Install - Highly Available (ON-demand)

Kubernetes .

: , etcd .

etcd : , etcd .

Kubernetes 1.11 .

.

Kubeadm 3 .

Kubeadm 3 .

.

SSH .

SUDU .

Etcd 3 .

SSH

ssh-agent .

```
# eval $(ssh-agent)
```

SSH ID .

```
# ssh-add ~/.ssh/path_to_private_key
```

SSH .

```
# ssh -A 10.0.0.7
```

sudo SSH .

```
# sudo -E -s
```

SSH

```
Master Node 1
$ rm -rf /root/.ssh/*
$ ssh <master ip 1> pwd
$ ssh <master ip 2> rm -rf /root/.ssh/*
$ ssh <master ip 3> rm -rf /root/.ssh/*
$ ssh <master ip 2> mkdir -p /root/.ssh/
$ ssh <master ip 3> mkdir -p /root/.ssh/

$ scp /root/.ssh/known_hosts root@<master ip 2>:/root/.ssh/
$ scp /root/.ssh/known_hosts root@<master ip 3>:/root/.ssh/

$ ssh-keygen -t rsa -P '' -f /root/.ssh/id_rsa
$ cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
$ scp /root/.ssh/authorized_keys root@<master ip 2>:/root/.ssh/

Master Node 2
$ ssh-keygen -t rsa -P '' -f /root/.ssh/id_rsa
$ cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
$ scp /root/.ssh/authorized_keys root@<master ip 3>:/root/.ssh/

Master Node 3
$ ssh-keygen -t rsa -P '' -f /root/.ssh/id_rsa
$ cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
$ scp /root/.ssh/authorized_keys root@<master ip 1>:/root/.ssh/
$ scp /root/.ssh/authorized_keys root@<master ip 2>:/root/.ssh/

# ssh -A <master ip 1>
# ssh -A <master ip 2>
# ssh -A <master ip 3>
```

)

Last login: Mon Oct 1 12:35:56 2018

kube-apiserver Load Balancer

  , .

DNS kube-apiserver . (TCP forwarding)

  .

apiserver kube-apiserver TCP (: 6443).

 apiserver , .

 master node haproxy keepalived (haproxy 1.5 SSL)

```
# yum install haproxy
# haproxy -v
# yum install keepalived
```

 master node chkconfig (on)

  .

```
# chkconfig haproxy on  && chkconfig keepalived on && chkconfig | egrep 'haproxy|keepalived'
```

 master ndoe non-local Virtual IPs binding

```
# echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf && sysctl -p
```

)

net.ipv4.tcp_max_syn_backlog = 4096

net.ipv4.conf.all.send_redirects = 0

net.ipv4.conf.all.accept_redirects = 0

net.ipv4.conf.all.accept_source_route = 0

net.ipv4.conf.all.forwarding = 0

net.ipv4.icmp_echo_ignore_broadcasts = 1

net.ipv4.ip_nonlocal_bind = 1

keepalived (All Master)

```
# cd /etc/keepalived //
# mv keepalived.conf keepalived.conf.org
```

Master Node 1

```
# vi /etc/keepalived/keepalived.conf

global_defs {
        notification_email {
        test@test.com
        test2 @test.com
}

notification_email_from lb1@test.com
        smtp_server localhost
        smtp_connect_timeout 30
}

# haproxy
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
}

vrrp_instance VI_1 {
        state MASTER
        interface eth0                          #    (ifconfig  )
        virtual_router_id 51         # Master Node 3    . ( 255)
        priority 101                           #   ( 255 )
        advert_int 1                           # VRRP    ( )
        authentication {
                auth_type PASS                 #
                auth_pass 1111                 #    (All Master  )
    }
        virtual_ipaddress {
                000.000.000.000         # VIP
    }

        track_script {
                chk_haproxy
    }
}
```

Master Node 2

Master Node 2 Master Node 1 priority .

```
# vi /etc/keepalived/keepalived.conf

global_defs {
        notification_email {
        test@test.com
        test2 @test.com
}

notification_email_from lb1@test.com
        smtp_server localhost
        smtp_connect_timeout 30
}

# haproxy
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
}

vrrp_instance VI_2 {
        state BACKUP
        interface eth0
        virtual_router_id 51
        priority 100
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1111
    }
        virtual_ipaddress {
                000.000.000.000
    }

        track_script {
                chk_haproxy
    }
}
```

Master Node 3

Master Node 3 Master Node 1,2 priority .

```
# vi /etc/keepalived/keepalived.conf

global_defs {
        notification_email {
        test@test.com
        test2 @test.com
}

notification_email_from lb1@test.com
        smtp_server localhost
        smtp_connect_timeout 30
}

# haproxy
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
}

vrrp_instance VI_3 {
        state BACKUP
        interface eth0
        virtual_router_id 51
        priority 99
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1111
    }
        virtual_ipaddress {
                000.000.000.000
    }

        track_script {
                chk_haproxy
    }
}
```

keepalived (All Master)

```
# systemctl enable keepalived
# systemctl start keepalived
```

)

keepalived.service - LVS and VRRP High Availability Monitor

Loaded: loaded (/usr/lib/systemd/system/keepalived.service; enabled; vendor preset: disabled)

Active: active (running) since 2018-10-01 14:11:44 KST; 7s ago

Process: 26099 ExecStart=/usr/sbin/keepalived $KEEPALIVED_OPTIONS (code=exited, status=0/SUCCESS)

Main PID: 26100 (keepalived)

Tasks: 3

Memory: 1.6M

CGroup: /system.slice/keepalived.service

26100 /usr/sbin/keepalived -D

26101 /usr/sbin/keepalived -D

26102 /usr/sbin/keepalived -D

10 01 14:11:46 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

10 01 14:11:46 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: /usr/bin/killall -0 haproxy exited with status 1

10 01 14:11:48 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: /usr/bin/killall -0 haproxy exited with status 1

10 01 14:11:50 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: /usr/bin/killall -0 haproxy exited with status 1

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: VRRP_Instance(VI_1) Sending/queueing gratuitous ARPs on eth0 for 22....50.9

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

10 01 14:11:51 sb-k8s-MASTER-STBY77 Keepalived_vrrp[26102]: Sending gratuitous ARP on eth0 for 222.231.50.9

Hint: Some lines were ellipsized, use -l to show in full.

Master Node

```
# ip addr show eth0
```

)

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

link/ether 52:54:00:21:46:f3 brd ff:ff:ff:ff:ff:ff

inet < IP> brd 222.231.50.255 scope global eth0

valid_lft forever preferred_lft forever

inet < IP> scope global eth0

valid_lft forever preferred_lft forever

inet6 fe80::5054:ff:fe21:46f3/64 scope link

valid_lft forever preferred_lft forever

// client node vip ping test arp cahce

// VIP node 1 mac cahce

```
# arp -a
```

)

? (192.168.000.000) at 00:50:56:fd:6b:11 [ether] on eth1

? (192.168.000.000) at 00:50:56:c0:00:08 [ether] on eth1

? (192.168.000.100) at 00:0c:29:c4:76:05 [ether] on eth1 <----------VIP

? (192.168.000.101) at 00:0c:29:c4:76:05 [ether] on eth1 <----------Node 1 IP

? (192.168.000.000) at 00:0c:29:bf:fb:82 [ether] on eth1

haproxy (All Master)

```
# mv /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.org  //
# vi /etc/haproxy/haproxy.cfg

global
        log 127.0.0.1 local2
        maxconn 2000
        uid 0
        gid 0
        daemon                                          # background process

defaults
    log     global                          # global
    mode    tcp                                     # SSL   TCP (http SSL )
    option  tcplog
    option  dontlognull                 #
    retries 3                                       #
    maxconn 2000                                     #option redispatch
    #timeout http-request   10s
    #timeout queue          1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m

frontend ssl_front
    bind 000.000.000.000:16443        #VIP (kube-master     port kube-api   )
    default_backend ssl_backend

backend ssl_backend
    balance roundrobin
    option tcp-check                            # ssl-hello-chk option   - ssl3.0 protocol  k8s api    (TLS
1.2  )
    server hostname1 000.000.000.000:6443 check
    server hostname2 000.000.000.001:6443 check
        server hostname3 000.000.000.002:6443 check
```

haproxy (All Master)

```
# systemctl enable haproxy
# systemctl start haproxy
```

)

)

haproxy.service - HAProxy Load Balancer

Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor preset: disabled)

Active: active (running) since 2018-10-01 14:53:58 KST; 5s ago

Main PID: 6841 (haproxy-systemd)

Tasks: 3

Memory: 1.6M

CGroup: /system.slice/haproxy.service

6841 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

6842 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds

6843 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds

10 01 14:53:58 sb-k8s-MASTER-STBY77 systemd[1]: Started HAProxy Load Balancer.

10 01 14:53:58 sb-k8s-MASTER-STBY77 systemd[1]: Starting HAProxy Load Balancer...

10 01 14:53:58 sb-k8s-MASTER-STBY77 haproxy-systemd-wrapper[6841]: haproxy-systemd-wrapper: executing /usr/sbin/haproxy -f /etc/... -Ds

Hint: Some lines were ellipsized, use -l to show in full.

kubeadm-config.yaml

kubeadm init kubeadm-config.yaml .

Master Node .

, HAProxy timeout . api-server .

Master Node 1

kubeadm-config.yaml (root)

Master Node Hostname . IP etcd kubeadm init .

podSubnet: "192.168.0.0/16" : IP ( )

```
# v1.11.0
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.3
apiServerCertSANs:
- "{VIP}"
api:
    controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 1 IP}:2379"
      advertise-client-urls: "https://{Master Node 1 IP}:2379"
      listen-peer-urls: "https://{Master Node 1 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 1 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 1 IP}:2380"
    serverCertSANs:
      - {Master Node 1 Host Name}
      - {Master Node 1 IP}
    peerCertSANs:
      - {Master Node 1 Host Name}
      - {Master Node 1 IP}
networking:
    podSubnet: "10.32.0.0/12"

================================================================================================

# v1.12.0
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: v1.12.1
apiServerCertSANs:
- "{VIP}"
controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 1 IP}:2379"
      advertise-client-urls: "https://{Master Node 1 IP}:2379"
      listen-peer-urls: "https://{Master Node 1 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 1 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 1 IP}:2380"
    serverCertSANs:
      - {Master Node 1 Host Name}
      - {Master Node 1 IP}
    peerCertSANs:
      - {Master Node 1 Host Name}
      - {Master Node 1 IP}
networking:
    podSubnet: "10.32.0.0/12"
```

Master Node 2

```
# v1.11.0
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.3
apiServerCertSANs:
- "{VIP}"
api:
    controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 2 IP}:2379"
      advertise-client-urls: "https://{Master Node 2 IP}:2379"
      listen-peer-urls: "https://{Master Node 2 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 2 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 1 IP},{Master Node 2 Host Name}
=https://{Master Node 2 IP}:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - {Master Node 2 Host Name}
      - {Master Node 2 IP}
    peerCertSANs:
      - {Master Node 2 Host Name}
      - {Master Node 2 IP}
networking:
    podSubnet: "10.32.0.0/12"

============================================================================================

#v1.12.0
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: v1.12.1
apiServerCertSANs:
- "{VIP}"
controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 2 IP}:2379"
      advertise-client-urls: "https://{Master Node 2 IP}:2379"
      listen-peer-urls: "https://{Master Node 2 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 2 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 2 IP}:2380,{Master Node 2 Host Name}
=https://{Master Node 2 IP}:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - {Master Node 2 Host Name}
      - {Master Node 2 IP}
    peerCertSANs:
      - {Master Node 2 Host Name}
      - {Master Node 2 IP}
networking:
    podSubnet: "10.32.0.0/12"
```

Master Node 3

```
# v1.11.0
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.3
apiServerCertSANs:
- "{VIP}"
api:
    controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 3 IP}:2379"
      advertise-client-urls: "https://{Master Node 3 IP}:2379"
      listen-peer-urls: "https://{Master Node 3 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 3 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 1 IP}:2380,{Master Node 2 Host Name}
=https://{Master Node 2 IP}:2380,{Master Node 3 Host Name}=https://{Master Node 3 IP}:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - {Master Node 3 Host Name}
      - {Master Node 3 IP}
    peerCertSANs:
      - {Master Node 3 Host Name}
      - {Master Node 3 IP}
networking:
    podSubnet: "10.32.0.0/12"


================================================================================================

# v1.12.0
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: v1.12.1
apiServerCertSANs:
- "{VIP}"
controlPlaneEndpoint: "{VIP}:16443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://{Master Node 3 IP}:2379"
      advertise-client-urls: "https://{Master Node 3 IP}:2379"
      listen-peer-urls: "https://{Master Node 3 IP}:2380"
      initial-advertise-peer-urls: "https://{Master Node 3 IP}:2380"
      initial-cluster: "{Master Node 1 Host Name}=https://{Master Node 1 IP}:2380,{Master Node 2 Host Name}
=https://{Master Node 2 IP}:2380,{Master Node 3 Host Name}=https://{Master Node 3 IP}:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - {Master Node 3 Host Name}
      - {Master Node 3 IP}
    peerCertSANs:
      - {Master Node 3 Host Name}
      - {Master Node 3 IP}
networking:
    podSubnet: "10.32.0.0/12"
```

Kubeadm Init

```
# echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward //
# kubeadm init --config kubeadm-config.yaml
```

)

I1001 16:33:49.306235 8091 version.go:89] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable.txt": Get https://dl.k8s.io/release/stable.txt: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
I1001 16:33:49.306486 8091 version.go:94] falling back to the local client version: v1.12.0
[init] using Kubernetes version: v1.12.0
[preflight] running pre-flight checks
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names...
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names...
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names...
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] valid certificates and keys now exist in "/etc/kubernetes/pki"
[certificates] Generated sa key and public key.
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"
[controlplane] wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"
[init] waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests"
[init] this might take a minute or longer if the control plane images have to be pulled
[apiclient] All control plane components are healthy after 21.504056 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.12" in namespace kube-system with the configuration for the kubelets in the cluster
[markmaster] Marking the node ... as master by adding the label "node-role.kubernetes.io/master=""
[markmaster] Marking the node ... as master by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object...
[bootstraptoken] using token: ws83pk.dm7js0fvgkdkud3y
[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 ...

Master Node 1 Certification

Master Node 1 init Master Node 2, 3 .

 Master Node 1 Master Node 2,3 SSH .

```
//
/etc/kubernetes/pki/ca.crt
/etc/kubernetes/pki/ca.key
/etc/kubernetes/pki/sa.key
/etc/kubernetes/pki/sa.pub
/etc/kubernetes/pki/front-proxy-ca.crt
/etc/kubernetes/pki/front-proxy-ca.key
/etc/kubernetes/pki/etcd/ca.crt
/etc/kubernetes/pki/etcd/ca.key
/etc/kubernetes/admin.conf


//
# USER=ubuntu # root ? (/root/  )
# CONTROL_PLANE_IPS="10.0.0.7 10.0.0.8"
# for host in ${CONTROL_PLANE_IPS}; do
    scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:etcd-ca.crt
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:etcd-ca.key
    scp /etc/kubernetes/admin.conf "${USER}"@$host:
  done
```

Master Node 2 Bootstrap

Master Node 1 Master Node 2 Bootstrap .

copy move sh .

```
//
mkdir -p /etc/kubernetes/pki/etcd
mv /root/ca.crt /etc/kubernetes/pki/
mv /root/ca.key /etc/kubernetes/pki/
mv /root/sa.pub /etc/kubernetes/pki/
mv /root/sa.key /etc/kubernetes/pki/
mv /root/front-proxy-ca.crt /etc/kubernetes/pki/
mv /root/front-proxy-ca.key /etc/kubernetes/pki/
mv /root/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
mv /root/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
mv /root/admin.conf /etc/kubernetes/admin.conf

# kubeadm alpha phase certs all --config kubeadm-config.yaml
# kubeadm alpha phase kubelet config write-to-disk --config kubeadm-config.yaml
# kubeadm alpha phase kubelet write-env-file --config kubeadm-config.yaml
# kubeadm alpha phase kubeconfig kubelet --config kubeadm-config.yaml
# systemctl start kubelet
```

Master Node 2 Cluster

```
# export CP0_IP={Master Node 1 IP}
# export CP0_HOSTNAME={Master Node 1 Host Name}
# export CP1_IP={Master Node 2 IP}
# export CP1_HOSTNAME={Master Node 2 Host Name}
# export KUBECONFIG=/etc/kubernetes/admin.conf
# kubectl exec -n kube-system etcd-${CP0_HOSTNAME} -- etcdctl --ca-file /etc/kubernetes/pki/etcd/ca.crt --
cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc/kubernetes/pki/etcd/peer.key --
endpoints=https://${CP0_IP}:2379 member add ${CP1_HOSTNAME} https://${CP1_IP}:2380
# kubeadm alpha phase etcd local --config kubeadm-config.yaml

//Master Node 2  Master Node 1
# kubeadm alpha phase kubeconfig all --config kubeadm-config.yaml
# kubeadm alpha phase controlplane all --config kubeadm-config.yaml
# kubeadm alpha phase mark-master --config kubeadm-config.yaml
```

)

kubelet.service - kubelet: The Kubernetes Node Agent

Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset: disabled)

Drop-In: /etc/systemd/system/kubelet.service.d

10-kubeadm.conf

Active: active (running) since 2018-10-01 16:55:10 KST; 35s ago

Docs: https://kubernetes.io/docs/

Main PID: 16141 (kubelet)

Tasks: 26

Memory: 33.0M

CGroup: /system.slice/kubelet.service

16141 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/...

10 01 16:55:45... kubelet[16141]: E1001 16:55:45.360340 16141 kubelet.go:2236] node "..." not found

10 01 16:55:45... kubelet[16141]: E1001 16:55:45.460682 16141 kubelet.go:2236] node "..." not found

...

Hint: Some lines were ellipsized, use -l to show in full.

Master Node 3 Bootstrap

Master Node 3 Bootstrap Master Node 2 .

```
//
mkdir -p /etc/kubernetes/pki/etcd
mv /root/ca.crt /etc/kubernetes/pki/
mv /root/ca.key /etc/kubernetes/pki/
mv /root/sa.pub /etc/kubernetes/pki/
mv /root/sa.key /etc/kubernetes/pki/
mv /root/front-proxy-ca.crt /etc/kubernetes/pki/
mv /root/front-proxy-ca.key /etc/kubernetes/pki/
mv /root/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
mv /root/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
mv /root/admin.conf /etc/kubernetes/admin.conf

# kubeadm alpha phase certs all --config kubeadm-config.yaml
# kubeadm alpha phase kubelet config write-to-disk --config kubeadm-config.yaml
# kubeadm alpha phase kubelet write-env-file --config kubeadm-config.yaml
# kubeadm alpha phase kubeconfig kubelet --config kubeadm-config.yaml
# systemctl start kubelet
```

Master Node 3 Cluster

```
# export CP0_IP={Master Node 1 IP}
# export CP0_HOSTNAME={Master Node 1 Host Name}
# export CP2_IP={Master Node 3 IP}
# export CP2_HOSTNAME={Master Node 3 Host Name}
# export KUBECONFIG=/etc/kubernetes/admin.conf# kubectl exec -n kube-system etcd-${CP0_HOSTNAME} -- etcdctl
--ca-file /etc/kubernetes/pki/etcd/ca.crt --cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc
/kubernetes/pki/etcd/peer.key --endpoints=https://${CP0_IP}:2379 member add ${CP2_HOSTNAME}
https://${CP2_IP}:2380

# kubeadm alpha phase etcd local --config kubeadm-config.yaml

//Master Node 3  Master Node 1
# kubeadm alpha phase kubeconfig all --config kubeadm-config.yaml
# kubeadm alpha phase controlplane all --config kubeadm-config.yaml
# kubeadm alpha phase mark-master --config kubeadm-config.yaml
```

K8s Cluster Network Addon (only Master 1)

```
// /proc/sys/net/bridge/bridge-nf-call-iptables  1
# cat /proc/sys/net/bridge/bridge-nf-call-iptables
// 1 ,
# sysctl net.bridge.bridge-nf-call-iptables=1

# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
# kubectl get nodes
NAME       STATUS  ROLES     AGE       VERSION
hostname   Ready   master    15h       v1.11.2
hostname   Ready   master    15h       v1.11.2
hostname   Ready   master    15h       v1.11.2
# kubectl get pods --all-namespaces
kube-system   coredns-78fcdf6894-969pp                     1/1       Running   0          15h
```

K8s Sub Node Join

```
# echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
//
# cat /proc/sys/net/ipv4/ip_forward
# kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hash sha256:<hash>
```

)

[preflight] running pre-flight checks
[WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the following required kernel modules are not loaded:
[ip_vs ip_vs_rr ip_vs_wrr ip_vs_sh] or no builtin kernel ipvs support: map[ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{} ip_vs_sh:{} nf_conntrack_ipv4:{}]
you can solve this problem with following methods:
1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

I1010 13:06:51.855190 19045 kernel_validator.go:81] Validating kernel version
I1010 13:06:51.855343 19045 kernel_validator.go:96] Validating kernel config
[discovery] Trying to connect to API Server...
[discovery] Created cluster-info discovery client, requesting info from...
[discovery] Requesting info from ... 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...
[discovery] Successfully established connection with API Server ...
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.11" ConfigMap in the kube-system namespace
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object ... as an annotation

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

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

Reference

K8s HA : https://kubernetes.io/docs/setup/independent/high-availability/#first-steps-for-both-methods

HAproxy : https://cbonte.github.io/haproxy-dconv/1.8/configuration.html

Weavnet : https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#pod-network