# Kubeadm安装

## 机器划分

| 主机名 | IP地址 | 角色 | 配置 |
| --- | --- | --- | --- |
| k8s-master01 | 192.168.0.107 | Master | 2C2G 40G |
| k8s-master02 | 192.168.0.108 | Master | 2C2G 40G |
| k8s-master03 | 192.168.0.109 | Master | 2C2G 40G |
| k8s-master-lb | 192.168.0.236 | keepalived虚拟 IP | 2C2G 40G |
| k8s-node01 | 192.168.0.110 | Node-1 | 2C2G 40G |
| k8s-node02 | 192.168.0.111 | Node-2 | 2C2G 40G |

说明：不要使用中文的环境，还有不要克隆的虚拟机。生产环境，使用二进制安装。

VIP（虚拟IP）不要和公司内网IP重复，首先去ping一下，不通才可用。VIP需要和主机在同一个局域网内！

| 信息 | 备注 |
| --- | --- |
| 系统版本 | CentOS 7.9 |
| Docker版本 | 19.03.x |
| K8s版本 | 1.20.x |
| Pod网段 | 172.168.0.0/12 |
| Service网段 | 10.96.0.0/12 |

k8s官网：https://kubernetes.io/docs/setup/

最新版本高可用安装：https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/

## 配置

所有节点配置hosts,修改 /etc/hosts 如下：

```
[root@k8s-master01 ~]# cat /etc/hosts
192.168.0.107 k8s-master01
192.168.0.108 k8s-master02
192.168.0.109 k8s-master03
192.168.0.236 k8s-master-lb # 如果不是高可用集群，该IP为Master01的IP
192.168.0.110 k8s-node01
192.168.0.111 k8s-node02
```

yum源配置

```
curl -o /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-7.repo
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager --add-repo https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
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
repo_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
sed -i -e '/mirrors.cloud.aliyuncs.com/d' -e '/mirrors.aliyuncs.com/d'
/etc/yum.repos.d/CentOS-Base.repo
```

必备工具安装

```
yum install wget jq psmisc vim net-tools telnet yum-utils device-mapper-persistent-data
lvm2 git -y
```

所有节点关闭防火墙、selinux、dnsmasq、swap。服务器配置如下：

```
systemctl disable --now firewalld
systemctl disable --now dnsmasq
systemctl disable --now NetworkManager

setenforce 0
sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/sysconfig/selinux
sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/selinux/config
```

关闭swap分区

```
swapoff -a && sysctl -w vm.swappiness=0
sed -ri '/^[^#]*swap/s@^@#@' /etc/fstab
```

安装ntpdate

```
rpm -ivh http://mirrors.wlnmp.com/centos/wlnmp-release-centos.noarch.rpm
yum install ntpdate -y
```

所有节点同步时间。时间同步配置如下：

```
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
echo 'Asia/Shanghai' >/etc/timezone
ntpdate time2.aliyun.com
```

加入到crontab

```
*/5 * * * * ntpdate time2.aliyun.com
```

所有节点配置limit：

```
ulimit -SHn 65535

vim /etc/security/limits.conf
# 末尾添加如下内容
* soft nofile 655360
* hard nofile 131072
* soft nproc 655350
* hard nproc 655350
* soft memlock unlimited
* hard memlock unlimited
```

Master01节点免密钥登录其他节点：

```
ssh-keygen -t rsa
for i in k8s-master01 k8s-master02 k8s-master03 k8s-node01 k8s-node02;do ssh-copy-id -i
.ssh/id_rsa.pub $i;done
```

下载安装所有的源码文件：

```
cd /root/ ; git clone https://github.com/dotbalo/k8s-ha-install.git
```

所有节点升级系统并重启：

```
yum update -y --exclude=kernel*  && reboot

# 查看安装的版本
[root@k8s-master01 ~]# cat /etc/redhat-release
CentOS Linux release 7.9.2009 (Core)
```

# 内核配置

centos7 需要升级内核至4.18+， 本地神经的版本为4.19

在Master01节点下载内核

```
cd /root
wget http://193.49.22.109/elrepo/kernel/el7/x86_64/RPMS/kernel-ml-devel-4.19.12-1.el7.elrepo.x86_64.rpm
wget http://193.49.22.109/elrepo/kernel/el7/x86_64/RPMS/kernel-ml-4.19.12-1.el7.elrepo.x86_64.rpm
```

在master01拷贝到其他节点

```
for i in k8s-master01 k8s-master02 k8s-master03 k8s-node01 k8s-node02; do scp kernel-ml-devel-4.19.12-1.el7.elrepo.x86_64.rpm kernel-ml-4.19.12-1.el7.elrepo.x86_64.rpm $i:/root/ ; done
```

所有节点安装内核

```
cd /root/ && yum localinstall -y kernel-ml*
```

所有节点更改内核启动顺序

```
grub2-set-default 0 && grub2-mkconfig -o /etc/grub2.cfg

grubby --args="user_namespace.enable=1" --update-kernel="$(grubby --default-kernel)"
```

检查默认内核是不是4.19

```
[root@k8s-master01 ~]# grubby --default-kernel
/boot/vmlinuz-4.19.12-1.el7.elrepo.x86_64
```

所有节点重启，然后检查内核是不是4.19

```
[root@k8s-master01 ~]# uname -a
```

所有节点安装ipvsadm

```
yum install ipvsadm ipset sysstat conntrack libseccomp -y
```

所有节点配置ipvs模块, 在内核4.19+版本 nf_conntrack_ipv4 已经修改为 nf_conntrack， 4.18 以下使用 nf_conntrack_ipv4 即可

```
vim /etc/modules-load.d/ipvs.conf
  # 加入以下内容
ip_vs
ip_vs_lc
ip_vs_wlc
ip_vs_rr
ip_vs_wrr
ip_vs_lblc
ip_vs_lblcr
ip_vs_dh
ip_vs_sh
ip_vs_fo
ip_vs_nq
ip_vs_sed
ip_vs_ftp
ip_vs_sh
nf_conntrack
ip_tables
ip_set
xt_set
ipt_set
ipt_rpfilter
ipt_REJECT
ipip
```

然后执行

```
systemctl enable --now systemd-modules-load.service
```

开启一些k8s集群中必须的内核参数，所有节点配置k8s内核

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

```
vm.panic_on_oom=0
fs.inotify.max_user_watches=89100
fs.file-max=52706963
fs.nr_open=52706963
net.netfilter.nf_conntrack_max=2310720

net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_keepalive_probes = 3
net.ipv4.tcp_keepalive_intvl =15
net.ipv4.tcp_max_tw_buckets = 36000
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_max_orphans = 327680
net.ipv4.tcp_orphan_retries = 3
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 16384
net.ipv4.ip_conntrack_max = 65536
net.ipv4.tcp_max_syn_backlog = 16384
net.ipv4.tcp_timestamps = 0
net.core.somaxconn = 16384
EOF
sysctl --system
```

所有节点重启，检查是否加载

```
reboot
lsmod | grep --color=auto -e ip_vs -e nf_conntrack
```

# 基本组件安装

所有节点安装Docker-ce 19.03

```
yum install docker-ce-19.03.* -y
```

**温馨提示**

由于新版kubelet 建议使用systemd，所以可以把 docker 的 CgroupDriver 改成systemd

```
 mkdir /etc/docker
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```

所有节点设置开机自启动Docker

```
systemctl daemon-reload && systemctl enable --now docker
```

安装k8s组件

```
yum list kubeadm.x86_64 --showduplicates | sort -r
```

所有节点安装最新版本kubeadm

```
yum install kubeadm -y
```

默认配置的pause镜像使用gcr.io仓库，国内可能无法访问，所以这里配置Kubelet使用阿里云的pause镜像：

```
cat >/etc/sysconfig/kubelet<<EOF
KUBELET_EXTRA_ARGS="--pod-infra-container-image=registry.cn-
hangzhou.aliyuncs.com/google_containers/pause-amd64:3.2"
EOF
```

设置Kubelet开机自启动

```
systemctl daemon-reload
systemctl enable --now kubelet
```

# 高可用组件安装

所有Master节点通过yum安装HAProxy和KeepAlived：

```
yum install keepalived haproxy -y
```

所有Master节点配置HAProxy（详细配置参考HAProxy文档，所有Master节点的HAProxy配置相同）：

```
[root@k8s-master01 etc]# mkdir /etc/haproxy
[root@k8s-master01 etc]# vim /etc/haproxy/haproxy.cfg
global
  maxconn  2000
  ulimit-n  16384
  log  127.0.0.1 local0 err
  stats timeout 30s

defaults
  log global
  mode  http
  option  httplog
  timeout connect 5000
  timeout client  50000
```

```
  timeout server  50000
  timeout http-request 15s
  timeout http-keep-alive 15s

frontend monitor-in
  bind *:33305
  mode http
  option httplog
  monitor-uri /monitor

frontend k8s-master
  bind 0.0.0.0:16443
  bind 127.0.0.1:16443
  mode tcp
  option tcplog
  tcp-request inspect-delay 5s
  default_backend k8s-master

backend k8s-master
  mode tcp
  option tcplog
  option tcp-check
  balance roundrobin
  default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250
maxqueue 256 weight 100
  server k8s-master01 192.168.0.107:6443  check
  server k8s-master02 192.168.0.108:6443  check
  server k8s-master03 192.168.0.109:6443  check
```

所有Master节点配置KeepAlived，配置不一样，注意区分
注意每个节点的IP和网卡（interface参数）
Master01节点的配置：

```
[root@k8s-master01 etc]# mkdir /etc/keepalived

[root@k8s-master01 ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
global_defs {
    router_id LVS_DEVEL
script_user root
    enable_script_security
}
vrrp_script chk_apiserver {
    script "/etc/keepalived/check_apiserver.sh"
    interval 5
    weight -5
    fall 2
rise 1
}
```

```
vrrp_instance VI_1 {
    state MASTER
    interface ens33
    mcast_src_ip 192.168.0.107
    virtual_router_id 51
    priority 101
    advert_int 2
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
    }
    virtual_ipaddress {
        192.168.0.236
    }
    track_script {
       chk_apiserver
    }
}
```

Master02节点的配置：

```
! Configuration File for keepalived
global_defs {
    router_id LVS_DEVEL
script_user root
    enable_script_security
}
vrrp_script chk_apiserver {
    script "/etc/keepalived/check_apiserver.sh"
   interval 5
    weight -5
    fall 2
rise 1
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    mcast_src_ip 192.168.0.108
    virtual_router_id 51
    priority 100
    advert_int 2
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
    }
    virtual_ipaddress {
        192.168.0.236
    }
    track_script {
```

```
       chk_apiserver
    }
}
```

Master03节点的配置：

```
! Configuration File for keepalived
global_defs {
    router_id LVS_DEVEL
script_user root
    enable_script_security
}
vrrp_script chk_apiserver {
    script "/etc/keepalived/check_apiserver.sh"
 interval 5
    weight -5
    fall 2
rise 1
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    mcast_src_ip 192.168.0.109
    virtual_router_id 51
    priority 100
    advert_int 2
    authentication {
        auth_type PASS
        auth_pass K8SHA_KA_AUTH
    }
    virtual_ipaddress {
        192.168.0.236
    }
    track_script {
       chk_apiserver
    }
}
```

所有master节点，配置KeepAlived健康检查文件：

```
[root@k8s-master01 keepalived]# cat /etc/keepalived/check_apiserver.sh
#!/bin/bash

err=0
for k in $(seq 1 3)
do
    check_code=$(pgrep haproxy)
```

```
    if [[ $check_code == "" ]]; then
        err=$(expr $err + 1)
        sleep 1
        continue
    else
        err=0
        break
    fi
done

if [[ $err != "0" ]]; then
    echo "systemctl stop keepalived"
    /usr/bin/systemctl stop keepalived
    exit 1
else
    exit 0
fi


chmod +x /etc/keepalived/check_apiserver.sh
启动haproxy和keepalived
[root@k8s-master01 keepalived]# systemctl daemon-reload
[root@k8s-master01 keepalived]# systemctl enable --now haproxy
[root@k8s-master01 keepalived]# systemctl enable --now keepalived
```

测试VIP

```
[root@k8s-master01 ~]# ping 192.168.0.236 -c 4
PING 192.168.0.236 (192.168.0.236) 56(84) bytes of data.
64 bytes from 192.168.0.236: icmp_seq=1 ttl=64 time=0.464 ms
64 bytes from 192.168.0.236: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 192.168.0.236: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 192.168.0.236: icmp_seq=4 ttl=64 time=0.063 ms
```

```
# 日志查看
tail -f /var/log/messages
```

# 集群初始化

Master01节点创建new.yaml配置文件如下：

```
vim /root/new.yaml
```

```
apiVersion: kubeadm.k8s.io/v1beta2
```

```
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: 7t2weq.bjbawausm0jaxury
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.0.107
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: k8s-master01
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  certSANs:
  - 192.168.0.236
  timeoutForControlPlane: 4m0s
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controlPlaneEndpoint: 192.168.0.236:16443
controllerManager: {}
dns:
  type: CoreDNS
etcd:
  local:
    dataDir: /var/lib/etcd
imageRepository: registry.cn-hangzhou.aliyuncs.com/google_containers
kind: ClusterConfiguration
kubernetesVersion: v1.20.0
networking:
  dnsDomain: cluster.local
  podSubnet: 172.168.0.0/12
  serviceSubnet: 10.96.0.0/12
scheduler: {}
```

**注意：如果不是高可用集群，192.168.0.236:16443改为master01的地址，16443改为apiserver的端口，默认是6443，注意更改v1.20.0为自己服务器kubeadm的版本：kubeadm version**

将new.yaml文件复制到其他master节点，之后所有Master节点提前下载镜像，可以节省初始化时间：

```
kubeadm config images pull --config /root/new.yaml
```

所有节点设置开机自启动kubelet

```
systemctl enable --now kubelet（如果启动失败无需管理，初始化成功以后即可启动）
```

Master01节点初始化，初始化以后会在/etc/kubernetes目录下生成对应的证书和配置文件，之后其他Master节点加入Master01即可：

```
kubeadm init --config /root/new.yaml  --upload-certs
```

初始化成功以后，会产生Token值，用于其他节点加入时使用，因此要记录下初始化成功生成的token值（令牌值）：

```
Your Kubernetes control-plane 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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 the control-plane node running the following command on
each as root:

  kubeadm join 192.168.0.236:16443 --token 7t2weq.bjbawausm0jaxury \
    --discovery-token-ca-cert-hash
sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908 \
    --control-plane --certificate-key
ac2854de93aaabdf6dc440322d4846fc230b290c818c32d6ea2e500fc930b0aa

Please note that the certificate-key gives access to cluster sensitive data, keep it
secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.236:16443 --token 7t2weq.bjbawausm0jaxury \
    --discovery-token-ca-cert-hash
sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908
```

Master01节点配置环境变量，用于访问Kubernetes集群：

```
cat <<EOF >> /root/.bashrc
export KUBECONFIG=/etc/kubernetes/admin.conf
EOF
source /root/.bashrc
```

查看节点状态：

```
 [root@k8s-master01 ~]# kubectl get nodes
NAME           STATUS     ROLES                  AGE   VERSION
k8s-master01   NotReady   control-plane,master   74s   v1.20.0
```

采用初始化安装方式，所有的系统组件均以容器的方式运行并且在kube-system命名空间内，此时可以查看Pod状态：

```
[root@k8s-master01 ~]# kubectl get pods -n kube-system -o wide
NAME                                   READY     STATUS    RESTARTS   AGE       IP
         NODE
coredns-777d78ff6f-kstsz               0/1       Pending   0          14m       <none>
         <none>
coredns-777d78ff6f-rlfr5               0/1       Pending   0          14m       <none>
         <none>
etcd-k8s-master01                      1/1       Running   0          14m
192.168.0.107   k8s-master01
kube-apiserver-k8s-master01            1/1       Running   0          13m
192.168.0.107   k8s-master01
kube-controller-manager-k8s-master01   1/1       Running   0          13m
192.168.0.107   k8s-master01
kube-proxy-8d4qc                       1/1       Running   0          14m
192.168.0.107   k8s-master01
kube-scheduler-k8s-master01            1/1       Running   0          13m
192.168.0.107   k8s-master01
```

# 高可用Master

初始化其他master加入集群, master02, 03加上

```
kubeadm join 192.168.0.236:16443 --token 7t2weq.bjbawausm0jaxury \
    --discovery-token-ca-cert-hash
sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908 \
    --control-plane --certificate-key
ac2854de93aaabdf6dc440322d4846fc230b290c818c32d6ea2e500fc930b0aa
```

下面是补充：

```
# Token过期后生成新的token:
[root@k8s-master01 ~]# kubeadm token create --print-join-command
kubeadm join 192.168.1.236:16443 --token 8k8qzk.d43ed9gfgw1st3xi     --discovery-token-ca-cert-hash sha256:3aa4cf3c52c1956cb86d2911fe0f6b8898bfa43c06966b2f1095e5000a00d1a4

# Master需要生成--certificate-key
[root@k8s-master01 ~]# kubeadm init phase upload-certs  --upload-certs
[upload-certs] Storing the certificates in Secret "kubeadm-certs" in the "kube-system" Namespace
[upload-certs] Using certificate key:
43c5695789c0dc4433f480a05683d55887e836b71b452b407138d8dd54cad937
```

# 添加Node节点

```
kubeadm join 192.168.0.236:16443 --token 7t2weq.bjbawausm0jaxury \
    --discovery-token-ca-cert-hash
sha256:8c92ecb336be2b9372851a9af2c7ca1f7f60c12c68f6ffe1eb513791a1b8a908
```

查看集群状态:

```
[root@k8s-master01]# kubectl  get node
NAME           STATUS     ROLES                  AGE     VERSION
k8s-master01   NotReady   control-plane,master   8m53s   v1.20.0
k8s-master02   NotReady   control-plane,master   2m25s   v1.20.0
k8s-master03   NotReady   control-plane,master   31s     v1.20.0
k8s-node01     NotReady   <none>                 32s     v1.20.0
k8s-node02     NotReady   <none>                 88s     v1.20.0
```

# Calico安装

以下步骤只在master01执行

```
cd /root/k8s-ha-install && git checkout manual-installation-v1.20.x && cd calico/
```

修改calico-etcd.yaml的以下位置

```
sed -i 's#etcd_endpoints: "http://<ETCD_IP>:<ETCD_PORT>"#etcd_endpoints: "https://192.168.0.107:2379,https://192.168.0.108:2379,https://192.168.0.109:2379"#g' calico-etcd.yaml


ETCD_CA=`cat /etc/kubernetes/pki/etcd/ca.crt | base64 | tr -d '\n'`
ETCD_CERT=`cat /etc/kubernetes/pki/etcd/server.crt | base64 | tr -d '\n'`
ETCD_KEY=`cat /etc/kubernetes/pki/etcd/server.key | base64 | tr -d '\n'`
```

```
sed -i "s@# etcd-key: null@etcd-key: ${ETCD_KEY}@g; s@# etcd-cert: null@etcd-cert: ${ETCD_CERT}@g; s@# etcd-ca: null@etcd-ca: ${ETCD_CA}@g" calico-etcd.yaml


sed -i 's#etcd_ca: ""#etcd_ca: "/calico-secrets/etcd-ca"#g; s#etcd_cert: ""#etcd_cert: "/calico-secrets/etcd-cert"#g; s#etcd_key: "" #etcd_key: "/calico-secrets/etcd-key" #g' calico-etcd.yaml

POD_SUBNET=`cat /etc/kubernetes/manifests/kube-controller-manager.yaml | grep cluster-cidr= | awk -F= '{print $NF}'`

sed -i 's@# - name: CALICO_IPV4POOL_CIDR@- name: CALICO_IPV4POOL_CIDR@g; s@#   value: "192.168.0.0/16"@  value: '"${POD_SUBNET}"'@g' calico-etcd.yaml
```

创建calico

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

查看集群状态 (**需等待几分钟**)

```
[root@k8s-master01 dashboard]# kubectl  get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-5f6d4b864b-khq4h   1/1     Running   0          13m
calico-node-5tvxh                          1/1     Running   0          13m
calico-node-kffn7                          1/1     Running   0          13m
calico-node-lltfs                          1/1     Running   0          13m
calico-node-nhgn8                          1/1     Running   0          13m
coredns-54d67798b7-8w5hd                   1/1     Running   0          117m
coredns-54d67798b7-vb2ll                   1/1     Running   0          117m
etcd-k8s-master01                          1/1     Running   0          117m
etcd-k8s-master02                          1/1     Running   0          104m
kube-apiserver-k8s-master01                1/1     Running   0          117m
kube-apiserver-k8s-master02                1/1     Running   0          104m
kube-controller-manager-k8s-master01       1/1     Running   1          117m
kube-controller-manager-k8s-master02       1/1     Running   0          104m
kube-proxy-5bws8                           1/1     Running   0          117m
kube-proxy-pbqjc                           1/1     Running   0          104m
kube-proxy-tpwbt                           1/1     Running   0          86m
kube-proxy-vbpc5                           1/1     Running   0          86m
kube-scheduler-k8s-master01                1/1     Running   1          117m
kube-scheduler-k8s-master02                1/1     Running   0          104m
metrics-server-545b8b99c6-hkgnz            1/1     Running   0          2m38s
```

# Metrics Server部署

在新版的Kubernetes中系统资源的采集均使用Metrics-server，可以通过Metrics采集节点和Pod的内存、磁盘、CPU和网络的使用率。
将Master01节点的front-proxy-ca.crt复制到所有Node节点

```
scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node01:/etc/kubernetes/pki/front-proxy-ca.crt
scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node(其他节点自行拷贝):/etc/kubernetes/pki/front-proxy-ca.crt

# 其他节点自行拷贝
scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node01:/etc/kubernetes/pki/front-proxy-ca.crt
scp /etc/kubernetes/pki/front-proxy-ca.crt k8s-node02:/etc/kubernetes/pki/front-proxy-ca.crt
```

安装metrics server

```
cd /root/k8s-ha-install/metrics-server-0.4.x-kubeadm/

[root@k8s-master01 metrics-server-0.4.x-kubeadm]# kubectl  create -f comp.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

等待kube-system命令空间下的Pod全部启动后，查看状态

```
[root@k8s-master01 metrics-server-0.4.x-kubeadm]# kubectl  top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
k8s-master01   109m         2%     1296Mi          33%
k8s-master02   99m          2%     1124Mi          29%
k8s-master03   104m         2%     1082Mi          28%
k8s-node01     55m          1%     761Mi           19%
k8s-node02     53m          1%     663Mi           17%
```

# Dashboard部署

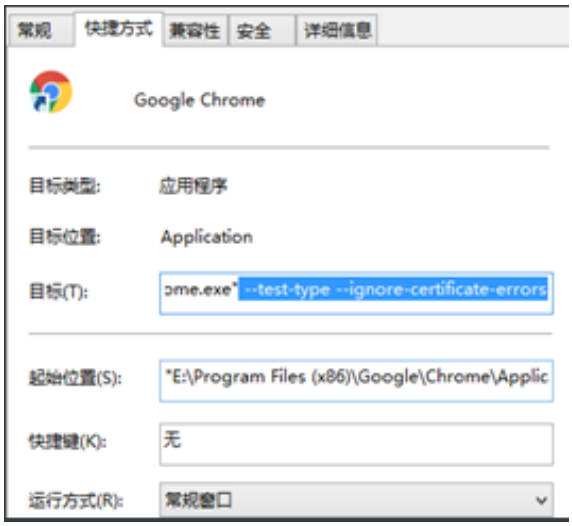
master01

```
cd /root/k8s-ha-install/dashboard/
```

```
[root@k8s-master01 dashboard]# kubectl  create -f .
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

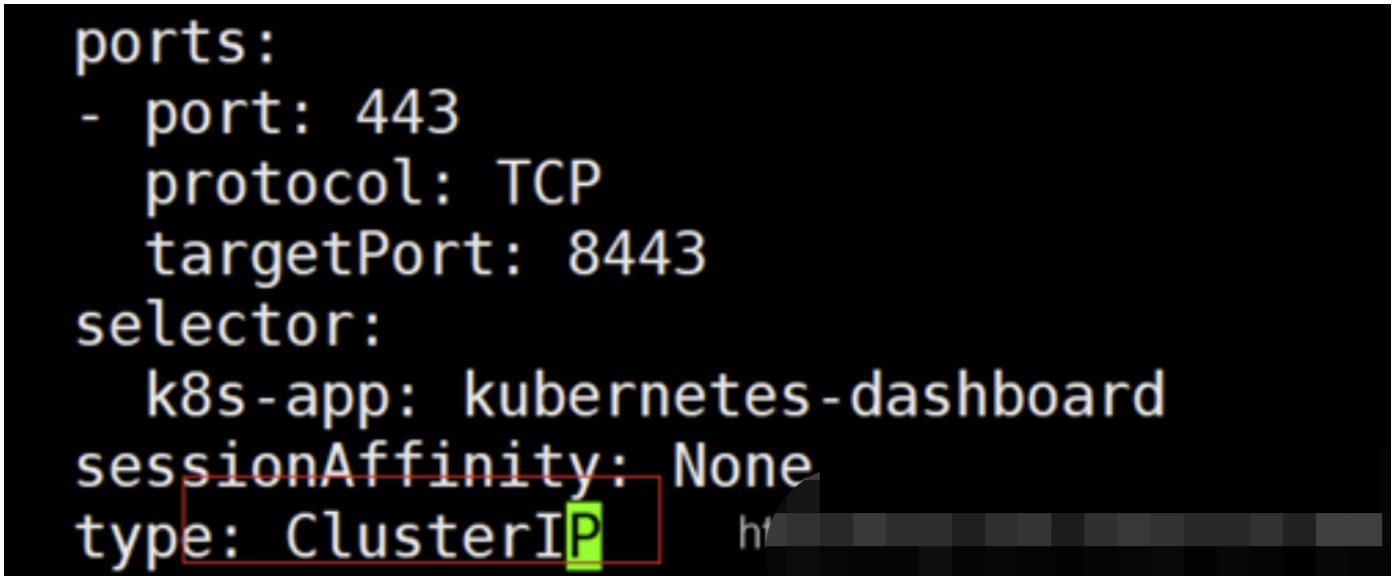
在谷歌浏览器（Chrome）启动文件中加入启动参数，用于解决无法访问Dashboard的问题，参考图：

```
--test-type --ignore-certificate-errors
```




更改dashboard的svc为NodePort：

```
kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard
```




将ClusterIP更改为NodePort（如果已经为NodePort忽略此步骤）：
查看端口号：

```
kubectl get svc kubernetes-dashboard -n kubernetes-dashboard
```

根据自己的实例端口号，通过任意安装了kube-proxy的宿主机或者VIP的IP+端口即可访问到dashboard：
访问Dashboard：https://192.168.0.236:18282（请更改18282为自己的端口），选择登录方式为令牌（即token方式）


查看token值：

```
[root@k8s-master01 1.1.1]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
Name:         admin-user-token-r4vcp
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: admin-user
              kubernetes.io/service-account.uid: 2112796c-1c9e-11e9-91ab-000c298bf023

Type:  kubernetes.io/service-account-token
```

```
Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZX
JuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc
2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXI0dmNwIiwia3ViZXJuZXRlcy5p
by9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLml
vL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiIyMTEyNzk2Yy0xYzllLTExZTktOTFhYi0wMD
BjMjk4YmYwMjMiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.
bWYmwgRb-90ydQmyjkbjJjFt8CdO8u6zxVZh-19rdlL_T-
n35nKyQIN7hCtNAt46u6gfJ5XXefC9HsGNBHtvo_Ve6oF7EXhU772aLAbXWkU1xOwQTQynixaypbRIas_kiO2MH
HxXfeeL_yYZRrgtatsDBxcBRg-
nUQv4TahzaGSyK42E_4YGpLa3X3Jc4t1z0SQXge7lrwlj8ysmqgO4ndlFjwPfvg0eoYqu9Qsc5Q7tazzFf9mVKM
mcS1ppPutdyqNYWL62P1prw_wclP0TezW1CsypjWSVT4AuJU8YmH8nTNR1EXn8mJURLSjINv6YbZpnhBIPgUGk1
JYVLcn47w
```

将token值输入到令牌后，单击登录即可访问Dashboard


```
# 查看所有容器状态
[root@k8s-master01 dashboard]# kubectl get po -A
NAMESPACE              NAME                                       READY   STATUS
RESTARTS   AGE
kube-system            calico-kube-controllers-5f6d4b864b-khq4h   1/1     Running   0
          16m
kube-system            calico-node-5tvxh                          1/1     Running   0
          16m
kube-system            calico-node-kffn7                          1/1     Running   0
          16m
kube-system            calico-node-lltfs                          1/1     Running   0
          16m
```

```
kube-system             calico-node-nhgn8                          1/1     Running   0
          16m
kube-system             coredns-54d67798b7-8w5hd                   1/1     Running   0
          121m
kube-system             coredns-54d67798b7-vb2ll                   1/1     Running   0
          121m
kube-system             etcd-k8s-master01                          1/1     Running   0
          121m
kube-system             etcd-k8s-master02                          1/1     Running   0
          107m
kube-system             kube-apiserver-k8s-master01                1/1     Running   0
          121m
kube-system             kube-apiserver-k8s-master02                1/1     Running   0
          107m
kube-system             kube-controller-manager-k8s-master01       1/1     Running   1
          121m
kube-system             kube-controller-manager-k8s-master02       1/1     Running   0
          107m
kube-system             kube-proxy-5bws8                           1/1     Running   0
          121m
kube-system             kube-proxy-pbqjc                           1/1     Running   0
          107m
kube-system             kube-proxy-tpwbt                           1/1     Running   0
          90m
kube-system             kube-proxy-vbpc5                           1/1     Running   0
          90m
kube-system             kube-scheduler-k8s-master01                1/1     Running   1
          121m
kube-system             kube-scheduler-k8s-master02                1/1     Running   0
          107m
kube-system             metrics-server-545b8b99c6-hkgnz            1/1     Running   0
          6m25s
kubernetes-dashboard    dashboard-metrics-scraper-7645f69d8c-hftvd 1/1     Running   0
          2m31s
kubernetes-dashboard    kubernetes-dashboard-78cb679857-zksm9      1/1     Running   0
          2m32s
```

更改dashboard的svc为NodePort

```
# 将type的ClusterIP更改为NodePort（如果已经为NodePort忽略此步骤）：
[root@k8s-master01 dashboard]# kubectl edit svc kubernetes-dashboard -n kubernetes-
dashboard
type: NodePort
```

# 一些必须的配置更改

将 Kube-proxy 改为ipvs模式，因为在初始化集群的时候注释了 ipvs配置，所以需要自行修改一下：

在master01节点执行

```
# 查看原来的mode模式
[root@k8s-master01 ~]# curl 127.0.0.1:10249/proxyMode
iptables

# 修改
[root@k8s-master01 ~]# kubectl edit cm kube-proxy -n kube-system
mode: "ipvs"  # 44行

# 更新Kube-Proxy的Pod
[root@k8s-master01 ~]# kubectl patch daemonset kube-proxy -p "{\"spec\":{\"template\":
{\"metadata\":{\"annotations\":{\"date\":\"`date +'%s'`\"}}}}}" -n kube-system
daemonset.apps/kube-proxy patched

# 再次查看mode模式
[root@k8s-master01 ~]# curl 127.0.0.1:10249/proxyMode
ipvs
```

# 注意事项

注意：kubeadm安装的集群，证书有效期默认是一年。master节点的kube-apiserver、kube-scheduler、kube-controller-manager、etcd都是以容器运行的。可以通过kubectl get po -n kube-system查看。

启动和二进制不同的是，

kubelet的配置文件在/etc/sysconfig/kubelet和/var/lib/kubelet/config.yaml

其他组件的配置文件在/etc/Kubernetes/manifests目录下，比如kube-apiserver.yaml，该yaml文件更改后，kubelet会自动刷新配置，也就是会重启pod。不能再次创建该文件

Kubeadm安装后，master节点默认不允许部署pod，可以通过以下方式打开：

```
# 查看Taints:
[root@k8s-master01 ~]# kubectl describe node -l node-role.kubernetes.io/master= | grep
Taints
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:       node-role.kubernetes.io/master:NoSchedule

# 删除Taint:

[root@k8s-master01 ~]# kubectl taint node -l node-role.kubernetes.io/master node-
role.kubernetes.io/master:NoSchedule-
node/k8s-master01 untainted
node/k8s-master02 untainted
```

```
node/k8s-master03 untainted

[root@k8s-master01 ~]# kubectl describe node -l node-role.kubernetes.io/master= | grep
Taints
Taints:             <none>
Taints:             <none>
Taints:             <none>
```

## 集群检测

```
kubectl get po --al-namespaces

# 监控数据
kubectl top po -n kube-system

# 网络
kubectl get svc
kubectl get svc -n kube-system

telnet 10.96.0.1 443

telnet 10.96.0.10 53

# 获取 k8s-master01的信息
kubectl get po --all-namespaces -owide
## 然后其他节点去访问
ping 172.168.32.130

# k8s-master03
kubectl get po --all-namespaces -owide
kubectl exec -it calico-node-qxs56 -n kube-system -- sh # 进入容器内访问
ping 172.168.32.130

# dashboard
kubectl get po -n kubernetes-dashboard
kubectl edit svc  kubernetes-dashboard -n !$
type: NodePort #修改为 NodePort
kubectl get po -n kubernetes-dashboard
https://192.168.0.107:32534 #浏览器访问
```

## dashboard的另一种选择

www.kuboard.cn

https://www.kuboard.cn/install/v3/install-in-k8s.html#%E5%AE%89%E8%A3%85%E6%AD%A5%E9%AA%A4