

K8S 高可用集群搭建

```
#配置/etc/hosts
#####
cat >> /etc/hosts <<EOF
10.3.7.241 k8s-master1
10.3.7.242 k8s-master2
10.3.7.243 k8s-node1
EOF

name=(k8s-master1 k8s-master2 k8s-node1)
for i in ${name[@]};do scp /etc/hosts root@$i:/etc/;done
#####
#免密登录
#####
yum install -y expect
ssh-keygen -t rsa -P "" -f /root/.ssh/id_rsa
export mypass=DHMPzzd388
name=(k8s-master1 k8s-master2 k8s-node1)

for i in ${name[@]};do expect -c "
spawn ssh-copy-id -i /root/.ssh/id_rsa.pub root@$i
expect {
    \"*yes/no*" {send "\r"; exp_continue}
    \"*password*" {send "\$mypass\r"; exp_continue}
    \"*Password*" {send "\$mypass\r";}
}";done

#####
#关闭防火墙、selinux、swap
#####
systemctl stop firewalld
systemctl disable firewalld
setenforce 0
sed -i 's/^SELINUX=.*SELINUX=disabled/' /etc/selinux/config
iptables -F && iptables -X && iptables -F -t nat && iptables -X -t nat
iptables -P FORWARD ACCEPT
swapoff -a
sed -i '/ swap / s/^(.*)$/#\1/g' /etc/fstab

#####
#查看操作系统内核版本，尽量大于4.1，未达到需要升级
#####
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
yum --enablerepo=elrepo-kernel install -y kernel-lt
grub2-set-default 0
grub2-mkconfig -o /etc/grub2.cfg
grubby --default-kernel
reboot
```

```
#####
```

```
#一些配置
```

```
#####
```

```
cat >/etc/sysctl.d/k8s.conf <<EOF
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
EOF
sysctl --system
```

```
#####
```

```
ipvs
```

```
#####
```

```
yum install -y conntrack ipvsadm ipset
cat >/etc/modules-load.d/ipvs.conf << EOF
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
nf_conntrack
EOF
```

```
modprobe ip_vs_rr nf_conntrack
lsmod | egrep "ip_vs_rr|nf_conntrack"
```

```
#####
```

```
#时间同步
```

```
#####
```

```
yum install -y ntpdate
echo "*/30 * * * * ntpdate time7.aliyun.com >/dev/null 2>&1" >> /
var/spool/cron/root
ntpdate time7.aliyun.com
```

```
#####
```

```
#证书分发工具
```

```
#####
```

```
curl -L -o /usr/local/bin/cfssl https://pkg.cfssl.org/R1.2/
cfssl_linux-amd64
curl -L -o /usr/local/bin/cfssljson https://pkg.cfssl.org/R1.2/
cfssljson_linux-amd64
chmod +x /usr/local/bin/{cfssl,cfssljson}
```

```
#####
```

```
#制作etcd证书,分发证书
```

```
#####
```

```
mkdir ~/cfssl && cd ~/cfssl
```

```
#CA根证书json
```

```
cat > ca-config.json <<EOF
{
```

```

    "signing": {
      "default": {
        "expiry": "87600h"
      },
      "profiles": {
        "kubernetes": {
          "usages": [
            "signing",
            "key encipherment",
            "server auth",
            "client auth"
          ],
          "expiry": "87600h"
        }
      }
    }
  }
}
EOF

```

#基于CA的证书分发证书

```
cat > ca-csr.json <<EOF
```

```

{
  "CN": "kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "GuangZhou",
      "L": "GuangZhou",
      "O": "k8s",
      "OU": "System"
    }
  ],
  "ca": {
    "expiry": "87600h"
  }
}
EOF

```

#制作证书

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

#etcd证书json

```
cat > etcd-csr.json <<EOF
```

```

{
  "CN": "etcd",
  "hosts": [
    "127.0.0.1",
    "10.3.7.241",
    "10.3.7.242",
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  }
}

```

```

    },
    "names": [
        {
            "C": "CN",
            "ST": "GuangZhou",
            "L": "GuangZhou",
            "O": "k8s",
            "OU": "System"
        }
    ]
}
EOF
#制作etcd证书
cfssl gencert -ca=ca.pem \
    -ca-key=ca-key.pem \
    -config=ca-config.json \
    -profile=kubernetes etcd-csr.json | cfssljson -bare etcd

#####
#安装etcd
#####
mkdir -p /data/etcd
mkdir -p /etc/etcd/ssl
cp ~/cfssl/{ca.pem,etcd.pem,etcd-key.pem} /etc/etcd/ssl/
cd /usr/local/src
wget https://github.com/etcd-io/etcd/releases/download/v3.4.1/etcd-
v3.4.1-linux-amd64.tar.gz
tar xf etcd-v3.4.1-linux-amd64.tar.gz
cp etcd-v3.4.1-linux-amd64/{etcd,etcdctl} /usr/local/bin/

#分发etcd etcdctl 给master
ETCDHOSTS=(10.3.7.241 10.3.7.242)
for i in ${ETCDHOSTS[@]}; do scp /usr/local/bin/etcd* root@$i:/usr/
local/bin/; done

#分发etcd证书给master,备master节点需要创建目录mkdir -p /data/etcd mkdir
-p /etc/etcd/ssl

cd ~/csssl
ETCDHOSTS=(10.3.7.241 10.3.7.242)
for i in ${ETCDHOSTS[@]}; do scp -r /etc/etcd root@$i:/etc/; done

#配置etcd service文件

ETCDHOSTS=(10.3.7.241 10.3.7.242)
NAMES=("etcd-0" "etcd-1")

for i in "${!ETCDHOSTS[@]}"; do
HOST=${ETCDHOSTS[$i]}
NAME=${NAMES[$i]}

```

```

cat << EOF > /tmp/etcd.service.${HOST}
[Unit]
Description=Etcd Server
After=network.target
After=network-online.target
Wants=network-online.target
[Service]
Type=notify
ExecStart=/usr/local/bin/etcd \\\
--data-dir=/data/etcd \\\
--name=${NAME} \\\
--trusted-ca-file=/etc/etcd/ssl/ca.pem \\\
--cert-file=/etc/etcd/ssl/etcd.pem \\\
--key-file=/etc/etcd/ssl/etcd-key.pem \\\
--peer-trusted-ca-file=/etc/etcd/ssl/ca.pem \\\
--peer-cert-file=/etc/etcd/ssl/etcd.pem \\\
--peer-key-file=/etc/etcd/ssl/etcd-key.pem \\\
--peer-client-cert-auth \\\
--client-cert-auth \\\
--listen-peer-urls=https://${HOST}:2380 \\\
--initial-advertise-peer-urls=https://${HOST}:2380 \\\
--listen-client-urls=https://${HOST}:2379,http://127.0.0.1:2379 \\\
--advertise-client-urls=https://${HOST}:2379 \\\
--initial-cluster-token=etcd-cluster-0 \\\
--initial-cluster=${NAMES[0]}=https://${ETCDHOSTS[0]}:2380,$
{NAMES[1]}=https://${ETCDHOSTS[1]}:2380 \\\
--initial-cluster-state=new
Restart=on-failure
RestartSec=5
LimitNOFILE=65536
[Install]
WantedBy=multi-user.target
EOF
done

```

#分发service文件

```

ETCDHOSTS=(10.3.7.241 10.3.7.242)
for i in ${ETCDHOSTS[@]};do
    scp /tmp/etcd.service.$i root@$i:/etc/systemd/system/
    etcd.service;
done

```

#到每个master去启动etcd服务，如果配置成功可通 ps -ef|grep etcd查看进程

```

systemctl daemon-reload
systemctl enable etcd
systemctl start etcd

```

#验证每个etcd节点的健康情况

```

ETCDCTL_API=3 etcdctl \
--cacert=/etc/etcd/ssl/ca.pem \
--cert=/etc/etcd/ssl/etcd.pem \
--key=/etc/etcd/ssl/etcd-key.pem \
--endpoints=https://${ETCDHOSTS[0]}:2379,https://${
ETCDHOSTS[1]}:2379 endpoint health

```

```

#安装haproxy 负责LB的一个服务
yum install -y haproxy
cat > /etc/haproxy/haproxy.cfg <<EOF
global
    log                127.0.0.1 local2
    chroot              /var/lib/haproxy
    pidfile             /var/run/haproxy.pid
    maxconn             4000
    user                haproxy
    group                haproxy
    daemon
    stats socket /var/lib/haproxy/stats

defaults
    mode                http
    log                  global
    option                httplog
    option                dontlognull
    option http-server-close
    option forwardfor    except 127.0.0.0/8
    option                redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn              3000

frontend kubernetes-apiserver
    mode                tcp
    bind                *:8443
    option                tcplog
    default_backend      kubernetes-apiserver

backend kubernetes-apiserver
    mode                tcp
    balance              roundrobin
    server               k8s01 10.3.7.241:6443 check
    server               k8s02 10.3.7.242:6443 check

listen stats
    bind                *:1080
    stats auth           admin:sunday
    stats refresh        5s
    stats realm          HAProxy\ Statistics
    stats uri            /haproxy-stats
EOF

```

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

#分发

```
ETCDHOSTS=(10.3.7.241 10.3.7.242)
```

```
for i in ${ETCDHOSTS[@]}; do scp /etc/haproxy/haproxy.cfg root@$i:/
etc/haproxy/haproxy.cfg; done
```

#安装keepalived,使用虚拟IP实现高可用无感切换, 虚拟IP使用同网段不能ping通的IP即可

#MASTER 为主节点标识 BACKUP为备节点标识, priority 为权重 由权重选举新的master

```
yum install -y keepalived psmisc
```

```
cat > /etc/keepalived/keepalived.conf <<EOF
```

```
! Configuration File for keepalived
```

```
global_defs {
```

```
    router_id keepalived
```

```
}
```

```
vrrp_script check_haproxy {
```

```
    script "killall -0 haproxy"
```

```
    interval 2
```

```
    weight -3
```

```
    fall 2
```

```
    rise 2
```

```
}
```

```
vrrp_instance VI_1 {
```

```
    state MASTER
```

```
    interface ens192
```

```
    virtual_router_id 51
```

```
    priority 100
```

```
    advert_int 1
```

```
    authentication {
```

```
        auth_type PASS
```

```
        auth_pass Sunday
```

```
    }
```

```
    virtual_ipaddress {
```

```
        10.3.7.245
```

```
    }
```

```
    track_script {
```

```
        check_haproxy
```

```
    }
```

```
}
```

```
EOF
```

#修改参数后分发

```
scp /etc/keepalived/keepalived.conf root@10.3.7.242:/etc/keepalived/
keepalived.conf
```

```
ssh root@10.3.7.242 "sed -i -e 's#MASTER#BACKUP#g' -e 's#priority
100#priority 99#g' /etc/keepalived/keepalived.conf"
```

```
#ETCDHOSTS=(10.3.7.241 10.3.7.242)
#for i in ${ETCDHOSTS[@]}; do
#   scp /etc/keepalived/keepalived.conf root@$i:/etc/keepalived/
keepalived.conf
#   ssh root>${ETCDHOSTS[1]} "sed -i -e 's#MASTER#BACKUP#g' -e
's#priority 100#priority 99#g' /etc/keepalived/#keepalived.conf"
#done
```

```
systemctl start keepalived.service
systemctl enable keepalived.service
```

```
#####
#安装docker 18.09.8
#####
```

```
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/
linux/centos/docker-ce.repo
yum install -y docker-ce-18.09.8
```

#添加cgroup,据说会更快

```
mkdir /etc/docker
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "registry-mirrors": ["https://hub-mirror.c.163.com"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
systemctl enable docker --now
```

```
#####
#安装k8s 1.15.4
#####
```

```
cat << EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-
el7-x86_64/
enabled=1
gpgcheck=1
gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg
https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg
EOF
systemctl enable --now kubelet
```



```
#从外部添加etcd
cat << EOF > kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
controlPlaneEndpoint: "10.3.7.245:8443"
etcd:
  external:
    endpoints:
      - https://10.3.7.241:2379
      - https://10.3.7.242:2379
    caFile: /etc/etcd/ssl/ca.pem
    certFile: /etc/etcd/ssl/etcd.pem
    keyFile: /etc/etcd/ssl/etcd-key.pem
kubernetesVersion: v1.15.4
imageRepository: gcr.azk8s.cn/google_containers
networking:
  #podSubnet: 192.168.0.0/16 #默认calico网段与内网有重叠
  podSubnet: 10.244.0.0/16
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
mode: ipvs
EOF
```

#初始化k8s, 这里会生成两个命令, 上面是加入master的, 下面是加入node的, 注意时效是2小时, 超过两小时要重新生成命令

```
kubeadm init --config kubeadm-config.yaml --upload-certs
```

```
kubeadm join 10.3.7.245:8443 --token q5frmj.9a9kihce2ph286nw      --
discovery-token-ca-cert-hash
sha256:7ae1437378fb44cd50756764674706d8f5177cdf53f3125515e5353848c3a
c34      --control-plane --certificate-key
34f6bcc0b3185f837f576938d9246524660de484e4fe8700231e3fd0b5f88738
```

```
kubeadm join 10.3.7.245:8443 --token q5frmj.9a9kihce2ph286nw \
--discovery-token-ca-cert-hash
sha256:7ae1437378fb44cd50756764674706d8f5177cdf53f3125515e5353848c3a
c34
```

完成以后, 在master节点查看

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8s-master1	NotReady	master	7m1s	v1.15.4
k8s-master2	NotReady	master	3m17s	v1.15.4
k8s-node1	NotReady	<none>	8s	v1.15.4

可以看到这里的状态是NotReady, 安装网络插件以后就会正常了, 这里我使用calico

```
#####
```

#安装calico,这里将默认的网段替换一下,防止抢占

#####

```
wget https://docs.projectcalico.org/v3.8/manifests/calico.yaml
```

```
POD_CIDR="10.244.0.0/16"
```

```
sed -i -e "s?192.168.0.0/16?$POD_CIDR?g" calico.yaml
```

```
kubectl apply -f calico.yaml
```

等个几分钟,视网络状况而定

```
k8s-master1    Ready    master    15m    v1.15.4
```

```
k8s-master2    Ready    master    11m    v1.15.4
```

```
k8s-node1      Ready    <none>    8m22s  v1.15.4
```

OK

这样一个高可用K8S集群就搭建好了,下面开始安装一些基础服务

#####

#这里一个node太少了,而我的服务器又不够用,所以启动master作为节点

#取消限制 是master成为node

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

#禁止master部署pod

```
kubectl taint nodes k8s node-role.kubernetes.io/
```

```
master=true:NoSchedule
```

#####

#####

#安装dashboard 一个带UI界面查看k8s集群的工具

#####

```
wget https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/
src/deploy/recommended/kubernetes-dashboard.yaml
```

打开这个文件,修改镜像,新增NodePort,我配置的31001,所以使用VIP去访问这个端口就可以了

```
images:gcr.azk8s.cn/google_containers/kubernetes-dashboard-
amd64:v1.10.1
```

```
kubectl apply -f kubernetes-dashboard.yaml
```

```
kind: Service
```

```
apiVersion: v1
```

```
metadata:
```

```
  labels:
```

```
    k8s-app: kubernetes-dashboard
```

```
  name: kubernetes-dashboard
```

```
  namespace: kube-system
```

```
spec:
```

```
  type: NodePort
```

```
  ports:
```

```
    - port: 443
```



```
#####
#配置指标服务，可用于POD伸缩
#####
```

```
git clone https://github.com/kubernetes-incubator/metrics-server.git
```

```
vim metrics-server/deploy/1.8+/metrics-server-deployment.yaml
```

改个镜像地址,新增几个配置

```
command:
- /metrics-server
- --metric-resolution=30s #kubelet 采集数据的周期
- --kubelet-preferred-address-
```

types=InternalIP,Hostname,InternalDNS,ExternalDNS,ExternalIP # 优先使用 InternalIP 来访问 kubelet,避免节点名称没有 DNS 解析

```
- --kubelet-insecure-tls
image: gcr.azk8s.cn/google_containers/metrics-server-
amd64:v0.3.5
```

```
kubectl create -f metrics-server/deploy/1.8+/  
等几十秒
```

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
k8s-master1	234m	11%	1762Mi	7%
k8s-master2	233m	11%	1366Mi	6%
k8s-node1	117m	5%	929Mi	4%

```
#####
efk日志分析系统
#####
```

```
mkdir elk && cd elk
```

```
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/es-service.yaml
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/es-statefulset.yaml
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/fluentd-es-configmap.yaml
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/fluentd-es-ds.yaml
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/kibana-deployment.yaml
wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/
cluster/addons/fluentd-elasticsearch/kibana-service.yaml
```

```
kubectl apply -f .
```

```
kubectl get pods -n kube-system -o wide|grep -E 'elasticsearch|
fluentd|kibana'
```

等待容器构建完成

elasticsearch-logging-0			1/1	Running	0
27m	10.244.36.69	k8s-node1	<none>	<none>	

elasticsearch-logging-1		1/1	Running	0
10m	10.244.159.130	k8s-master1	<none>	<none>
fluentd-es-v2.7.0-2tbb9		1/1	Running	0
27m	10.244.36.70	k8s-node1	<none>	<none>
fluentd-es-v2.7.0-4kdh6		1/1	Running	0
27m	10.244.224.4	k8s-master2	<none>	<none>
fluentd-es-v2.7.0-gbf6s		1/1	Running	0
27m	10.244.159.129	k8s-master1	<none>	<none>
kibana-logging-7b97c764f6-jbz7p		1/1	Running	8
27m	10.244.36.71	k8s-node1	<none>	

kubectl cluster-info可用看到

Kubernetes master is running at https://10.3.7.245:8443
 Elasticsearch is running at https://10.3.7.245:8443/api/v1/
 namespaces/kube-system/services/elasticsearch-logging/proxy
 Kibana is running at https://10.3.7.245:8443/api/v1/namespaces/kube-
 system/services/kibana-logging/proxy
 KubeDNS is running at https://10.3.7.245:8443/api/v1/namespaces/
 kube-system/services/kube-dns:dns/proxy
 Metrics-server is running at https://10.3.7.245:8443/api/v1/
 namespaces/kube-system/services/https:metrics-server:/proxy

kubectl proxy --address='10.3.7.245' --port=8086 --accept-
 hosts='^*\$' &

再用网页访问8086端口就行了

http://10.3.7.245:8086/api/v1/namespaces/kube-system/services/
 kibana-logging/proxy

#需要用ingress-nginx就自行搭建，也可以用NodePort，但是不推荐使用，还是用LB好点