可参考我的微信公众号文章：

https://mp.weixin.qq.com/s?\_\_biz=MzU0NjEwMTg4Mg==&mid=2247484264&idx=1&sn=4d74de0cb117c07ba7c73ea21c42bf94&chksm=fb638c7dcc14056b9f0261e05e4d36a057c4511fb5806222f71b7965c3573fed86672b2c93b7&token=1373889250&lang=zh\_CN#rd

技术交流

微信：

luckylucky421302

加微信备注k8s即可进群交流学习



**前言**

这篇文章会带领大家去安装k8s1.18的高可用集群，如果你是初学小白，只要跟着做，也能保证100%完成安装，下面开始我们的安装之旅吧，内容较多，都是干货

**心灵鸡汤**

如果你觉得累，请看看下面这段话：我们想要去的地方，永远没有捷径，只有脚踏实地，一步一个脚印的才能走向诗和远方！

**资料下载**

**1.下文需要的yaml文件所在的github地址如下：**

https://github.com/luckylucky421/kubernetes1.17.3/tree/master

大家可以把我的github仓库fork到你们自己的仓库里，这样就可以永久保存了，下面提供的yaml访问地址如果不能访问，那么就把这个github上的内容clone和下载到自己电脑。

**下面实验用到yaml文件大家需要从上面的github上clone和下载到本地，然后把yaml文件传到k8s集群的master节点，如果直接复制粘贴格式可能会有问题**

**2.下文里提到的初始化k8s集群需要的镜像获取方式：镜像在百度网盘，链接如下：**

链接：<https://pan.baidu.com/s/1k1heJy8lLnDk2JEFyRyJdA>

提取码：udkj

**正文**

**一、准备实验环境**

1.准备四台centos7虚拟机，用来安装k8s集群，下面是四台虚拟机的配置情况

**master1（192.168.0.6）配置：**

操作系统：centos7.6以及更高版本都可以配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**master2（192.168.0.16）配置：**

操作系统：centos7.6以及更高版本都可以配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**master3（192.168.0.26）配置：**

操作系统：centos7.6以及更高版本都可以配置：4核cpu，6G内存，两块60G硬盘

网络：桥接网络

**node1（192.168.0.56）配置：**

操作系统：centos7.6以及更高版本都可以配置：4核cpu，4G内存，两块60G硬盘

网络：桥接网络

**二、初始化实验环境**

**1.配置静态ip**

把虚拟机或者物理机配置成静态ip地址，这样机器重新启动后ip地址也不会发生改变。

**1.1 在master1节点配置网络**

**注：**/etc/sysconfig/network-scripts/ifcfg-ens33文件里的配置说明：

NAME=ens33 #网卡名字，跟DEVICE名字保持一致即可

DEVICE=ens33 #网卡设备名，大家ip addr可看到自己的这个网卡设备名，每个人的机器可能这个名字不一样，需要写自己的

BOOTPROTO=static #static表示静态ip地址

ONBOOT=yes #开机自启动网络，必须是yes

修改/etc/sysconfig/network-scripts/ifcfg-ens33文件，变成如下：

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

IPADDR=192.168.0.6

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=192.168.0.1

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens33

DEVICE=ens33

ONBOOT=yes

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

service network restart

**1.2 在master2节点配置网络**

修改/etc/sysconfig/network-scripts/ifcfg-ens33文件，变成如下：

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

IPADDR=192.168.0.16

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=192.168.0.1

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens33

DEVICE=ens33

ONBOOT=yes

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

service network restart

**1.3 在master3节点配置网络**

修改/etc/sysconfig/network-scripts/ifcfg-ens33文件，变成如下：

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

IPADDR=192.168.0.26

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=192.168.0.1

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens33

DEVICE=ens33

ONBOOT=yes

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

service network restart

**注：**ifcfg-ens33文件配置解释：

IPADDR=192.168.0.6

#ip地址，需要跟自己电脑所在网段一致

NETMASK=255.255.255.0

#子网掩码，需要跟自己电脑所在网段一致

GATEWAY=192.168.0.1

#网关，在自己电脑打开cmd，输入ipconfig /all可看到

DNS1=192.168.0.1

#DNS，在自己电脑打开cmd，输入ipconfig /all可看到

**1.4 在node1节点配置网络**

修改/etc/sysconfig/network-scripts/ifcfg-ens33文件，变成如下：

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

IPADDR=192.168.0.56

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

DNS1=192.168.0.1

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

IPV6\_ADDR\_GEN\_MODE=stable-privacy

NAME=ens33

DEVICE=ens33

ONBOOT=yes

修改配置文件之后需要重启网络服务才能使配置生效，重启网络服务命令如下：

service network restart

**2.修改yum源，各个节点操作**

（1）备份原来的yum源

mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup

（2）下载阿里的yum源

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

（3）生成新的yum缓存

yum makecache fast

（4）配置安装k8s需要的yum源

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=0

EOF

（5）清理yum缓存

yum clean all

（6）生成新的yum缓存

yum makecache fast

（7）更新yum源

yum -y update

 （8）安装软件包

yum -y install yum-utils device-mapper-persistent-data lvm2

 （9）添加新的软件源

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

yum clean all

yum makecache fast

**3.安装基础软件包，各个节点操作**

yum -y install wget net-tools nfs-utils lrzsz gcc gcc-c++ make cmake libxml2-devel openssl-devel curl curl-devel unzip sudo ntp libaio-devel wget vim ncurses-devel autoconf automake zlib-devel python-devel epel-release openssh-server socat ipvsadm conntrack ntpdate

**4.关闭firewalld防火墙，各个节点操作，centos7系统默认使用的是firewalld防火墙，停止firewalld防火墙，并禁用这个服务**

systemctl stop firewalld && systemctl disable firewalld

**5.安装iptables，各个节点操作，如果你用firewalld不是很习惯，可以安装iptables，这个步骤可以不做，根据大家实际需求**

5.1 安装iptables

yum install iptables-services -y

5.2 禁用iptables

service iptables stop && systemctl disable iptables

**6.时间同步，各个节点操作**

6.1 时间同步

ntpdate cn.pool.ntp.org

6.2 编辑计划任务，每小时做一次同步

1）crontab -e

\* \*/1 \* \* \* /usr/sbin/ntpdate cn.pool.ntp.org

2）重启crond服务：

service crond restart

**7. 关闭selinux，各个节点操作**

关闭selinux，设置永久关闭，这样重启机器selinux也处于关闭状态

修改/etc/sysconfig/selinux和/etc/selinux/config文件，把

SELINUX=enforcing变成SELINUX=disabled，也可用下面方式修改：

sed -i  's/SELINUX=enforcing/SELINUX=disabled/'  /etc/sysconfig/selinux

sed -i  's/SELINUX=enforcing/SELINUX=disabled/g'  /etc/selinux/config

上面文件修改之后，需要重启虚拟机，可以强制重启：

reboot -f

**8.关闭交换分区，各个节点操作**

swapoff -a

#永久禁用，打开/etc/fstab注释掉swap那一行。

sed -i 's/.\*swap.\*/#&/' /etc/fstab

**9.修改内核参数，各个节点操作**

cat <<EOF > /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sysctl --system

**10.修改主机名**

在192.168.0.6上：

hostnamectl set-hostname master1

在192.168.0.16上：

hostnamectl set-hostname master2

在192.168.0.26上：

hostnamectl set-hostname master3

在192.168.0.56上：

hostnamectl set-hostname node1

**11.配置hosts文件，各个节点操作**

在/etc/hosts文件增加如下几行：

192.168.0.6 master1

192.168.0.16 master2

192.168.0.26 master3

192.168.0.56 node1

**12.配置master1到node无密码登陆，配置master1到master2、master3无密码登陆**

在master1上操作

ssh-keygen -t rsa

#一直回车就可以

ssh-copy-id -i .ssh/id\_rsa.pub root@master2

#上面需要输入yes之后，输入密码，输入master2物理机密码即可

ssh-copy-id -i .ssh/id\_rsa.pubroot@master3

#上面需要输入yes之后，输入密码，输入master3物理机密码即可

ssh-copy-id -i .ssh/id\_rsa.pubroot@node1

#上面需要输入yes之后，输入密码，输入node1物理机密码即可

**三、安装kubernetes1.18.2高可用集群**

**1.安装docker19.03，各个节点操作**

**1.1 查看支持的docker版本**

yum list docker-ce --showduplicates |sort -r

**1.2 安装19.03.7版本**

yum install -y docker-ce-19.03.7-3.el7

systemctl enable docker && systemctl start docker

#查看docker状态，如果状态是active（running），说明docker是正常运行状态

systemctl status docker

**1.3 修改docker配置文件**

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

{

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

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2",

"storage-opts": [

"overlay2.override\_kernel\_check=true"

]

}

EOF

**1.4 重启docker使配置生效**

systemctl daemon-reload && systemctl restart docker

**1.5 设置网桥包经IPTables，core文件生成路径，配置永久生效**

echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables

echo 1 >/proc/sys/net/bridge/bridge-nf-call-ip6tables

echo """

vm.swappiness = 0

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip\_forward = 1

net.bridge.bridge-nf-call-ip6tables = 1

""" > /etc/sysctl.conf

sysctl -p

**1.6 开启ipvs，不开启ipvs将会使用iptables，但是效率低，所以官网推荐需要开通ipvs内核**

cat > /etc/sysconfig/modules/ipvs.modules <<EOF

#!/bin/bash

ipvs\_modules="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 nf\_conntrack"

for kernel\_module in \${ipvs\_modules}; do

/sbin/modinfo -F filename \${kernel\_module} > /dev/null 2>&1

if [ $? -eq 0 ]; then

/sbin/modprobe \${kernel\_module}

fi

done

EOF

chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep ip\_vs

**2.安装kubernetes1.18.2**

**2.1在master1、master2、master3和node1上安装kubeadm和kubelet**

yum install kubeadm-1.18.2 kubelet-1.18.2 -y

systemctl enable kubelet

**2.2上传镜像到master1、master2、master3和node1节点之后，按如下方法通过docker load -i手动解压镜像，镜像在百度网盘，文章最上面附有镜像所在的百度网盘地址，我是从官方下载的镜像，大家可以放心使用。**

docker load -i 1-18-kube-apiserver.tar.gz

docker load -i 1-18-kube-scheduler.tar.gz

docker load -i 1-18-kube-controller-manager.tar.gz

docker load -i 1-18-pause.tar.gz

docker load -i 1-18-cordns.tar.gz

docker load -i 1-18-etcd.tar.gz

docker load -i 1-18-kube-proxy.tar.gz

**说明：**

pause版本是3.2，用到的镜像是k8s.gcr.io/pause:3.2

etcd版本是3.4.3，用到的镜像是k8s.gcr.io/etcd:3.4.3-0

cordns版本是1.6.7，用到的镜像是k8s.gcr.io/coredns:1.6.7

apiserver、scheduler、controller-manager、kube-proxy版本是1.18.2，用到的镜像分别是

k8s.gcr.io/kube-apiserver:v1.18.2

k8s.gcr.io/kube-controller-manager:v1.18.2

k8s.gcr.io/kube-scheduler:v1.18.2

k8s.gcr.io/kube-proxy:v1.18.2

为什么手动解压镜像？

1）因为很多同学的公司是内网环境，或者访问不了dockerhub镜像仓库，所以需要我们把镜像上传到各个机器手动解压，很多同学会问，如果机器很多，怎么办，难道还要把镜像拷贝到很多机器，这岂不是很费时间，的确，如果机器很多，我们只需要把这些镜像传到我们的内部私有镜像仓库即可，这样我们在kubeadm初始化kubernetes时可以通过"--image-repository=私有镜像仓库地址"的方式进行镜像拉取，这样不需要手动传到镜像到每个机器，后面会介绍；

2）镜像存到百度网盘可以永久使用，防止官方不在维护，我们无从下载镜像，所以有私有仓库的同学可以把这些镜像传到自己私有镜像仓库。

**2.3 部署keepalive+lvs实现master节点高可用-对apiserver做高可用**

（1）部署keepalived+lvs，在各master节点操作

yum install -y socat keepalived ipvsadm conntrack

（2）修改master1的keepalived.conf文件，按如下修改

修改/etc/keepalived/keepalived.conf

master1节点修改之后的keepalived.conf如下所示:

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state BACKUP

nopreempt

interface ens33

virtual\_router\_id 80

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass just0kk

}

virtual\_ipaddress {

192.168.0.199

}

}

virtual\_server 192.168.0.199 6443 {

delay\_loop 6

lb\_algo loadbalance

lb\_kind DR

net\_mask 255.255.255.0

persistence\_timeout 0

protocol TCP

real\_server 192.168.0.6 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.16 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.26 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

（3）修改master2的keepalived.conf文件，按如下修改

修改/etc/keepalived/keepalived.conf

master2节点修改之后的keepalived.conf如下所示:

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state BACKUP

nopreempt

interface ens33

virtual\_router\_id 80

priority 50

advert\_int 1

authentication {

auth\_type PASS

auth\_pass just0kk

}

virtual\_ipaddress {

192.168.0.199

}

}

virtual\_server 192.168.0.199 6443 {

delay\_loop 6

lb\_algo loadbalance

lb\_kind DR net\_mask 255.255.255.0

persistence\_timeout 0

protocol TCP

real\_server 192.168.0.6 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.16 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.26 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

（4）修改master3的keepalived.conf文件，按如下修改

修改/etc/keepalived/keepalived.conf

master3节点修改之后的keepalived.conf如下所示:

global\_defs {

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state BACKUP

nopreempt

interface ens33

virtual\_router\_id 80

priority 30

advert\_int 1

authentication {

auth\_type PASS

auth\_pass just0kk

}

virtual\_ipaddress {

192.168.0.199

}

}

virtual\_server 192.168.0.199 6443 {

delay\_loop 6

lb\_algo loadbalance

lb\_kind DR

net\_mask 255.255.255.0

persistence\_timeout 0

protocol TCP

real\_server 192.168.0.6 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.16 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

real\_server 192.168.0.26 6443 {

weight 1

SSL\_GET {

url {

path /healthz

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

**重要知识点，必看，否则生产会遇到巨大的坑**

keepalive需要配置BACKUP，而且是非抢占模式nopreempt，假设master1宕机，启动之后vip不会自动漂移到master1，这样可以保证k8s集群始终处于正常状态，因为假设master1启动，apiserver等组件不会立刻运行，如果vip漂移到master1，那么整个集群就会挂掉，这就是为什么我们需要配置成非抢占模式了

启动顺序master1->master2->master3，在master1、master2、master3依次执行如下命令

systemctl enable keepalived && systemctl start keepalived && systemctl status keepalived

keepalived启动成功之后，在master1上通过ip addr可以看到vip已经绑定到ens33这个网卡上了

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:9d:7b:09 brd ff:ff:ff:ff:ff:ff

inet 192.168.0.6/24 brd 192.168.0.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.0.199/32 scope global ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::e2f9:94cd:c994:34d9/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default

link/ether 02:42:61:b0:6f:ca brd ff:ff:ff:ff:ff:ff

inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0

valid\_lft forever preferred\_lft forever

**2.4 在master1节点初始化k8s集群，在master1上操作如下**

**如果按照我在2.2节手动上传镜像到各个节点，通过docker load -i方式解压镜像，那么用下面的yaml文件初始化，大家都统一按照这种方法上传镜像到各个机器，手动解压，这样后面实验才会正常进行。**

kubeadm-config.yaml内容如下：

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.18.2

controlPlaneEndpoint: 192.168.0.199:6443

apiServer:

certSANs:

- 192.168.0.6

- 192.168.0.16

- 192.168.0.26

- 192.168.0.56

- 192.168.0.199

networking:

podSubnet: 10.244.0.0/16

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

初始化k8s集群

kubeadm init --config kubeadm-config.yaml

注：如果没有按照2.2节的方法上传镜像到各个节点，那么用下面的kubeadm-config.yaml文件，多了imageRepository: registry.aliyuncs.com/google\_containers参数，表示走的是阿里云镜像，我们可以直接访问，这个方法更简单，但是在这里了解即可，先不使用这种方法，使用的话在后面手动加节点到k8s集群会有问题。

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.18.2

controlPlaneEndpoint: 192.168.0.199:6443

imageRepository: registry.aliyuncs.com/google\_containers

apiServer:

certSANs:

- 192.168.0.6

- 192.168.0.16

- 192.168.0.26

- 192.168.0.56

- 192.168.0.199

networking:

podSubnet: 10.244.0.0/16

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

**kubeadm init --config kubeadm-config.yaml初始化命令执行成功之后显示如下内容，说明初始化成功了**

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 control-plane nodes by copying certificate authorities  
and service account keys on each node and then running the following as root:  
  
  kubeadm join 192.168.0.199:6443 --token 7dwluq.x6nypje7h55rnrhl \  
    --discovery-token-ca-cert-hash sha256:fa75619ab0bb6273126350a9dbda9aa6c89828c2c4650299fe1647ab510a7e6c \  
    --control-plane  
  
Then you can join any number of worker nodes by running the following on each as root:  
  
kubeadm join 192.168.0.199:6443 --token 7dwluq.x6nypje7h55rnrhl \  
    --discovery-token-ca-cert-hash sha256:fa75619ab0bb6273126350a9dbda9aa6c89828c2c4650299fe1647ab510a7e6c

**注：**kubeadm join ... 这条命令需要记住，我们把k8s的master2、master3，node1节点加入到集群需要在这些节点节点输入这条命令，每次执行这个结果都是不一样的，大家记住自己执行的结果，在下面会用到

**2.5 在master1节点执行如下，这样才能有权限操作k8s资源**

mkdir -p $HOME/.kube

sudo cp -i  /etc/kubernetes/admin.conf  $HOME/.kube/config

sudo chown $(id -u):$(id -g)  $HOME/.kube/config

在master1节点执行

kubectl get nodes

显示如下，master1节点是NotReady

NAME STATUS ROLES AGE VERSION master1   NotReady   master   8m11s   v1.18.2

kubectl get pods -n kube-system

显示如下，可看到cordns也是处于pending状态

coredns-7ff77c879f-j48h6 0/1 Pending 0 3m16scoredns-7ff77c879f-lrb77 0/1 Pending 0 3m16s

上面可以看到STATUS状态是NotReady，cordns是pending，是因为没有安装网络插件，需要安装calico或者flannel，接下来我们安装calico，在master1节点安装calico网络插件：

安装calico需要的镜像是quay.io/calico/cni:v3.5.3和quay.io/calico/node:v3.5.3，镜像在文章开头处的百度网盘地址

手动上传上面两个镜像的压缩包到各个节点，通过docker load -i解压

docker load -i   cni.tar.gz  
docker load -i   calico-node.tar.gz

在master1节点执行如下：

kubectl apply -f calico.yaml

calico.yaml文件内容在如下提供的地址，打开下面链接可复制内容：

<https://raw.githubusercontent.com/luckylucky421/kubernetes1.17.3/master/calico.yaml>

如果打不开上面的链接,可以访问下面的github地址，把下面的目录clone和下载下来，解压之后，在把文件传到master1节点即可

https://github.com/luckylucky421/kubernetes1.17.3/tree/master

在master1节点执行

kubectl get nodes

显示如下，看到STATUS是Ready

NAME STATUS ROLES AGE VERSION

master1 Ready master 98m v1.18.2

kubectl get pods -n kube-system

看到cordns也是running状态，说明master1节点的calico安装完成

NAME READY STATUS RESTARTS AGE

calico-node-6rvqm 1/1 Running 0 17m

coredns-7ff77c879f-j48h6 1/1 Running 0 97m

coredns-7ff77c879f-lrb77 1/1 Running 0 97m

etcd-master1 1/1 Running 0 97m

kube-apiserver-master1 1/1 Running 0 97m

kube-controller-manager-master1 1/1 Running 0 97m

kube-proxy-njft6 1/1 Running 0 97m

kube-scheduler-master1 1/1 Running 0 97m

**2.6 把master1节点的证书拷贝到master2和master3上**

**（1）在master2和master3上创建证书存放目录**

cd /root && mkdir -p /etc/kubernetes/pki/etcd &&mkdir -p ~/.kube/

**（2）在master1节点把证书拷贝到master2和master3上，在master1上操作如下，下面的scp命令大家最好一行一行复制，这样不会出错：**

scp /etc/kubernetes/pki/ca.crt master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/ca.key master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/sa.key master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/sa.pub master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/front-proxy-ca.crt master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/front-proxy-ca.key master2:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/etcd/ca.crt master2:/etc/kubernetes/pki/etcd/

scp /etc/kubernetes/pki/etcd/ca.key master2:/etc/kubernetes/pki/etcd/

scp /etc/kubernetes/pki/ca.crt master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/ca.key master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/sa.key master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/sa.pub master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/front-proxy-ca.crt master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/front-proxy-ca.key master3:/etc/kubernetes/pki/

scp /etc/kubernetes/pki/etcd/ca.crt master3:/etc/kubernetes/pki/etcd/

scp /etc/kubernetes/pki/etcd/ca.key master3:/etc/kubernetes/pki/etcd/

证书拷贝之后在master2和master3上执行如下命令，大家复制自己的，这样就可以把master2和master3加入到集群

kubeadm join 192.168.0.199:6443 --token 7dwluq.x6nypje7h55rnrhl \  
    --discovery-token-ca-cert-hash sha256:fa75619ab0bb6273126350a9dbda9aa6c89828c2c4650299fe1647ab510a7e6c   --control-plane

--control-plane：这个参数表示加入到k8s集群的是master节点

在master2和master3上操作：

    mkdir -p $HOME/.kube

    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

    sudo chown $(id -u):$(id -g)$HOME/.kube/config

kubectl get nodes

显示如下：

NAME STATUS ROLES AGE VERSION

master1 Ready master 39m v1.18.2

master2 Ready master 5m9s v1.18.2

master3 Ready master 2m33s v1.18.2

**2.7 把node1节点加入到k8s集群，在node1节点操作**

kubeadm join 192.168.0.199:6443 --token 7dwluq.x6nypje7h55rnrhl \  
    --discovery-token-ca-cert-hash sha256:fa75619ab0bb6273126350a9dbda9aa6c89828c2c4650299fe1647ab510a7e6c

**注：**上面的这个加入到k8s节点的一串命令kubeadm join就是在2.4初始化的时候生成的

**2.8 在master1节点查看集群节点状态**

kubectl get nodes

显示如下：

NAME STATUS ROLES AGE VERSION

master1 Ready master 3m36s v1.18.2

master2 Ready master 3m36s v1.18.2

master3 Ready master 3m36s v1.18.2

node1 Ready <none> 3m36s v1.18.2

说明node1节点也加入到k8s集群了，通过以上就完成了k8s多master高可用集群的搭建

**2.9 安装traefik**

把traefik镜像上传到各个节点，按照如下方法通过docker load -i解压，镜像地址在文章开头处的百度网盘里，可自行下载

docker load -i  traefik\_1\_7\_9.tar.gz

traefik用到的镜像是k8s.gcr.io/traefik:1.7.9

1）生成traefik证书，在master1上操作

mkdir  ~/ikube/tls/ -p

echo """

[req]

distinguished\_name = req\_distinguished\_name

prompt = yes

[ req\_distinguished\_name ]

countryName                     = Country Name (2 letter code)

countryName\_value               = CN

stateOrProvinceName             = State or Province Name (full name)

stateOrProvinceName\_value       = Beijing

localityName                    = Locality Name (eg, city)

localityName\_value              = Haidian

organizationName                = Organization Name (eg, company)

organizationName\_value          = Channelsoft

organizationalUnitName          = Organizational Unit Name (eg, section)

organizationalUnitName\_value    = R & D Department

commonName                      = Common Name (eg, your name or your server\'s hostname)

commonName\_value                = \*.multi.io

emailAddress                    = Email Address

emailAddress\_value              = lentil1016@gmail.com

""" > ~/ikube/tls/openssl.cnf

openssl req -newkey rsa:4096 -nodes -config ~/ikube/tls/openssl.cnf -days 3650 -x509 -out ~/ikube/tls/tls.crt -keyout ~/ikube/tls/tls.key

kubectl create -n kube-system secret tls ssl --cert ~/ikube/tls/tls.crt --key ~/ikube/tls/tls.key

2）执行yaml文件，创建traefik

kubectl apply -f traefik.yaml

traefik.yaml文件内容在如下链接地址处复制：

https://raw.githubusercontent.com/luckylucky421/kubernetes1.17.3/master/traefik.yaml

上面如果访问不了，可以访问下面的链接，然后把下面的分支克隆和下载，手动把yaml文件传到master1上即可：

https://github.com/luckylucky421/kubernetes1.17.3

3）查看traefik是否部署成功：

kubectl get pods -n kube-system

traefik-ingress-controller-csbp8 1/1 Running 0 5s

traefik-ingress-controller-hqkwf 1/1 Running 0 5s

traefik-ingress-controller-wtjqd 1/1 Running 0 5s

**3.安装kubernetes-dashboard-2版本（kubernetes的web ui界面）**

把kubernetes-dashboard镜像上传到各个节点，按照如下方法通过docker load -i解压，镜像地址在文章开头处的百度网盘里，可自行下载

docker load -i dashboard\_2\_0\_0.tar.gz

docker load -i metrics-scrapter-1-0-1.tar.gz

解压出来的镜像是kubernetesui/dashboard:v2.0.0-beta8和kubernetesui/metrics-scraper:v1.0.1

在master1节点操作

kubectl apply -f kubernetes-dashboard.yaml

kubernetes-dashboard.yaml文件内容在如下链接地址处复制<https://raw.githubusercontent.com/luckylucky421/kubernetes1.17.3/master/kubernetes-dashboard.yaml>

上面如果访问不了，可以访问下面的链接，然后把下面的分支克隆和下载，手动把yaml文件传到master1上即可：

https://github.com/luckylucky421/kubernetes1.17.3

查看dashboard是否安装成功：

kubectl get pods -n kubernetes-dashboard

显示如下，说明dashboard安装成功了

NAME READY STATUS RESTARTS AGE

dashboard-metrics-scraper-694557449d-8xmtf 1/1 Running 0 60s

kubernetes-dashboard-5f98bdb684-ph9wg 1/1 Running 2 60s

查看dashboard前端的service

kubectl get svc -n kubernetes-dashboard

显示如下：

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

dashboard-metrics-scraper ClusterIP 10.100.23.9 <none> 8000/TCP 50s

kubernetes-dashboard ClusterIP 10.105.253.155 <none> 443/TCP 50s

修改service type类型变成NodePort：

kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard

把 type: ClusterIP变成 type: NodePort，保存退出即可。

kubectl get svc -n kubernetes-dashboard

显示如下：

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

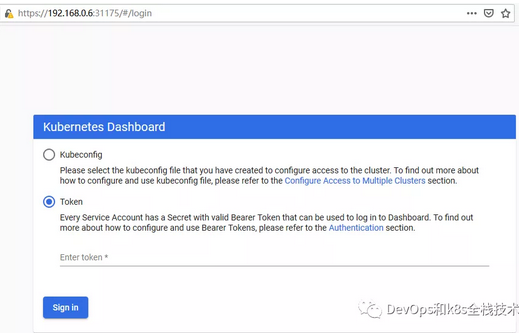
dashboard-metrics-scraper ClusterIP 10.100.23.9 <none> 8000/TCP 3m59s

kubernetes-dashboard NodePort 10.105.253.155 <none> 443:31175/TCP 4m

上面可看到service类型是NodePort，访问master1节点ip:31175端口即可访问kubernetes dashboard，我的环境需要输入如下地址

 https://192.168.0.6:31775/

可看到出现了dashboard界面



**3.1通过yaml文件里指定的默认的token登陆dashboard**

**1）查看kubernetes-dashboard名称空间下的secret**

kubectl get secret -n kubernetes-dashboard

**显示如下：**

NAME TYPE DATA AGE

default-token-vxd7t kubernetes.io/service-account-token 3 5m27s

kubernetes-dashboard-certs Opaque 0 5m27s

kubernetes-dashboard-csrf Opaque 1 5m27s

kubernetes-dashboard-key-holder Opaque 2 5m27s

kubernetes-dashboard-token-ngcmg kubernetes.io/service-account-token 3 5m27s

**2）找到对应的带有token的kubernetes-dashboard-token-ngcmg**

kubectl  describe  secret  kubernetes-dashboard-token-ngcmg  -n   kubernetes-dashboard

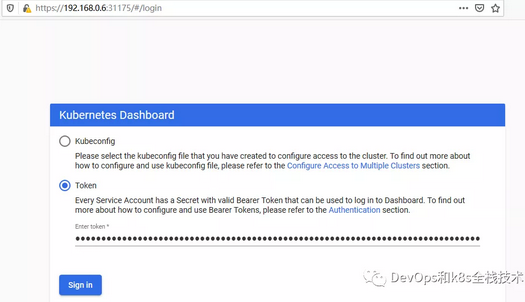
**显示如下：**

...

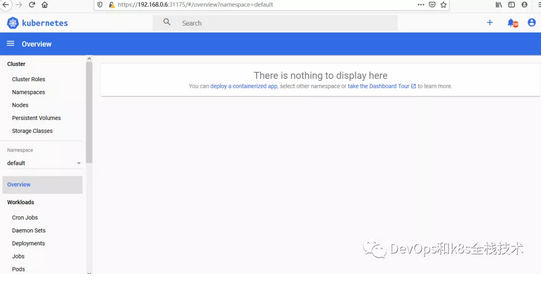
...  
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IjZUTVVGMDN4enFTREpqV0s3cDRWa254cTRPc2xPRTZ3bk8wcFJBSy1JSzgifQ..WQFE0ygYdKkUjaQjFFU-BeWqys07J98N24R\_azv6f-o9AB8Zy1bFWZcNrOlo6WYQuh-xoR8tc5ZDuLQlnZMBSwl2jo9E9FLZuEt7klTfXf4TkrQGLCxzDMD5c2nXbdDdLDtRbSwQMcQwePwp5WTAfuLyqJPFs22Xi2awpLRzbHn3ei\_czNuamWUuoGHe6kP\_rTnu6OUpVf1txi9C1Tg\_3fM2ibNy-NWXLvrxilG3x3SbW1A3G6Y2Vbt1NxqVNtHRRQsYCvTnp3NZQqotV0-TxnvRJ3SLo\_X6oxdUVnqt3DZgebyIbmg3wvgAzGmuSLlqMJ-mKQ7cNYMFR2Z8vnhhtA

**记住token后面的值，把下面的token值复制到浏览器token登陆处即可登陆：**

eyJhbGciOiJSUzI1NiIsImtpZCI6IjZUTVVGMDN4enFTREpqV0s3cDRWa254cTRPc2xPRTZ3bk8wcFJBSy1JSzgifQ..WQFE0ygYdKkUjaQjFFU-BeWqys07J98N24R\_azv6f-o9AB8Zy1bFWZcNrOlo6WYQuh-xoR8tc5ZDuLQlnZMBSwl2jo9E9FLZuEt7klTfXf4TkrQGLCxzDMD5c2nXbdDdLDtRbSwQMcQwePwp5WTAfuLyqJPFs22Xi2awpLRzbHn3ei\_czNuamWUuoGHe6kP\_rTnu6OUpVf1txi9C1Tg\_3fM2ibNy-NWXLvrxilG3x3SbW1A3G6Y2Vbt1NxqVNtHRRQsYCvTnp3NZQqotV0-TxnvRJ3SLo\_X6oxdUVnqt3DZgebyIbmg3wvgAzGmuSLlqMJ-mKQ7cNYMFR2Z8vnhhtA



点击sing in登陆，显示如下，默认是只能看到default名称空间内容



**3.2 创建管理员token，可查看任何空间权限**

kubectl create clusterrolebinding dashboard-cluster-admin --clusterrole=cluster-admin --serviceaccount=kubernetes-dashboard:kubernetes-dashboard

**1）查看kubernetes-dashboard名称空间下的secret**

kubectl get secret -n kubernetes-dashboard

**显示如下：**

NAME TYPE DATA AGE

default-token-vxd7t kubernetes.io/service-account-token 3 5m27s

kubernetes-dashboard-certs Opaque 0 5m27s

kubernetes-dashboard-csrf Opaque 1 5m27s

kubernetes-dashboard-key-holder Opaque 2 5m27s

kubernetes-dashboard-token-ngcmg kubernetes.io/service-account-token 3 5m27s

**2）找到对应的带有token的kubernetes-dashboard-token-ngcmg**

kubectl  describe  secret  kubernetes-dashboard-token-ngcmg  -n   kubernetes-dashboard

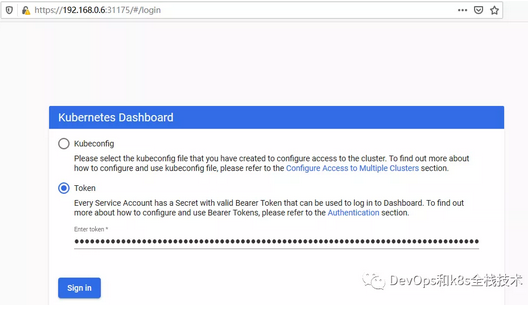
**显示如下：**

...

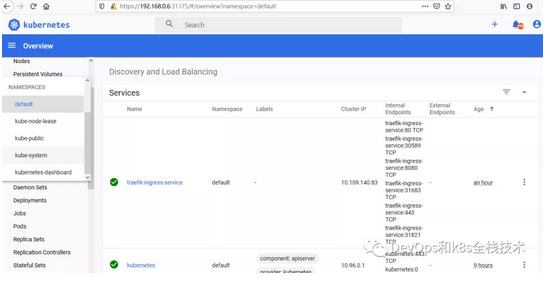
...  
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IjZUTVVGMDN4enFTREpqV0s3cDRWa254cTRPc2xPRTZ3bk8wcFJBSy1JSzgifQ..WQFE0ygYdKkUjaQjFFU-BeWqys07J98N24R\_azv6f-o9AB8Zy1bFWZcNrOlo6WYQuh-xoR8tc5ZDuLQlnZMBSwl2jo9E9FLZuEt7klTfXf4TkrQGLCxzDMD5c2nXbdDdLDtRbSwQMcQwePwp5WTAfuLyqJPFs22Xi2awpLRzbHn3ei\_czNuamWUuoGHe6kP\_rTnu6OUpVf1txi9C1Tg\_3fM2ibNy-NWXLvrxilG3x3SbW1A3G6Y2Vbt1NxqVNtHRRQsYCvTnp3NZQqotV0-TxnvRJ3SLo\_X6oxdUVnqt3DZgebyIbmg3wvgAzGmuSLlqMJ-mKQ7cNYMFR2Z8vnhhtA

**记住token后面的值，把下面的token值复制到浏览器token登陆处即可登陆：**

eyJhbGciOiJSUzI1NiIsImtpZCI6IjZUTVVGMDN4enFTREpqV0s3cDRWa254cTRPc2xPRTZ3bk8wcFJBSy1JSzgifQ..WQFE0ygYdKkUjaQjFFU-BeWqys07J98N24R\_azv6f-o9AB8Zy1bFWZcNrOlo6WYQuh-xoR8tc5ZDuLQlnZMBSwl2jo9E9FLZuEt7klTfXf4TkrQGLCxzDMD5c2nXbdDdLDtRbSwQMcQwePwp5WTAfuLyqJPFs22Xi2awpLRzbHn3ei\_czNuamWUuoGHe6kP\_rTnu6OUpVf1txi9C1Tg\_3fM2ibNy-NWXLvrxilG3x3SbW1A3G6Y2Vbt1NxqVNtHRRQsYCvTnp3NZQqotV0-TxnvRJ3SLo\_X6oxdUVnqt3DZgebyIbmg3wvgAzGmuSLlqMJ-mKQ7cNYMFR2Z8vnhhtA



点击sing in登陆，显示如下，这次就可以看到和操作任何名称空间的资源了



**4.安装metrics插件**

把metrics-server-amd64\_0\_3\_1.tar.gz和addon.tar.gz镜像上传到各个节点，按照如下方法通过docker load -i解压，镜像地址在文章开头处的百度网盘里，可自行下载

docker load -i metrics-server-amd64\_0\_3\_1.tar.gz

docker load -i addon.tar.gz

metrics-server版本0.3.1，用到的镜像是k8s.gcr.io/metrics-server-amd64:v0.3.1

addon-resizer版本是1.8.4，用到的镜像是k8s.gcr.io/addon-resizer:1.8.4

在k8s-master节点操作

kubectl apply -f metrics.yaml

metrics.yaml文件内容在如下链接地址处复制

<https://raw.githubusercontent.com/luckylucky421/kubernetes1.17.3/master/metrics.yaml>

上面如果访问不了，可以访问下面的链接，然后把下面的分支克隆和下载，手动把yaml文件传到master1上即可：

https://github.com/luckylucky421/kubernetes1.17.3

上面组件都安装之后，kubectl  get  pods  -n kube-system  -o wide，查看组件安装是否正常，STATUS状态是Running，说明组件正常，如下所示

NAME READY STATUS RESTARTS AGE

calico-node-6rvqm 1/1 Running 10 14h

calico-node-cbrvw 1/1 Running 4 14h

calico-node-l6628 1/1 Running 0 9h

coredns-7ff77c879f-j48h6 1/1 Running 2 16h

coredns-7ff77c879f-lrb77 1/1 Running 2 16h

etcd-master1 1/1 Running 37 16h

etcd-master2 1/1 Running 7 9h

kube-apiserver-master1 1/1 Running 52 16h

kube-apiserver-master2 1/1 Running 11 14h

kube-controller-manager-master1 1/1 Running 42 16h

kube-controller-manager-master2 1/1 Running 13 14h

kube-proxy-dq6vc 1/1 Running 2 14h

kube-proxy-njft6 1/1 Running 2 16h

kube-proxy-stv52 1/1 Running 0 9h

kube-scheduler-master1 1/1 Running 37 16h

kube-scheduler-master2 1/1 Running 15 14h

kubernetes-dashboard-85f499b587-dbf72 1/1 Running 1 8h

metrics-server-8459f8db8c-5p59m 2/2 Running 0 33s

traefik-ingress-controller-csbp8 1/1 Running 0 8h

traefik-ingress-controller-hqkwf 1/1 Running 0 8h

traefik-ingress-controller-wtjqd 1/1 Running 0 8h

上面如果看到metrics-server-8459f8db8c-5p59m是running状态，说明metrics-server组件部署成功了，接下来就可以在master1节点上使用kubectl top pods -n kube-system或者kubectl top nodes命令