目录

**[CentOS-7.6 安装k8s](#_Toc18291_WPSOffice_Level1)** **[3](#_Toc18291_WPSOffice_Level1)**

**[第一部分 环境初始化](#_Toc21065_WPSOffice_Level1)** **[3](#_Toc21065_WPSOffice_Level1)**

[1、 环境准备：](#_Toc21065_WPSOffice_Level2) [3](#_Toc21065_WPSOffice_Level2)

[2、 设置SSH免密码登录](#_Toc17319_WPSOffice_Level2) [3](#_Toc17319_WPSOffice_Level2)

[3、 配置内核参数](#_Toc7965_WPSOffice_Level2) [3](#_Toc7965_WPSOffice_Level2)

[4、 安装Docker（各个节点都要）](#_Toc15869_WPSOffice_Level2) [3](#_Toc15869_WPSOffice_Level2)

[5、 准备部署目录（各个节点都要）](#_Toc19909_WPSOffice_Level2) [4](#_Toc19909_WPSOffice_Level2)

**[第二部分 证书制作](#_Toc17319_WPSOffice_Level1)** **[4](#_Toc17319_WPSOffice_Level1)**

[1、 下载并安装CFSSL](#_Toc6306_WPSOffice_Level2) [4](#_Toc6306_WPSOffice_Level2)

[2、 创建CA证书](#_Toc19228_WPSOffice_Level2) [4](#_Toc19228_WPSOffice_Level2)

[3、 创建ETCD证书](#_Toc13672_WPSOffice_Level2) [5](#_Toc13672_WPSOffice_Level2)

[4、 创建kubernetes证书](#_Toc135_WPSOffice_Level2) [6](#_Toc135_WPSOffice_Level2)

[5、 创建admin证书](#_Toc5012_WPSOffice_Level2) [7](#_Toc5012_WPSOffice_Level2)

[6、 创建kube-proxy证书](#_Toc1503_WPSOffice_Level2) [7](#_Toc1503_WPSOffice_Level2)

[7、 创建Flannel证书](#_Toc21146_WPSOffice_Level2) [8](#_Toc21146_WPSOffice_Level2)

**[第三部分 ETCD集群部署](#_Toc7965_WPSOffice_Level1)** **[8](#_Toc7965_WPSOffice_Level1)**

[1、 准备ETCD软件](#_Toc20417_WPSOffice_Level2) [8](#_Toc20417_WPSOffice_Level2)

[2、 配置ETCD参数](#_Toc951_WPSOffice_Level2) [8](#_Toc951_WPSOffice_Level2)

[3、 创建ETCD数据目录](#_Toc9539_WPSOffice_Level2) [9](#_Toc9539_WPSOffice_Level2)

[4、 创建ETCD系统服务](#_Toc2379_WPSOffice_Level2) [9](#_Toc2379_WPSOffice_Level2)

[5、 文件分发到两个worker节点](#_Toc16473_WPSOffice_Level2) [9](#_Toc16473_WPSOffice_Level2)

[6、 修改k8s-worker1的etcd.conf文件](#_Toc31274_WPSOffice_Level2) [10](#_Toc31274_WPSOffice_Level2)

[7、 修改k8s-worker2的etcd.conf文件](#_Toc22138_WPSOffice_Level2) [10](#_Toc22138_WPSOffice_Level2)

[8、 加载并启动系统服务](#_Toc28723_WPSOffice_Level2) [10](#_Toc28723_WPSOffice_Level2)

[9、 验证集群](#_Toc29500_WPSOffice_Level2) [10](#_Toc29500_WPSOffice_Level2)

**[第四部分 Master节点部署](#_Toc15869_WPSOffice_Level1)** **[10](#_Toc15869_WPSOffice_Level1)**

[第一节 准备kubernetes软件包](#_Toc9460_WPSOffice_Level2) [10](#_Toc9460_WPSOffice_Level2)

[第二节 部署Kubernetes API Server](#_Toc7426_WPSOffice_Level2) [11](#_Toc7426_WPSOffice_Level2)

[第三节 部署Controller Manager服务](#_Toc5159_WPSOffice_Level2) [13](#_Toc5159_WPSOffice_Level2)

[第四节 部署Kubernetes Scheduler](#_Toc15349_WPSOffice_Level2) [13](#_Toc15349_WPSOffice_Level2)

[第五节 部署kubectl 命令行工具](#_Toc23366_WPSOffice_Level2) [14](#_Toc23366_WPSOffice_Level2)

[第六节 准备部署node节点](#_Toc15832_WPSOffice_Level2) [15](#_Toc15832_WPSOffice_Level2)

**[第五部分 Node节点部署](#_Toc19909_WPSOffice_Level1)** **[16](#_Toc19909_WPSOffice_Level1)**

[第一节 部署kubelet](#_Toc1670_WPSOffice_Level2) [16](#_Toc1670_WPSOffice_Level2)

[第二节 部署Kubernetes Proxy](#_Toc1406_WPSOffice_Level2) [18](#_Toc1406_WPSOffice_Level2)

**[第六部分 Flannel网络部署](#_Toc6306_WPSOffice_Level1)** **[20](#_Toc6306_WPSOffice_Level1)**

[1. 部署flannel软件包](#_Toc16128_WPSOffice_Level2) [20](#_Toc16128_WPSOffice_Level2)

[2. 配置flannel](#_Toc27369_WPSOffice_Level2) [20](#_Toc27369_WPSOffice_Level2)

[3. 创建Flannel系统服务](#_Toc10963_WPSOffice_Level2) [20](#_Toc10963_WPSOffice_Level2)

[4. 复制配置文件到其它节点](#_Toc31062_WPSOffice_Level2) [21](#_Toc31062_WPSOffice_Level2)

[5. 安装CNI插件](#_Toc6007_WPSOffice_Level2) [21](#_Toc6007_WPSOffice_Level2)

[6. 在master节点创建Etcd的key](#_Toc17097_WPSOffice_Level2) [21](#_Toc17097_WPSOffice_Level2)

[7. 启动flannel](#_Toc1955_WPSOffice_Level2) [21](#_Toc1955_WPSOffice_Level2)

[8. 查看网络配置](#_Toc7379_WPSOffice_Level2) [21](#_Toc7379_WPSOffice_Level2)

[9. 配置Docker使用Flannel](#_Toc18805_WPSOffice_Level2) [22](#_Toc18805_WPSOffice_Level2)

[10. 所有节点重启Docker服务](#_Toc30887_WPSOffice_Level2) [23](#_Toc30887_WPSOffice_Level2)

[11. 部署应用测试](#_Toc7866_WPSOffice_Level2) [23](#_Toc7866_WPSOffice_Level2)

**[第七部分 CoreDNS和Dashboard部署](#_Toc19228_WPSOffice_Level1)** **[24](#_Toc19228_WPSOffice_Level1)**

[第一节 部署CoreDNS](#_Toc22714_WPSOffice_Level2) [24](#_Toc22714_WPSOffice_Level2)

[第二节 部署Dashboard](#_Toc4226_WPSOffice_Level2) [25](#_Toc4226_WPSOffice_Level2)

**[第八部分 部署traefik Ingress](#_Toc13672_WPSOffice_Level1)** **[26](#_Toc13672_WPSOffice_Level1)**

[1. 理解Ingress](#_Toc14465_WPSOffice_Level2) [26](#_Toc14465_WPSOffice_Level2)

[2. 理解Ingress Controller](#_Toc4209_WPSOffice_Level2) [26](#_Toc4209_WPSOffice_Level2)

[3. 介绍traefik Ingress](#_Toc7949_WPSOffice_Level2) [26](#_Toc7949_WPSOffice_Level2)

[4. 部署Traefik Ingress](#_Toc23906_WPSOffice_Level2) [26](#_Toc23906_WPSOffice_Level2)

# CentOS-7.6 安装k8s

## 第一部分 环境初始化

1. 环境准备：

k8s-master1 10.3.8.101 etcd/docker/kube-apiserver/kube-controller-manager/kube-scheduler/flannel

k8s-worker1 10.3.8.104 etcd/docker/kube-proxy/kubelet/flannel

k8s-worker2 10.3.8.105 etcd/docker/kube-proxy/kubelet/flannel

以上系统最小化安装，修改好主机名，禁用selinux，关闭防火墙，并写好/etc/hosts：

# cat /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

10.3.8.101 k8s-master1

10.3.8.104 k8s-worker1

10.3.8.105 k8s-worker2

1. 设置SSH免密码登录

这里k8s-master1兼作部署节点：

[root@k8s-master1 ~]# ssh-keygen -t rsa

[root@k8s-master1 ~]# ssh-copy-id k8s-master1

[root@k8s-master1 ~]# ssh-copy-id k8s-worker1

[root@k8s-master1 ~]# ssh-copy-id k8s-worker2

1. 配置内核参数

[root@k8s-master1 ~]# cat > /etc/sysctl.d/kubernetes.conf << EOF

net.ipv4.ip\_forward = 1

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

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

EOF

[root@k8s-master1 ~]# sysctl -p /etc/sysctl.d/kubernetes.conf >& /dev/null

1. 安装Docker（各个节点都要）

# wget https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo -P /etc/yum.repos.d/

# yum install -y docker-ce

# mkdir -p /etc/docker

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

{

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

"registry-mirrors": ["https://hub-mirror.c.163.com", "https://docker.mirrors.ustc.edu.cn"],

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

"log-opts": {

"max-size": "100m",

"max-file": "3"

},

"storage-driver": "overlay2",

"max-concurrent-downloads": 20

}

EOF

# systemctl enable docker

# systemctl start docker

# docker info

......

Registry Mirrors:

https://hub-mirror.c.163.com/

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

1. 准备部署目录（各个节点都要）

# mkdir -p /opt/kubernetes/{cfg,bin/cni,ssl,log}

#vim /etc/profile

export PATH=/opt/kubernetes/bin/:$PATH

# source /etc/profile

## 证书制作

1. 下载并安装CFSSL

[root@k8s-master1 ~]# cd /opt/kubernetes/bin/

[root@k8s-master1 bin]# wget <https://pkg.cfssl.org/R1.2/cfssl_linux-amd64> -O cfssl

[root@k8s-master1 bin]# wget <https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64> -O cfssljson

[root@k8s-master1 bin]# wget <https://pkg.cfssl.org/R1.2/cfssl-certinfo_linux-amd64> -O cfssl-certinfo

[root@k8s-master1 bin]# chmod +x cfssl\*

1. 创建CA证书

[root@k8s-master1 ~]# mkdir /usr/local/src/ssl && cd /usr/local/src/ssl

[root@k8s-master1 ssl]# cfssl print-defaults config > ca-config.json

[root@k8s-master1 ssl]# cfssl print-defaults csr > ca-csr.json

修改CA配置文件

[root@k8s-master1 ssl]# vi ca-config.json

{

"signing": {

"default": {

"expiry": "87600h"

},

"profiles": {

"kubernetes": {

"expiry": "87600h",

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth"

]

}

}

}

}

修改CA请求文件

[root@k8s-master1 ssl]# vi ca-csr.json

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "k8s",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert -initca ca-csr.json | cfssljson -bare ca

[root@k8s-master1 ssl]# ls -l

total 20

-rw-r--r-- 1 root root 387 Mar 20 17:00 ca-config.json

-rw-r--r-- 1 root root 1005 Mar 20 17:05 ca.csr

-rw-r--r-- 1 root root 269 Mar 20 17:03 ca-csr.json

-rw------- 1 root root 1679 Mar 20 17:05 ca-key.pem

-rw-r--r-- 1 root root 1371 Mar 20 17:05 ca.pem

1. 创建ETCD证书

[root@k8s-master1 ssl]# cat etcd-csr.json

{

"CN": "etcd",

"hosts": [

"127.0.0.1",

"10.3.8.101",

"10.3.8.104",

"10.3.8.105"

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "k8s",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert \

-ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes \

etcd-csr.json | cfssljson -bare etcd

[root@k8s-master1 ssl]# ls -l etcd\*

-rw-r--r-- 1 root root 1086 Mar 20 17:27 etcd.csr

-rw-r--r-- 1 root root 402 Mar 20 17:27 etcd-csr.json

-rw------- 1 root root 1679 Mar 20 17:27 etcd-key.pem

-rw-r--r-- 1 root root 1460 Mar 20 17:27 etcd.pem

1. 创建kubernetes证书

[root@k8s-master1 ssl]# vi kubernetes-csr.json

{

"CN": "kubernetes",

"hosts": [

"127.0.0.1",

"10.3.8.101",

"10.3.8.102",

"10.3.8.103",

"10.1.0.1",

"10.254.0.2",

"kubernetes",

"kubernetes.default",

"kubernetes.default.svc",

"kubernetes.default.svc.cluster",

"kubernetes.default.svc.cluster.local"],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "k8s",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kubernetes-csr.json | cfssljson -bare kubernetes

1. 创建admin证书

[root@k8s-master1 ssl]# vi admin-csr.json

{

"CN": "admin",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "system:masters",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes admin-csr.json | cfssljson -bare admin

1. 创建kube-proxy证书

[root@k8s-master1 ssl]# vi kube-proxy-csr.json

{

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

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "k8s",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kube-proxy-csr.json | cfssljson -bare kube-proxy

1. 创建Flannel证书

[root@k8s-master1 ssl]# vi flanneld-csr.json

{

"CN": "flanneld",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "Guangdong",

"L": "Guangzhou",

"O": "k8s",

"OU": "System"

}

]

}

[root@k8s-master1 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes flanneld-csr.json | cfssljson -bare flanneld

将证书复制到/opt/kubernetes/ssl目录下：

[root@k8s-master1 ssl]# cp \*.pem /opt/kubernetes/ssl

[root@k8s-master1 ssl]# scp \*.pem k8s-worker1:/opt/kubernetes/ssl

[root@k8s-master1 ssl]# scp \*.pem k8s-worker2:/opt/kubernetes/ssl

查看校验证书：

openssl x509 -noout -text -in kubernetes.pem

cfssl-certinfo -cert kubernetes.pem

## ETCD集群部署

1. 准备ETCD软件

# cd /usr/local/src

# wget https://github.com/etcd-io/etcd/releases/download/v3.3.12/etcd-v3.3.12-linux-amd64.tar.gz

# tar zxf etcd-v3.3.12-linux-amd64.tar.gz

# cd etcd-v3.3.12-linux-amd64/

# cp etcd etcdctl /opt/kubernetes/bin/

# scp etcd etcdctl k8s-worker1:/opt/kubernetes/bin

#

1. 配置ETCD参数

[root@k8s-master1 ~]# vi /opt/kubernetes/cfg/etcd.conf

#[member]

ETCD\_NAME="k8s-master1"

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

#ETCD\_SNAPSHOT\_COUNTER="10000"

#ETCD\_HEARTBEAT\_INTERVAL="100"

#ETCD\_ELECTION\_TIMEOUT="1000"

ETCD\_LISTEN\_PEER\_URLS="https://10.3.8.101:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://10.3.8.101:2379,https://127.0.0.1:2379"

#ETCD\_MAX\_SNAPSHOTS="5"

#ETCD\_MAX\_WALS="5"

#ETCD\_CORS=""

#[cluster]

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://10.3.8.101:2380"

# if you use different ETCD\_NAME (e.g. test),

# set ETCD\_INITIAL\_CLUSTER value for this name, i.e. "test=http://..."

ETCD\_INITIAL\_CLUSTER="k8s-master1=https://10.3.8.101:2380,k8s-worker1=https://10.3.8.104:2380,k8s-worker2=https://10.3.8.105:2380"

ETCD\_INITIAL\_CLUSTER\_STATE="new"

ETCD\_INITIAL\_CLUSTER\_TOKEN="k8s-etcd-cluster"

ETCD\_ADVERTISE\_CLIENT\_URLS="https://10.3.8.101:2379"

#[security]

CLIENT\_CERT\_AUTH="true"

ETCD\_CA\_FILE="/opt/kubernetes/ssl/ca.pem"

ETCD\_CERT\_FILE="/opt/kubernetes/ssl/etcd.pem"

ETCD\_KEY\_FILE="/opt/kubernetes/ssl/etcd-key.pem"

PEER\_CLIENT\_CERT\_AUTH="true"

ETCD\_PEER\_CA\_FILE="/opt/kubernetes/ssl/ca.pem"

ETCD\_PEER\_CERT\_FILE="/opt/kubernetes/ssl/etcd.pem"

ETCD\_PEER\_KEY\_FILE="/opt/kubernetes/ssl/etcd-key.pem"

1. 创建ETCD数据目录

[root@k8s-master1 ~]# mkdir -p /var/lib/etcd/default.etcd

1. 创建ETCD系统服务

[root@k8s-master1 ~]# vi /etc/systemd/system/etcd.service

[Unit]

Description=Etcd Server

After=network.target

[Service]

Type=notify

WorkingDirectory=/var/lib/etcd

EnvironmentFile=-/opt/kubernetes/cfg/etcd.conf

# set GOMAXPROCS to number of processors

ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /opt/kubernetes/bin/etcd"

[Install]

WantedBy=multi-user.target

1. 文件分发到两个worker节点

[root@k8s-master1 ~]# scp /opt/kubernetes/cfg/etcd.conf k8s-worker1:/opt/kubernetes/cfg/

[root@k8s-master1 ~]# scp /opt/kubernetes/cfg/etcd.conf k8s-worker2:/opt/kubernetes/cfg/

[root@k8s-master1 ~]# scp /etc/systemd/system/etcd.service k8s-worker1:/etc/systemd/system/

[root@k8s-master1 ~]# scp /etc/systemd/system/etcd.service k8s-worker2:/etc/systemd/system/

1. 修改k8s-worker1的etcd.conf文件

[root@k8s-worker1 ~]# vi /opt/kubernetes/cfg/etcd.conf，未改动部分没有列出来

ETCD\_NAME="k8s-worker1"

ETCD\_LISTEN\_PEER\_URLS="https://10.3.8.104:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://10.3.8.104:2379,https://127.0.0.1:2379"

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://10.3.8.104:2380"

ETCD\_ADVERTISE\_CLIENT\_URLS="https://10.3.8.104:2379"

创建ETCD数据目录

[root@k8s-worker1 ~]# mkdir -p /var/lib/etcd/default.etcd

1. 修改k8s-worker2的etcd.conf文件

[root@k8s-worker2 ~]# vi /opt/kubernetes/cfg/etcd.conf

ETCD\_NAME="k8s-worker2"

ETCD\_LISTEN\_PEER\_URLS="https://10.3.8.105:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://10.3.8.105:2379,https://127.0.0.1:2379"

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://10.3.8.105:2380"

ETCD\_ADVERTISE\_CLIENT\_URLS="https://10.3.8.105:2379"

创建ETCD数据目录

[root@k8s-worker2 ~]# mkdir -p /var/lib/etcd/default.etcd

1. 加载并启动系统服务

# systemctl daemon-reload

# systemctl enable etcd

# systemctl start etcd

# systemctl status etcd

1. 验证集群

在所有节点上：

# vi /etc/profile，在末尾添加

export ETCDCTL\_CERT\_FILE=/opt/kubernetes/ssl/etcd.pem

export ETCDCTL\_KEY\_FILE=/opt/kubernetes/ssl/etcd-key.pem

export ETCDCTL\_CA\_FILE=/opt/kubernetes/ssl/ca.pem

export ETCDCTL\_ENDPOINTS=https://10.3.8.101:2379,https://10.3.8.104:2379,https://10.3.8.105:2379

# source /etc/profile

# etcdctl cluster-health

member 68901cd2c39ac88 is healthy: got healthy result from https://10.3.8.104:2379

member 5b4bf4f7034bb829 is healthy: got healthy result from https://10.3.8.105:2379

member ce825ba3add8b819 is healthy: got healthy result from https://10.3.8.101:2379

cluster is healthy

## Master节点部署

1. **准备kubernetes软件包**

登录https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG.md，选择相应的版本下载：

[root@k8s-master1 ~]# cd /usr/local/src/

下载官方脚本，执行脚本自动下载

[root@k8s-master1 src]# wget [https://dl.k8s.io/](https://dl.k8s.io/v1.13.4/kubernetes.tar.gz)[v1.13.4/kubernetes.tar.gz](https://dl.k8s.io/v1.13.4/kubernetes.tar.gz)

[root@k8s-master1 src]# tar zxf kubernetes.tar.gz

[root@k8s-master1 src]# cd kubernetes/cluster

[root@k8s-master1 cluster]# ./get-kube-binaries.sh

或者手工下载指定软件包

[root@k8s-master1 src]# wget <https://dl.k8s.io/v1.13.4/kubernetes-server-linux-amd64.tar.gz>

还有client和node版的，不需要下载了，server版的已经全包含进来了。

解压kubernetes-server-linux-amd64.tar.gz并拷贝到适当的目录：

[root@k8s-master1 src]# tar xzf kubernetes-server-linux-amd64.tar.gz

[root@k8s-master1 src]# cd kubernetes/server/bin

[root@k8s-master1 bin]# cp kube-apiserver kube-controller-manager kube-scheduler /opt/kubernetes/bin/

创建 kube-apiserver 使用的客户端 token 文件

[root@k8s-master1 ~]# export BOOTSTRAP\_TOKEN=$(head -c 16 /dev/urandom | od -An -t x | tr -d ' ')

[root@k8s-master1 ~]# cat > /opt/kubernetes/ssl/bootstrap-token.csv <<EOF

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

EOF

在token-auth.csv中拥有以列为单位的认证信息，格式为token，username，uid

创建基础用户名/密码认证配置

[root@k8s-master1 ~]# vi /opt/kubernetes/ssl/basic-auth.csv

admin,admin,1

readonly,readonly,2

格式为：密码,用户名,ui，为后面创建dashborad后用户认证。

1. **部署Kubernetes API Server**

[root@k8s-master1 ~]# vi /usr/lib/systemd/system/kube-apiserver.service

[Unit]

Description=Kubernetes API Server

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

After=network.target

[Service]

ExecStart=/opt/kubernetes/bin/kube-apiserver \

--admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota,NodeRestriction \

--bind-address=0.0.0.0 \

--insecure-bind-address=127.0.0.1 \

--authorization-mode=Node,RBAC \

--runtime-config=rbac.authorization.k8s.io/v1 \

--kubelet-https=true \

--anonymous-auth=false \

--basic-auth-file=/opt/kubernetes/ssl/basic-auth.csv \

--enable-bootstrap-token-auth \

--token-auth-file=/opt/kubernetes/ssl/bootstrap-token.csv \

--service-cluster-ip-range=10.1.0.0/16 \

--service-node-port-range=30000-32767 \

--tls-cert-file=/opt/kubernetes/ssl/kubernetes.pem \

--tls-private-key-file=/opt/kubernetes/ssl/kubernetes-key.pem \

--client-ca-file=/opt/kubernetes/ssl/ca.pem \

--service-account-key-file=/opt/kubernetes/ssl/ca-key.pem \

--etcd-cafile=/opt/kubernetes/ssl/ca.pem \

--etcd-certfile=/opt/kubernetes/ssl/kubernetes.pem \

--etcd-keyfile=/opt/kubernetes/ssl/kubernetes-key.pem \

--etcd-servers=https://10.3.8.101:2379,https://10.3.8.104:2379,https://10.3.8.105:2379 \

--enable-swagger-ui=true \

--allow-privileged=true \

--audit-log-maxage=30 \

--audit-log-maxbackup=3 \

--audit-log-maxsize=100 \

--audit-log-path=/opt/kubernetes/log/api-audit.log \

--event-ttl=1h \

--v=2 \

--logtostderr=false \

--log-dir=/opt/kubernetes/log

Restart=on-failure

RestartSec=5

Type=notify

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

启动API Server服务

[root@k8s-master1 ~]# systemctl daemon-reload

[root@k8s-master1 ~]# systemctl enable kube-apiserver

[root@k8s-master1 ~]# systemctl start kube-apiserver

[root@k8s-master1 ~]# systemctl status kube-apiserver

查看API版本

[root@k8s-master1 ~]# curl localhost:8080/api

{

"kind": "APIVersions",

"versions": [

"v1"

],

"serverAddressByClientCIDRs": [

{

"clientCIDR": "0.0.0.0/0",

"serverAddress": "10.3.8.101:6443"

}

]

}

1. **部署Controller Manager服务**

[root@k8s-master1 ~]# vi /usr/lib/systemd/system/kube-controller-manager.service

[Unit]

Description=Kubernetes Controller Manager

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

[Service]

ExecStart=/opt/kubernetes/bin/kube-controller-manager \

--address=127.0.0.1 \

--master=http://127.0.0.1:8080 \

--allocate-node-cidrs=true \

--service-cluster-ip-range=10.1.0.0/16 \

--cluster-cidr=10.2.0.0/16 \

--cluster-name=kubernetes \

--cluster-signing-cert-file=/opt/kubernetes/ssl/ca.pem \

--cluster-signing-key-file=/opt/kubernetes/ssl/ca-key.pem \

--service-account-private-key-file=/opt/kubernetes/ssl/ca-key.pem \

--root-ca-file=/opt/kubernetes/ssl/ca.pem \

--leader-elect=true \

--v=2 \

--logtostderr=false \

--log-dir=/opt/kubernetes/log

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

启动Controller Manager

[root@k8s-master1 ~]# systemctl daemon-reload

[root@k8s-master1 ~]# systemctl enable kube-controller-manager

[root@k8s-master1 ~]# systemctl start kube-controller-manager

[root@k8s-master1 ~]# systemctl status kube-controller-manager

1. **部署Kubernetes Scheduler**

[root@k8s-master1 ~]# vi /usr/lib/systemd/system/kube-scheduler.service

[Unit]

Description=Kubernetes Scheduler

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

[Service]

ExecStart=/opt/kubernetes/bin/kube-scheduler \

--address=127.0.0.1 \

--master=http://127.0.0.1:8080 \

--leader-elect=true \

--v=2 \

--logtostderr=false \

--log-dir=/opt/kubernetes/log

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

启动Kubernetes Scheduler

[root@k8s-master1 ~]# systemctl daemon-reload

[root@k8s-master1 ~]# systemctl enable kube-scheduler

[root@k8s-master1 ~]# systemctl start kube-scheduler

[root@k8s-master1 ~]# systemctl status kube-scheduler

1. **部署kubectl 命令行工具**

[root@k8s-master1 ~]# cp /usr/local/src/kubernetes/server/bin/kubectl /opt/kubernetes/bin/

配置命令补全：

[root@k8s-master1 ~]# yum install bash-completion

[root@k8s-master1 ~]# source /usr/share/bash-completion/bash\_completion

[root@k8s-master1 ~]# source <(kubectl completion bash)

命令kubectl在默认情况下（即未指定--kubeconfig=参数时），会到$HOME/.kube目录下寻找名为config的配置文件，配置文件中包含集群ip地址、端口号、用户名、密码、证书、名称空间等信息，kubectl据此建构访问集群的上下文。以下命令kubectl config均未指定--kubeconfig=参数。

1. 设置集群参数

[root@k8s-master1 ~]# kubectl config set-cluster kubernetes \

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

--embed-certs=true \

--server=https://10.3.8.101:6443

1. 设置客户端认证参数

[root@k8s-master1 ~]# kubectl config set-credentials admin \

--client-certificate=/opt/kubernetes/ssl/admin.pem \

--embed-certs=true \

--client-key=/opt/kubernetes/ssl/admin-key.pem

1. 设置上下文参数

[root@k8s-master1 ~]# kubectl config set-context kubernetes \

--cluster=kubernetes \

--user=admin

1. 设置默认上下文

[root@k8s-master1 ~]# kubectl config use-context kubernetes

1. 查看kubeconfig内容

[root@k8s-master1 ~]# kubectl config view

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: DATA+OMITTED

server: https://10.3.8.101:6443

name: kubernetes

contexts:

- context:

cluster: kubernetes

user: admin

name: kubernetes

current-context: kubernetes

kind: Config

preferences: {}

users:

- name: admin

user:

client-certificate-data: REDACTED

client-key-data: REDACTED

1. 验证master节点功能

[root@k8s-master1 ~]# kubectl get cs

1. **准备部署node节点**
2. 将相关软件包复制到node节点中

[root@k8s-master1 ~]# cd /usr/local/src/kubernetes/server/bin/

[root@k8s-master1 bin]# scp kubelet kube-proxy k8s-worker1:/opt/kubernetes/bin/

[root@k8s-master1 bin]# scp kubelet kube-proxy k8s-worker2:/opt/kubernetes/bin/

1. 创建角色绑定

[root@k8s-master1 ssl]# kubectl create clusterrolebinding kubelet-bootstrap \

--clusterrole=system:node-bootstrapper \

--user=kubelet-bootstrap

1. 创建kubeconfig 文件，设置集群参数

[root@k8s-master1 ssl]# kubectl config set-cluster kubernetes \

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

--embed-certs=true \

--server=https://10.3.8.101:6443 \

--kubeconfig=bootstrap.kubeconfig

[root@k8s-master1 ssl]# kubectl config set-cluster kubernetes \

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

--embed-certs=true \

--server=https://10.3.8.101:6443 \

--kubeconfig=kube-proxy.kubeconfig

1. 设置客户端认证参数，token值为之前生成的

[root@k8s-master1 ssl]# kubectl config set-credentials kubelet-bootstrap \

--token=${BOOTSTRAP\_TOKEN} \

--kubeconfig=bootstrap.kubeconfig

[root@k8s-master1 ssl]# kubectl config set-credentials kube-proxy \

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

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

--embed-certs=true \

--kubeconfig=kube-proxy.kubeconfig

1. 设置上下文参数

[root@k8s-master1 ssl]# kubectl config set-context default \

--cluster=kubernetes \

--user=kubelet-bootstrap \

--kubeconfig=bootstrap.kubeconfig

[root@k8s-master1 ssl]# kubectl config set-context default \

--cluster=kubernetes \

--user=kube-proxy \

--kubeconfig=kube-proxy.kubeconfig

1. 选择默认上下文并向node节点分发在master端生成的bootstrap.kubeconfig文件

[root@k8s-master1 ssl]# kubectl config use-context default --kubeconfig=bootstrap.kubeconfig

[root@k8s-master1 ssl]# kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig

[root@k8s-master1 ssl]# cp \*.kubeconfig /opt/kubernetes/cfg

[root@k8s-master1 ssl]# scp \*.kubeconfig k8s-worker1:/opt/kubernetes/cfg

[root@k8s-master1 ssl]# scp \*.kubeconfig k8s-worker2:/opt/kubernetes/cfg

## Node节点部署

Node节点是Kubernetes集群中的工作负载节点，每个node都会被master分配一些工作负载，每个node节点都运行以下关键服务进程：  
Kubelet：负责pod对应的容器的创建、启停等任务，同时与master节点密切协作，实现集群管理的基本功能。  
Kube-proxy：实现kubernetes service的通信与负载均衡机制的重要组件。

Docker Engine(docker)：Docker引擎，负责本机的容器创建和管理工作。

1. **部署kubelet**

在k8s集群中，每个Node节点都会启动kubelet进程，用来处理Master节点下发到本节点的任务，管理Pod和pod中的容器。kubelet会在API Server上注册节点信息，定期向Master汇报节点资源使用情况，并通过cAdvisor监控容器和节点资源。

1. 设置CNI支持

[root@k8s-worker1 ~]# mkdir -p /etc/cni/net.d

[root@k8s-worker1 ~]# vi /etc/cni/net.d/10-default.conf

{

"name": "flannel",

"type": "flannel",

"delegate": {

"bridge": "docker0",

"isDefaultGateway": true,

"mtu": 1400

}

}

1. 创建kubelet服务配置

[root@k8s-worker1 ~]# mkdir /var/lib/kubelet

[root@k8s-worker1 ~]# vi /usr/lib/systemd/system/kubelet.service

[Unit]

Description=Kubernetes Kubelet

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

After=docker.service

Requires=docker.service

[Service]

WorkingDirectory=/var/lib/kubelet

ExecStart=/opt/kubernetes/bin/kubelet \

--address=10.3.8.104 \

--hostname-override=10.3.8.104 \

--pod-infra-container-image=mirrorgooglecontainers/pause-amd64 \

--experimental-bootstrap-kubeconfig=/opt/kubernetes/cfg/bootstrap.kubeconfig \

--kubeconfig=/opt/kubernetes/cfg/kubelet.kubeconfig \

--cert-dir=/opt/kubernetes/ssl \

--network-plugin=cni \

--cni-conf-dir=/etc/cni/net.d \

--cni-bin-dir=/opt/kubernetes/bin/cni \

--cluster-dns=10.1.0.2 \

--cluster-domain=cluster.local. \

--hairpin-mode hairpin-veth \

--allow-privileged=true \

--fail-swap-on=false \

--cgroup-driver=systemd \

--runtime-cgroups=/systemd/system.slice \

--kubelet-cgroups=/systemd/system.slice \

--v=2 \

--logtostderr=false \

--log-dir=/opt/kubernetes/log

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

1. 启动Kubelet

[root@k8s-worker1 ~]# systemctl daemon-reload

[root@k8s-worker1 ~]# systemctl enable kubelet

[root@k8s-worker1 ~]# systemctl start kubelet

[root@k8s-worker1 ~]# systemctl status kubelet

1. 查看csr请求（注意是在k8s-maste上执行）

[root@k8s-master1 ~]# kubectl get csr

NAME AGE REQUESTOR CONDITION

node-csr-THuGyzjc4RyGvpPH3iiutbvegrRZX-Zyf\_KJGhd1WhA 28s kubelet-bootstrap Pending

1. 批准kubelet 的 TLS 证书请求

[root@k8s-master1 ~]# kubectl get csr|grep 'Pending' | awk 'NR>0{print $1}'| xargs kubectl certificate approve

执行完毕后，查看节点状态如果是Ready的状态就说明一切正常：

[root@k8s-master1 ~]# kubectl get node

NAME STATUS ROLES AGE VERSION

10.3.8.104 Ready <none> 54s v1.13.4

1. **部署Kubernetes Proxy**

从Kubernetes 1.12版本起，kube-proxy服务默认使用ipvs实现，取消了之前的iptables。这有助于提升K8s大规模集群环境下的性能和稳定性。

1. 配置kube-proxy使用IPVS

[root@k8s-worker1 ~]# yum install -y ipvsadm ipset conntrack

[root@k8s-worker1 ~]# lsmod|grep ip\_vs

ip\_vs\_sh 12688 0

ip\_vs\_wrr 12697 0

ip\_vs\_rr 12600 1

ip\_vs 145497 7 ip\_vs\_rr,ip\_vs\_sh,ip\_vs\_wrr

nf\_conntrack 133095 7 ip\_vs,......,nf\_conntrack\_ipv4

libcrc32c 12644 4 xfs,ip\_vs,nf\_nat,nf\_conntrack

1. 创建kube-proxy服务配置

[root@k8s-worker1 ~]# mkdir /var/lib/kube-proxy

[root@k8s-worker1 ~]# vi /usr/lib/systemd/system/kube-proxy.service

[Unit]

Description=Kubernetes Kube-Proxy Server

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

After=network.target

[Service]

WorkingDirectory=/var/lib/kube-proxy

ExecStart=/opt/kubernetes/bin/kube-proxy \

--bind-address=10.3.8.104 \

--hostname-override=10.3.8.104 \

--kubeconfig=/opt/kubernetes/cfg/kube-proxy.kubeconfig \

--masquerade-all \

--feature-gates=SupportIPVSProxyMode=true \

--proxy-mode=ipvs \

--ipvs-min-sync-period=5s \

--ipvs-sync-period=5s \

--ipvs-scheduler=rr \

--logtostderr=true \

--v=2 \

--logtostderr=false \

--log-dir=/opt/kubernetes/log

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

1. 启动Kubernetes Proxy

[root@k8s-worker1 ~]# systemctl daemon-reload

[root@k8s-worker1 ~]# systemctl enable kube-proxy

[root@k8s-worker1 ~]# systemctl start kube-proxy

[root@k8s-worker1 ~]# systemctl status kube-proxy

虽然status结果显示绿色的active (running)，但也存在问题：

Failed to execute iptables-restore for nat: exit status 1 (iptables-restore: line 7 failed

1. 检查LVS状态

[root@k8s-worker1 ~]# ipvsadm -L -n

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 10.1.0.1:443 rr

-> 10.3.8.101:6443 Masq 1 0 0

至此，在k8s-worker1节点上部署Kubelet和proxy完成，在K8s-worker2上重复上述过程，部署完后，回到k8s-master1节点上查看集群状态：

[root@k8s-master1 ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

10.3.8.104 Ready <none> 65m v1.13.4

10.3.8.105 Ready <none> 14s v1.13.4

因为Master中没有装kubelet，所以kubectl get nodes就看不到Master的。

给节点打标签：

[root@k8s-master1 docker]# kubectl label node 10.3.8.104 node-role.kubernetes.io/node='node'

[root@k8s-master1 docker]# kubectl label node 10.3.8.105 node-role.kubernetes.io/node='node'

[root@k8s-master1 docker]# kubectl get node

NAME STATUS ROLES AGE VERSION

10.3.8.104 Ready node 12d v1.13.4

10.3.8.105 Ready node 12d v1.13.4

## Flannel网络部署

1. 部署flannel软件包

[root@k8s-master1 ~]# cd /usr/local/src

[root@k8s-master1 src]# wget \

https://github.com/coreos/flannel/releases/download/v0.11.0/flannel-v0.11.0-linux-amd64.tar.gz

[root@k8s-master1 src]# tar zxvf flannel-v0.11.0-linux-amd64.tar.gz

[root@k8s-master1 src]# cp flanneld mk-docker-opts.sh /opt/kubernetes/bin/

[root@k8s-master1 src]# scp flanneld mk-docker-opts.sh k8s-worker1:/opt/kubernetes/bin/

[root@k8s-master1 src]# scp flanneld mk-docker-opts.sh k8s-worker2:/opt/kubernetes/bin/

[root@k8s-master1 src]# cd kubernetes/cluster/centos/node/bin/

[root@k8s-master1 bin]# cp remove-docker0.sh /opt/kubernetes/bin/

[root@k8s-master1 bin]# scp remove-docker0.sh k8s-worker1:/opt/kubernetes/bin/

[root@k8s-master1 bin]# scp remove-docker0.sh k8s-worker2:/opt/kubernetes/bin/

1. 配置flannel

[root@k8s-master1 bin]# vi /opt/kubernetes/cfg/flannel

FLANNEL\_ETCD="-etcd-endpoints=https://10.3.8.101:2379,https://10.3.8.104:2379,https://10.3.8.105:2379"

FLANNEL\_ETCD\_KEY="-etcd-prefix=/kubernetes/network"

FLANNEL\_ETCD\_CAFILE="-etcd-cafile=/opt/kubernetes/ssl/ca.pem"

FLANNEL\_ETCD\_CERTFILE="-etcd-certfile=/opt/kubernetes/ssl/flanneld.pem"

FLANNEL\_ETCD\_KEYFILE="-etcd-keyfile=/opt/kubernetes/ssl/flanneld-key.pem"

1. 创建Flannel系统服务

[root@k8s-master1 ~]# vi /usr/lib/systemd/system/flannel.service

[Unit]

Description=Flanneld overlay address etcd agent

After=network.target

Before=docker.service

[Service]

EnvironmentFile=-/opt/kubernetes/cfg/flannel

ExecStartPre=/opt/kubernetes/bin/remove-docker0.sh

ExecStart=/opt/kubernetes/bin/flanneld ${FLANNEL\_ETCD} ${FLANNEL\_ETCD\_KEY} ${FLANNEL\_ETCD\_CAFILE} ${FLANNEL\_ETCD\_CERTFILE} ${FLANNEL\_ETCD\_KEYFILE}

ExecStartPost=/opt/kubernetes/bin/mk-docker-opts.sh -d /run/flannel/docker

Type=notify

[Install]

WantedBy=multi-user.target

RequiredBy=docker.service

* mk-docker-opts.sh 脚本将分配给 flanneld 的 Pod 子网网段信息写入 /run/flannel/docker 文件，后续 docker 启动时使用这个文件中的环境变量配置 docker0 网桥；
* flanneld 使用系统缺省路由所在的接口与其它节点通信，对于有多个网络接口（如内网和公网）的节点，可以用 -iface 参数指定通信接口，如上面的 eth0 接口;
* flanneld 运行时需要 root 权限；

1. 复制配置文件到其它节点

[root@k8s-master1 ~]# scp /opt/kubernetes/cfg/flannel k8s-worker1:/opt/kubernetes/cfg/

[root@k8s-master1 ~]# scp /opt/kubernetes/cfg/flannel k8s-worker2:/opt/kubernetes/cfg/

[root@k8s-master1 ~]# scp /usr/lib/systemd/system/flannel.service k8s-worker1:/usr/lib/systemd/system/

[root@k8s-master1 ~]# scp /usr/lib/systemd/system/flannel.service k8s-worker2:/usr/lib/systemd/system/

1. 安装CNI插件

CNI插件官网：<https://github.com/containernetworking/plugins/releases>

[root@k8s-master1 ~]# cd /usr/local/src/

[root@k8s-master1 src]# wget \

<https://github.com/containernetworking/plugins/releases/download/v0.7.5/cni-plugins-amd64-v0.7.5.tgz>

[root@k8s-master1 src]# tar zxf cni-plugins-amd64-v0.7.5.tgz -C /opt/kubernetes/bin/cni/

[root@k8s-master1 src]# scp -r /opt/kubernetes/bin/cni/\* k8s-worker1:/opt/kubernetes/bin/cni/

[root@k8s-master1 src]# scp -r /opt/kubernetes/bin/cni/\* k8s-worker2:/opt/kubernetes/bin/cni/

1. 在master节点创建Etcd的key

[root@k8s-master1 ~]# etcdctl --ca-file /opt/kubernetes/ssl/ca.pem \

--cert-file /opt/kubernetes/ssl/flanneld.pem --key-file /opt/kubernetes/ssl/flanneld-key.pem \

--no-sync -C https://10.3.8.101:2379,https://10.3.8.104:2379,https://10.3.8.105:2379 \

mk /kubernetes/network/config '{ "Network": "10.2.0.0/16", "Backend": { "Type": "vxlan", "VNI": 1 }}'

1. 启动flannel

[root@k8s-master1 ~]# systemctl daemon-reload

[root@k8s-master1 ~]# systemctl enable flannel

[root@k8s-master1 ~]# systemctl start flannel

[root@k8s-master1 ~]# systemctl status flannel

1. 查看网络配置

[root@k8s-master1 ~]# etcdctl ls /kubernetes/network -r

/kubernetes/network/config

/kubernetes/network/subnets

/kubernetes/network/subnets/10.2.42.0-24

/kubernetes/network/subnets/10.2.52.0-24

/kubernetes/network/subnets/10.2.63.0-24

查看路由

[root@k8s-master1 ~]# etcdctl get /kubernetes/network/subnets/10.2.52.0-24

{"PublicIP":"10.3.8.104","BackendType":"vxlan","BackendData":{"VtepMAC":"36:3a:ec:77:84:66"}}

[root@k8s-master1 ~]# etcdctl get /kubernetes/network/subnets/10.2.42.0-24

{"PublicIP":"10.3.8.105","BackendType":"vxlan","BackendData":{"VtepMAC":"12:ef:62:03:5c:cb"}}

[root@k8s-master1 ~]# etcdctl get /kubernetes/network/subnets/10.2.63.0-24

{"PublicIP":"10.3.8.101","BackendType":"vxlan","BackendData":{"VtepMAC":"96:33:67:a3:f4:b0"}}

flannel服务启动时主要做了以下几步的工作：

* 从etcd中获取network的配置信息
* 划分subnet，并在etcd中进行注册
* 将子网信息记录到/run/flannel/subnet.env中

[root@k8s-master1 ~]# cat /run/flannel/subnet.env

FLANNEL\_NETWORK=10.2.0.0/16

FLANNEL\_SUBNET=10.2.63.1/24

FLANNEL\_MTU=1450

FLANNEL\_IPMASQ=false

* 之后将会有一个脚本将subnet.env转写成一个docker的环境变量文件/run/flannel/docker

[root@k8s-master1 ~]# cat /run/flannel/docker

DOCKER\_OPT\_BIP="--bip=10.2.63.1/24"

DOCKER\_OPT\_IPMASQ="--ip-masq=true"

DOCKER\_OPT\_MTU="--mtu=1450"

DOCKER\_OPTS=" --bip=10.2.63.1/24 --ip-masq=true --mtu=1450"

1. 配置Docker使用Flannel
2. 在Unit段中的After后面添加flannel.service参数，在Wants下面添加Requires=flannel.service.
3. [Service]段中Type后面添加EnvironmentFile=-/run/flannel/docker段，在ExecStart后面添加$DOCKER\_OPTS参数  
   配置如下：

[root@k8s-master1 ~]# vi /usr/lib/systemd/system/docker.service

[Unit]

Description=Docker Application Container Engine

Documentation=https://docs.docker.com

After=network-online.target firewalld.service containerd.service flannel.service

Wants=network-online.target

Requires=docker.socket flannel.service

[Service]

Type=notify

EnvironmentFile=-/run/flannel/docker

ExecStart=/usr/bin/dockerd $DOCKER\_OPTS

...

将配置分发到另外两个节点中(源和目标机器都要安装rsync)

[root@k8s-master1 ~]# rsync -av /usr/lib/systemd/system/docker.service k8s-worker1:/usr/lib/systemd/system/docker.service

[root@k8s-master1 ~]# rsync -av /usr/lib/systemd/system/docker.service k8s-worker2:/usr/lib/systemd/system/docker.service

1. 所有节点重启Docker服务

# systemctl daemon-reload

# systemctl restart docker

运行ip a命令，如果docker0和flannel.1在一个网段，则表示正常：

[root@k8s-master1 ~]# ip a | egrep "flannel|docker" | grep inet

inet 10.2.63.0/32 scope global flannel.1

inet 10.2.63.1/24 brd 10.2.63.255 scope global docker0

查看主机路由表：

[root@k8s-master1 ~]# ip route

default via 10.3.8.254 dev ens192 proto static metric 100

10.2.42.0/24 via 10.2.42.0 dev flannel.1 onlink

10.2.52.0/24 via 10.2.52.0 dev flannel.1 onlink

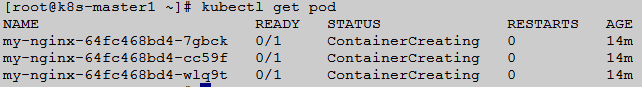
10.2.63.0/24 dev docker0 proto kernel scope link src 10.2.63.1

10.3.8.0/24 dev ens192 proto kernel scope link src 10.3.8.101 metric 100

至此flannel网络配置完成,k8s的集群也部署完成，下面我们来建立pod测试集群之间网络的连通性。

1. 部署应用测试

[root@k8s-master1 ~]# kubectl run my-nginx --image=nginx --port=80 --replicas=3



都14分钟了还没下载完nginx镜像？有问题，查看事件：

[root@k8s-master1 ~]# kubectl describe pod my-nginx-64fc468bd4-7gbck

......

...... mirrorgooglecontainers/pause-amd64:latest not found

可以看到是pause-amd64这个镜像拉取不下来，切换到node节点，从阿里云上拉取再改名：

[root@k8s-worker1 ~]# docker pull registry.cn-beijing.aliyuncs.com/zhoujun/pause-amd64:3.1

[root@k8s-worker1 ~]# docker tag registry.cn-beijing.aliyuncs.com/zhoujun/pause-amd64:3.1 mirrorgooglecontainers/pause-amd64:latest

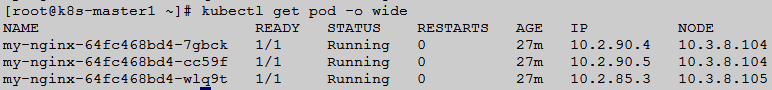
[root@k8s-worker1 ~]# systemctl restart docker

节点k8s-worker2上做同样的操作。

[root@k8s-worker1 ~]# docker ps -a

可以看到两个容器，一个是nginx，一个是pause-amd64。

到k8s-master1上查看pod的IP：



测试连通性：

[root@k8s-master1 ~]# ping 10.2.90.5 -c 2

[root@k8s-master1 ~]# ping 10.2.85.3 -c 2

暴露服务，创建service

[root@k8s-master1 ~]# kubectl expose deployment my-nginx --port=8080 --target-port=80 --external-ip=10.3.8.104

这个external-ip就是某个node节点的对外IP。

[root@k8s-master1 ~]# curl -I <http://10.3.8.104:8080>

HTTP/1.1 200 OK

至此，kubernetes集群大功告成。

删除pod：

[root@k8s-master1 ~]# kubectl scale deployment/my-nginx --replicas=0

[root@k8s-master1 ~]# kubectl delete deployment/my-nginx

下面的curl命令，分别返回集群中的Pod列表、Service列表、RC列表：

curl localhost:8080/api/v1/pods

curl localhost:8080/api/v1/services

curl localhost:8080/api/v1/replicationcontrollers

## [CoreDNS和Dashboard部署](https://www.cnblogs.com/saneri/p/9132556.html)

1. 部署CoreDNS

[root@k8s-master1 ~]# cd /usr/local/src/kubernetes/cluster/addons/dns/coredns

[root@k8s-master1 coredns]# cp coredns.yaml.base coredns.yaml

[root@k8s-master1 coredns]# vi coredns.yaml，修改如下两个地方为自己的domain和cluster ip地址

1.kubernetes \_\_PILLAR\_\_DNS\_\_DOMAIN\_\_

改为 kubernetes cluster.local.

2.clusterIP: \_\_PILLAR\_\_DNS\_\_SERVER\_\_

改为：

clusterIP: 10.1.0.2

创建coredns服务：

[root@k8s-master1 coredns]# kubectl apply -f coredns.yaml

[root@k8s-master1 coredns]# kubectl get pod -n kube-system -o wide

NAME READY STATUS RESTARTS AGE IP NODE

coredns-fff89c9b9-5tttj 0/1 ImagePullBackOff 0 3m10s 10.2.52.6 10.3.8.104

状态ImagePullBackOff，查看事件：

[root@k8s-master1 coredns]# kubectl describe pod coredns-fff89c9b9-5tttj -n kube-system

最后几行可以看到： 10.3.8.104 Back-off pulling image "k8s.gcr.io/coredns:1.2.6"

到node节点（包括k8s-worker1和k8s-worker2），下载其它的coredns再改名：

# docker pull coredns/coredns:1.2.6

# docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6

过一会就能看到pod是running状态了。

[root@k8s-master1 coredns]# kubectl scale deploy coredns --replicas=2 -n kube-system

[root@k8s-master1 coredns]# kubectl get pod -o wide -n kube-system

NAME READY STATUS RESTARTS AGE IP NODE

coredns-fff89c9b9-5tttj 1/1 Running 0 50m 10.2.52.6 10.3.8.104

coredns-fff89c9b9-lv65z 1/1 Running 0 26s 10.2.42.6 10.3.8.105

[root@k8s-master1 coredns]# kubectl get svc --all-namespaces

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

default kubernetes ClusterIP 10.1.0.1 <none> 443/TCP 5d23h

kube-system kube-dns ClusterIP 10.1.0.2 <none> 53/UDP,53/TCP 32m

CoreDNS解析测试

不要用image=docker.io/busybox，这个镜像的nslookup测试会失败。

[root@k8s-master1 coredns]# kubectl run dig --rm -it --image=docker.io/azukiapp/dig /bin/sh

kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.

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

/ # nslookup www.baidu.com

Server: 10.1.0.2

Address: 10.1.0.2#53

Non-authoritative answer:

www.baidu.com canonical name = www.a.shifen.com.

Name: www.a.shifen.com

Address: 163.177.151.109

Name: www.a.shifen.com

Address: 163.177.151.110

1. 部署Dashboard

下载dashborad文件地址，大神已经修改好了我们直接执行就可以：

[root@k8s-master1 ~]# mkdir /opt/kubernetes/dashboard && cd /opt/kubernetes/dashboard

[root@k8s-master1 dashboard]# git clone https://github.com/unixhot/salt-kubernetes.git

[root@k8s-master1 dashboard]# cd salt-kubernetes/addons

[root@k8s-master1 addons]# kubectl apply -f dashboard/

[root@k8s-master1 addons]# kubectl cluster-info

Kubernetes master is running at https://10.3.8.101:6443

CoreDNS is running at https://10.3.8.101:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

kubernetes-dashboard is running at https://10.3.8.101:6443/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy

访问dashboard的方式有三种：

1. 通过kube-apiserver访问，见前面kubectl cluster-info输出；
2. 通过 kubectl proxy 访问；
3. 通过http://NodeIP:nodePort访问；

查看dashborad对外映射端口

[root@k8s-master1 addons]# kubectl get svc --all-namespaces

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

default kubernetes ClusterIP 10.1.0.1 <none> 443/TCP 6d

kube-system kube-dns ClusterIP 10.1.0.2 <none> 53/UDP,53/TCP 75m

kube-system kubernetes-dashboard NodePort 10.1.146.241 <none> 443:30001/TCP 3m1s

那么可以通过https://10.3.8.104:30001/ 或者https://10.3.8.105:30001访问。

登录的时候，选择令牌。然后在master端执行如下命令，生成认证token登录：

[root@k8s-master1 ~]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

将token:一行后面的内容复制到令牌里，即可登录仪表板。

参考：<https://www.cnblogs.com/saneri/p/9119336.html>

<https://www.cnblogs.com/saneri/category/923407.html>

## 部署traefik Ingress

1. 理解Ingress

简单的说，ingress就是从kubernetes集群外访问集群的入口，将用户的URL请求转发到不同的service上。Ingress相当于nginx、apache等负载均衡方向代理服务器，其中还包括规则定义，即URL的路由信息，路由信息得的刷新由Ingress controller来提供。

1. 理解Ingress Controller

Ingress Controller 实质上可以理解为是个监视器，Ingress Controller 通过不断地跟 kubernetes API 打交道，实时的感知后端 service、pod 等变化，比如新增和减少 pod，service 增加与减少等；当得到这些变化信息后，Ingress Controller 再结合下文的 Ingress 生成配置，然后更新反向代理负载均衡器，并刷新其配置，达到服务发现的作用。不过traefik出现后，它就要废弃了，毕竟Ingress Controller不是原生的工具。

1. 介绍traefik Ingress

Traefik是一款开源的反向代理与负载均衡工具。它最大的优点是能够与常见的微服务系统直接整合，可以实现自动化动态配置。目前支持Docker, Swarm, Mesos/Marathon, Mesos, Kubernetes, Consul, Etcd, Zookeeper, BoltDB, Rest API等等后端模型。

1. 部署Traefik Ingress

本文将采用daemonset方式部署Traefik Ingress来进行服务发布。

部署Traefik的配置文件可以在如下github仓库中找到：

<https://github.com/rootsongjc/kubernetes-handbook/tree/master/manifests/traefik-ingress>

下载相关yaml文件：

mkdir /opt/kubernetes/Traefik/ && cd /opt/kubernetes/Traefik/

traefik\_url=”https://raw.githubusercontent.com/rootsongjc/kubernetes-handbook/master/manifests/traefik-ingress”

wget $traefik\_url/ingress-rbac.yaml

wget $traefik\_url/ingress.yaml

wget $traefik\_url/traefik.yaml

wget $traefik\_url/ui.yaml

其中，ingress-rbac.yaml用于service account验证，不需要修改，内容如下：

[root@k8s-master Traefik]# vim ingress-rbac.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: ingress

namespace: kube-system

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1beta1

metadata:

name: ingress

subjects:

- kind: ServiceAccount

name: ingress

namespace: kube-system

roleRef:

kind: ClusterRole

name: cluster-admin

apiGroup: rbac.authorization.k8s.io

[root@k8s-master Traefik]# kubectl apply -f ingress-rbac.yaml

创建DaemonSet

由于是指定边缘节点来部署Traefik，所以要给指定的节点打上label：

[root@k8s-master ~]# kubectl get nodes --show-labels

[root@k8s-master ~]# kubectl label node 10.3.8.104 traefik=proxy

[root@k8s-master ~]# kubectl label node 10.3.8.105 traefik=proxy

[root@k8s-master ~]# kubectl get nodes --show-labels

通过DaemonSet方式部署Traefik服务：

[root@k8s-master Traefik]# vi traefik.yaml，将文件最后一行改下。

apiVersion: extensions/v1beta1

kind: DaemonSet

metadata:

name: traefik-ingress-lb

namespace: kube-system

labels:

k8s-app: traefik-ingress-lb

spec:

template:

metadata:

labels:

k8s-app: traefik-ingress-lb

name: traefik-ingress-lb

spec:

terminationGracePeriodSeconds: 60

hostNetwork: true

restartPolicy: Always

serviceAccountName: ingress

containers:

- image: traefik

name: traefik-ingress-lb

resources:

limits:

cpu: 200m

memory: 30Mi

requests:

cpu: 100m

memory: 20Mi

ports:

- name: http

containerPort: 80

hostPort: 80

- name: admin

containerPort: 8580

hostPort: 8580

args:

- --web

- --web.address=:8580

- --kubernetes

nodeSelector:

# edgenode: "true"

traefik: "proxy"

其中 traefik 监听 node 的 80 和 8580 端口，80 提供正常服务，8580 是其自带的 UI 界面，原本默认是 8080，因为环境里端口冲突了，所以这里临时改一下。

[root@k8s-master Traefik]# kubectl apply -f traefik.yaml

Traefik UI部署

[root@k8s-master Traefik]# cat ui.yaml，不需要修改

apiVersion: v1

kind: Service

metadata:

name: traefik-web-ui

namespace: kube-system

spec:

selector:

k8s-app: traefik-ingress-lb

ports:

- name: web

port: 80

targetPort: 8580

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: traefik-web-ui

namespace: kube-system

spec:

rules:

- host: traefik-ui.local

http:

paths:

- path: /

backend:

serviceName: traefik-web-ui

servicePort: web

[root@k8s-master Traefik]# kubectl apply -f ui.yaml

准备二个服务实例

实例一：

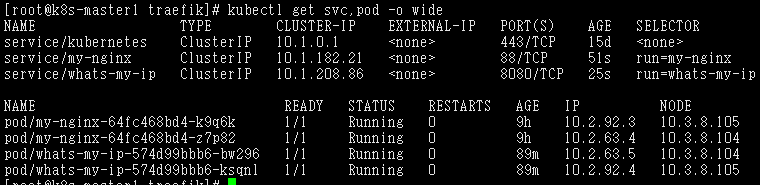
kubectl run my-nginx --image=nginx --replicas=2 #默认80端口

kubectl expose deploy my-nginx --port=88 --target-port=80 --name=my-nginx

实例二：

kubectl run whats-my-ip --image=cloudnativelabs/whats-my-ip --replicas=2 #默认8080端口

kubectl expose deploy whats-my-ip --target-port=8080 --port=8080 --name=whats-my-ip



创建规则ingress.yaml

Ingress有两种代理方法，一是域名，二是路径。域名方式形如xxx.domain.com，yyy.domain.com等，域名部分不同。路径方式形如name.domain.com/path1，name.domain.com/path2，路径部分不同，这里用第一种方式。

[root@k8s-master Traefik]# vi ingress.yaml

cat ingress.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: traefik-ingress

namespace: default

spec:

rules:

- host: mynginx.linuxs.top #要访问的域名

http:

paths:

- path: /

backend:

serviceName: my-nginx #关联服务名及端口

servicePort: 88

- host: whatsmyip.linuxs.top

http:

paths:

- path: /

backend:

serviceName: whats-my-ip

servicePort: 8080

有新service增加时，修改该文件后可以使用kubectl replace -f ingress.yaml来更新。

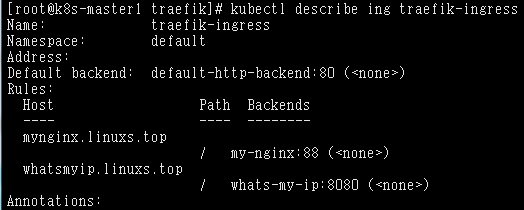
[root@k8s-master Traefik]# kubectl apply -f ingress.yaml

查看traefik关联了哪些服务：

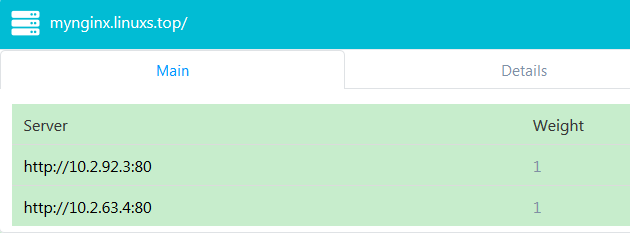
[root@k8s-master1 traefik]# kubectl get ing

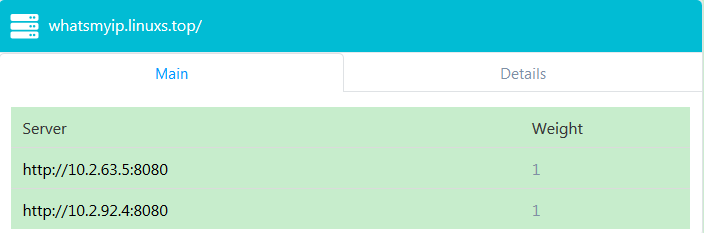
NAME HOSTS ADDRESS PORTS AGE

traefik-ingress mynginx.linuxs.top,whatsmyip.linuxs.top 80 56m



也可通过UI：http://10.3.8.104:8580/dashboard/ 查看traefik关联了哪些服务：





测试：

在集群的任意一个节点上执行：

[root@k8s-master1 traefik]# curl -I -H Host:mynginx.linuxs.top http://10.3.8.104

HTTP/1.1 200 OK

Accept-Ranges: bytes

Content-Length: 612

Content-Type: text/html

Date: Thu, 04 Apr 2019 13:25:47 GMT

Etag: "5c9a3176-264"

Last-Modified: Tue, 26 Mar 2019 14:04:38 GMT

Server: nginx/1.15.10

[root@k8s-master1 traefik]# curl -I -H Host:whatsmyip.linuxs.top http://10.3.8.104

HTTP/1.1 200 OK

Content-Length: 51

Content-Type: text/plain; charset=utf-8

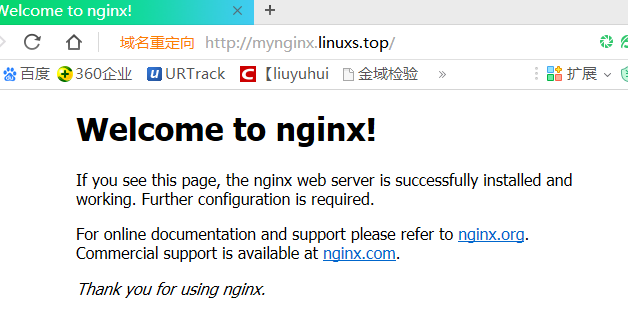
Date: Thu, 04 Apr 2019 13:26:15 GMT

如果在kubernetes集群以外访问就需要设置DNS，或者修改本机的hosts文件，在其中加入：

10.3.8.104 mynginx.linuxs.top

10.3.8.104 whatsmyip.linuxs.top

 浏览器访问mynginx.linuxs.top：



 浏览器访问whatsmyip.linuxs.top：



刷新一下就能看到轮询到另一个服务：



来源：

<https://www.cnblogs.com/saneri/p/9437490.html>