kubeadm安装多master节点的k8s集群-1.20以上稳定版本

加我微信可进技术群学习交流-获取最新课件资料

微信号：

luckylucky421302

也可通过扫描下面二维码添加



课程更新的知识点会通过微信公众号免费分享给大家，可以关注我的公众号



推荐课程如下：

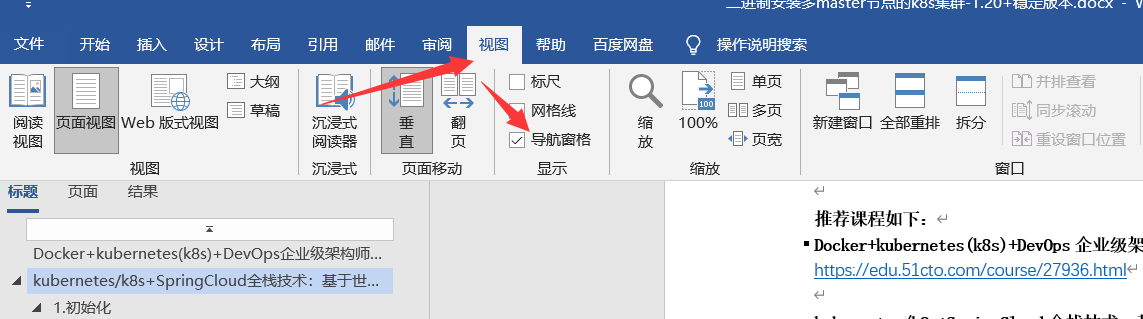
# Docker+kubernetes(k8s)+DevOps企业级架构师实战培训

<https://edu.51cto.com/course/27936.html>

# kubernetes/k8s+SpringCloud全栈技术：基于世界500强的企业实战课程

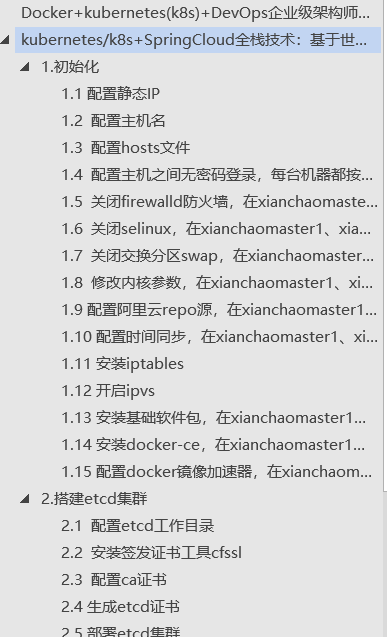
https://edu.51cto.com/course/26635.html

文档使用方法：



打开试图🡪勾选导航窗格

会看到左侧有详细的目录大纲



k8s环境规划：

podSubnet（pod网段） 10.244.0.0/16

serviceSubnet（service网段）: 10.10.0.0/16

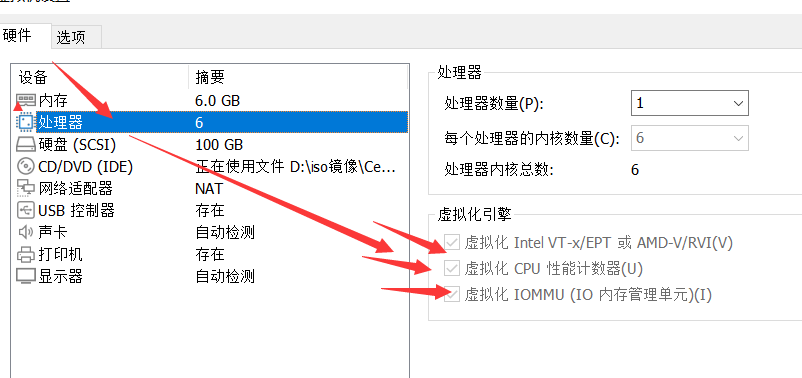
实验环境规划：

操作系统：centos7.6

配置： 4Gib内存/6vCPU/100G硬盘

网络：NAT

开启虚拟机的虚拟化：





kubeadm和二进制安装k8s适用场景分析

kubeadm是官方提供的开源工具，是一个开源项目，用于快速搭建kubernetes集群，目前是比较方便和推荐使用的。kubeadm init 以及 kubeadm join 这两个命令可以快速创建 kubernetes 集群。Kubeadm初始化k8s，所有的组件都是以pod形式运行的，具备故障自恢复能力。

kubeadm是工具，可以快速搭建集群，也就是相当于用程序脚本帮我们装好了集群，属于自动部署，简化部署操作，自动部署屏蔽了很多细节，使得对各个模块感知很少，如果对k8s架构组件理解不深的话，遇到问题比较难排查。

kubeadm适合需要经常部署k8s，或者对自动化要求比较高的场景下使用。

二进制：在官网下载相关组件的二进制包，如果手动安装，对kubernetes理解也会更全面。

Kubeadm和二进制都适合生产环境，在生产环境运行都很稳定，具体如何选择，可以根据实际项目进行评估。

### 1、初始化安装k8s集群的实验环境

#### 1.1 修改机器IP，变成静态IP

vim /etc/sysconfig/network-scripts/ifcfg-ens33文件

TYPE=Ethernet

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=static

IPADDR=192.168.40.180

NETMASK=255.255.255.0

GATEWAY=192.168.40.2

DNS1=192.168.40.2

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：/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

IPADDR=192.168.40.180 #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=192.168.40.2 #网关，在自己电脑打开cmd，输入ipconfig /all可看到

DNS1=192.168.40.2 #DNS，在自己电脑打开cmd，输入ipconfig /all可看到

#### 1.2 配置机器主机名

在192.168.40.180上执行如下：

hostnamectl set-hostname xianchaomaster1 && bash

在192.168.40.181上执行如下：

hostnamectl set-hostname xianchaomaster2 && bash

在192.168.40.182上执行如下：

hostnamectl set-hostname xianchaonode1 && bash

#### 1.3 配置主机hosts文件，相互之间通过主机名互相访问

修改每台机器的/etc/hosts文件，增加如下三行：

192.168.40.180 xianchaomaster1

192.168.40.181 xianchaomaster2

192.168.40.182 xianchaonode1

#### 1.4 配置主机之间无密码登录

[root@xianchaomaster1 ~]# ssh-keygen #一路回车，不输入密码

把本地生成的密钥文件和私钥文件拷贝到远程主机

[root@xianchaomaster1 ~]# ssh-copy-id xianchaomaster1





[root@xianchaomaster1 ~]# ssh-copy-id xianchaomaster2

[root@xianchaomaster1 ~]# ssh-copy-id xianchaonode1

[root@xianchaomaster2 ~]# ssh-keygen #一路回车，不输入密码

把本地的生成的密钥文件和私钥文件拷贝到远程主机

[root@xianchaomaster2 ~]# ssh-copy-id xianchaomaster1

[root@xianchaomaster2 ~]# ssh-copy-id xianchaomaster2

[root@xianchaomaster2 ~]# ssh-copy-id xianchaonode1

[root@xianchaonode1~]# ssh-keygen #一路回车，不输入密码

把本地的生成的密钥文件和私钥文件拷贝到远程主机

[root@xianchaonode1~]# ssh-copy-id xianchaomaster1

[root@xianchaonode1 ~]# ssh-copy-id xianchaomaster2

[root@xianchaonode1 ~]# ssh-copy-id xianchaonode1

#### 1.5 关闭交换分区swap，提升性能

#临时关闭

[root@xianchaomaster1 ~]# swapoff -a

[root@xianchaomaster2 ~]# swapoff -a

[root@xianchaonode1~]# swapoff -a

#永久关闭：注释swap挂载，给swap这行开头加一下注释

[root@xianchaomaster1 ~]# vim /etc/fstab

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

#如果是克隆的虚拟机，需要删除UUID

[root@xianchaomaster2 ~]# vim /etc/fstab

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

#如果是克隆的虚拟机，需要删除UUID

[root@xianchaonode1 ~]# vim /etc/fstab #给swap这行开头加一下注释#

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

#如果是克隆的虚拟机，需要删除UUID

问题1：为什么要关闭swap交换分区？

**Swap是交换分区，如果机器内存不够，会使用swap分区，但是swap分区的性能较低，k8s设计的时候为了能提升性能，默认是不允许使用交换分区的。Kubeadm初始化的时候会检测swap是否关闭，如果没关闭，那就初始化失败。如果不想要关闭交换分区，安装k8s的时候可以指定--ignore-preflight-errors=Swap来解决。**

#### 1.6 修改机器内核参数

[root@xianchaomaster1 ~]# modprobe br\_netfilter

[root@xianchaomaster1 ~]# echo "modprobe br\_netfilter" >> /etc/profile

[root@xianchaomaster1 ~]# cat > /etc/sysctl.d/k8s.conf <<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

[root@xianchaomaster1 ~]# sysctl -p /etc/sysctl.d/k8s.conf

[root@xianchaomaster2 ~]# modprobe br\_netfilter

[root@xianchaomaster2 ~]# echo "modprobe br\_netfilter" >> /etc/profile

[root@xianchaomaster2 ~]# cat > /etc/sysctl.d/k8s.conf <<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

[root@xianchaomaster2 ~]# sysctl -p /etc/sysctl.d/k8s.conf

[root@xianchaonode1 ~]# modprobe br\_netfilter

[root@xianchaonode1 ~]# echo "modprobe br\_netfilter" >> /etc/profile

[root@xianchaonode1 ~]# cat > /etc/sysctl.d/k8s.conf <<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

[root@xianchaonode1 ~]# sysctl -p /etc/sysctl.d/k8s.conf

问题1：sysctl是做什么的？

在运行时配置内核参数

  -p   从指定的文件加载系统参数，如不指定即从/etc/sysctl.conf中加载

问题2：为什么要执行modprobe br\_netfilter？

修改/etc/sysctl.d/k8s.conf文件，增加如下三行参数：

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

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

net.ipv4.ip\_forward = 1

sysctl -p /etc/sysctl.d/k8s.conf出现报错：

sysctl: cannot stat /proc/sys/net/bridge/bridge-nf-call-ip6tables: No such file or directory

sysctl: cannot stat /proc/sys/net/bridge/bridge-nf-call-iptables: No such file or directory

解决方法：

modprobe br\_netfilter

问题3：为什么开启net.bridge.bridge-nf-call-iptables内核参数？

在centos下安装docker，执行docker info出现如下警告：

WARNING: bridge-nf-call-iptables is disabled

WARNING: bridge-nf-call-ip6tables is disabled

解决办法：

vim /etc/sysctl.d/k8s.conf

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

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

问题4：为什么要开启net.ipv4.ip\_forward = 1参数？

kubeadm初始化k8s如果报错：



就表示没有开启ip\_forward，需要开启。

net.ipv4.ip\_forward是数据包转发：

出于安全考虑，Linux系统默认是禁止数据包转发的。所谓转发即当主机拥有多于一块的网卡时，其中一块收到数据包，根据数据包的目的ip地址将数据包发往本机另一块网卡，该网卡根据路由表继续发送数据包。这通常是路由器所要实现的功能。

要让Linux系统具有路由转发功能，需要配置一个Linux的内核参数net.ipv4.ip\_forward。这个参数指定了Linux系统当前对路由转发功能的支持情况；其值为0时表示禁止进行IP转发；如果是1,则说明IP转发功能已经打开。

#### 1.7 关闭firewalld防火墙

[root@xianchaomaster1 ~]# systemctl stop firewalld ; systemctl disable firewalld

[root@xianchaomaster2 ~]# systemctl stop firewalld ; systemctl disable firewalld

[root@xianchaonode1 ~]# systemctl stop firewalld ; systemctl disable firewalld

#### 1.7 关闭selinux

[root@xianchaomaster1 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

#修改selinux配置文件之后，重启机器，selinux配置才能永久生效

[root@xianchaomaster2 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

#修改selinux配置文件之后，重启机器，selinux配置才能永久生效

[root@xianchaonode1 ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

#修改selinux配置文件之后，重启机器，selinux配置才能永久生效

[root@xianchaomaster1 ~]#getenforce

Disabled

#显示Disabled说明selinux已经关闭

[root@xianchaomaster2 ~]#getenforce

Disabled

#显示Disabled说明selinux已经关闭

[root@xianchaonode1 ~]#getenforce

Disabled

#显示Disabled说明selinux已经关闭

#### 1.8 配置阿里云的repo源

在xianchaomaster1上操作：

安装rzsz命令

[root@xianchaomaster1]# yum install lrzsz -y

安装scp：

[root@xianchaomaster1]#yum install openssh-clients

#备份基础repo源

[root@xianchaomaster1 ~]# mkdir /root/repo.bak

[root@xianchaomaster1 ~]# cd /etc/yum.repos.d/

[root@xianchaomaster1]# mv \* /root/repo.bak/

#下载阿里云的repo源

把CentOS-Base.repo文件上传到xianchaomaster1主机的/etc/yum.repos.d/目录下

在xianchaomaster2上操作：

安装rzsz命令

[root@xianchaomaster2]# yum install lrzsz -y

安装scp：

[root@xianchaomaster2]#yum install openssh-clients

#备份基础repo源

[root@xianchaomaster2 ~]# mkdir /root/repo.bak

[root@xianchaomaster2 ~]# cd /etc/yum.repos.d/

[root@xianchaomaster2]# mv \* /root/repo.bak/

#下载阿里云的repo源

把CentOS-Base.repo文件上传到xianchaomaster2主机的/etc/yum.repos.d/目录下

在xianchaonode1上操作：

安装rzsz命令

[root@xianchaonode1]# yum install lrzsz -y

安装scp：

[root@xianchaonode1]#yum install openssh-clients

#备份基础repo源

[root@ xianchaonode1 ~]# mkdir /root/repo.bak

[root@ xianchaonode1 ~]# cd /etc/yum.repos.d/

[root@ xianchaonode1]# mv \* /root/repo.bak/

#下载阿里云的repo源

把CentOS-Base.repo文件上传到xianchaonode1主机的/etc/yum.repos.d/目录下

#配置国内阿里云docker的repo源

[root@xianchaomaster1 ~]# yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

[root@xianchaomaster2 ~]# yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

[root@xianchaonode1 ~]# yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

#### 1.9 配置安装k8s组件需要的阿里云的repo源

[root@xianchaomaster1 ~]#vim /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

#将xianchaomaster1上Kubernetes的repo源复制给xianchaomaster2和xianchaonode1

[root@xianchaomaster1 ~]# scp /etc/yum.repos.d/kubernetes.repo xianchaomaster2:/etc/yum.repos.d/

[root@xianchaomaster1 ~]# scp /etc/yum.repos.d/kubernetes.repo xianchaonode1:/etc/yum.repos.d/

#### 1.10 配置时间同步

#安装ntpdate命令

[root@xianchaomaster1 ~]# yum install ntpdate -y

#跟网络时间做同步

[root@xianchaomaster1 ~]# ntpdate cn.pool.ntp.org

#把时间同步做成计划任务

[root@xianchaomaster1 ~]# crontab -e

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

#重启crond服务

[root@xianchaomaster1 ~]#service crond restart

在xianchaomaster2上执行如下：

#安装ntpdate命令

[root@xianchaomaster2 ~]# yum install ntpdate -y

#跟网络时间做同步

[root@xianchaomaster2 ~]#ntpdate cn.pool.ntp.org

#把时间同步做成计划任务

[root@xianchaomaster2 ~]#crontab -e

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

#重启crond服务

[root@xianchaomaster2 ~]#service crond restart

在xianchaonode1上执行如下：

#安装ntpdate命令

[root@xianchaonode1 ~]# yum install ntpdate -y

#跟网络时间做同步

[root@xianchaonode1 ~]#ntpdate cn.pool.ntp.org

#把时间同步做成计划任务

[root@xianchaonode1 ~]#crontab -e

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

#重启crond服务

[root@xianchaonode1 ~]#service crond restart

#### 1.11开启ipvs

#把ipvs.modules上传到xianchaomaster1机器的/etc/sysconfig/modules/目录下

[root@xianchaomaster1 docker]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep ip\_vs

ip\_vs\_ftp 13079 0

nf\_nat 26583 1 ip\_vs\_ftp

ip\_vs\_sed 12519 0

ip\_vs\_nq 12516 0

ip\_vs\_sh 12688 0

ip\_vs\_dh 12688 0

[root@xianchaomaster1 ~]# scp /etc/sysconfig/modules/ipvs.modules xianchaomaster2:/etc/sysconfig/modules/

[root@xianchaomaster1 ~]# scp /etc/sysconfig/modules/ipvs.modules xianchaonode1:/etc/sysconfig/modules/

[root@xianchaomaster2]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep ip\_vs

ip\_vs\_ftp 13079 0

nf\_nat 26583 1 ip\_vs\_ftp

ip\_vs\_sed 12519 0

ip\_vs\_nq 12516 0

ip\_vs\_sh 12688 0

ip\_vs\_dh 12688 0

[root@xianchaonode1]# chmod 755 /etc/sysconfig/modules/ipvs.modules && bash /etc/sysconfig/modules/ipvs.modules && lsmod | grep ip\_vs

ip\_vs\_ftp 13079 0

nf\_nat 26583 1 ip\_vs\_ftp

ip\_vs\_sed 12519 0

ip\_vs\_nq 12516 0

ip\_vs\_sh 12688 0

ip\_vs\_dh 12688 0

问题1：ipvs是什么？

ipvs (IP Virtual Server) 实现了传输层负载均衡，也就是我们常说的4层LAN交换，作为 Linux 内核的一部分。ipvs运行在主机上，在真实服务器集群前充当负载均衡器。ipvs可以将基于TCP和UDP的服务请求转发到真实服务器上，并使真实服务器的服务在单个 IP 地址上显示为虚拟服务。

问题2：ipvs和iptable对比分析

kube-proxy支持 iptables 和 ipvs 两种模式， 在kubernetes v1.8 中引入了 ipvs 模式，在 v1.9 中处于 beta 阶段，在 v1.11 中已经正式可用了。iptables 模式在 v1.1 中就添加支持了，从 v1.2 版本开始 iptables 就是 kube-proxy 默认的操作模式，ipvs 和 iptables 都是基于netfilter的，但是ipvs采用的是hash表，因此当service数量达到一定规模时，hash查表的速度优势就会显现出来，从而提高service的服务性能。那么 ipvs 模式和 iptables 模式之间有哪些差异呢？

1、ipvs 为大型集群提供了更好的可扩展性和性能

2、ipvs 支持比 iptables 更复杂的复制均衡算法（最小负载、最少连接、加权等等）

3、ipvs 支持服务器健康检查和连接重试等功能

#### 1.12 安装基础软件包

[root@xianchaomaster1 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2 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 telnet ipvsadm

[root@xianchaomaster2 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2 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 telnet ipvsadm

[root@xianchaonode1 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2 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 telnet ipvsadm

#### 1.13 安装iptables

如果用firewalld不习惯，可以安装iptables ，在xianchaomaster1、xianchaomaster2、xianchaonode1上操作：

#安装iptables

yum install iptables-services -y

#禁用iptables

service iptables stop && systemctl disable iptables

#清空防火墙规则

iptables -F

### 2、安装docker服务

#### 2.1 安装docker-ce

[root@xianchaomaster1 ~]# yum install docker-ce-20.10.6 docker-ce-cli-20.10.6 containerd.io -y

[root@xianchaomaster1 ~]# systemctl start docker && systemctl enable docker && systemctl status docker

[root@xianchaomaster2 ~]# yum install docker-ce-20.10.6 docker-ce-cli-20.10.6 containerd.io -y

[root@xianchaomaster2 ~]# systemctl start docker && systemctl enable docker && systemctl status docker

[root@xianchaonode1 ~]# yum install docker-ce-20.10.6 docker-ce-cli-20.10.6 containerd.io -y

[root@xianchaonode1 ~]# systemctl start docker && systemctl enable docker && systemctl status docker

#### 2.2 配置docker镜像加速器和驱动

[root@xianchaomaster1 ~]#vim /etc/docker/daemon.json

{

"registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hub-mirror.c.163.com","<http://qtid6917.mirror.aliyuncs.com>", "https://rncxm540.mirror.aliyuncs.com"],

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

}

#修改docker文件驱动为systemd，默认为cgroupfs，kubelet默认使用systemd，两者必须一致才可以。

[root@xianchaomaster1 ~]# systemctl daemon-reload && systemctl restart docker

[root@xianchaomaster1 ~]# systemctl status docker

[root@xianchaomaster2 ~]#vim /etc/docker/daemon.json

{

"registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hub-mirror.c.163.com","<http://qtid6917.mirror.aliyuncs.com>", "https://rncxm540.mirror.aliyuncs.com"],

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

}

[root@xianchaomaster2 ~]# systemctl daemon-reload

[root@xianchaomaster2 ~]# systemctl restart docker

[root@xianchaomaster2 ~]# systemctl status docke

[root@ xianchaonode1 ~]#vim /etc/docker/daemon.json

{

"registry-mirrors":["https://rsbud4vc.mirror.aliyuncs.com","https://registry.docker-cn.com","https://docker.mirrors.ustc.edu.cn","https://dockerhub.azk8s.cn","http://hub-mirror.c.163.com","<http://qtid6917.mirror.aliyuncs.com>", "https://rncxm540.mirror.aliyuncs.com"],

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

}

[root@xianchaonode1 ~]# systemctl daemon-reload

[root@xianchaonode1 ~]# systemctl restart docker

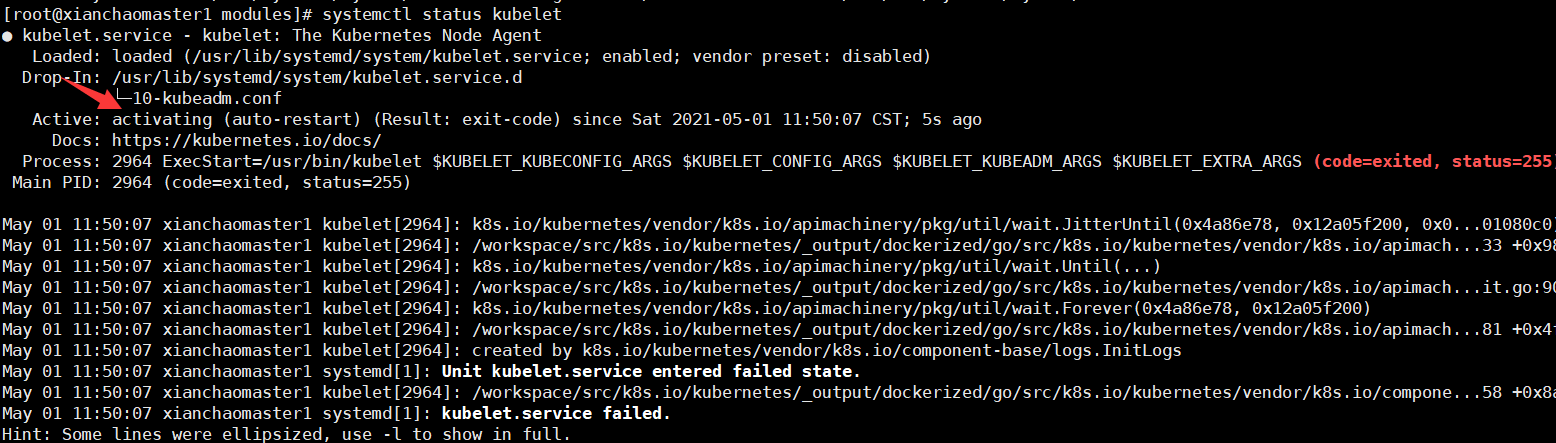
[root@xianchaonode1 ~]# systemctl status docker

### 3、安装初始化k8s需要的软件包

[root@xianchaomaster1 ~]# yum install -y kubelet-1.20.6 kubeadm-1.20.6 kubectl-1.20.6

[root@xianchaomaster1 ~]# systemctl enable kubelet && systemctl start kubelet

[root@xianchaomaster1]# systemctl status kubelet



#上面可以看到kubelet状态不是running状态，这个是正常的，不用管，等k8s组件起来这个kubelet就正常了。

[root@xianchaomaster2 ~]# yum install -y kubelet-1.20.6 kubeadm-1.20.6 kubectl-1.20.6

[root@xianchaomaster2~]# systemctl enable kubelet && systemctl start kubelet

[root@xianchaomaster1 modules]# systemctl enable kubelet && systemctl start kubelet

[root@xianchaomaster2]# systemctl status kubelet

[root@xianchaonode1 ~]# yum install -y kubelet-1.20.6 kubeadm-1.20.6 kubectl-1.20.6

[root@xianchaonode1 ~]# systemctl enable kubelet && systemctl start kubelet

[root@xianchaonode1]# systemctl status kubelet

注：每个软件包的作用

Kubeadm: kubeadm是一个工具，用来初始化k8s集群的

kubelet: 安装在集群所有节点上，用于启动Pod的

kubectl: 通过kubectl可以部署和管理应用，查看各种资源，创建、删除和更新各种组件

### 4、通过keepalive+nginx实现k8s apiserver节点高可用

#配置epel源

把epel.repo上传到xianchaomaster1的/etc/yum.repos.d目录下，这样才能安装keepalived和nginx

#把epel.repo拷贝到远程主机xianchaomaster2和xianchaonode1上

[root@xianchaomaster1 ~]# scp /etc/yum.repos.d/epel.repo xianchaomaster2:/etc/yum.repos.d/

[root@xianchaomaster1 ~]# scp /etc/yum.repos.d/epel.repo xianchaonode1:/etc/yum.repos.d/

1、安装nginx主备：

在xianchaomaster1和xianchaomaster2上做nginx主备安装

[root@xianchaomaster1 ~]# yum install nginx keepalived -y

[root@xianchaomaster2 ~]# yum install nginx keepalived -y

2、修改nginx配置文件。主备一样

[root@xianchaomaster1 ~]# vim /etc/nginx/nginx.conf

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

# 四层负载均衡，为两台Master apiserver组件提供负载均衡

stream {

log\_format main '$remote\_addr $upstream\_addr - [$time\_local] $status $upstream\_bytes\_sent';

access\_log /var/log/nginx/k8s-access.log main;

upstream k8s-apiserver {

server 192.168.40.180:6443; # Master1 APISERVER IP:PORT

server 192.168.40.181:6443; # Master2 APISERVER IP:PORT

}

server {

listen 16443; # 由于nginx与master节点复用，这个监听端口不能是6443，否则会冲突

proxy\_pass k8s-apiserver;

}

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

server {

listen 80 default\_server;

server\_name \_;

location / {

}

}

}

[root@xianchaomaster2 ~]# vim /etc/nginx/nginx.conf

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

# 四层负载均衡，为两台Master apiserver组件提供负载均衡

stream {

log\_format main '$remote\_addr $upstream\_addr - [$time\_local] $status $upstream\_bytes\_sent';

access\_log /var/log/nginx/k8s-access.log main;

upstream k8s-apiserver {

server 192.168.40.180:6443; # Master1 APISERVER IP:PORT

server 192.168.40.181:6443; # Master2 APISERVER IP:PORT

}

server {

listen 16443; # 由于nginx与master节点复用，这个监听端口不能是6443，否则会冲突

proxy\_pass k8s-apiserver;

}

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

server {

listen 80 default\_server;

server\_name \_;

location / {

}

}

}

3、keepalive配置

主keepalived

[root@xianchaomaster1 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_MASTER

}

vrrp\_script check\_nginx {

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

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33 # 修改为实际网卡名

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 100 # 优先级，备服务器设置 90

advert\_int 1 # 指定VRRP 心跳包通告间隔时间，默认1秒

authentication {

auth\_type PASS

auth\_pass 1111

}

# 虚拟IP

virtual\_ipaddress {

192.168.40.199/24

}

track\_script {

check\_nginx

}

}

#vrrp\_script：指定检查nginx工作状态脚本（根据nginx状态判断是否故障转移）

#virtual\_ipaddress：虚拟IP（VIP）

[root@xianchaomaster1 ~]# vim /etc/keepalived/check\_nginx.sh

#!/bin/bash

count=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$")

if [ "$count" -eq 0 ];then

systemctl stop keepalived

fi

[root@xianchaomaster1 ~]# chmod +x /etc/keepalived/check\_nginx.sh

备keepalive

[root@xianchaomaster2 ~]# vim /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_BACKUP

}

vrrp\_script check\_nginx {

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

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens33

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.40.199/24

}

track\_script {

check\_nginx

}

}

[root@xianchaomaster2 ~]# vim /etc/keepalived/check\_nginx.sh

#!/bin/bash

count=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$")

if [ "$count" -eq 0 ];then

systemctl stop keepalived

fi

[root@xianchaomaster2 ~]# chmod +x /etc/keepalived/check\_nginx.sh

#注：keepalived根据脚本返回状态码（0为工作正常，非0不正常）判断是否故障转移。

4、启动服务：

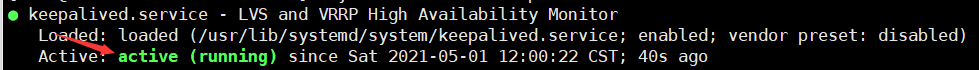
[root@xianchaomaster1 ~]# systemctl daemon-reload

[root@xianchaomaster1 ~]# systemctl start nginx

[root@xianchaomaster1 ~]# systemctl start keepalived

[root@xianchaomaster1 ~]# systemctl enable nginx keepalived

[root@xianchaomaster1]# systemctl status keepalived



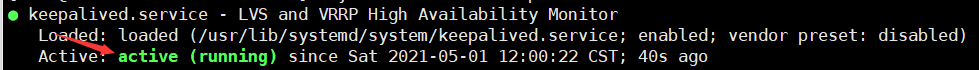
[root@xianchaomaster2 ~]# systemctl daemon-reload

[root@xianchaomaster2 ~]# systemctl start nginx

[root@xianchaomaster2 ~]# systemctl start keepalived

[root@xianchaomaster2 ~]# systemctl enable nginx keepalived

[root@xianchaomaster2]# systemctl status keepalived



5、测试vip是否绑定成功

[root@xianchaomaster1 ~]# ip addr

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:29:79:9e:36 brd ff:ff:ff:ff:ff:ff

inet 192.168.40.180/24 brd 192.168.40.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.40.199/24 scope global secondary ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::b6ef:8646:1cfc:3e0c/64 scope link noprefixroute

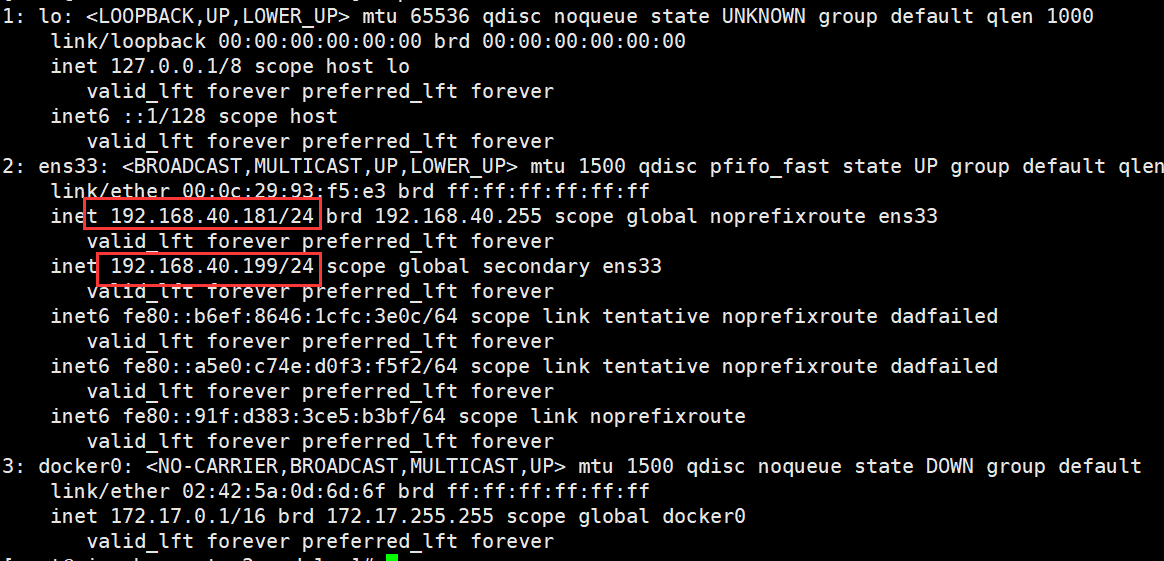
valid\_lft forever preferred\_lft forever

6、测试keepalived：

停掉xianchaomaster1上的nginx。Vip会漂移到xianchaomaster2

[root@xianchaomaster1 ~]# service nginx stop

[root@xianchaomaster2]# ip addr



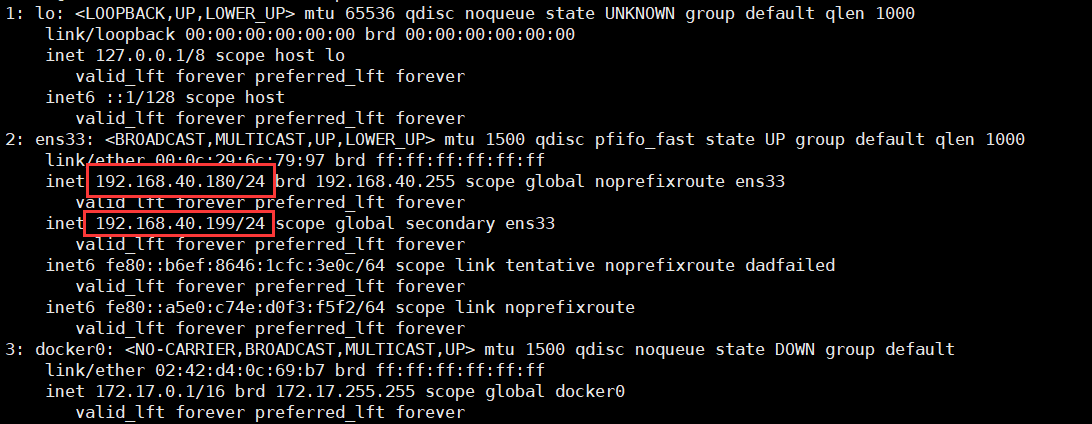
#启动xianchaomaster1上的nginx和keepalived，vip又会漂移回来

[root@xianchaomaster1 ~]# systemctl daemon-reload

[root@xianchaomaster1 ~]# systemctl start nginx

[root@xianchaomaster1 ~]# systemctl start keepalived

[root@xianchaomaster1]# ip addr



### 5、kubeadm初始化k8s集群

在xianchaomaster1上创建kubeadm-config.yaml文件：

[root@xianchaomaster1 ~]# cd /root/

[root@xianchaomaster1]# vim kubeadm-config.yaml

apiVersion: kubeadm.k8s.io/v1beta2

kind: ClusterConfiguration

kubernetesVersion: v1.20.6

controlPlaneEndpoint: 192.168.40.199:16443

imageRepository: registry.aliyuncs.com/google\_containers

apiServer:

certSANs:

- 192.168.40.180

- 192.168.40.181

- 192.168.40.182

- 192.168.40.199

networking:

podSubnet: 10.244.0.0/16

serviceSubnet: 10.10.0.0/16

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

#使用kubeadm初始化k8s集群

#把初始化k8s集群需要的离线镜像包上传到xianchaomaster1、xianchaomaster2、xianchaonode1机器上，手动解压：

[root@xianchaomaster1 ~]# docker load -i k8simage-1-20-6.tar.gz

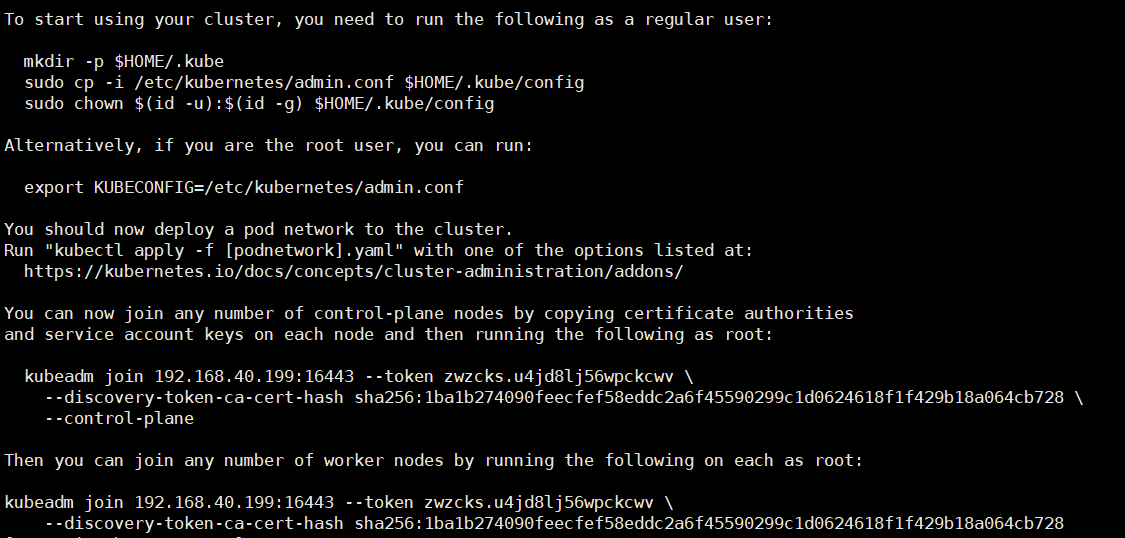
[root@xianchaomaster2 ~]# docker load -i k8simage-1-20-6.tar.gz

[root@xianchaonode1 ~]# docker load -i k8simage-1-20-6.tar.gz

[root@xianchaomaster1]# kubeadm init --config kubeadm-config.yaml

注：--image-repository registry.aliyuncs.com/google\_containers：手动指定仓库地址为registry.aliyuncs.com/google\_containers。kubeadm默认从k8s.grc.io拉取镜像，但是k8s.gcr.io访问不到，所以需要指定从registry.aliyuncs.com/google\_containers仓库拉取镜像。

显示如下，说明安装完成：



kubeadm join 192.168.40.199:16443 --token zwzcks.u4jd8lj56wpckcwv \

--discovery-token-ca-cert-hash sha256:1ba1b274090feecfef58eddc2a6f45590299c1d0624618f1f429b18a064cb728 \

--control-plane

#上面命令是把master节点加入集群，需要保存下来，每个人的都不一样

kubeadm join 192.168.40.199:16443 --token zwzcks.u4jd8lj56wpckcwv \

--discovery-token-ca-cert-hash sha256:1ba1b274090feecfef58eddc2a6f45590299c1d0624618f1f429b18a064cb728

#上面命令是把node节点加入集群，需要保存下来，每个人的都不一样

#配置kubectl的配置文件config，相当于对kubectl进行授权，这样kubectl命令可以使用这个证书对k8s集群进行管理

[root@xianchaomaster1 ~]# mkdir -p $HOME/.kube

[root@xianchaomaster1 ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@xianchaomaster1 ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config

[root@xianchaomaster1 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

xianchaomaster1 NotReady control-plane,master 60s v1.20.6

此时集群状态还是NotReady状态，因为没有安装网络插件。

### 6、扩容k8s集群-添加master节点

#把xianchaomaster1节点的证书拷贝到xianchaomaster2上

在xianchaomaster2创建证书存放目录：

[root@xianchaomaster2 ~]# cd /root && mkdir -p /etc/kubernetes/pki/etcd &&mkdir -p ~/.kube/

#把xianchaomaster1节点的证书拷贝到xianchaomaster2上：

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/ca.crt xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/ca.key xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/sa.key xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/sa.pub xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/front-proxy-ca.crt xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/front-proxy-ca.key xianchaomaster2:/etc/kubernetes/pki/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/etcd/ca.crt xianchaomaster2:/etc/kubernetes/pki/etcd/

[root@xianchaomaster1 ~]# scp /etc/kubernetes/pki/etcd/ca.key xianchaomaster2:/etc/kubernetes/pki/etcd/

#证书拷贝之后在xianchaomaster2上执行如下命令，大家复制自己的，这样就可以把xianchaomaster2和加入到集群，成为控制节点：

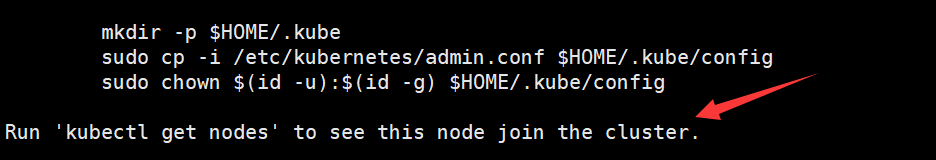
在xianchaomaster1上查看加入节点的命令：

[root@xianchaomaster1 ~]# kubeadm token create --print-join-command

kubeadm join 192.168.40.199:16443 --token zwzcks.u4jd8lj56wpckcwv \

--discovery-token-ca-cert-hash sha256:1ba1b274090feecfef58eddc2a6f45590299c1d0624618f1f429b18a064cb728 \

--control-plane



#看到上面说明xianchaomaster2节点已经加入到集群了

在xianchaomaster1上查看集群状况：

[root@xianchaomaster1 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

xianchaomaster1 NotReady control-plane,master 49m v1.20.6

xianchaomaster2 NotReady <none> 39s v1.20.6

上面可以看到xianchaomaster2已经加入到集群了

### 7、扩容k8s集群-添加node节点

在xianchaomaster1上查看加入节点的命令：

[root@xianchaomaster1 ~]# kubeadm token create --print-join-command

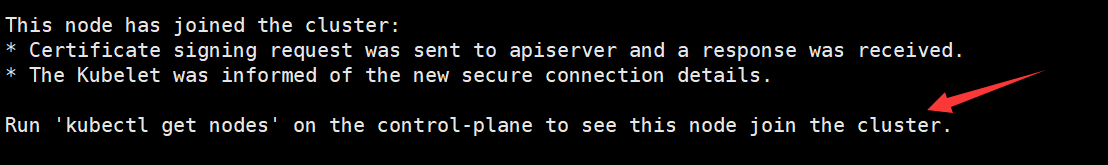
#显示如下：

kubeadm join 192.168.40.199:16443 --token y23a82.hurmcpzedblv34q8 --discovery-token-ca-cert-hash sha256:1ba1b274090feecfef58eddc2a6f45590299c1d0624618f1f429b18a064cb728

把xianchaonode1加入k8s集群：

[root@xianchaonode1~]# kubeadm token create --print-join-command

kubeadm join 192.168.40.199:16443 --token y23a82.hurmcpzedblv34q8 --discovery-token-ca-cert-hash sha256:1ba1b274090feecfef58eddc2a6f45590299c1d0624618f1f429b18a064cb728



#看到上面说明xianchaonode1节点已经加入到集群了,充当工作节点

#在xianchaomaster1上查看集群节点状况：

[root@xianchaomaster1 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

xianchaomaster1 NotReady control-plane,master 53m v1.20.6

xianchaomaster2 NotReady control-plane,master 5m13s v1.20.6

xianchaonode1 NotReady <none> 59s v1.20.6

#可以看到xianchaonode1的ROLES角色为空，<none>就表示这个节点是工作节点。

#可以把xianchaonode1的ROLES变成work，按照如下方法：

[root@xianchaomaster1 ~]# kubectl label node xianchaonode1 node-role.kubernetes.io/worker=worker

注意：上面状态都是NotReady状态，说明没有安装网络插件

[root@xianchaomaster1 ~]# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

coredns-7f89b7bc75-lh28j 0/1 Pending 0 18h

coredns-7f89b7bc75-p7nhj 0/1 Pending 0 18h

etcd-xianchaomaster1 1/1 Running 0 18h

etcd-xianchaomaster2 1/1 Running 0 15m

kube-apiserver-xianchaomaster1 1/1 Running 0 18h

kube-apiserver-xianchaomaster2 1/1 Running 0 15m

kube-controller-manager-xianchaomaster1 1/1 Running 1 18h

kube-controller-manager-xianchaomaster2 1/1 Running 0 15m

kube-proxy-n26mf 1/1 Running 0 4m33s

kube-proxy-sddbv 1/1 Running 0 18h

kube-proxy-sgqm2 1/1 Running 0 15m

kube-scheduler-xianchaomaster1 1/1 Running 1 18h

kube-scheduler-xianchaomaster2 1/1 Running 0 15m

coredns-7f89b7bc75-lh28j是pending状态，这是因为还没有安装网络插件，等到下面安装好网络插件之后这个cordns就会变成running了

### 8、安装kubernetes网络组件-Calico

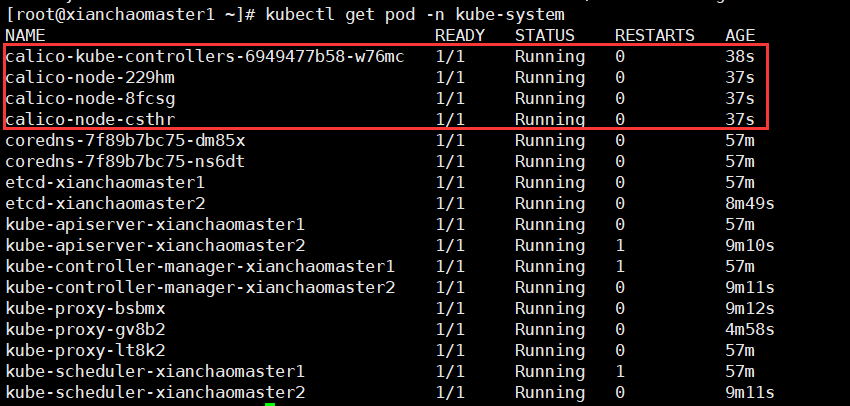
上传calico.yaml到xianchaomaster1上，使用yaml文件安装calico 网络插件 。

[root@xianchaomaster1 ~]# kubectl apply -f calico.yaml

注：在线下载配置文件地址是： https://docs.projectcalico.org/manifests/calico.yaml

。

[root@xianchaomaster1 ~]# kubectl get pod -n kube-system



coredns-这个pod现在是running状态，运行正常

再次查看集群状态。

[root@xianchaomaster1 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

xianchaomaster1 Ready control-plane,master 58m v1.20.6

xianchaomaster2 Ready control-plane,master 10m v1.20.6

xianchaonode1 Ready <none> 5m46s v1.20.6

#STATUS状态是Ready，说明k8s集群正常运行了

### 9、测试在k8s创建pod是否可以正常访问网络

#把busybox-1-28.tar.gz上传到xianchaonode1节点，手动解压

[root@xianchaonode1 ~]# docker load -i busybox-1-28.tar.gz

[root@xianchaomaster1 ~]# kubectl run busybox --image busybox:1.28 --restart=Never --rm -it busybox -- sh

/ # ping www.baidu.com

PING www.baidu.com (39.156.66.18): 56 data bytes

64 bytes from 39.156.66.18: seq=0 ttl=127 time=39.3 ms

#通过上面可以看到能访问网络，说明calico网络插件已经被正常安装了

### 10、测试k8s集群中部署tomcat服务

#把tomcat.tar.gz上传到xianchaonode1，手动解压

[root@xianchaonode1 ~]# docker load -i tomcat.tar.gz

[root@xianchaomaster1 ~]# kubectl apply -f tomcat.yaml

[root@xianchaomaster1 ~]# kubectl get pods

NAME READY STATUS RESTARTS AGE

demo-pod 1/1 Running 0 10s

[root@xianchaomaster1 ~]# kubectl apply -f tomcat-service.yaml

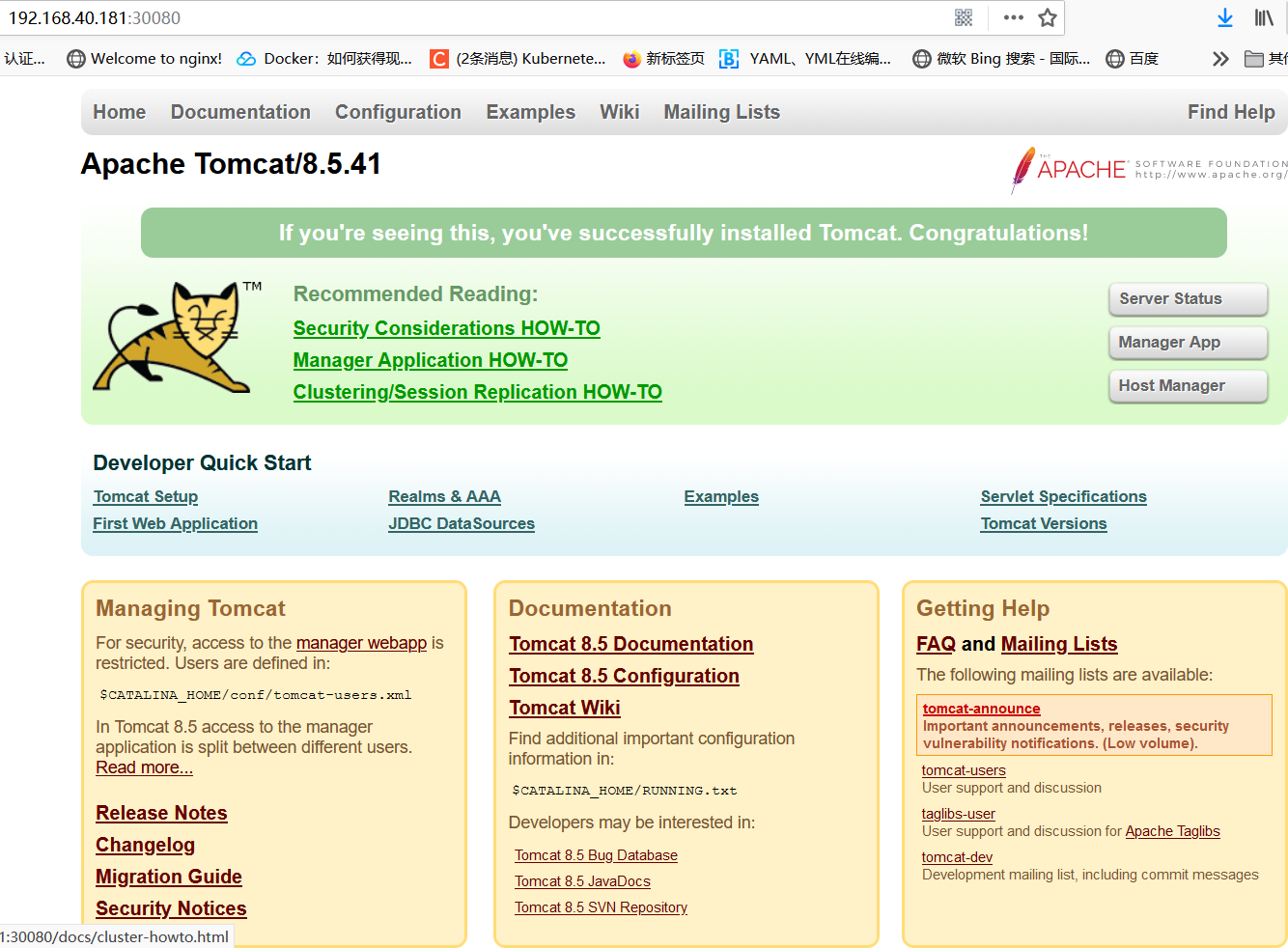
[root@xianchaomaster1 ~]# kubectl get svc

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

kubernetes ClusterIP 10.255.0.1 <none> 443/TCP 158m

tomcat NodePort 10.255.227.179 <none> 8080:30080/TCP 19m

在浏览器访问xianchaonode1节点的ip:30080即可请求到浏览器



### 11、测试coredns是否正常

[root@xianchaomaster1 ~]# kubectl run busybox --image busybox:1.28 --restart=Never --rm -it busybox -- sh

If you don't see a command prompt, try pressing enter.

/ # nslookup kubernetes.default.svc.cluster.local

Server: 10.10.0.10

Address 1: 10.10.0.10 kube-dns.kube-system.svc.cluster.local

Name: kubernetes.default.svc.cluster.local

Address 1: 10.10.0.1 kubernetes.default.svc.cluster.local

/ # nslookup tomcat.default.svc.cluster.local

Server: 10.10.0.10

Address 1: 10.10.0.10 kube-dns.kube-system.svc.cluster.local

Name: tomcat.default.svc.cluster.local

Address 1: 10.10.13.88 tomcat.default.svc.cluster.local

10.10.13.88就是我们coreDNS的clusterIP，说明coreDNS配置好了。

解析内部Service的名称，是通过coreDNS去解析的。

10.10.0.10是创建的tomcat的service ip

#注意：

busybox要用指定的1.28版本，不能用最新版本，最新版本，nslookup会解析不到dns和ip