再也不踩坑的Kubernetes实战指南

请不要使用带中文的服务器和克隆的虚拟机

生产环境建议使用二进制安装方式

请将该文档复制一份，然后进行更改安装，并记录每一个步骤的返回信息，有问题可以直接发送部署文档进行问答，解决更加迅速

第1章

Kubernetes高可用安装

* 1. kubeadm高可用安装k8s集群1.18.x

### 1.1.1 基本环境配置

1、Kubectl debug 设置一个临时容器

2、Sidecar

3、Volume：更改目录权限，fsGroup

4、ConfigMap和Secret

K8S官网：<https://kubernetes.io/docs/setup/>

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

表1-1 高可用Kubernetes集群规划

|  |  |  |
| --- | --- | --- |
| 主机名 | IP地址 | 说明 |
| k8s-master01 ~ 03 | 192.168.0.106 ~ 20 | master节点 \* 3 |
| k8s-master-lb | 192.168.0.200 | keepalived虚拟IP |
| k8s-node01 ~ 02 | 192.168.0.108 ~ 22 | worker节点 \* 2 |

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！

所有节点配置hosts，修改/etc/hosts如下：

[root@k8s-master01 ~]# cat /etc/hosts

192.168.0.100 k8s-master01

192.168.0.106 k8s-master02

192.168.0.107 k8s-master03

192.168.0.200 k8s-master-lb

192.168.0.108 k8s-node01

192.168.0.109 k8s-node02

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 #CentOS8无需关闭

setenforce 0

[root@k8s-master01 ~]# cat !$

cat /etc/sysconfig/selinux

SELINUX=disabled

swapoff -a && sysctl -w vm.swappiness=0

[root@k8s-master01 ~]# vi /etc/fstab

[root@k8s-master01 ~]# cat /etc/fstab

#

# /etc/fstab

# Created by anaconda on Fri Nov 1 23:02:53 2019

#

# Accessible filesystems, by reference, are maintained under '/dev/disk/'.

# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.

#

# After editing this file, run 'systemctl daemon-reload' to update systemd

# units generated from this file.

#

/dev/mapper/cl-root / xfs defaults 0 0

UUID=6897cd7b-9b3a-42b0-a827-57991141b297 /boot ext4 defaults 1 2

#/dev/mapper/cl-swap swap swap defaults 0 0

安装ntpdate（CentOS 7 无需安装，自带ntpdate命令）

rpm -ivh http://mirrors.wlnmp.com/centos/wlnmp-release-centos.noarch.rpm

yum install wntp -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

# 加入到开机自动同步，/etc/rc.local

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，安装过程中生成配置文件和证书均在Master01上操作，集群管理也在Master01上操作，阿里云或者AWS上需要单独一台kubectl服务器。密钥配置如下：

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

在源码中的repo目录配置使用的是国内仓库源，将其复制到所有节点：

git clone <https://github.com/dotbalo/k8s-ha-install.git>

CentOS 7安装yum源如下：

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

CentOS 8 安装源如下：

curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-8.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

所有节点升级系统并重启，此处升级没有升级内核，下节会单独升级内核：

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

yum update -y --exclude=kernel\* && reboot #CentOS7需要升级，8不需要

### 1.1.2 内核配置

CentOS7 需要升级内核至4.18+

https://www.kernel.org/ 和 <https://elrepo.org/linux/kernel/el7/x86_64/>

CentOS 7 dnf可能无法安装内核

~~dnf --disablerepo=\\* --enablerepo=elrepo -y install kernel-ml kernel-ml-devel~~

~~grubby --default-kernel~~

使用如下方式安装最新版内核

rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org

rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm

查看最新版内核yum --disablerepo="\*" --enablerepo="elrepo-kernel" list available

[root@k8s-node01 ~]# yum --disablerepo="\*" --enablerepo="elrepo-kernel" list available

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

\* elrepo-kernel: mirrors.neusoft.edu.cn

elrepo-kernel | 2.9 kB 00:00:00

elrepo-kernel/primary\_db | 1.9 MB 00:00:00

Available Packages

elrepo-release.noarch 7.0-5.el7.elrepo elrepo-kernel

kernel-lt.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-devel.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-doc.noarch 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-headers.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-tools.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-tools-libs.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-lt-tools-libs-devel.x86\_64 4.4.229-1.el7.elrepo elrepo-kernel

kernel-ml.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-devel.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-doc.noarch 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-headers.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-tools.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-tools-libs.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

kernel-ml-tools-libs-devel.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

perf.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

python-perf.x86\_64 5.7.7-1.el7.elrepo elrepo-kernel

安装最新版：

yum --enablerepo=elrepo-kernel install kernel-ml kernel-ml-devel -y

安装完成后reboot

更改内核顺序：

grub2-set-default 0 && grub2-mkconfig -o /etc/grub2.cfg && grubby --args="user\_namespace.enable=1" --update-kernel="$(grubby --default-kernel)" && reboot

开机后查看内核

[appadmin@k8s-node01 ~]$ uname -a

Linux k8s-node01 5.7.7-1.el7.elrepo.x86\_64 #1 SMP Wed Jul 1 11:53:16 EDT 2020 x86\_64 x86\_64 x86\_64 GNU/Linux

CentOS 8按需升级:

可以采用dnf升级，也可使用上述同样步骤升级（使用上述步骤注意elrepo-release-8.1版本）

rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org

dnf install <https://www.elrepo.org/elrepo-release-8.1-1.el8.elrepo.noarch.rpm>

dnf --disablerepo=\\* --enablerepo=elrepo -y install kernel-ml kernel-ml-devel

grubby --default-kernel && reboot

本所有节点安装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已经改为nf\_conntrack，本例安装的内核为4.18，使用nf\_conntrack\_ipv4即可：

modprobe -- ip\_vs

modprobe -- ip\_vs\_rr

modprobe -- ip\_vs\_wrr

modprobe -- ip\_vs\_sh

modprobe -- nf\_conntrack\_ipv4

cat /etc/modules-load.d/ipvs.conf

ip\_vs

ip\_vs\_rr

ip\_vs\_wrr

ip\_vs\_sh

nf\_conntrack\_ipv4

ip\_tables

ip\_set

xt\_set

ipt\_set

ipt\_rpfilter

ipt\_REJECT

ipip

然后执行systemctl enable --now systemd-modules-load.service即可

检查是否加载：

[root@k8s-master01 ~]# lsmod | grep -e ip\_vs -e nf\_conntrack\_ipv4

nf\_conntrack\_ipv4 16384 23

nf\_defrag\_ipv4 16384 1 nf\_conntrack\_ipv4

nf\_conntrack 135168 10 xt\_conntrack,nf\_conntrack\_ipv6,nf\_conntrack\_ipv4,nf\_nat,nf\_nat\_ipv6,ipt\_MASQUERADE,nf\_nat\_ipv4,xt\_nat,nf\_conntrack\_netlink,ip\_vs

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

所有节点配置完内核后，重启服务器，保证重启后内核依旧加载

reboot

lsmod | grep --color=auto -e ip\_vs -e nf\_conntrack

### 1.1.3 基本组件安装

本节主要安装的是集群中用到的各种组件，比如Docker-ce、Kubernetes各组件等。

查看可用docker-ce版本：

yum list docker-ce.x86\_64 --showduplicates | sort -r

[root@k8s-master01 k8s-ha-install]# wget https://download.docker.com/linux/centos/7/x86\_64/edge/Packages/containerd.io-1.2.13-3.2.el7.x86\_64.rpm

[root@k8s-master01 k8s-ha-install]# yum install containerd.io-1.2.13-3.2.el7.x86\_64.rpm -y

安装指定版本的Docker：

yum -y install docker-ce-17.09.1.ce-1.el7.centos

安装最新版本的Docker

yum install docker-ce -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

cat > /etc/docker/daemon.json <<EOF

{

"exec-opts": ["native.cgroupdriver=systemd"]

}

EOF

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

所有节点安装指定版本k8s组件：

yum install -y kubeadm-1.12.3-0.x86\_64 kubectl-1.12.3-0.x86\_64 kubelet-1.12.3-0.x86\_64

所有节点设置开机自启动Docker：

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

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：

DOCKER\_CGROUPS=$(docker info | grep 'Cgroup' | cut -d' ' -f4)

cat >/etc/sysconfig/kubelet<<EOF

KUBELET\_EXTRA\_ARGS="--cgroup-driver=systemd --pod-infra-container-image=registry.cn-hangzhou.aliyuncs.com/google\_containers/pause-amd64:3.1"

EOF

设置Kubelet开机自启动：

systemctl daemon-reload

systemctl enable --now kubelet

### 1.1.4 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 192.168.0.100:6443 check

server k8s-master02 192.168.0.106:6443 check

server k8s-master03 192.168.0.107:6443 check

　所有Master节点配置KeepAlived，配置不一样，注意区分 [root@k8s-master01 pki]# vim /etc/keepalived/keepalived.conf ，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

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 10

weight -5

fall 3

rise 2

}

vrrp\_instance VI\_1 {

state MASTER

interface ens160

mcast\_src\_ip 192.168.0.100

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

192.168.0.200

}

# track\_script {

# chk\_apiserver

# }

}

Master02节点的配置：

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 10

weight -5

fall 3

rise 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens160

mcast\_src\_ip 192.168.0.106

virtual\_router\_id 51

priority 101

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

192.168.0.200

}

# track\_script {

# chk\_apiserver

# }

}

Master03节点的配置：

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 10

weight -5

fall 3

rise 2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens160

mcast\_src\_ip 192.168.0.107

virtual\_router\_id 51

priority 102

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

192.168.0.200

}

# track\_script {

# chk\_apiserver

# }

}

注意上述的健康检查是关闭的，集群建立完成后再开启：

# track\_script {

# chk\_apiserver

# }

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 5)

do

check\_code=$(pgrep kube-apiserver)

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

启动haproxy和keepalived

[root@k8s-master01 keepalived]# systemctl enable --now haproxy

[root@k8s-master01 keepalived]# systemctl enable --now keepalived

重要：如果安装了keepalived和haproxy，需要测试keepalived是否是正常的

telnet 192.168.0.200 16443

如果不通或者非常快速的自动中断，则认为VIP不可以，不可在继续往下执行，需要排查keepalived的问题，比如防火墙和selinux，haproxy和keepalived的状态，监听端口等

所有节点查看防火墙状态必须为disable和inactive：systemctl status firewalld

所有节点查看selinux状态，必须为disable：getenforce

master节点查看haproxy和keepalived状态：systemctl status keepalived haproxy

master节点查看监听端口：netstat -lntp

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/>

各Master节点的kubeadm-config.yaml配置文件如下：

Master01：

daocloud.io/daocloud

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: 192.168.0.100

bindPort: 6443

nodeRegistration:

criSocket: /var/run/dockershim.sock

name: k8s-master01

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

---

apiServer:

certSANs:

- 192.168.0.200

timeoutForControlPlane: 4m0s

apiVersion: kubeadm.k8s.io/v1beta2

certificatesDir: /etc/kubernetes/pki

clusterName: kubernetes

controlPlaneEndpoint: 192.168.0.200:16443

controllerManager: {}

dns:

type: CoreDNS

etcd:

local:

dataDir: /var/lib/etcd

imageRepository: registry.cn-hangzhou.aliyuncs.com/google\_containers

kind: ClusterConfiguration

kubernetesVersion: v1.18.5

networking:

dnsDomain: cluster.local

podSubnet: 172.168.0.0/16

serviceSubnet: 10.96.0.0/12

scheduler: {}

更新kubeadm文件

kubeadm config migrate --old-config kubeadm-config.yaml --new-config new.yaml

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/kubeadm-config.yaml

所有节点设置开机自启动kubelet

systemctl enable --now kubelet

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/kubeadm-config.yaml --upload-certs

不用配置文件初始化：kubeadm init --control-plane-endpoint "LOAD\_BALANCER\_DNS:LOAD\_BALANCER\_PORT" --upload-certs

如果初始化失败，重置后再次初始化，命令如下：

kubeadm reset

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

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 192.168.0.200:16443 --token 5joxsb.zo1vh747wljgzrlt \

--discovery-token-ca-cert-hash sha256:86ee9b6a65c6d8641507e9e56e66dad47cfa15b41b52a11e175c5f9588a485b8 \

--control-plane --certificate-key bc4726d06255be0cd54592e29068e32c5a49eb8fd30a691342412cf79b3d47c7

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 192.168.0.200:16443 --token 5joxsb.zo1vh747wljgzrlt \

--discovery-token-ca-cert-hash sha256:86ee9b6a65c6d8641507e9e56e66dad47cfa15b41b52a11e175c5f9588a485b8

所有Master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 master 14m v1.12.3

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 192.168.0.100 k8s-master01

kube-apiserver-k8s-master01 1/1 Running 0 13m 192.168.0.100 k8s-master01

kube-controller-manager-k8s-master01 1/1 Running 0 13m 192.168.0.100 k8s-master01

kube-proxy-8d4qc 1/1 Running 0 14m 192.168.0.100 k8s-master01

kube-scheduler-k8s-master01 1/1 Running 0 13m 192.168.0.100 k8s-master01

### 1.1.5 Calico组件的安装

注意：如果国内用户下载Calico较慢，所有节点可以配置加速器(如果该文件有其他配置，别忘了加上去)

vim /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"registry-mirrors": [

"https://registry.docker-cn.com",

"http://hub-mirror.c.163.com",

"https://docker.mirrors.ustc.edu.cn"

]

}

systemctl daemon-reload

systemctl restart docker

Calico：<https://www.projectcalico.org/>

<https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises>

curl https://docs.projectcalico.org/manifests/calico.yaml -O

- name: CALICO\_IPV4POOL\_CIDR

value: "172.168.0.0/16"

kubectl apply -f calico.yaml

### 1.1.6 高可用Master

Token过期后生成新的token：

kubeadm token create --print-join-command

Master需要生成--certificate-key

kubeadm init phase upload-certs --upload-certs

初始化其他master加入集群

kubeadm join 192.168.0.200:16443 --token 9zp1xe.h5kpi1b9kd5blk76 --discovery-token-ca-cert-hash sha256:6ba6e5205ac27e39e03d3b89a639ef70f6503fb877b1cf8a332b399549471740 \

--control-plane --certificate-key 309f945f612dd7f0d830b11868edd5135e6cf358ed503107eb645dc8d7c84405

### 1.1.7 Node节点的配置

Node节点上主要部署公司的一些业务应用，生产环境中不建议Master节点部署系统组件之外的其他Pod，测试环境可以允许Master节点部署Pod以节省系统资源。

kubeadm join 192.168.0.200:16443 --token 9zp1xe.h5kpi1b9kd5blk76 --discovery-token-ca-cert-hash sha256:6ba6e5205ac27e39e03d3b89a639ef70f6503fb877b1cf8a332b399549471740

### 1.1.8 Metrics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。Heapster

更改metrics的部署文件证书，将metrics-server-3.6.1/metrics-server-deployment.yaml的front-proxy-ca.pem改为front-proxy-ca.crt



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

安装metrics server

kubectl apply -f metrics-server-3.6.1/

[root@k8s-master01 k8s-ha-install]# cd metrics-server-3.6.1/

[root@k8s-master01 metrics-server-3.6.1]#

[root@k8s-master01 metrics-server-3.6.1]# ls

aggregated-metrics-reader.yaml auth-delegator.yaml auth-reader.yaml metrics-apiservice.yaml metrics-server-deployment.yaml metrics-server-service.yaml resource-reader.yaml

[root@k8s-master01 metrics-server-3.6.1]# kubectl apply -f .

clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created

clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created

rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created

apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created

serviceaccount/metrics-server created

deployment.apps/metrics-server created

service/metrics-server created

clusterrole.rbac.authorization.k8s.io/system:metrics-server created

clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created

### 1.1.9 Dashboard部署

官方GitHub：<https://github.com/kubernetes/dashboard>

Dashboard用于展示集群中的各类资源，同时也可以通过Dashboard实时查看Pod的日志和在容器中执行一些命令等。

可以在官方dashboard查看到最新版dashboard



kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml>

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1-1：

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



图1-1 谷歌浏览器 Chrome的配置

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：



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://192.168.0.200:18282（请更改18282为自己的端口）>，选择登录方式为令牌（即token方式），参考图1-2



图1-2 Dashboard登录方式

创建管理员用户vim admin.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: admin-user

namespace: kube-system

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

annotations:

rbac.authorization.kubernetes.io/autoupdate: "true"

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: admin-user

namespace: kube-system

kubectl create -f admin.yaml -n kube-system

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.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi11c2VyLXRva2VuLXI0dmNwIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiIyMTEyNzk2Yy0xYzllLTExZTktOTFhYi0wMDBjMjk4YmYwMjMiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlciJ9.bWYmwgRb-90ydQmyjkbjJjFt8CdO8u6zxVZh-19rdlL\_T-n35nKyQIN7hCtNAt46u6gfJ5XXefC9HsGNBHtvo\_Ve6oF7EXhU772aLAbXWkU1xOwQTQynixaypbRIas\_kiO2MHHxXfeeL\_yYZRrgtatsDBxcBRg-nUQv4TahzaGSyK42E\_4YGpLa3X3Jc4t1z0SQXge7lrwlj8ysmqgO4ndlFjwPfvg0eoYqu9Qsc5Q7tazzFf9mVKMmcS1ppPutdyqNYWL62P1prw\_wclP0TezW1CsypjWSVT4AuJU8YmH8nTNR1EXn8mJURLSjINv6YbZpnhBIPgUGk1JYVLcn47w**

将token值输入到令牌后，单击登录即可访问Dashboard，参考图1-3：



图1-3 Dashboard页面

1.1.10 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

kubectl edit cm kube-proxy -n kube-system

mode: “ipvs”

更新Kube-Proxy的Pod：

kubectl patch daemonset kube-proxy -p "{\"spec\":{\"template\":{\"metadata\":{\"annotations\":{\"date\":\"`date +'%s'`\"}}}}}" -n kube-system

验证Kube-Proxy模式

[root@k8s-master01 1.1.1]# curl 127.0.0.1:10249/proxyMode

ipvs

在所有的master节点

开启keepalived的健康检查：

[root@k8s-master01 etc]# vim /etc/keepalived

[root@k8s-master01 ~]# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_script chk\_apiserver {

script "/etc/keepalived/check\_apiserver.sh"

interval 10

weight -5

fall 3

rise 2

}

vrrp\_instance VI\_1 {

state MASTER

interface ens160

mcast\_src\_ip 192.168.0.100

virtual\_router\_id 51

priority 100

advert\_int 2

authentication {

auth\_type PASS

auth\_pass K8SHA\_KA\_AUTH

}

virtual\_ipaddress {

192.168.0.200

}

# 将此块注释打开

track\_script {

chk\_apiserver

}

}

重启keepalived

systemctl daemon-reload

systemctl restart keepalived

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

其他组件的配置文件在/etc/kubernetes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