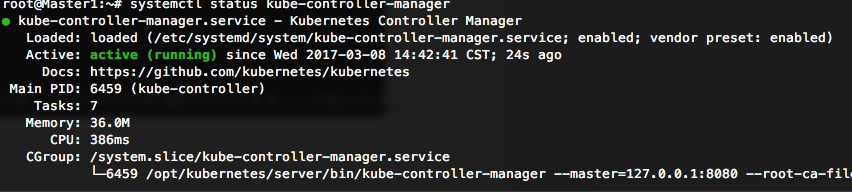
$ for name in kube-apiserver kube-controller-manager kube-scheduler; do

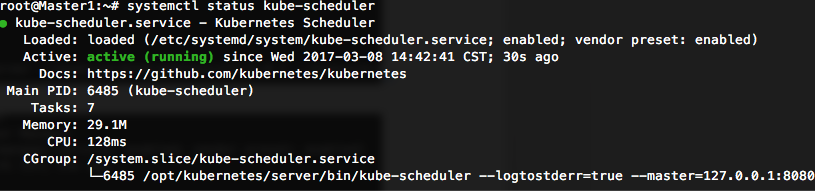
systemctl enable $name

systemctl start $name

done







OpenStack: config your security group to allow 6443 port

* 1. Install kube-apiserver, kube-controller-manager and kube-scheduler on Master2

$ curl -L 'https://github.com/kubernetes/kubernetes/releases/download/v1.4.9/kubernetes.tar.gz' -O kubernetes.tar.gz

$ tar xvf  kubernetes.tar.gz && cd kubernetes

$ tar xf ./server/kubernetes-server-linux-amd64.tar.gz -C /opt/

$ cat <<EOF | sudo tee /etc/systemd/system/kube-apiserver.service

[Unit]

Description=Kubernetes API Server

Documentation=https://github.com/kubernetes/kubernetes

After=network.target

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-apiserver \

--insecure-bind-address=127.0.0.1 \

--insecure-port=8080 \

--etcd-servers=http://192.168.1.100:2379,http://192.168.1.101:2379 \

--logtostderr=true \

--allow-privileged=false \

--service-cluster-ip-range=192.168.100.0/24 \

--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,SecurityContextDeny,ResourceQuota \

--service-node-port-range=30000-32767 \

--advertise-address=192.168.1.105 \

--client-ca-file=/srv/kubernetes/ca.crt \

--tls-cert-file=/srv/kubernetes/server.crt \

--tls-private-key-file=/srv/kubernetes/server.key

Restart=on-failure

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/kube-controller-manager.service

[Unit]

Description=Kubernetes Controller Manager

Documentation=https://github.com/kubernetes/kubernetes

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-controller-manager \

--master=127.0.0.1:8080 \

--root-ca-file=/srv/kubernetes/ca.crt \

--service-account-private-key-file=/srv/kubernetes/server.key \

--logtostderr=true

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

$ cat <<EOF | sudo tee /etc/systemd/system/kube-scheduler.service

[Unit]

Description=Kubernetes Scheduler

Documentation=https://github.com/kubernetes/kubernetes

[Service]

User=root

ExecStart=/opt/kubernetes/server/bin/kube-scheduler \

--logtostderr=true \

--master=127.0.0.1:8080

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

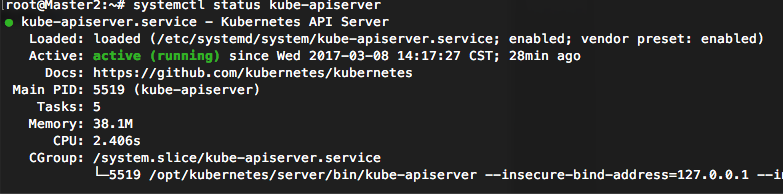
$ systemctl daemon-reload

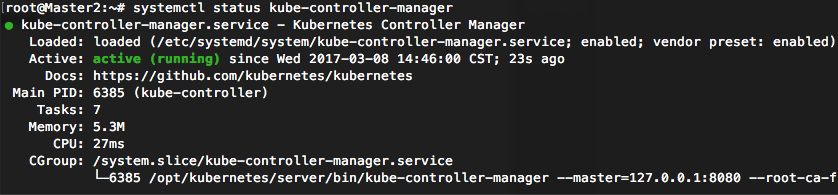
$ for name in kube-apiserver kube-controller-manager kube-scheduler; do

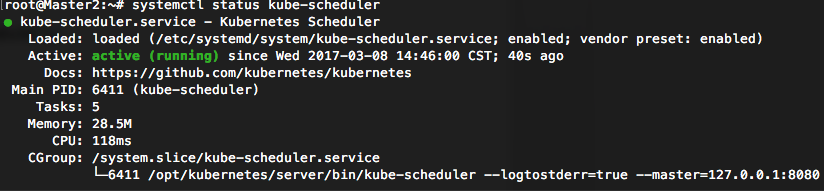
systemctl enable $name

systemctl start $name

done







OpenStack: config your security group to allow 6443 port

1. Install HA Proxy

2-1. Running haproxy on docker

$ apt-get install docker.io

$ cat <<EOF > /opt/haproxy.cfg

global

log 127.0.0.1 local0

log 127.0.0.1 local1 notice

maxconn 4096

maxpipes 1024

daemon

defaults

log global

mode tcp

option tcplog

option dontlognull

option redispatch

option http-server-close

retries 3

timeout connect 5000

timeout client 50000

timeout server 50000

frontend default\_frontend

bind \*:6443

default\_backend master-cluster

backend master-cluster

server master1 192.168.1.100

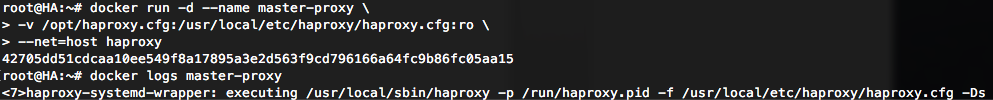
server master2 192.168.1.101

EOF

$ docker run -d --name master-proxy \

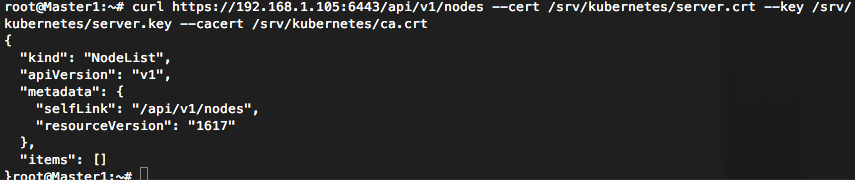
-v /opt/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro \

--net=host haproxy



2-2. Test

$ curl https://192.168.1.105:6443/pi/v1/nodes --cert /srv/kubernetes/server.crt --key /srv/kubernetes/server.key --cacert /srv/kubernetes/ca.crt



OpenStack: config your security group to allow 6443 port

1. Install Worker Node

3-1. create node’s key for Node1

$ cat <<EOF | sudo tee node-openssl.cnf

[req]

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 192.168.1.102

EOF

$ openssl genrsa -out node.key 2048

$ openssl req -new -key node.key -subj "/CN=192.168.1.102" -out node.csr -config node-openssl.cnf

$ openssl x509 -req -in node.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out node.crt -days 10000 -extensions v3\_req -extfile node-openssl.cnf

$ openssl x509 -noout -text -in node.crt

3-2. create node’s key for Node2

$ cat <<EOF | sudo tee node-openssl.cnf

[req]

req\_extensions = v3\_req

distinguished\_name = req\_distinguished\_name

[req\_distinguished\_name]

[ v3\_req ]

basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment

subjectAltName = @alt\_names

[alt\_names]

IP.1 = 192.168.1.103

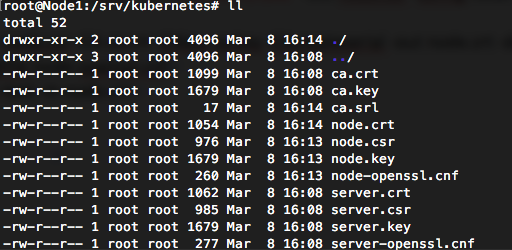
EOF

$ openssl genrsa -out node.key 2048

$ openssl req -new -key node.key -subj "/CN=192.168.1.103" -out node.csr -config node-openssl.cnf

$ openssl x509 -req -in node.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out node.crt -days 10000 -extensions v3\_req -extfile node-openssl.cnf

$ openssl x509 -noout -text -in node.crt



3-3. install docker, kubelet, kube-proxy and flannel on Node1

# docker

$ apt-get install -y docker.io

# download

$ curl -L 'https://github.com/kubernetes/kubernetes/releases/download/v1.4.9/kubernetes.tar.gz' -O kubernetes.tar.gz

$ tar xvf  kubernetes.tar.gz && cd kubernetes

$ tar xf ./server/kubernetes-server-linux-amd64.tar.gz -C /opt/

# prepare kubeconfig

$ cat <<EOF | sudo tee /srv/kubernetes/kubeconfig

apiVersion: v1

kind: Config

clusters:

- name: local

cluster:

certificate-authority: /srv/kubernetes/ca.crt

users:

- name: controllermanager

user:

client-certificate: /srv/kubernetes/node.crt

client-key: /srv/kubernetes/node.key

contexts:

- context:

cluster: local

user: controllermanager

name: kubelet-context

current-context: kubelet-context

EOF

# kubelet config

$ cat <<EOF | sudo tee /etc/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet

After=docker.service

Requires=docker.service

[Service]

ExecStart=/opt/kubernetes/server/bin/kubelet \

--hostname-override=192.168.1.103 \

--api-servers=https://192.168.1.105:6443 \

--logtostderr=true \

--tls-cert-file=/srv/kubernetes/node.crt \

--tls-private-key-file=/srv/kubernetes/node.key \

--kubeconfig=/srv/kubernetes/kubeconfig

Restart=on-failure

KillMode=process

[Install]

WantedBy=multi-user.target

EOF

# kube-proxy config

$ cat <<EOF | sudo tee /etc/systemd/system/kube-proxy.service

[Unit]

Description=Kubernetes Proxy

After=network.target

[Service]

ExecStart=/opt/kubernetes/server/bin/kube-proxy \

--hostname-override=192.168.1.103 \

--master=https://192.168.1.105:6443 \

--kubeconfig=/srv/kubernetes/kubeconfig \

--logtostderr=true

Restart=on-failure

[Install]

WantedBy=multi-user.target

EOF

# download flannel

$ curl -L https://github.com/coreos/flannel/releases/download/v0.6.1/flannel-v0.6.1-linux-amd64.tar.gz -o flannel.tar.gz

$ mkdir -p /opt/flannel

$ tar xzf flannel.tar.gz -C /opt/flannel

# flannel config

$ cat <<EOF | sudo tee /etc/systemd/system/flanneld.service

[Unit]

Description=Flanneld

Documentation=https://github.com/coreos/flannel

After=network.target

Before=docker.service

[Service]

User=root

ExecStart=/opt/flannel/flanneld \

--etcd-endpoints="http://192.168.1.100:2379,http://192.168.1.101:2379" \

--iface=192.168.1.102 \

--ip-masq

ExecStartPost=/bin/bash /opt/flannel/update\_docker.sh

Restart=on-failure

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

# add update\_docker.sh

$ cat <<EOF | sudo tee /opt/flannel/update\_docker.sh

source /run/flannel/subnet.env

sed -i "s|ExecStart=.\*|ExecStart=\/usr\/bin\/dockerd -H tcp:\/\/127.0.0.1:4243 -H unix:\/\/\/var\/run\/docker.sock --bip=\${FLANNEL\_SUBNET} --mtu=\${FLANNEL\_MTU}|g" /lib/systemd/system/docker.service

rc=0

ip link show docker0 >/dev/null 2>&1 || rc="$?"

if [[ "$rc" -eq "0" ]]; then

ip link set dev docker0 down

ip link delete docker0

fi

systemctl daemon-reload

EOF

# start service

$ systemctl daemon-reload

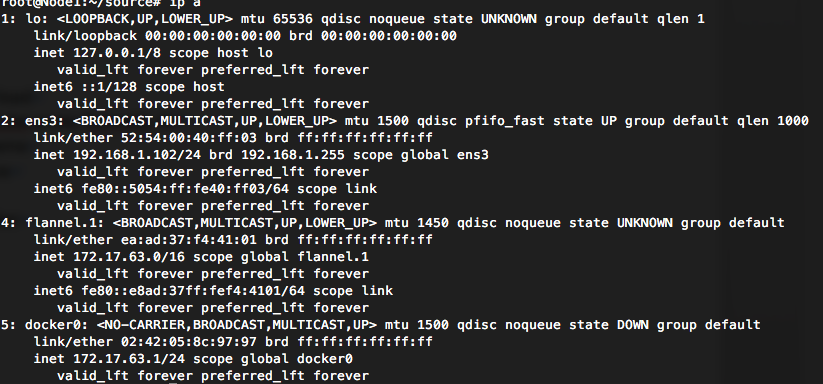
$ for name in kubelet kube-proxy flanneld; do

systemctl enable $name

systemctl start $name

done

$ systemctl restart docker



3-4. install docker, kubelet, kube-proxy and flannel on Node2

# docker

$ apt-get install -y docker.io

# download

$ curl -L 'https://github.com/kubernetes/kubernetes/releases/download/v1.4.9/kubernetes.tar.gz' -O kubernetes.tar.gz

$ tar xvf  kubernetes.tar.gz && cd kubernetes

$ tar xf ./server/kubernetes-server-linux-amd64.tar.gz -C /opt/

# prepare kubeconfig

$ cat <<EOF | sudo tee /srv/kubernetes/kubeconfig

apiVersion: v1

kind: Config

clusters:

- name: local

cluster:

certificate-authority: /srv/kubernetes/ca.crt

users:

- name: controllermanager

user:

client-certificate: /srv/kubernetes/node.crt

client-key: /srv/kubernetes/node.key

contexts:

- context:

cluster: local

user: controllermanager

name: kubelet-context

current-context: kubelet-context

EOF

# kubelet config

$ cat <<EOF | sudo tee /etc/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet

After=docker.service

Requires=docker.service

[Service]

ExecStart=/opt/kubernetes/server/bin/kubelet \

--hostname-override=192.168.1.104 \

--api-servers=https://192.168.1.105:6443 \

--logtostderr=true \

--tls-cert-file=/srv/kubernetes/node.crt \

--tls-private-key-file=/srv/kubernetes/node.key \

--kubeconfig=/srv/kubernetes/kubeconfig

Restart=on-failure

KillMode=process

[Install]

WantedBy=multi-user.target

EOF

# kube-proxy config

$ cat <<EOF | sudo tee /etc/systemd/system/kube-proxy.service

[Unit]

Description=Kubernetes Proxy

After=network.target

[Service]

ExecStart=/opt/kubernetes/server/bin/kube-proxy \

--hostname-override=192.168.1.103 \

--master=https://192.168.1.105:6443 \

--kubeconfig=/srv/kubernetes/kubeconfig \

--logtostderr=true

Restart=on-failure

[Install]

WantedBy=multi-user.target

EOF

# download flannel

$ curl -L https://github.com/coreos/flannel/releases/download/v0.6.1/flannel-v0.6.1-linux-amd64.tar.gz -o flannel.tar.gz

$ mkdir -p /opt/flannel

$ tar xzf flannel.tar.gz -C /opt/flannel

# flannel config

$ cat <<EOF | sudo tee /etc/systemd/system/flanneld.service

[Unit]

Description=Flanneld

Documentation=https://github.com/coreos/flannel

After=network.target

Before=docker.service

[Service]

User=root

ExecStart=/opt/flannel/flanneld \

--etcd-endpoints="http://192.168.1.100:2379,http://192.168.1.101:2379" \

--iface=192.168.1.103 \

--ip-masq

ExecStartPost=/bin/bash /opt/flannel/update\_docker.sh

Restart=on-failure

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

# add update\_docker.sh

$ cat <<EOF | sudo tee /opt/flannel/update\_docker.sh

source /run/flannel/subnet.env

sed -i "s|ExecStart=.\*|ExecStart=\/usr\/bin\/dockerd -H tcp:\/\/127.0.0.1:4243 -H unix:\/\/\/var\/run\/docker.sock --bip=\${FLANNEL\_SUBNET} --mtu=\${FLANNEL\_MTU}|g" /lib/systemd/system/docker.service

rc=0

ip link show docker0 >/dev/null 2>&1 || rc="$?"

if [[ "$rc" -eq "0" ]]; then

ip link set dev docker0 down

ip link delete docker0

fi

systemctl daemon-reload

EOF

# start service

$ systemctl daemon-reload

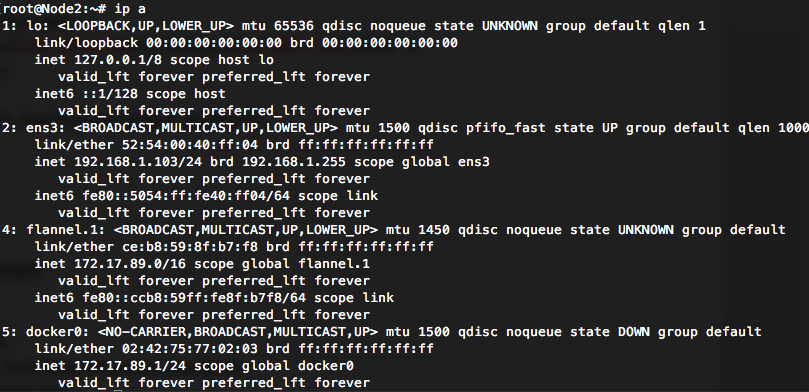
$ for name in kubelet kube-proxy flanneld; do

systemctl enable $name

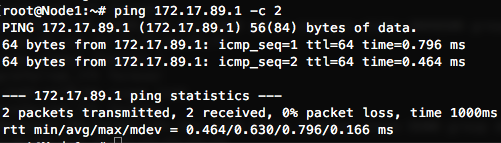
systemctl start $name

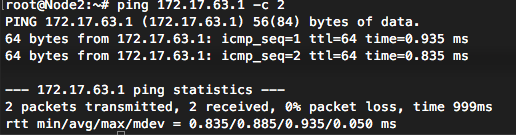
done

$ systemctl restart docker

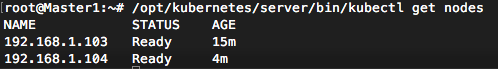


3-5. Ping test





3-6. List node on Master



1. Deploy Dashboard

$ cat <<EOF | sudo tee kubernetes-dashboard.yaml

kind: Deployment

apiVersion: extensions/v1beta1

metadata:

labels:

app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

replicas: 1

selector:

matchLabels:

app: kubernetes-dashboard

template:

metadata:

labels:

app: kubernetes-dashboard

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

annotations:

scheduler.alpha.kubernetes.io/tolerations: |

[

{

"key": "dedicated",

"operator": "Equal",

"value": "master",

"effect": "NoSchedule"

}

]

spec:

containers:

- name: kubernetes-dashboard

image: gcr.io/google\_containers/kubernetes-dashboard-amd64:v1.5.1

imagePullPolicy: Always

ports:

- containerPort: 9090

protocol: TCP

args:

- --apiserver-host=https://192.168.1.105:6443

- --kubeconfig=/srv/kubernetes/kubeconfig

livenessProbe:

httpGet:

path: /

port: 9090

initialDelaySeconds: 30

timeoutSeconds: 30

volumeMounts:

- mountPath: /srv/kubernetes

name: certs

readOnly: true

volumes:

- hostPath:

path: /srv/kubernetes

name: certs

---

kind: Service

apiVersion: v1

metadata:

labels:

app: kubernetes-dashboard

name: kubernetes-dashboard

namespace: kube-system

spec:

type: NodePort

ports:

- port: 80

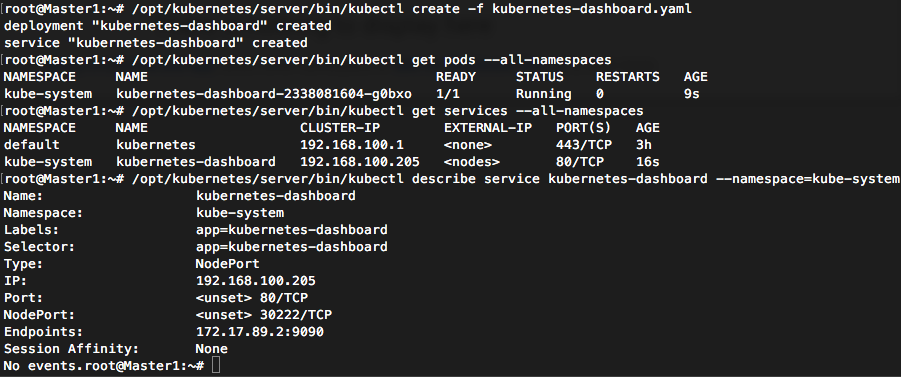
targetPort: 9090

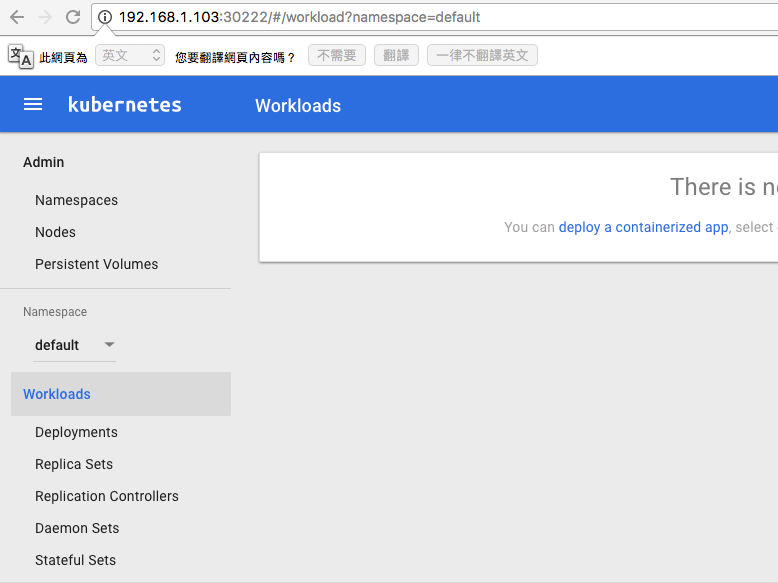
selector:

app: kubernetes-dashboard

EOF

$ /opt/kubernetes/server/bin/kubectl create -f kubernetes-dashboard.yaml





OpenStack: Config your security group to allow dashboard’s port

1. Using install\_k8s

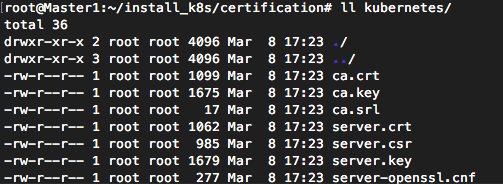
5-1. Create master key

$ cd install\_k8s/certification

$ bash create\_ca.sh

$ bach create\_master\_key.sh

# scp install\_k8s to master and node, include keys



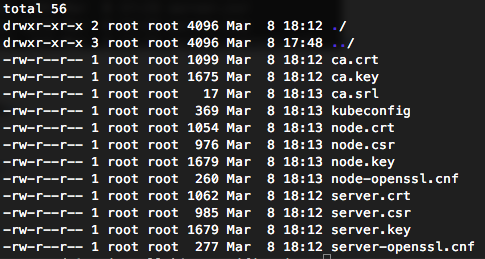
5-2. Install Master

$ bash install\_master.sh

5-3. Create node key

$ cd install\_k8s/certification

$ bash create\_node\_key.sh



5-4. Install Node

$ apt-get install docker.io

$ bash install\_node.sh

5-5. Install HA Proxy

$ apt-get install docker.io

5-6. Deploy Dashboard

$ kubectl create -f kubernetes-dashboard.yaml

# remember to modify your ip address

Other Services

1. Skydns

Using kubernetes/cluster/addons/dns/skydns-rc.yaml.base and skydns-svc.yaml.base

Modify skydns-rc.yaml

1. replace \_\_PILLAR\_\_DNS\_\_REPLICAS\_\_ to 1
2. replace \_\_PILLAR\_\_DNS\_\_DOMAIN\_\_ to your domain name (ex. k8s.uat …)
3. replace \_\_PILLAR\_\_FEDERATIONS\_\_DOMAIN\_\_MAP\_\_ to - --kube-master-url=https://${MASTER\_LB\_IP}:6443
4. If you are using ssl

add --kubecfg\_file=/srv/kubernetes/kubeconfig under - --kuber-master-url

add “volumeMounts:

- mountPath: /srv/kubernetes

name: certs

readOnly: true” in the end of containers section

add “volumes:

- hostPath:

path: /srv/kubernetes

name: certs” in the endo of spec section

Modify skydns-rc.yaml

1. choose a IP from your SERVICE\_CLUSTER\_IP\_RANGE to replace \_\_PILLAR\_\_DNS\_\_SERVER\_\_

Modify kubelet config

1. add --cluster-dns=${DNS\_IP} --cluster-domain=${YOUR\_DOMAIN}

$ kubectl create -f skydns-rc.yaml.base

$ kubectl create -f skydns-svc.yaml.base

1. Heapster
2. Install flannel on Master. Dashboar will use Master as proxy to get heapster data
3. $ git clone <https://github.com/GoogleCloudPlatform/heapster.git>
4. kubectl create -f deploy/kube-config/influxdb/

Using kubernetes/cluster/addons/cluster-monitoring/influxdb

Modify

Modify skydns-rc.yam

OpenStack

1. attached a volume for etcd

# assume your volume attached to /dev/vdb

# partition and format

$ fdisk /dev/vdb (press n then w)

$ mkfs.ext4 /dev/vdb1

# add to fstab

$ blkid | grep vdb1

*# /dev/vdb1: UUID="5a497ac2-c250-446d-876b-5cd043741f5a" TYPE="ext4" PARTUUID="68f42975-01"*

$ echo “UUID=5a497ac2-c250-446d-876b-5cd043741f5a /opt ext4 defaults 0 0” >> /etc/fstab

# modify your etcd config

# ETCD\_DATA\_DIR=/var/lib/etcd

Notes:

1. When you write overwrite-hostname = $hostname in kubelet’s config

Add node in hosts on Master