环境说明

podSubnet（pod网段） 10.244.0.0/16

serviceSubnet（service网段）: 10.96.0.0/16

物理机网段：192.168.40.0/24

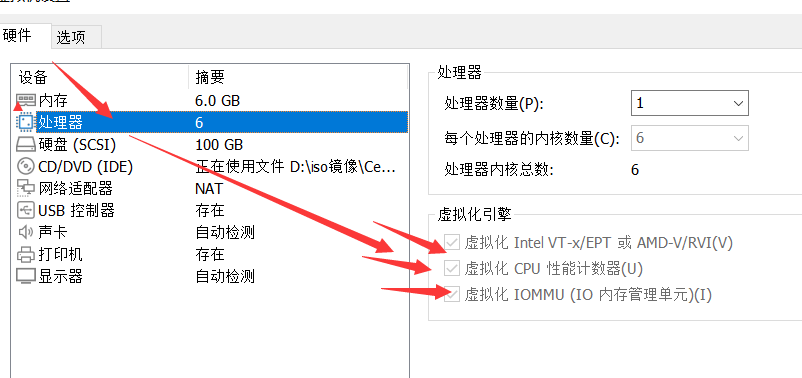
实验环境规划：

操作系统：centos7.9

配置： 4Gib内存/4vCPU/60G硬盘

网络：NAT

开启虚拟机的虚拟化：





### 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

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@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@xianchaonode2 ~]# 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

reboot -f

[root@xianchaomaster1 ~]#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

[root@xianchaonode2~]#getenforce

Disabled

#显示Disabled说明selinux已经关闭

#### 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 xianchaonode1 && bash

在192.168.40.182上执行如下：

hostnamectl set-hostname xianchaonode2 && bash

升级操作系统，k8s集群每台机器都操作：

[root@localhost ~]# yum update -y

#### 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 xianchaonode1

192.168.40.182 xianchaonode2

修改之后内容如下：

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.40.180 xianchaomaster1

192.168.40.181 xianchaonode1

192.168.40.182 xianchaonode2

#### 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 xianchaonode1

[root@xianchaomaster1 ~]# ssh-copy-id xianchaonode2

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



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

[root@xianchaonode1 ~]# ssh-copy-id xianchaonode2

[root@ xianchaonode2 ~]# 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@ xianchaonode2 ~]# ssh-copy-id xianchaomaster1



[root@xianchaonode2 ~]# ssh-copy-id xianchaonode1

[root@xianchaonode2 ~]# ssh-copy-id xianchaonode2

#### 1.5 关闭交换分区swap，提升性能

#临时关闭

[root@xianchaomaster1 ~]# swapoff -a

[root@xianchaonode1 ~]# swapoff -a

[root@xianchaonode2~]# 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

[root@ xianchaonode1 ~]# vim /etc/fstab

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

[root@ xianchaonode2 ~]# vim /etc/fstab #给swap这行开头加一下注释#

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

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@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

[root@xianchaonode2 ~]# modprobe br\_netfilter

[root@xianchaonode2 ~]# echo "modprobe br\_netfilter" >> /etc/profile

[root@xianchaonode2 ~]# 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@xianchaonode2 ~]# 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@xianchaonode1 ~]# systemctl stop firewalld ; systemctl disable firewalld

[root@xianchaonode2 ~]# systemctl stop firewalld ; systemctl disable firewalld

#### 1.8 配置阿里云的repo源

#配置国内阿里云docker的repo源

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

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

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

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

[root@xianchaonode2 ~]# yum install -y yum-utils

[root@xianchaonode2 ~]# yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

[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@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

[root@xianchaonode2 ~]# 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

#### 1.9 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xianchaonode1和xianchaonode2

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

[root@xianchaomaster1 ~]# scp /etc/yum.repos.d/kubernetes.repo xianchaonode2:/etc/yum.repos.d/

#### 1.10 配置时间同步

在xianchaomaster1上执行如下：

#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

在xianchaonode2上执行如下：

#安装ntpdate命令

[root@xianchaonode2 ~]# yum install ntpdate -y

#跟网络时间做同步

[root@xianchaonode2 ~]#ntpdate cn.pool.ntp.org

#把时间同步做成计划任务

[root@xianchaonode2 ~]#crontab -e

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

#重启crond服务

[root@xianchaonode2 ~]#service crond restart

### 2、安装docker服务

#### 2.1 安装docker-ce

[root@xianchaomaster1 ~]# yum install docker-ce-20.10.6 -y

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

[root@xianchaonode1 ~]# yum install docker-ce-20.10.6 -y

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

[root@xianchaonode2 ~]# yum install docker-ce-20.10.6 -y

[root@xianchaonode2 ~]# systemctl start docker && systemctl enable docker.service

#### 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"],

"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@ 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"],

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

}

[root@xianchaonode1 ~]# systemctl daemon-reload

[root@xianchaonode1 ~]# systemctl restart docker

[root@xianchaonode1 ~]# systemctl status docker

[root@ xianchaonode2 ~]#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"],

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

}

[root@xianchaonode2 ~]# systemctl daemon-reload

[root@xianchaonode2 ~]# systemctl restart docker

[root@xianchaonode2 ~]# 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

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

[root@xianchaonode1 ~]# systemctl enable kubelet

[root@xianchaonode2 ~]# yum install -y kubelet-1.20.6 kubeadm-1.20.6 kubectl-1.20.6

[root@xianchaonode2 ~]# systemctl enable 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

### 5、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、xianchaonode1、xianchaonode2机器上，手动解压：

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

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

[root@xianchaonode2 ~]# docker load -i k8simage-1-20-6.tar.gz

使用kubeadm初始化k8s集群

[root@xianchaomaster1~]# kubeadm config print init-defaults > kubeadm.yaml

根据我们自己的需求修改配置，比如修改 imageRepository 的值，kube-proxy 的模式为 ipvs，初始化节点的时候需要指定cgroupDriver为systemd

kubeadm.yaml配置如下：

apiVersion: kubeadm.k8s.io/v1beta2

bootstrapTokens:

- groups:

- system:bootstrappers:kubeadm:default-node-token

token: abcdef.0123456789abcdef

ttl: 24h0m0s

usages:

- signing

- authentication

kind: InitConfiguration

localAPIEndpoint:

advertiseAddress: 192.168.40.180 #控制节点的ip

bindPort: 6443

nodeRegistration:

criSocket: /var/run/dockershim.sock

name: xianchaomaster1 #控制节点主机名

taints:

- effect: NoSchedule

key: node-role.kubernetes.io/master

---

apiServer:

timeoutForControlPlane: 4m0s

apiVersion: kubeadm.k8s.io/v1beta2

certificatesDir: /etc/kubernetes/pki

clusterName: kubernetes

controllerManager: {}

dns:

type: CoreDNS

etcd:

local:

dataDir: /var/lib/etcd

imageRepository: registry.aliyuncs.com/google\_containers

kind: ClusterConfiguration

kubernetesVersion: v1.20.6

networking:

dnsDomain: cluster.local

serviceSubnet: 10.96.0.0/12

podSubnet: 10.244.0.0/16 #指定pod网段， 需要新增加这个

scheduler: {}

#追加如下几行

---

apiVersion: kubeproxy.config.k8s.io/v1alpha1

kind: KubeProxyConfiguration

mode: ipvs

---

apiVersion: kubelet.config.k8s.io/v1beta1

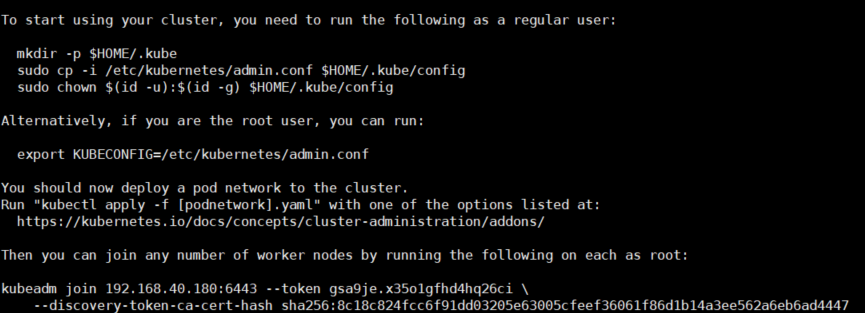
kind: KubeletConfiguration

cgroupDriver: systemd

#基于kubeadm.yaml文件初始化k8s

[root@xianchaomaster1~]# kubeadm init --config=kubeadm.yaml --ignore-preflight-errors=SystemVerification

显示如下，说明安装完成：

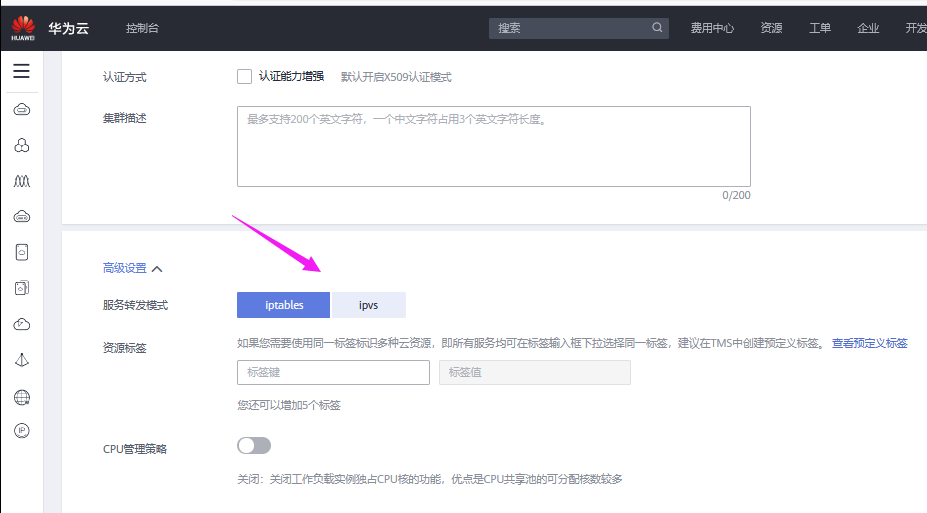


特别提醒：

--image-repository registry.aliyuncs.com/google\_containers为保证拉取镜像不到国外站点拉取，手动指定仓库地址为registry.aliyuncs.com/google\_containers。kubeadm默认从k8s.gcr.io拉取镜像。 我们本地有导入到的离线镜像，所以会优先使用本地的镜像。

mode: ipvs 表示kube-proxy代理模式是ipvs，如果不指定ipvs，会默认使用iptables，但是iptables效率低，所以我们生产环境建议开启ipvs，阿里云和华为云托管的K8s，也提供ipvs模式，如下：





#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

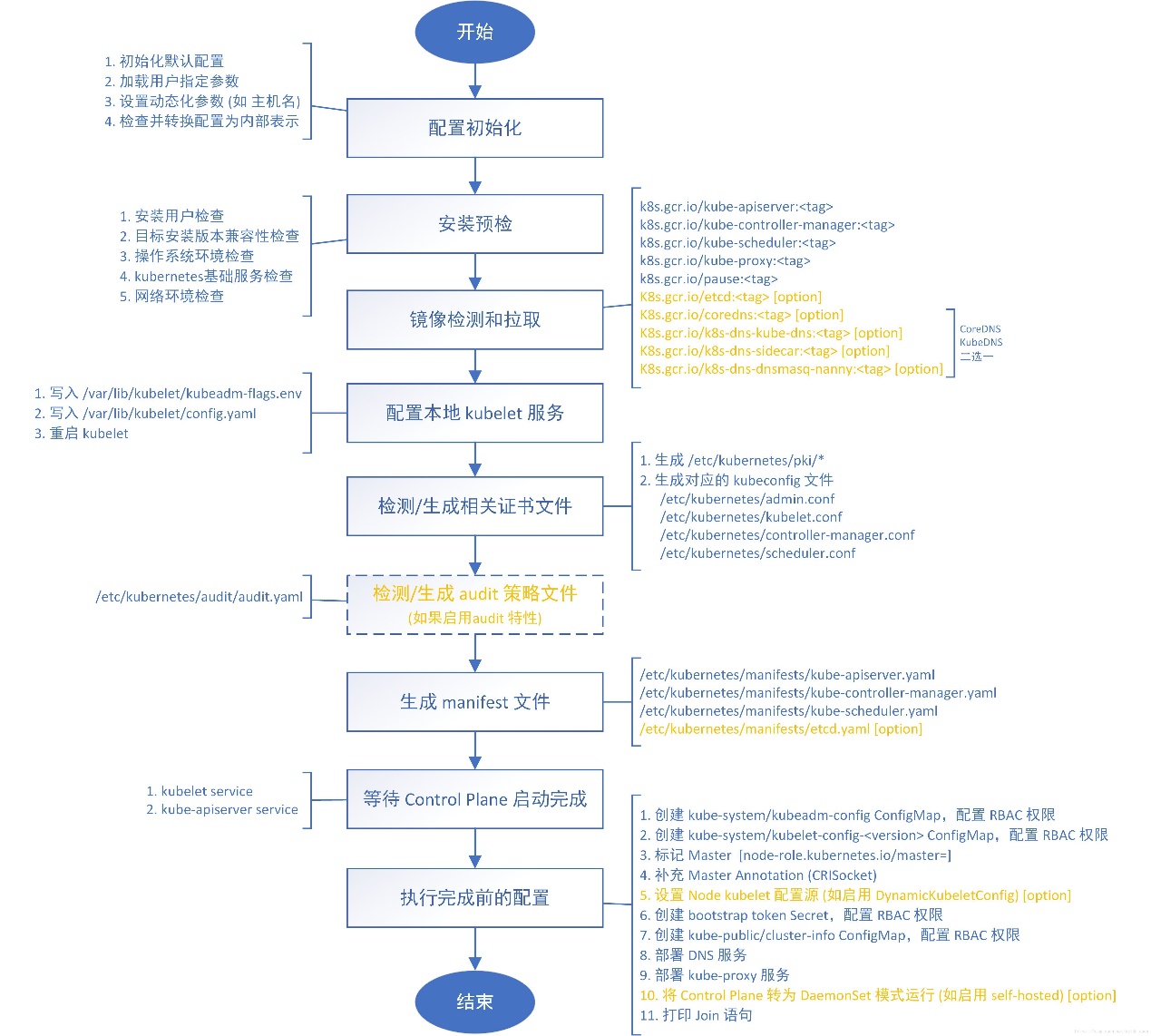
[root@xianchaomaster1 ~]# kubectl get nodes

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。

扩展：kubeadm init初始化流程分析



注意：操作系统需要我们手动配置好，包括防火墙、swap、selinux关掉

kubeadm 在执行安装之前进行了相当细致的环境检测，下面看一看：

1) 检查执行 init 命令的用户是否为 root，如果不是 root，直接快速失败（fail fast）；

2) 检查待安装的 k8s 版本是否被当前版本的 kubeadm 支持（kubeadm 版本 >= 待安装 k8s 版本）；

3) 检查防火墙，如果防火墙未关闭，提示开放端口 10250；

4) 检查端口是否已被占用，6443（或你指定的监听端口）、10257、10259；

5) 检查文件是否已经存在，/etc/kubernetes/manifests/\*.yaml；

6) 检查是否存在代理，连接本机网络、服务网络、Pod网络，都会检查，目前不允许代理；

7) 检查容器运行时，使用 CRI 还是 Docker，如果是 Docker，进一步检查 Docker 服务是否已启动，是否设置了开机自启动；

8) 对于 Linux 系统，会额外检查以下内容：

8.1) 检查以下命令是否存在：crictl、ip、iptables、mount、nsenter、ebtables、ethtool、socat、tc、touch；

8.2) 检查 /proc/sys/net/bridge/bridge-nf-call-iptables、/proc/sys/net/ipv4/ip-forward 内容是否为 1；

8.3) 检查 swap 是否是关闭状态；

9) 检查内核是否被支持，Docker 版本及后端存储 GraphDriver 是否被支持；

对于 Linux 系统，还需检查 OS 版本和 cgroup 支持程度（支持哪些资源的隔离）；

10) 检查主机名访问可达性；

11) 检查 kubelet 版本，要高于 kubeadm 需要的最低版本，同时不高于待安装的 k8s 版本；

12) 检查 kubelet 服务是否开机自启动；

13) 检查 10250 端口是否被占用；

14) 如果开启 IPVS 功能，检查系统内核是否加载了 ipvs 模块；

15) 对于 etcd，如果使用 Local etcd，则检查 2379 端口是否被占用， /var/lib/etcd/ 是否为空目录；

如果使用 External etcd，则检查证书文件是否存在（CA、key、cert），验证 etcd 服务版本是否符合要求；

16) 如果使用 IPv6，

检查 /proc/sys/net/bridge/bridge-nf-call-iptables、/proc/sys/net/ipv6/conf/default/forwarding 内容是否为 1；

以上就是 kubeadm init 需要检查的所有项目了！

完成安装前的配置

1) 在 kube-system 命名空间创建 ConfigMap kubeadm-config，同时对其配置 RBAC 权限；

2) 在 kube-system 命名空间创建 ConfigMap kubelet-config-<version>，同时对其配置 RBAC 权限；

3) 为当前节点（Master）打标记：node-role.kubernetes.io/master=；

4) 为当前节点（Master）补充 Annotation；

5) 如果启用了 DynamicKubeletConfig 特性，设置本节点 kubelet 的配置数据源为 ConfigMap 形式；

6) 创建 BootStrap token Secret，并对其配置 RBAC 权限；

7) 在 kube-public 命名空间创建 ConfigMap cluster-info，同时对其配置 RBAC 权限；

8) 与 apiserver 通信，部署 DNS 服务；

9) 与 apiserver 通信，部署 kube-proxy 服务；

10) 如果启用了 self-hosted 特性，将 Control Plane 转为 DaemonSet 形式运行；

11) 打印 join 语句；

扩展：Kubeadm生成的k8s证书内容说明：

一、证书分组

Kubernetes把证书放在了两个文件夹中

/etc/kubernetes/pki

/etc/kubernetes/pki/etcd

二、Kubernetes 集群根证书

Kubernetes 集群根证书CA(Kubernetes集群组件的证书签发机构)

/etc/kubernetes/pki/ca.crt

/etc/kubernetes/pki/ca.key

以上这组证书为签发其他Kubernetes组件证书使用的根证书, 可以认为是Kubernetes集群中证书签发机构之一

由此根证书签发的证书有:

1、kube-apiserver apiserver证书

/etc/kubernetes/pki/apiserver.crt

/etc/kubernetes/pki/apiserver.key

2、kubelet客户端证书, 用作 kube-apiserver 主动向 kubelet 发起请求时的客户端认证

/etc/kubernetes/pki/apiserver-kubelet-client.crt

/etc/kubernetes/pki/apiserver-kubelet-client.key

三、kube-apiserver 代理根证书(客户端证书)

用在requestheader-client-ca-file配置选项中, kube-apiserver 使用该证书来验证客户端证书是否为自己所签发

/etc/kubernetes/pki/front-proxy-ca.crt

/etc/kubernetes/pki/front-proxy-ca.key

由此根证书签发的证书只有一组:

代理层(如汇聚层aggregator)使用此套代理证书来向 kube-apiserver 请求认证

代理端使用的客户端证书, 用作代用户与 kube-apiserver 认证

/etc/kubernetes/pki/front-proxy-client.crt

/etc/kubernetes/pki/front-proxy-client.key

三、etcd 集群根证书

etcd集群所用到的证书都保存在/etc/kubernetes/pki/etcd这路径下, 很明显, 这一套证书是用来专门给etcd集群服务使用的, 设计以下证书文件

etcd 集群根证书CA(etcd 所用到的所有证书的签发机构)

/etc/kubernetes/pki/etcd/ca.crt

/etc/kubernetes/pki/etcd/ca.key

由此根证书签发机构签发的证书有:

1、etcd server 持有的服务端证书

/etc/kubernetes/pki/etcd/server.crt

/etc/kubernetes/pki/etcd/server.key

2、peer 集群中节点互相通信使用的客户端证书

/etc/kubernetes/pki/etcd/peer.crt

/etc/kubernetes/pki/etcd/peer.key

注: Peer：对同一个etcd集群中另外一个Member的称呼

3、pod 中定义 Liveness 探针使用的客户端证书

kubeadm 部署的 Kubernetes 集群是以 pod 的方式运行 etcd 服务的, 在该 pod 的定义中, 配置了 Liveness 探活探针

/etc/kubernetes/pki/etcd/healthcheck-client.crt

/etc/kubernetes/pki/etcd/healthcheck-client.key

当你 describe etcd 的 pod 时, 会看到如下一行配置:

Liveness: exec [/bin/sh -ec ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key get foo] delay=15s timeout=15s period=10s #success=1 #failure=8

4、配置在 kube-apiserver 中用来与 etcd server 做双向认证的客户端证书

/etc/kubernetes/pki/apiserver-etcd-client.crt

/etc/kubernetes/pki/apiserver-etcd-client.key

### 6、扩容k8s集群-添加第一个工作节点

在xianchaomaster1上查看加入节点的命令：

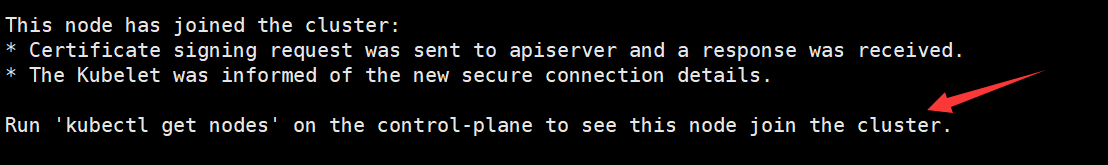
[root@xianchaomaster1 ~]# kubeadm token create --print-join-command

显示如下：

kubeadm join 192.168.40.180:6443 --token vulvta.9ns7da3saibv4pg1 --discovery-token-ca-cert-hash sha256:72a0896e27521244850b8f1c3b600087292c2d10f2565adb56381f1f4ba7057a

把xianchaonode1加入k8s集群：

[root@xianchaonode1~]# kubeadm join 192.168.40.180:6443 --token vulvta.9ns7da3saibv4pg1 --discovery-token-ca-cert-hash sha256:72a0896e27521244850b8f1c3b600087292c2d10f2565adb56381f1f4ba7057a --ignore-preflight-errors=SystemVerification



#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

xianchaonode1 NotReady <none> 59s v1.20.6

### 7、扩容k8s集群-添加第二个工作节点

在xianchaomaster1上查看加入节点的命令：

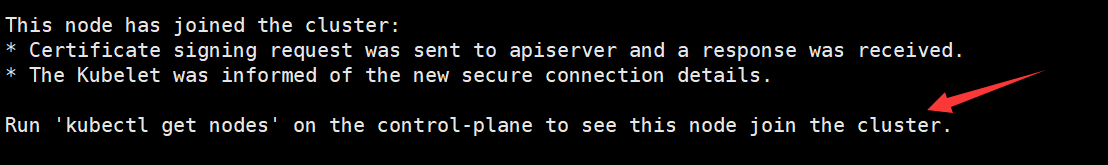
[root@xianchaomaster1 ~]# kubeadm token create --print-join-command

显示如下：

kubeadm join 192.168.40.180:6443 --token i3u8gu.n1d6fy40jdxgqjpu --discovery-token-ca-cert-hash sha256:72a0896e27521244850b8f1c3b600087292c2d10f2565adb56381f1f4ba7057a

把xianchaonode2加入k8s集群：

[root@xianchaonode2~]# kubeadm join 192.168.40.180:6443 --token i3u8gu.n1d6fy40jdxgqjpu --discovery-token-ca-cert-hash sha256:72a0896e27521244850b8f1c3b600087292c2d10f2565adb56381f1f4ba7057a --ignore-preflight-errors=SystemVerification



#看到上面说明xianchaonode2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

xianchaonode1 NotReady <none> 59s v1.20.6

xianchaonode2 NotReady <none> 59s v1.20.6

#可以看到xianchaonode1、xianchaonode2的ROLES角色为空，<none>就表示这个节点是工作节点。

#可以把xianchaonode1和xianchaonode2的ROLES变成work，按照如下方法：

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

[root@xianchaomaster1 ~]# kubectl label node xianchaonode2 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

### 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

再次查看集群状态。

[root@xianchaomaster1 ~]# kubectl get nodes

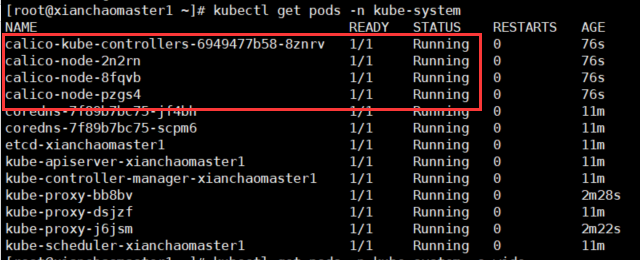
NAME STATUS ROLES AGE VERSION

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

xianchaonode1 Ready <none> 5m46s v1.20.6

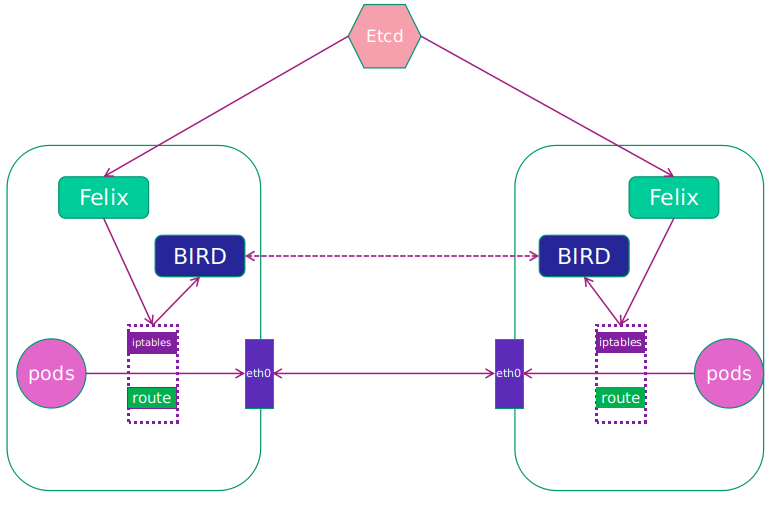
xianchaonode2 Ready <none> 5m46s v1.20.6

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



#STATUS状态是Ready，说明k8s集群正常运行了

#### 8.1 Calico架构图



Calico网络模型主要工作组件：

1.Felix：运行在每一台 Host 的 agent 进程，主要负责网络接口管理和监听、路由、ARP 管理、ACL 管理和同步、状态上报等。保证跨主机容器网络互通。

2.etcd：分布式键值存储，相当于k8s集群中的数据库，存储着Calico网络模型中IP地址等相关信息。主要负责网络元数据一致性，确保 Calico 网络状态的准确性；

3.BGP Client（BIRD）：Calico 为每一台 Host 部署一个 BGP Client，即每台host上部署一个BIRD。 主要负责把 Felix 写入 Kernel 的路由信息分发到当前 Calico 网络，确保 Workload 间的通信的有效性；

4.BGP Route Reflector：在大型网络规模中，如果仅仅使用 BGP client 形成 mesh 全网互联的方案就会导致规模限制，因为所有节点之间俩俩互联，需要 N^2 个连接，为了解决这个规模问题，可以采用 BGP 的 Router Reflector 的方法，通过一个或者多个 BGP Route Reflector 来完成集中式的路由分发。

#### 8.2 calico网络插件配置文件说明

1、Daemonset配置

……

containers:

        # Runs calico-node container on each Kubernetes node. This

        # container programs network policy and routes on each

        # host.

        - name: calico-node

          image: docker.io/calico/node:v3.18.0

……

          env:

            # Use Kubernetes API as the backing datastore.

            - name: DATASTORE\_TYPE

              value: "kubernetes"

            # Cluster type to identify the deployment type

            - name: CLUSTER\_TYPE

              value: "k8s,bgp"

            # Auto-detect the BGP IP address.

            - name: IP

              value: "autodetect"

#pod网段

- name: CALICO\_IPV4POOL\_CIDR

value: "10.244.0.0/16"

            # Enable IPIP

            - name: CALICO\_IPV4POOL\_IPIP

              value: "Always"

calico-node服务的主要参数如下:

CALICO\_IPV4POOL\_IPIP：是否启用IPIP模式。启用IPIP模式时，Calico将在Node上创建一个名为tunl0的虚拟隧道。IP Pool可以使用两种模式：BGP或IPIP。使用IPIP模式时，设置CALICO\_IPV4POOL\_IPIP="Always"，不使用IPIP模式时，设置CALICO\_IPV4POOL\_IPIP="Off"，此时将使用BGP模式。

IP\_AUTODETECTION\_METHOD：获取Node IP地址的方式，默认使用第1个网络接口的IP地址，对于安装了多块网卡的Node，可以使用正则表达式选择正确的网卡，例如"interface=eth.\*"表示选择名称以eth开头的网卡的IP地址。

- name: IP\_AUTODETECTION\_METHOD

  value: "interface=ens33"

扩展：calico的IPIP模式和BGP模式对比分析

1）IPIP

把一个IP数据包又套在一个IP包里，即把IP层封装到IP层的一个 tunnel，它的作用其实基本上就相当于一个基于IP层的网桥，一般来说，普通的网桥是基于mac层的，根本不需要IP，而这个ipip则是通过两端的路由做一个tunnel，把两个本来不通的网络通过点对点连接起来；

calico以ipip模式部署完毕后，node上会有一个tunl0的网卡设备，这是ipip做隧道封装用的,也是一种overlay模式的网络。当我们把节点下线，calico容器都停止后，这个设备依然还在，执行 rmmodipip命令可以将它删除。

2）BGP

BGP模式直接使用物理机作为虚拟路由路（vRouter），不再创建额外的tunnel

边界网关协议（BorderGateway Protocol, BGP）是互联网上一个核心的去中心化的自治路由协议。它通过维护IP路由表或‘前缀’表来实现自治系统（AS）之间的可达性，属于矢量路由协议。BGP不使用传统的内部网关协议（IGP）的指标，而是基于路径、网络策略或规则集来决定路由。因此，它更适合被称为矢量性协议，而不是路由协议，通俗的说就是将接入到机房的多条线路（如电信、联通、移动等）融合为一体，实现多线单IP；

BGP 机房的优点：服务器只需要设置一个IP地址，最佳访问路由是由网络上的骨干路由器根据路由跳数与其它技术指标来确定的，不会占用服务器的任何系统；

官方提供的calico.yaml模板里，默认打开了ip-ip功能，该功能会在node上创建一个设备tunl0，容器的网络数据会经过该设备被封装一个ip头再转发。这里，calico.yaml中通过修改calico-node的环境变量：CALICO\_IPV4POOL\_IPIP来实现ipip功能的开关：默认是Always，表示开启；Off表示关闭ipip。

- name: CLUSTER\_TYPE

value: "k8s,bgp"

# Auto-detect the BGP IP address.

- name: IP

value: "autodetect"

# Enable IPIP

- name: CALICO\_IPV4POOL\_IPIP

value: "Always"

总结：

calico BGP通信是基于TCP协议的，所以只要节点间三层互通即可完成，即三层互通的环境bird就能生成与邻居有关的路由。但是这些路由和flannel host-gateway模式一样，需要二层互通才能访问的通，因此如果在实际环境中配置了BGP模式生成了路由但是不同节点间pod访问不通，可能需要再确认下节点间是否二层互通。

为了解决节点间二层不通场景下的跨节点通信问题，calico也有自己的解决方案——IPIP模式

### 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、xianchaonode2节点，手动解压

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

[root@xianchaonode2 ~]# docker load -i busybox-1-28.tar.gz

[root@xianchaomaster1 ~]# [root@xianchaomaster1 ~]# kubectl run busybox --image busybox:1.28 --image-pull-policy=IfNotPresent --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coredns是否正常

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

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

Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: kubernetes.default.svc.cluster.local

Address 1: 10.96.0.1 kubernetes.default.svc.cluster.local

10.96.0.10 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。

#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

### 11、kubeadm初始化k8s证书过期解决方案

查看证书有效时间：

openssl x509 -in /etc/kubernetes/pki/ca.crt -noout -text |grep Not

openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text |grep Not

延长证书过期时间

1.把update-kubeadm-cert.sh文件上传到xianchaomaster1节点

2.在xianchaomaster1上执行如下：

1）给update-kubeadm-cert.sh证书授权可执行权限

[root@xianchaomaster1~]#chmod +x update-kubeadm-cert.sh

2）执行下面命令，修改证书过期时间，把时间延长到10年

[root@xianchaomaster1~]# ./update-kubeadm-cert.sh all

3）在xianchaomaster1节点查询Pod是否正常,能查询出数据说明证书签发完成

kubectl get pods -n kube-system

4）再次查看证书有效期，可以看到会延长到10年

openssl x509 -in /etc/kubernetes/pki/ca.crt -noout -text |grep Not

openssl x509 -in /etc/kubernetes/pki/apiserver.crt -noout -text |grep Not