K8S 部署要点、流程、经验

主机：

Master ：192.168.182.132 centos7.5-1804（apollo1）

Minion1：192.168.182.133 centos7.5-1804（apollo2）

Minion2：192.168.182.134 centos7.5-1804（apollo3）

1. 部署docker

Yum –y install docker

1. 部署harbor服务器
   1. 安装一些必要的系统工具

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

* 1. 添加docker镜像源

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

* 1. 安装 Docker-CE

yum -y install docker-ce

* 1. 开启Docker服务

systemctl start docker

问题：Failed to start docker.server.service: Unit not found.

* 1. 安装Docker-Compose

#curl -L https://github.com/docker/compose/releases/download/version/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

* 1. 下载harbor

<https://github.com/vmware/harbor/releases>

* 1. 在解压后的路径下，找到harbor.cfg进行修改

1. 安装k8s

3.1、创建 TLS 证书和秘钥

1)安装CFSSL工具

wget https://pkg.cfssl.org/R1.2/cfssl\_linux-amd64

chmod +x cfssl\_linux-amd64

mv cfssl\_linux-amd64 /usr/local/bin/cfssl

wget https://pkg.cfssl.org/R1.2/cfssljson\_linux-amd64

chmod +x cfssljson\_linux-amd64

mv cfssljson\_linux-amd64 /usr/local/bin/cfssljson

wget https://pkg.cfssl.org/R1.2/cfssl-certinfo\_linux-amd64

chmod +x cfssl-certinfo\_linux-amd64

mv cfssl-certinfo\_linux-amd64 /usr/local/bin/cfssl-certinfo

export PATH=/usr/local/bin:$PATH

2)创建CA

mkdir /root/ssl

cd /root/ssl

cfssl print-defaults config > config.json

cfssl print-defaults csr > csr.json

# 根据config.json文件的格式创建如下的ca-config.json文件

# 过期时间设置成了 87600h

cat > ca-config.json <<EOF

{

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": {

"kubernetes": {

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth"

],

"expiry": "87600h"

}

}

}

}

EOF

字段说明

ca-config.json：可以定义多个 profiles，分别指定不同的过期时间、使用场景等参数；后续在签名证书时使用某个 profile；

signing：表示该证书可用于签名其它证书；生成的 ca.pem 证书中 CA=TRUE；

server auth：表示client可以用该 CA 对server提供的证书进行验证；

client auth：表示server可以用该CA对client提供的证书进行验证；

[复制代码](javascript:void(0);)

3)创建CA证书签名请求

创建 ca-csr.json 文件：

cat >ca-csr.json << EOF

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "shenzhen",

"L": "shenzhen",

"O": "k8s",

"OU": "System"

}

]

}  
EOF

4）生成CA证书私钥

　　 cfssl gencert -initca ca-csr.json | cfssljson -bare ca

5)创建kubernetes证书

　　hosts字段填写上所有你要用到的节点ip，创建 kubernetes 证书签名请求文件 kubernetes-csr.json：

{

"CN": "kubernetes",

"hosts": [

"127.0.0.1",

"192.168.182.132",

"192.168.182.133",

"192.168.182.134",  
　　　 "kubernetes",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "shenzhen",

"L": "shenzhen",

"O": "k8s",

"OU": "System"

}

]

}

6)生成kubernetes证书和私钥

cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kubernetes-csr.json | cfssljson -bare kubernetes

问题：invalid character 'ã' looking for beginning of value

Failed to parse input: unexpected end of JSON input

以上2步可以合并成一个步骤，少生成1个kubernetes-csr.json文件，直接在命令行中输入参数代理了文件输入。

echo '{"CN":"kubernetes","hosts":[""],"key":{"algo":"rsa","size":2048}}' | cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes -hostname="127.0.0.1,192.168.182.128, 192.168.182.130, 192.168.182.131,kubernetes,kubernetes.default" - | cfssljson -bare kubernetes

7)创建admin证书

vim  admin-csr.json

[复制代码](javascript:void(0);)

{

"CN": "admin",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "shenzhen",

"L": "shenzhen",

"O": "system:masters",

"OU": "System"

}

]

}

8)生成admin证书和私钥

cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes admin-csr.json | cfssljson -bare admin

截止目前15个文件，admin开头的4个

　　9）创建kuber-proxy证书

vim kube-proxy-csr.json

{

"CN": "system:kube-proxy",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "shenzhen",

"L": "shenzhen",

"O": "k8s",

"OU": "System"

}

]

}

[复制代码](javascript:void(0);)

　　10)生成相关证书和私钥

$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kube-proxy-csr.json | cfssljson -bare kube-proxy

$ ls kube-proxy\*

kube-proxy.csr kube-proxy-csr.json kube-proxy-key.pem kube-proxy.pem

截止到目前19个文件，kube-proxy开头的4个

　　11)校验证书，举例校验kubernetes.pem证书，2个方法都可以，看输出内容可json定义是否一致。

openssl x509 -noout -text -in kubernetes.pem

cfssl-certinfo -cert kubernetes.pem

3.2、创建kubeconfig 文件

前提：安装kubelet工具，参考：https://jimmysong.io/kubernetes-handbook/practice/kubectl-installation.html

1、创建TLS bootstrapping Token，即token.csv文件

export BOOTSTRAP\_TOKEN=$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ')

cat > token.csv <<EOF

${BOOTSTRAP\_TOKEN},kubelet-bootstrap,10001,"system:kubelet-bootstrap"

EOF

注意事项：

1. 更新 token.csv 文件，分发到所有机器 (master 和 node）的 /etc/kubernetes/ 目录下，分发到node节点上非必需；
2. 重新生成 bootstrap.kubeconfig 文件，分发到所有 node 机器的 /etc/kubernetes/ 目录下；
3. 重启 kube-apiserver 和 kubelet 进程；
4. 重新 approve kubelet 的 csr 请求；

 拷贝文件到配置文件夹

cp token.csv /etc/kubernetes

2、创建kubelet（node节点）要用的bootstrap.kubeconfig文件

**注意：以下都是一条条执行的命令，不是复制这些代码到文件里。。。注意。。有网友已经错误的这样做了。**

[复制代码](javascript:void(0);)

cd /etc/kubernetes

export KUBE\_APISERVER="https://192.168.182.132:6443"

# 设置集群参数

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=bootstrap.kubeconfig

# 设置客户端认证参数

kubectl config set-credentials kubelet-bootstrap \

--token=${BOOTSTRAP\_TOKEN} \

--kubeconfig=bootstrap.kubeconfig

# 设置上下文参数

kubectl config set-context default \

--cluster=kubernetes \

--user=kubelet-bootstrap \

--kubeconfig=bootstrap.kubeconfig

# 设置默认上下文

kubectl config use-context default --kubeconfig=bootstrap.kubeconfig

[复制代码](javascript:void(0);)

3、创建节点要用的kube-proxy.kubeconfig文件

[复制代码](javascript:void(0);)

export KUBE\_APISERVER="https:// 192.168.182.132:6443"

# 设置集群参数

kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=kube-proxy.kubeconfig

# 设置客户端认证参数

kubectl config set-credentials kube-proxy \

--client-certificate=/etc/kubernetes/ssl/kube-proxy.pem \

--client-key=/etc/kubernetes/ssl/kube-proxy-key.pem \

--embed-certs=true \

--kubeconfig=kube-proxy.kubeconfig

# 设置上下文参数

kubectl config set-context default \

--cluster=kubernetes \

--user=kube-proxy \

--kubeconfig=kube-proxy.kubeconfig

# 设置默认上下文

kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig

[复制代码](javascript:void(0);)

4、分发文件

将两个 kubeconfig 文件分发到所有 Node 机器的 /etc/kubernetes/ 目录，本机就是生产的那些

scp bootstrap.kubeconfig kube-proxy.kubeconfig root@192.168.182.130:/etc/kubernetes/

scp bootstrap.kubeconfig kube-proxy.kubeconfig root@192.168.182.131:/etc/kubernetes/

3.3、创建高可用etcd集群

1、TLS认证文件分发：etcd集群认证用，除了本机有，分发到其他node节点

scp ca.pem kubernetes-key.pem kubernetes.pem root@192.168.182.133:/etc/kubernetes/ssl

scp ca.pem kubernetes-key.pem kubernetes.pem root@192.168.182.134:/etc/kubernetes/ssl

2、安装Etcd，这里使用的yum安装方式

#查询版本是否合适，我这里是3.2.9版本

yum info etcd

#安装

yum install etcd

若使用yum安装，默认etcd命令将在/usr/bin目录下，注意修改下面的etcd.service文件中的启动命令地址为/usr/bin/etcd

3、创建etcd的systemd unit文件（既centos7下的服务定义文件）

文件位置：/usr/lib/systemd/system/etcd.service ，默认该文件存在，删除重建即可。

[复制代码](javascript:void(0);)

[Unit]

Description=Etcd Server

After=network.target

After=network-online.target

Wants=network-online.target

Documentation=https://github.com/coreos

[Service]

Type=notify

WorkingDirectory=/var/lib/etcd/

EnvironmentFile=-/etc/etcd/etcd.conf

ExecStart=/usr/bin/etcd \

--name apollo1 \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

--peer-cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--peer-key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

--trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--peer-trusted-ca-file=/etc/kubernetes/ssl/ca.pem \

--initial-advertise-peer-urls https://192.168.182.132:2380 \

--listen-peer-urls https://192.168.182.132:2380 \

--listen-client-urls https://192.168.182.128:2379,http://127.0.0.1:2379 \

--advertise-client-urls https://192.168.182.132:2379 \

--initial-cluster-token etcd-cluster-0 \

--initial-cluster apollo1=https://192.168.182.132:2380,apollo2=https://192.168.182.133:2380,apollo3=https://192.168.182.34:2380 \

--initial-cluster-state new \

--data-dir=/var/lib/etcd

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

配置注意事项：所有节点都必须配置此文件，并且注意下面4个注意事项。

　　1、IP地址除了initial-cluster 配置项是配置集群内3个地址的ip外，其他IP均为本机的IP。

　　2、配置下--name必须与--initial-cluster配置项里的的对应，比如图上配置就是我master的配置，name是etcd-host0，下面的IP对应的名称也是这个。

　　3、通过不同方式安装的软件Execstart配置项下的程序启动命令路径注意修改

　　4、WorkingDirectory工作目录需要实现创建，否则启动会报错。

4、创建etcd环境变量文件

文件位置：/etc/etcd/etcd.conf，yum安装完之后该文件会存在，删除重建即可。

[复制代码](javascript:void(0);)

# [member]

ETCD\_NAME=infra1

ETCD\_DATA\_DIR="/var/lib/etcd"

ETCD\_LISTEN\_PEER\_URLS="https:// 192.168.182.132:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https:// 192.168.182.132:2379"

#[cluster]

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https:// 192.168.182.132:2380"

ETCD\_INITIAL\_CLUSTER\_TOKEN="etcd-cluster"

ETCD\_ADVERTISE\_CLIENT\_URLS="https:// 192.168.182.132:2379"

[复制代码](javascript:void(0);)

注意事项：

　　1、再次提醒ETCD\_DATA\_DIR一定要存在，其他的IP地址替换为本机的即可，maser及node节点都需要配置

　　2、ETCD\_NAME按照etcd系统服务里面的配置一一对应，分别是infra1,infra2,infra3

5、设置开机启动及启动etcd

systemctl daemon-reload

systemctl enable etcd

systemctl start etcd

systemctl status etcd

6、检测集群工作情况

在任意一个节点，master或者node都可以，执行以下命令

etcdctl \

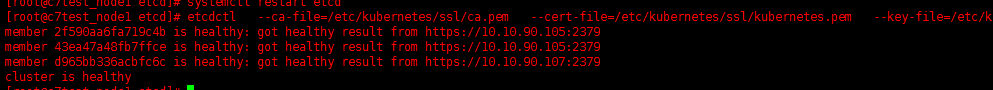
--ca-file=/etc/kubernetes/ssl/ca.pem \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

cluster-health

如果输出类似如下如的情况，代表成功：



结果最后一行为 cluster is healthy 时表示集群服务正常

目前在学习K8S, etcd 集群搭建，启动时候报错：

master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)   
master-16 etcd[25461]: request cluster ID mismatch (got bdd7c7c32bde1208 want 698851d67fea3c17)

检查启动service 和配置etcd.conf ,问题应该没有出在配置上。

解决办法：删除了etcd集群所有节点中的--data\_dir的内容  
分析： 因为集群搭建过程，单独启动过单一etcd,做为测试验证，集群内第一次启动其他etcd服务时候，是通过发现服务引导的，所以需要删除旧的成员信息

注意事项：

　　1、建议所有节点都运行一次进行检测，我在maser检点进行检测的时候发现master本身的几点都链接上，报unhealthy错误，查看报错后发现是使用了代理上网设置

当初为了在线软件设置的上网代理，需要关闭代理，取消配置参数，重启了服务器才检测成功。

　　2、防火墙务必关闭，否则会有链接不到etcd的问题。。

　　3、以后使用etcd查询数据都需要使用认证文件，查询格式：

etcdctl \

--ca-file=/etc/kubernetes/ssl/ca.pem \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

cluster-health

否则出错，例如：

[复制代码](javascript:void(0);)

[root@kube\_master ssl]# etcdctl cluster-health

failed to check the health of member 2f590aa6fa719c4b on https://10.10.90.105:2379: Get https://10.10.90.105:2379/health: x509: certificate signed by unknown authority

member 2f590aa6fa719c4b is unreachable: [https://10.10.90.105:2379] are all unreachable

failed to check the health of member 43ea47a48fb7ffce on https://10.10.90.106:2379: Get https://10.10.90.106:2379/health: x509: certificate signed by unknown authority

member 43ea47a48fb7ffce is unreachable: [https://10.10.90.106:2379] are all unreachable

failed to check the health of member d965bb336acbfc6c on https://10.10.90.107:2379: Get https://10.10.90.107:2379/health: x509: certificate signed by unknown authority

member d965bb336acbfc6c is unreachable: [https://10.10.90.107:2379] are all unreachable

cluster is unhealthy

3.4、部署master节点

作为k8s的核心，master节点主要包含三个组件，分别是：

三个组件：  
kube-apiserver

kube-scheduler

kube-controller-manager

1、创建TLS证书

这些证书我们在第一篇文章中已经创建，共8个，这里核对一下数量是否正确，至于证书是否正确参考第一篇文章的注释实现。位置：105虚拟机master节点

# ls /etc/kubernetes/ssl

admin-key.pem admin.pem ca-key.pem ca.pem kube-proxy-key.pem kube-proxy.pem kubernetes-key.pem kubernetes.pem

2、获取k8s server端文件并安装

我们采用在github上下载的方式获得tar包，解压或者二进制程序。说明:这里使用的是最新的1.9版本的。

wget https://dl.k8s.io/v1.10.0/kubernetes-server-linux-amd64.tar.gz

tar -xzvf kubernetes-server-linux-amd64.tar.gz

cd kubernetes

tar -xzvf kubernetes-src.tar.gz

拷贝二进制文件到/usr/bin下，可能会提示overwrite，因为前面安装的kubectl会安装一部分，直接覆盖就好，下面的语句使用了-r去覆盖，不加-r会提示，并且这个server包含server和client文件，不用单独下载client包

cp -r server/bin/{kube-apiserver,kube-controller-manager,kube-scheduler,kubectl,kube-proxy,kubelet} /usr/local/bin/

至此一些必要的二进制命令文件获取完毕，下一部制作3个组件的服务程序和配置文件

3、制作apiserver的服务文件

/usr/lib/systemd/system/kube-apiserver.service内容：

[复制代码](javascript:void(0);)

[Unit]

Description=Kubernetes API Service

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=network.target

After=etcd.service

[Service]

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/apiserver

ExecStart=/usr/local/bin/kube-apiserver \

$KUBE\_LOGTOSTDERR \

$KUBE\_LOG\_LEVEL \

$KUBE\_ETCD\_SERVERS \

$KUBE\_API\_ADDRESS \

$KUBE\_API\_PORT \

$KUBELET\_PORT \

$KUBE\_ALLOW\_PRIV \

$KUBE\_SERVICE\_ADDRESSES \

$KUBE\_ADMISSION\_CONTROL \

$KUBE\_API\_ARGS

Restart=on-failure

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

制作/etc/kubernetes/config通用文件，的内容为：

[复制代码](javascript:void(0);)

###

# kubernetes system config

#

# The following values are used to configure various aspects of all

# kubernetes services, including

#

# kube-apiserver.service

# kube-controller-manager.service

# kube-scheduler.service

# kubelet.service

# kube-proxy.service

# logging to stderr means we get it in the systemd journal

KUBE\_LOGTOSTDERR="--logtostderr=true"

# journal message level, 0 is debug

KUBE\_LOG\_LEVEL="--v=0"

# Should this cluster be allowed to run privileged docker containers

KUBE\_ALLOW\_PRIV="--allow-privileged=true"

# How the controller-manager, scheduler, and proxy find the apiserver

#KUBE\_MASTER="--master=http://sz-pg-oam-docker-test-001.tendcloud.com:8080"

KUBE\_MASTER="--master=http://192.168.182.132:8080"

[复制代码](javascript:void(0);)

kube-apiserver的配置文件/etc/kubernetes/apiserver内容为：

[复制代码](javascript:void(0);)

###

# kubernetes system config

#

# The following values are used to configure the kube-apiserver

#

# The address on the local server to listen to.

KUBE\_API\_ADDRESS="--advertise-address=192.168.182.128 --bind-address=192.168.182.128 --insecure-bind-address=127.0.0.1"

# The port on the local server to listen on.

#KUBE\_API\_PORT="--port=8080"

# Port minions listen on

# KUBELET\_PORT="--kubelet-port=10250"

# Comma separated list of nodes in the etcd cluster

KUBE\_ETCD\_SERVERS="--etcd-servers=https://192.168.182.128:2379,https://192.168.182.130:2379,https://192.168.182.131:2379"

# Address range to use for services

KUBE\_SERVICE\_ADDRESSES="--service-cluster-ip-range=192.254.0.0/16"

# default admission control policies

KUBE\_ADMISSION\_CONTROL="--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota,NodeRestriction"

# Add your own!

KUBE\_API\_ARGS="--authorization-mode=RBAC,Node --runtime-config=rbac.authorization.k8s.io/v1beta1 --kubelet-https=true --enable-bootstrap-token-auth --token-auth-file=/etc/kubernetes/token.csv --service-node-port-range=30000-32767 --tls-cert-file=/etc/kubernetes/ssl/kubernetes.pem --tls-private-key-file=/etc/kubernetes/ssl/kubernetes-key.pem --client-ca-file=/etc/kubernetes/ssl/ca.pem --service-account-key-file=/etc/kubernetes/ssl/ca-key.pem --etcd-cafile=/etc/kubernetes/ssl/ca.pem --etcd-certfile=/etc/kubernetes/ssl/kubernetes.pem --etcd-keyfile=/etc/kubernetes/ssl/kubernetes-key.pem --enable-swagger-ui=true --apiserver-count=3 --audit-log-maxage=30 --audit-log-maxbackup=3 --audit-log-maxsize=100 --audit-log-path=/var/lib/audit.log --event-ttl=1h"

[复制代码](javascript:void(0);)

设置开机启动并启动apiserver组件：

systemctl daemon-reload

systemctl enable kube-apiserver

systemctl start kube-apiserver

systemctl status kube-apiserver

ss -tanl  检查端口，6443和8080端口应该监听成功，代表apiserver安装成功。

4、配置和启动 kube-controller-manager

服务定义文件/usr/lib/systemd/system/kube-controller-manager.service内容为：

说明，某些文件可能已经存在，我们只要核对内容即可。

[复制代码](javascript:void(0);)

[Unit]

Description=Kubernetes Controller Manager

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

[Service]

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/controller-manager

ExecStart=/usr/local/bin/kube-controller-manager \

$KUBE\_LOGTOSTDERR \

$KUBE\_LOG\_LEVEL \

$KUBE\_MASTER \

$KUBE\_CONTROLLER\_MANAGER\_ARGS

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

相关配置文件配置文件/etc/kubernetes/controller-manager内容：

[复制代码](javascript:void(0);)

###

# The following values are used to configure the kubernetes controller-manager

# defaults from config and apiserver should be adequate

# Add your own!

KUBE\_CONTROLLER\_MANAGER\_ARGS="--address=127.0.0.1 --service-cluster-ip-range=10.254.0.0/16 --cluster-name=kubernetes --cluster-signing-cert-file=/etc/kubernetes/ssl/ca.pem --cluster-signing-key-file=/etc/kubernetes/ssl/ca-key.pem --service-account-private-key-file=/etc/kubernetes/ssl/ca-key.pem --root-ca-file=/etc/kubernetes/ssl/ca.pem --leader-elect=true"

[复制代码](javascript:void(0);)

设置开机启动并启动controller-manager

systemctl daemon-reload

systemctl enable kube-controller-manager

systemctl start kube-controller-manager

5、配置和启动 kube-scheduler

服务定义文件/usr/lib/systemd/system/kube-scheduler.service内容为：

[复制代码](javascript:void(0);)

[Unit]

Description=Kubernetes Scheduler Plugin

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

[Service]

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/scheduler

User=kube

ExecStart=/usr/local/bin/kube-scheduler \

$KUBE\_LOGTOSTDERR \

$KUBE\_LOG\_LEVEL \

$KUBE\_MASTER \

$KUBE\_SCHEDULER\_ARGS

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

 相关的配置文件/etc/kubernetes/scheduler内容为：

[复制代码](javascript:void(0);)

###

# kubernetes scheduler config

# default config should be adequate

# Add your own!

KUBE\_SCHEDULER\_ARGS="--leader-elect=true --address=127.0.0.1"

[复制代码](javascript:void(0);)

设置开机启动并启动：

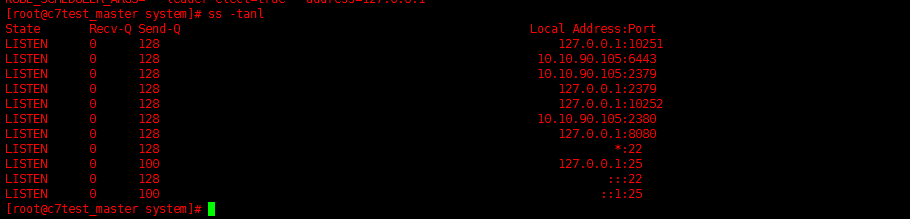
systemctl daemon-reload

systemctl enable kube-scheduler

systemctl start kube-scheduler

6、所有服务启动之后验证服务

首先ss -tanl查看端口：我的如下：



使用kubectl get命令获得组件信息：确保所有组件都是ok和healthy状态为true

[复制代码](javascript:void(0);)

[root@c7test\_master ~]# kubectl get componentstatuses

NAME STATUS MESSAGE ERROR

scheduler Healthy ok

controller-manager Healthy ok

etcd-2 Healthy {"health": "true"}

etcd-1 Healthy {"health": "true"}

etcd-0 Healthy {"health": "true"}

[复制代码](javascript:void(0);)

 报错：Unable to connect to the server: dial tcp 192.168.180.128:6443: getsockopt: no route to host 。

至此，master节点安装完成，在创建配置文件的过程中一定要信息，如果发现报错，使用journalctl -xe -u 服务名称  查看相关报错以及查看/var/log/message查看更详细的报错情况，具体情况具体解决即可。

注意事项：1、拷贝配置文件注意标点符号2、需要创建kube账户，否则scheduler启动不了

补充：

source <(kubectl completion bash)

执行以上命令可以执行kubectl命令的自动补全，因为kubectl太多子命令了。

3.5、安装flannel网络插件

1、安装flannel插件：

注意是2个node节点都需要安装，都需要修改service文件和配置文件。

yum install flannel -y

2、修改service文件/usr/lib/systemd/system/flanneld.service其内容为：

[复制代码](javascript:void(0);)

[Unit]

Description=Flanneld overlay address etcd agent

After=network.target

After=network-online.target

Wants=network-online.target

After=etcd.service

Before=docker.service

[Service]

Type=notify

EnvironmentFile=/etc/sysconfig/flanneld

EnvironmentFile=-/etc/sysconfig/docker-network

ExecStart=/usr/bin/flanneld-start \

-etcd-endpoints=${FLANNEL\_ETCD\_ENDPOINTS} \

-etcd-prefix=${FLANNEL\_ETCD\_PREFIX} \

$FLANNEL\_OPTIONS

ExecStartPost=/usr/libexec/flannel/mk-docker-opts.sh -k DOCKER\_NETWORK\_OPTIONS -d /run/flannel/docker

Restart=on-failure

[Install]

WantedBy=multi-user.target

RequiredBy=docker.service

[复制代码](javascript:void(0);)

提示:service文件中所有的变量参数都是读取相应的配置文件里面的，所有要保证前后对应。

修改其配置文件/etc/sysconfig/flanneld 内容如下：

[复制代码](javascript:void(0);)

# Flanneld configuration options

# etcd url location. Point this to the server where etcd runs

FLANNEL\_ETCD\_ENDPOINTS="https://192.168.182.132:2379,https://192.168.182.133:2379,https://192.168.182.134:2379"

# etcd config key. This is the configuration key that flannel queries

# For address range assignment

#FLANNEL\_ETCD\_PREFIX="/atomic.io/network"

FLANNEL\_ETCD\_PREFIX="/kube-centos/network"

# Any additional options that you want to pass

#FLANNEL\_OPTIONS=""

FLANNEL\_OPTIONS="-etcd-cafile=/etc/kubernetes/ssl/ca.pem -etcd-certfile=/etc/kubernetes/ssl/kubernetes.pem -etcd-keyfile=/etc/kubernetes/ssl/kubernetes-key.pem"

[复制代码](javascript:void(0);)

以上2步是2个node节点都需要做的。

3、在etcd中常见网络配置信息

[复制代码](javascript:void(0);)

etcdctl --endpoints=https://192.168.182.132:2379,https://192.168.182.133:2379,https://192.168.182.134:2379 \

--ca-file=/etc/kubernetes/ssl/ca.pem \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

mkdir /kube-centos/network

**#上下是2条命令，在任何节点上创建都行，因为etcd是集群的。**

etcdctl --endpoints=https://192.168.182.132:2379,https://192.168.182.133:2379,https://192.168.182.134:2379 \

--ca-file=/etc/kubernetes/ssl/ca.pem \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

mk /kube-centos/network/config '{"Network":"10.30.0.0/16","SubnetLen":24,"Backend":{"Type":"host-gw"}}'

[复制代码](javascript:void(0);)

4、启动flannel服务

systemctl daemon-reload

systemctl enable flanneld

systemctl start flanneld

systemctl status flanneld

5、核对相关配置信息：

[复制代码](javascript:void(0);)

#先声明个endpoint变量，后边好调用  
[root@c7test\_node1 ~]#ETCD\_ENDPOINTS='https://192.168.182.132:2379,https://192.168.182.133:2379,https://192.168.182.134:2379'

[root@c7test\_node1 ~]# etcdctl --endpoints=${ETCD\_ENDPOINTS} \

--ca-file=/etc/kubernetes/ssl/ca.pem \

--cert-file=/etc/kubernetes/ssl/kubernetes.pem \

--key-file=/etc/kubernetes/ssl/kubernetes-key.pem \

ls /kube-centos/network/subnets  
#输出

/kube-centos/network/subnets/172.30.87.0-24

/kube-centos/network/subnets/172.30.92.0-24

#说明，有几个node就有几个子网络，就有几条记录，我是2个node，分别安装了flannel插件

 [root@c7test\_node1 ~]# etcdctl --endpoints=${ETCD\_ENDPOINTS} \  
 --ca-file=/etc/kubernetes/ssl/ca.pem \  
 --cert-file=/etc/kubernetes/ssl/kubernetes.pem \  
 --key-file=/etc/kubernetes/ssl/kubernetes-key.pem \  
 get /kube-centos/network/config

 #输出  
 {"Network":"172.30.0.0/16","SubnetLen":24,"Backend":{"Type":"host-gw"}}

 #此处是查看主网络配置

 etcdctl --endpoints=${ETCD\_ENDPOINTS} --ca-file=/etc/kubernetes/ssl/ca.pem --cert-file=/etc/kubernetes/ssl/kubernetes.pem --key-file=/etc/kubernetes/ssl/kubernetes-key.pem get /kube-centos/network/subnets/172.30.92.0-24

 #输出

{"PublicIP":"10.10.90.106","BackendType":"vxlan","BackendData":{"VtepMAC":"26:af:ac:26:47:ad"}}

 etcdctl --endpoints=${ETCD\_ENDPOINTS} --ca-file=/etc/kubernetes/ssl/ca.pem --cert-file=/etc/kubernetes/ssl/kubernetes.pem --key-file=/etc/kubernetes/ssl/kubernetes-key.pem get /kube-centos/network/subnets/172.30.87.0-24

 #输出

 {"PublicIP":"10.10.90.107","BackendType":"vxlan","BackendData":{"VtepMAC":"82:6f:58:94:9f:59"}}

[复制代码](javascript:void(0);)

如有以上输出即代表flannel插件安装配置成功，下一篇文章开始安装node节点。

1、虚拟机时间同步不一致问题，导致etcd创建资源不成功

2、node节点无法自动创建kubelet.kubeconfig问题，这个是最严重的问题，原因是config文件没有拷贝到node的/etc/kubernetes文件夹内，因为kubelet启动调用

kubelet配置文件的时候也会同时调用这个文件，具体见kubelt的servier文件配置方法，这个文件是自动生成的。如果没有自动生产，检查所有配置参数和报错，特别是config和kublet文件。

3、有关config文件并不是你从客户端拷贝过来的时候就直接可以用了，需要里面修改master地址，因为apiserver的配置启动参数绑定的地址中安全的访问地址是10.10.90.105:6443,不安全是127.0.0.1:8080，这里可以简单理解为6443是安全端口，不过只监听在master的10.10.90.105的ip上，所以要修改node中config配置文件的master地址为 10.10.90.90.105:6443，而如果你master节点同时也是node节点的话，我测试了这个形式，那么你的config文件只能用127.0.0.1:8080访问，使用6443也是不行的，也就是说本地和其他机器访问apiserver的方式不同时的，否则log中会狂报错无法连接api，这里注意一下，如果node复用了master节点同事需要重启scheduler和control服务。

4、Failed at step CHDIR spawning /usr/local/bin/kubelet: No such file or directory 是没有创建 /var/lib/kubelt文件夹

5、配置过程中一定要关闭防火墙，selinux，防止虚拟机重启了这些服务业自动重启。

6、1.8后面的kubelet配置文件不需要--api-servers参数，请注释掉！！

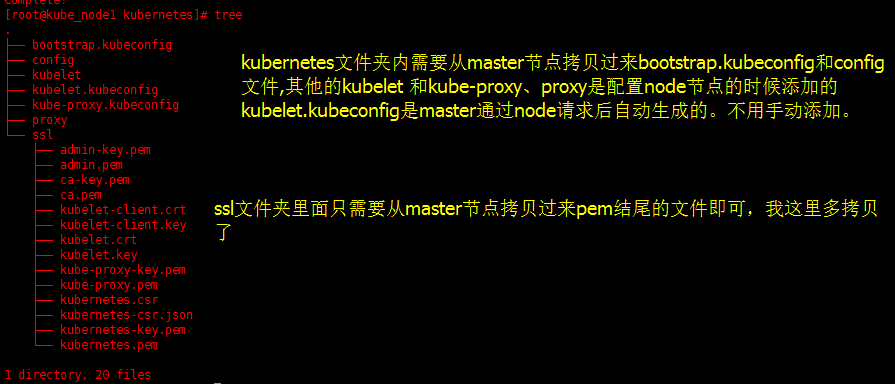
7、swap 分区请在/etc/fstab注释掉，并重启虚拟机和所有服务。

3.6、部署node节点

好，上面只是一些我还有印象的注意事项，其他的请多多查看log文件排错。

废话不多说，直接开始配置node节点了，再次感慨大家的支持和等待！。

1、检查2个node节点配置文件和ssl证书是否齐全,这一步很重要。



**注意ssl里面有几个kubelet开头的文件 ，是通过过自动生成的文件。**

2、配置docker的服务文件

因为需要docker联合flannel使用，所以需要修改docker的服务service文件

我们前面是flannel插件是通过yum方式安装的，修改方式如下：

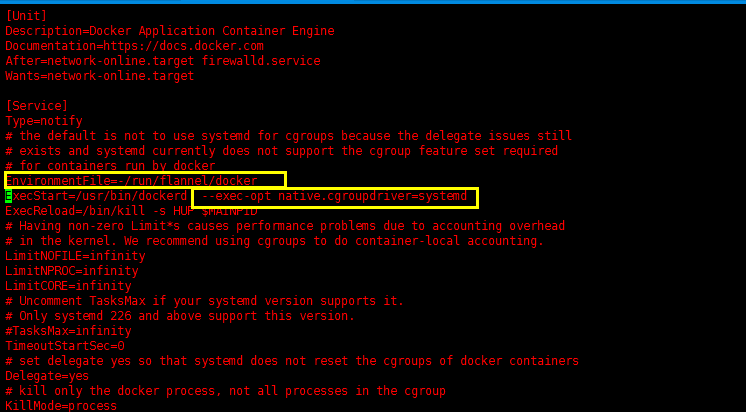
[复制代码](javascript:void(0);)

修改docker的配置文件/usr/lib/systemd/system/docker.service，增加一条环境变量配置：

EnvironmentFile=-/run/flannel/docker  
  
同时为start添加一个参数 --exec-opt native.cgroupdriver=systemd，这里的systemd和kubelet配置文件里面的--cgroup-drive相同即可，否则kubelet启动报错

[复制代码](javascript:void(0);)

如图：



 修改配置参数后，重启docker服务

systemctl restart docekr

 3、安装kubelet工具及配置

kubelet是node节点安装的工具，我们依然从我们前面下载的server包的bin里面可以找到，同时还需要kube-proxy问，可以同时上传的文件夹的/usr/local/bin文件

并且赋予可执行权限。

注意事项：swap分区务必注释掉并重启服务器。

配置前我们需要现在master节点上执行如下操作，创建认证角色：

cd /etc/kubernetes

kubectl create clusterrolebinding kubelet-bootstrap \

--clusterrole=system:node-bootstrapper \

--user=kubelet-bootstrap

created成功后，我们回到node节点操作：

我们已经获得了bin文件，开始配置相应的服务器文件

添加配置文件kubelt:

[复制代码](javascript:void(0);)

cd /etc/kubernetes

cat > kubelet << EOF

###

## kubernetes kubelet (minion) config

#

## The address for the info server to serve on (set to 0.0.0.0 or "" for all interfaces)

KUBELET\_ADDRESS="--address=192.168.182.133"

#

## The port for the info server to serve on

#KUBELET\_PORT="--port=10250"

#

## You may leave this blank to use the actual hostname

KUBELET\_HOSTNAME="--hostname-override=192.168.182.133"

#

## location of the api-server

## COMMENT THIS ON KUBERNETES 1.8+

#KUBELET\_API\_SERVER="--api-servers=http://172.20.0.113:8080"

#

## pod infrastructure container

KUBELET\_POD\_INFRA\_CONTAINER="--pod-infra-container-image=pause-amd64:3.1"

#

## Add your own!

KUBELET\_ARGS="--cgroup-driver=systemd --cluster-dns=10.254.0.2 --experimental-bootstrap-kubeconfig=/etc/kubernetes/bootstrap.kubeconfig --kubeconfig=/etc/kubernetes/kubelet.kubeconfig --cert-dir=/etc/kubernetes/ssl --cluster-domain=cluster.local --hairpin-mode promiscuous-bridge --serialize-image-pulls=false"

EOF

[复制代码](javascript:void(0);)

说明：里面的ip地址都为node节点的ip地址，其他节点相应就好就好，注意KUBELET\_API\_SERVER已经在1.8的时候不用了。注释掉。

KUBELET\_POD\_INFRA\_CONTAINER是指定pod运行的基础镜像，必须存在，我这里直接指定的是一个本地的镜像，镜像的或许地址为：

docker pull registry.cn-hangzhou.aliyuncs.com/google-containers/pause-amd64:3.0

下载到本地后tag一下，方便使用，当然你也可以添加其他的公共pod基础镜像，在线地址也行，注意不要被墙就好。

docker tag registry.cn-hangzhou.aliyuncs.com/google-containers/pause-amd64:3.0 pause-amd64:3.0

添加kubelt的服务文件/usr/lib/systemd/system/kubelet.service  
内容如下：

[复制代码](javascript:void(0);)

[Unit]

Description=Kubernetes Kubelet Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=docker.service

Requires=docker.service

[Service]

WorkingDirectory=/var/lib/kubelet

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/kubelet

ExecStart=/usr/local/bin/kubelet \

$KUBE\_LOGTOSTDERR \

$KUBE\_LOG\_LEVEL \

$KUBELET\_API\_SERVER \

$KUBELET\_ADDRESS \

$KUBELET\_PORT \

$KUBELET\_HOSTNAME \

$KUBE\_ALLOW\_PRIV \

$KUBELET\_POD\_INFRA\_CONTAINER \

$KUBELET\_ARGS

Restart=on-failure

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

添加工作目录：不添加启动报错

mkdir /var/lib/kubelet

启动kubelt：

systemctl daemon-reload

systemctl enable kubelet

systemctl start kubelet

systemctl status kubelet

4、接受node请求

启动后，如果政策会自动向master节点发送验证加入请求，我们在master节点操作：

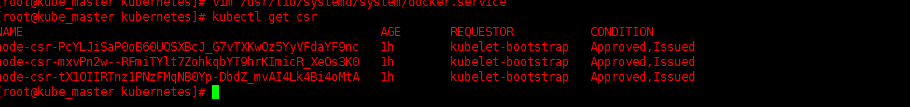
[复制代码](javascript:void(0);)

kubectl get csr

#此命令可以看到所有请求，所有为pending状态，则是需要批准的

kubectl certificate approve 节点name  
  
#此命令可以通过请求

[复制代码](javascript:void(0);)



我这是是已经approve过了，显示为approved和issued状态。就正常了

命令扩展：

kubectl delete csr 节点名称 #删除单个节点的请求

kubectl delete csr --all  #删除所有节点请求

kubectl  delete nodes  node名称  #删除加入的节点

kubectl  delete nodes --all   #删除所有节点

5、配置kube-proxy服务

现安装个工具conntrack，具体是干什么的还不是很清楚：

yum install -y conntrack-tools

创建 kube-proxy 的service配置文件，路径/usr/lib/systemd/system/kube-proxy.service，内容：

[复制代码](javascript:void(0);)

[Unit]

Description=Kubernetes Kube-Proxy Server

Documentation=https://github.com/GoogleCloudPlatform/kubernetes

After=network.target

[Service]

EnvironmentFile=-/etc/kubernetes/config

EnvironmentFile=-/etc/kubernetes/proxy

ExecStart=/usr/local/bin/kube-proxy \

$KUBE\_LOGTOSTDERR \

$KUBE\_LOG\_LEVEL \

$KUBE\_MASTER \

$KUBE\_PROXY\_ARGS

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

[复制代码](javascript:void(0);)

添加配置文件/etc/kubernetes/proxy:内容为：

[复制代码](javascript:void(0);)

proxy config

# default config should be adequate

# Add your own!

KUBE\_PROXY\_ARGS="--bind-address=10.10.90.106 --hostname-override=10.10.90.106 --kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig --cluster-cidr=10.254.0.0/16"

[复制代码](javascript:void(0);)

ip修改为本机ip即可。

注意事项：

--hostname-override 参数值必须与 kubelet 的值一致，否则 kube-proxy 启动后会找不到该 Node，从而不会创建任何 iptables 规则；  
kube-proxy 根据 --cluster-cidr 判断集群内部和外部流量，指定 --cluster-cidr 或 --masquerade-all 选项后 kube-proxy 才会对访问 Service IP 的请求做 SNAT；  
--kubeconfig 指定的配置文件嵌入了 kube-apiserver 的地址、用户名、证书、秘钥等请求和认证信息；  
预定义的 RoleBinding cluster-admin 将User system:kube-proxy 与 Role system:node-proxier 绑定，该 Role 授予了调用 kube-apiserver Proxy 相关 API 的权限；

启动proxy服务：

systemctl daemon-reload

systemctl enable kube-proxy

systemctl start kube-proxy

systemctl status kube-proxy

补充知识点：

[复制代码](javascript:void(0);)

**有关node节点名称修改：**  
  
在master上通过kubectl get node 获得的列表中，Name显示的名称是通过 客户端kubelet和proxy配置文件中hostname-override配置参数定义的，修改这2个参数为你想要的名称，并且删除**kubelet.kubeconfi**g(这个文件是master认证后客户端自动生成的，如果不删除会报node节点forbidden)文件，重新启动着2个服务，master端重新

kubectl certificate approve name名称 就可以看到新名称。

[复制代码](javascript:void(0);)

 修改配置文件，不删除kubelet.kubeconfig文件会报错误：

kubelet\_node\_status.go:106] Unable to register node "node2" with API server: nodes "node2" is forbidden: node "10.10.90.107" cannot modify node "node2"

6、验证测试

我们可以创建一个nginx部署验证集群是否正常：

[复制代码](javascript:void(0);)

#delete是清理不用的所有pods，service和deployment，非必须执行

kubectl delete pods --all

kubectl delete service --all

kubectl delete deployment --all

这里以下是测试一个nginx集群部署的

1、定义集群并启动

kubectl run nginx --replicas=3 --labels="run=load-balancer-example" --image=nginx --port=80

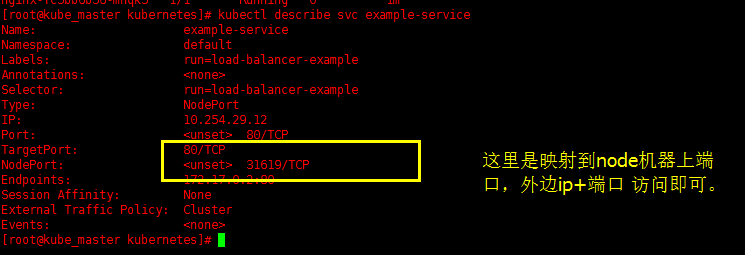
2、定义集群服务

kubectl expose deployment nginx --type=NodePort --name=example-service

3、查看服务信息

kubectl describe svc example-service  
4、查看pod状态,全为running既正常，否则使用kubectl describe pods {有问题的pod名称} 查看具体的报错。  
kubectl get pods

[复制代码](javascript:void(0);)



其他电脑访问node节点+ip是可以访问的，创建service的有其他类型可以选择。

3.7、安装kubedns插件

在/etc/kubernetes/yamlfile新增配置文件coredns.yaml,当然这个文件你随意放置，我只是归类而已，安装节点为master节点，node无需安装

1、预先下载corends的镜像

docker pull registry.docker-cn.com/coredns/coredns:0.9.10

2、新增配置文件coredns.yaml配置文件内容：

注意配置文件中红色指定的image在本地仓库一定要存在，按照第一步下载下来接口，且名称要对应上

[复制代码](javascript:void(0);)

apiVersion: v1

kind: ServiceAccount

metadata:

name: coredns

namespace: kube-system

labels:

kubernetes.io/cluster-service: "true"

addonmanager.kubernetes.io/mode: Reconcile

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

labels:

kubernetes.io/bootstrapping: rbac-defaults

addonmanager.kubernetes.io/mode: Reconcile

name: system:coredns

rules:

- apiGroups:

- ""

resources:

- endpoints

- services

- pods

- namespaces

verbs:

- list

- watch

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

annotations:

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

labels:

kubernetes.io/bootstrapping: rbac-defaults

addonmanager.kubernetes.io/mode: EnsureExists

name: system:coredns

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: system:coredns

subjects:

- kind: ServiceAccount

name: coredns

namespace: kube-system

---

apiVersion: v1

kind: ConfigMap

metadata:

name: coredns

namespace: kube-system

labels:

addonmanager.kubernetes.io/mode: EnsureExists

data:

Corefile: |

.:53 {

errors

log stdout

health

kubernetes cluster.local 10.254.0.0/16

prometheus

proxy . /etc/resolv.conf

cache 30

}

---

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: coredns

namespace: kube-system

labels:

k8s-app: coredns

kubernetes.io/cluster-service: "true"

addonmanager.kubernetes.io/mode: Reconcile

kubernetes.io/name: "CoreDNS"

spec:

replicas: 1

selector:

matchLabels:

k8s-app: coredns

template:

metadata:

labels:

k8s-app: coredns

spec:

serviceAccountName: coredns

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

- key: "CriticalAddonsOnly"

operator: "Exists"

containers:

- name: coredns

image: **registry.docker-cn.com/coredns/coredns:0.9.10**

imagePullPolicy: IfNotPresent

resources:

limits:

memory: 170Mi

requests:

cpu: 100m

memory: 70Mi

args: [ "-conf", "/etc/coredns/Corefile" ]

volumeMounts:

- name: config-volume

mountPath: /etc/coredns

ports:

- containerPort: 53

name: dns

protocol: UDP

- containerPort: 53

name: dns-tcp

protocol: TCP

- containerPort: 9153

name: metrics

protocol: TCP

livenessProbe:

httpGet:

path: /health

port: 8080

scheme: HTTP

initialDelaySeconds: 60

timeoutSeconds: 5

successThreshold: 1

failureThreshold: 5

dnsPolicy: Default

volumes:

- name: config-volume

configMap:

name: coredns

items:

- key: Corefile

path: Corefile

---

apiVersion: v1

kind: Service

metadata:

name: coredns

namespace: kube-system

labels:

k8s-app: coredns

kubernetes.io/cluster-service: "true"

addonmanager.kubernetes.io/mode: Reconcile

kubernetes.io/name: "CoreDNS"

spec:

selector:

k8s-app: coredns

clusterIP: 10.254.0.2

ports:

- name: dns

port: 53

protocol: UDP

- name: dns-tcp

port: 53

protocol: TCP

- name: metrics

port: 9153

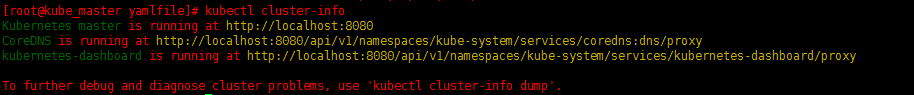
protocol: TCP

[复制代码](javascript:void(0);)

3、部署coredns

kubectl create -f coredns.yaml

执行完成后开始添加服务及启动，可以通过kubectl  cluster-info查看



以及

kubectl get services coredns -n kube-system



3.8、安装dashboard（**后面教程已经更换为coredns**）插件

3.9、安装heapster插件

3.10、安装EFK插件