etcd+tls集群部署文档

# 前言

kubeadm安装的集群，默认etcd是一个单机的容器化的etcd，这里改造成三节点的etcd集群。

首先我们需要先部署一个三节点的etcd集群，二进制部署，systemd守护进程，并且需要生成ca证书。

# ETCD集群详情

kuberntes 系统使用 etcd 存储所有数据，此外calico网络也使用该etcd集群，本文档介绍部署一个三节点高可用 etcd 集群，分别命名为etcd01、etcd02、etcd03

192.168.40.3 etcd01

192.168.40.4 etcd02

192.168.40.5 etcd03

本系列默认使用root用户操作。

# 创建 CA 证书和秘钥

本文档使用 CloudFlare 的 PKI 工具集 cfssl 来生成 Certificate Authority (CA) 证书和秘钥文件，CA 是自签名的证书，用来签名后续创建的其它 TLS 证书。

## 安装cfssl

如果不希望将cfssl工具安装到etcd集群节点主机上，可以在其他的主机上进行该操作，生成以后将证书拷贝到部署etcd的节点主机上即可。本文档就是采取这种方法，在一台测试机上执行下面操作。

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

$ chmod u+x cfssl\_linux-amd64

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

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

$ chmod u+x cfssljson\_linux-amd64

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

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

$ chmod u+x cfssl-certinfo\_linux-amd64

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

## 生成ETCD的TLS 秘钥和证书

为了保证通信安全，客户端(如 etcdctl) 与 etcd 集群、etcd 集群之间的通信需要使用 TLS 加密。开始创建 etcd TLS 加密所需的证书和私钥。

### 创建 CA 配置文件

$ cat > ca-config.json <<EOF

{

"signing": {

"default": {

"expiry": "8760h"

},

"profiles": {

"kubernetes": {

"usages": [

"signing",

"key encipherment",

"server auth",

"client auth"

],

"expiry": "8760h"

}

}

}

}

EOF

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指定的 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；

### 创建 CA 证书签名请求文件

$ cat > ca-csr.json <<EOF

{

"CN": "kubernetes",

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

EOF

"CN"：Common Name，kube-apiserver 从证书中提取该字段作为请求的用户名 (User Name)；浏览器使用该字段验证网站是否合法；

"O"：Organization，kube-apiserver 从证书中提取该字段作为请求用户所属的组 (Group)；

### 生成 CA 证书和私钥

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

$ ls ca\*

ca-config.json ca.csr ca-csr.json ca-key.pem ca.pem

$

### 创建 etcd 证书签名请求

$ cat > etcd-csr.json <<EOF

{

"CN": "etcd",

"hosts": [

"127.0.0.1",

"192.168.40.3",

"192.168.40.4",

"192.168.40.5",

"192.168.40.24",

"192.168.40.23",

"192.168.40.22"，

"etcd01.local.com",

" etcd02.local.com ",

" etcd02.local.com ",

" master01.local.com ",

" master01.local.com ",

" master01.local.com ",

],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

EOF

hosts 字段指定授权使用该证书的 etcd 节点 IP

### 生成 etcd 证书和私钥

$ cfssl gencert -ca=ca.pem \

-ca-key=ca-key.pem \

-config=ca-config.json \

-profile=kubernetes etcd-csr.json | cfssljson -bare etcd

$ ls etcd\*

etcd.csr etcd-csr.json etcd-key.pem etcd.pem ca-config.json ca.csr ca-csr.json ca-key.pem ca.pem

$ rm etcd.csr etcd-csr.json

将生成好的etcd.pem和etcd-key.pem以及ca.pem三个文件拷贝到目标节点主机的/etc/etcd/ssl目录下(没有就创建)

# 开始安装etcd

## 下载二进制文件

$ wget https://etcd.readthedocs.io/en/latest/

$ tar -xvf etcd-v3.3.2-linux-amd64.tar.gz

$ mv etcd-v3.2.11-linux-amd64/etcd\* /usr/local/bin/

## 创建 etcd 的 systemd unit 文件

$ mkdir -p /var/lib/etcd # 必须先创建工作目录

$ yum -y install ntpdate kernel-devel gcc-c++

$ntpdate ntp.pool.org

$ vi install\_etcd.sh

#!/bin/bash

# Copyright 2018 The Kubernetes Authors.

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

## Create etcd.conf, etcd.service, and start etcd service.

mkdir -p /usr/local/kubernetes/config

mkdir -p /usr/local/kubernetes/bin

etcd\_data\_dir=/var/lib/etcd

mkdir -p ${etcd\_data\_dir}

#ETCD\_NAME=${1:-"etcd01"}

#ETCD\_INITIAL\_CLUSTER=${3:-"etcd01=https://192.168.40.3:2380,etcd02=https://192.168.40.4:2380,etcd03=https://192.168.40.5:2380"}

#CURRENT\_HOST\_IP=`ip a | grep ens32 | grep 'inet ' | awk '{ print $2}'`

ETCD\_NAME="$1"

ETCD\_INITIAL\_CLUSTER="$2"

if [ ! $ETCD\_NAME ]; then

echo "ENTER ETCD\_NAME eg:etcd01"

exit 1

fi

if [ ! $ETCD\_INITIAL\_CLUSTER ]; then

echo "ENTER ETCD\_INITIAL\_CLUSTER eg:etcd01=https://192.168.40.3:2380,etcd02=https://192.168.40.4:2380,etcd03=https://192.168.40.5:2380"

exit 1

fi

cp -rf bin/\* /usr/local/kubernetes/bin

chmod +x /usr/local/kubernetes/bin/\*

ETCD\_LISTEN\_IP=`ip a | grep ens32 | grep 'inet ' | awk '{ print $2}' | awk -F / '{print $1}'`

ETCD\_DATA\_DIR="${etcd\_data\_dir}/default.etcd"

ETCD\_LISTEN\_PEER\_URLS="https:// ${ETCD\_LISTEN\_IP}:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://${ETCD\_LISTEN\_IP}:2379"

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://${ETCD\_LISTEN\_IP}:2380"

ETCD\_INITIAL\_CLUSTER\_STATE="new"

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

ETCD\_ADVERTISE\_CLIENT\_URLS="https://${ETCD\_LISTEN\_IP}:2379"

CERT\_FILE=/etc/etcd/ssl/etcd.pem

KEY\_FILE=/etc/etcd/ssl/etcd-key.pem

PERR\_CERT\_FILE=/etc/etcd/ssl/etcd.pem

PERR\_KEY\_FILE=/etc/etcd/ssl/etcd-key.pem

TRUSTED\_CA\_FILE=/etc/etcd/ssl/ca.pem

PERR\_TRUSTED\_CA\_FILE=/etc/etcd/ssl/ca.pem

cat <<EOF >/usr/local/kubernetes/config/etcd.conf

# [member]

ETCD\_NAME="${ETCD\_NAME}"

ETCD\_DATA\_DIR="${etcd\_data\_dir}/default.etcd"

ETCD\_LISTEN\_PEER\_URLS="https://192.168.40.3:2380"

ETCD\_LISTEN\_CLIENT\_URLS="https://192.168.40.3:2379"

#[cluster]

ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="https://${ETCD\_LISTEN\_IP}:2380"

ETCD\_INITIAL\_CLUSTER="${ETCD\_INITIAL\_CLUSTER}"

ETCD\_INITIAL\_CLUSTER\_STATE="new"

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

ETCD\_ADVERTISE\_CLIENT\_URLS="https://${ETCD\_LISTEN\_IP}:2379"

#[certs]

CERT\_FILE=/etc/etcd/ssl/etcd.pem

KEY\_FILE=/etc/etcd/ssl/etcd-key.pem

PERR\_CERT\_FILE=/etc/etcd/ssl/etcd.pem

PERR\_KEY\_FILE=/etc/etcd/ssl/etcd-key.pem

TRUSTED\_CA\_FILE=/etc/etcd/ssl/ca.pem

PERR\_TRUSTED\_CA\_FILE=/etc/etcd/ssl/ca.pem

EOF

cat <<EOF >/usr/lib/systemd/system/etcd.service

[Unit]

Description=Etcd Server

After=network.target

[Service]

Type=notify

WorkingDirectory=${etcd\_data\_dir}

EnvironmentFile=-/usr/local/kubernetes/config/etcd.conf

ExecStart=/usr/local/kubernetes/bin/etcd \\

--name=${ETCD\_NAME} \\

--data-dir=${ETCD\_DATA\_DIR} \\

--listen-peer-urls=${ETCD\_LISTEN\_PEER\_URLS} \\

--listen-client-urls=${ETCD\_LISTEN\_CLIENT\_URLS} \\

--advertise-client-urls=${ETCD\_ADVERTISE\_CLIENT\_URLS} \\

--initial-advertise-peer-urls=${ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS} \\

--initial-cluster=${ETCD\_INITIAL\_CLUSTER} \\

--initial-cluster-token=${ETCD\_INITIAL\_CLUSTER\_TOKEN} \\

--initial-cluster-state=${ETCD\_INITIAL\_CLUSTER\_STATE} \\

--cert-file=${CERT\_FILE} \\

--key-file=${KEY\_FILE} \\

--peer-cert-file=${PERR\_CERT\_FILE} \\

--peer-key-file=${PERR\_KEY\_FILE} \\

--trusted-ca-file=${TRUSTED\_CA\_FILE} \\

--peer-trusted-ca-file=${PERR\_TRUSTED\_CA\_FILE} \\

Restart=on-failure

RestartSec=5

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

EOF

systemctl daemon-reload

systemctl enable etcd

systemctl restart etcd

指定 etcd 的工作目录和数据目录为 /var/lib/etcd，需在启动服务前创建这个目录；

为了保证通信安全，需要指定 etcd 的公私钥(cert-file和key-file)、Peers 通信的公私钥和 CA 证书(peer-cert-file、peer-key-file、peer-trusted-ca-file)、客户端的CA证书（trusted-ca-file）；

--initial-cluster-state 值为 new 时，--name 的参数值必须位于 --initial-cluster 列表中；

## 启动 etcd 服务

$ mv etcd.service /etc/systemd/system/

$ systemctl daemon-reload

$ systemctl enable etcd

$ systemctl start etcd

$ systemctl status etcd

$./install\_etcd.sh etcd01 etcd01=https://192.168.40.3:2380,etcd02=https://192.168.40.4:2380,etcd03=https://192.168.40.5:2380

$./install\_etcd.sh etcd02 etcd01=https://192.168.40.3:2380,etcd02=https://192.168.40.4:2380,etcd03=https://192.168.40.5:2380

$./install\_etcd.sh etcd03 etcd01=https://192.168.40.3:2380,etcd02=https://192.168.40.4:2380,etcd03=https://192.168.40.5:2380

# 验证服务

部署完 etcd 集群后，在任意一台 etcd 集群节点上执行如下命令：

$ etcdctl \

--endpoints=https://192.168.40.3:2379 \

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

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

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

cluster-health

## 三台 etcd 的输出均为 healthy 时表示集群服务正常

member 71df888fdf6f0bb9 is healthy: got healthy result from https://192.168.40.3:2379

member 73b5207bc2491164 is healthy: got healthy result from https://192.168.40.4:2379

member 7a4ddb7c77253f4b is healthy: got healthy result from https://192.168.40.5:2379