etcd具有以下特点：

- 完全复制：集群中的每个节点都可以使用完整的存档
- 高可用性：Etcd可用于避免硬件的单点故障或网络问题
- 一致性：每次读取都会返回跨多主机的最新写入
- 简单：包括一个定义良好、面向用户的API（gRPC）
- 安全：实现了带有可选的客户端证书身份验证的自动化TLS
- 快速：每秒10000次写入的基准速度
- 可靠：使用Raft算法实现了强一致、高可用的服务存储目录

https://www.topgoer.com/%E6%95%B0%E6%8D%AE%E5%BA%93%E6%93%8D%E4%BD%9C/go%E6%93%8D%E4%BD%9Cetcd/etcd%E4%BB%8B%E7%BB%8D.html

etcd 是基于 Raft 的分布式 key-value 存储系统，由 CoreOS 开发，常用于服务发现、共享配置以及并发控制（如 leader 选举、分布式锁等）。kubernetes 使用 etcd 存储所有运行数据。

本文档介绍部署一个三节点高可用 etcd 集群的步骤：

- 下载和分发 etcd 二进制文件；
- 创建 etcd 集群各节点的 x509 证书，用于加密客户端(如 etcdctl) 与 etcd 集群、etcd 集群之间的数据流；
- 创建 etcd 的 systemd unit 文件，配置服务参数；
- 检查集群工作状态；

etcd 集群各节点的名称和 IP 如下：

- 192.168.110.100 k8s-master
- 192,168.110.101 k8s-node01
- 192.168.110.102 k8s-node02

关闭防火墙以及SELinux

```
systemctl stop iptables
systemctl stop firewalld
systemctl disable iptables
systemctl disable firewalld
vi /etc/selinux/config
SELINUX=disable
```

设置hosts

```
vim /etc/hosts
192.168.110.100 k8s-master192,168.110.101 k8s-node01192.168.110.102 k8s-node02
```

创建用户

```
useradd etcd -d /data/home -c "Etcd user" -r -s /sbin/nologin
```

下载和分发 etcd 二进制文件

在每个节点下载etcd二进制文件

```
wget https://github.com/coreos/etcd/releases/download/v3.3.7/etcd-v3.3.7-linux-amd64.tar.gz
tar -xvf etcd-v3.3.7-linux-amd64.tar.gz
```

cfssl安装

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

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

chmod +x cfssl_linux-amd64 cfssljson_linux-amd64

mv cfssl_linux-amd64 /usr/local/bin/cfssl

```
 mv cfssljson_linux-amd64 /usr/local/bin/cfssljson
```

创建 CA 证书配置，生成 CA 证书和私钥

```
mkdir /opt/ssl
cd /opt/ssl
cfssl print-defaults config > config.json
cfssl print-defaults csr > csr.json
```

然后分别修改这两个文件为如下内容

config.json

```
cat > config.json <<EOF{
  "signing": {
    "default": {
      "expiry": "87600h"
    },
    "profiles": {
      "kubernetes": {
        "usages": [
            "signing",
            "key encipherment",
            "server auth",
            "client auth"
        ],
        "expiry": "87600h"
      }
    }
  }
}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 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；

csr.json

```
cat > 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

```
cd /opt/ssl
cfssl gencert -initca csr.json | cfssljson -bare ca
```

CA 有关证书列表如下：

```
    [root@k8s-master ssl]# tree
    .
    ├── ca.csr
    ├── ca-key.pem
    ├── ca.pem
    ├── config.json
    └── csr.json
```

创建 etcd 证书配置，生成 etcd 证书和私钥

在 `/opt/ssl` 下添加文件 `etcd-csr.json`，内容如下

```
{
  "CN": "etcd",
```

```
  "hosts": [
    "127.0.0.1",
    "192.168.110.100",
    "192.168.110.101",
    "192.168.110.102"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "etcd",
      "OU": "Etcd Security"
    }
  ]
}
```

生成 etcd 证书和密钥

```
cd /opt/ssl
cfssl gencert -ca=/opt/ssl/ca.pem \
-ca-key=/opt/ssl/ca-key.pem \
-config=/opt/ssl/config.json \
-profile=kubernetes etcd-csr.json | cfssljson -bare etcd
```

etcd 有关证书证书列表如下

```
ls etcd*
etcd.csr  etcd-csr.json  etcd-key.pem  etcd.pem
```

将证书分别传到各个节点并设置权限

```
chmod 644 /opt/ssl/*
```

在三台上都安装 etcd

```
tar -xvf etcd-v3.3.4-linux-amd64.tar.gz
cd etcd-v3.3.4-linux-amd64
cp mv etcd* /opt/platform/etcd/
```

分别添加 etcd 配置

```
vim /opt/platform/etcd/etcd.conf
# [member]
ETCD_NAME="etcd1"  #不同机器需要做修改 与下面集群配置相对应
ETCD_DATA_DIR="/data/etcd"
ETCD_LISTEN_PEER_URLS="https://192.168.110.100:2380"
ETCD_LISTEN_CLIENT_URLS="https://192.168.110.100:2379,http://127.0.0.1:2379"

# [cluster]
```

```
ETCD_ADVERTISE_CLIENT_URLS="https://192.168.110.100:2379"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://192.168.110.100:2380"
ETCD_INITIAL_CLUSTER="etcd1=https://192.168.110.100:2380,etcd2=https://192.168.1
10.101:2380,etcd3=https://192.168.110.102:2380"
ETCD_INITIAL_CLUSTER_STATE=new
ETCD_INITIAL_CLUSTER_TOKEN=etcd-cluster
ETCD_ENABLE_V2="true"
```

配置说明

ETCD_NAME:

etcd 集群中的节点名，这里可以随意，可区分且不重复就行。

ETCD_LISTEN_PEER_URLS:

监听的用于节点之间通信的 URL，可监听多个，集群内部将通过这些 URL 进行数据交互（如选举、数据同步等）。

ETCD_LISTEN_CLIENT_URLS:

监听的用于客户端通信的 URL，同样可以监听多个。

ETCD_ADVERTISE_CLIENT_URLS:

建议使用的客户端通信 URL，该值用于 etcd 代理或 etcd 成员与 etcd 节点通信。

ETCD_INITIAL_ADVERTISE_PEER_URLS:

建议用于节点之间通信的 URL，节点间将以该值进行通信。

ETCD_INITIAL_CLUSTER:

也就是集群中所有的 initial--advertise-peer-urls 的合集。

ETCD_INITIAL_CLUSTER_STATE:

新建集群的标志。

ETCD_INITIAL_CLUSTER_TOKEN:

节点的 token 值，设置该值后集群将生成唯一 ID，并为每个节点也生成唯一 ID，当使用相同配置文件再启动一个集群时，只要该 token 值不一样，etcd 集群就不会相互影响。

flannel操作etcd使用的是v2的API，而kubernetes操作etcd使用的v3的API，为了兼容flannel，将默认开启v2版本，故配置文件中设置 ETCD_ENABLE_V2="true"

添加系统服务

```
vim /usr/lib/systemd/system/etcd.service
[Unit]
Description=Etcd Server
After=network.target
After=network-online.target
Wants=network-online.target

[Service]
Type=notify
EnvironmentFile=/opt/platform/etcd/etcd.conf
ExecStart=/usr/bin/etcd \
--cert-file=/opt/ssl/etcd.pem \
--key-file=/opt/ssl/etcd-key.pem \
```

```
--peer-cert-file=/opt/ssl/etcd.pem \
--peer-key-file=/opt/ssl/etcd-key.pem \
--trusted-ca-file=/opt/ssl/ca.pem \
--peer-trusted-ca-file=/opt/ssl/ca.pem
Restart=on-failure
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

创建 data 目录，然后启动 etcd 服务

```
mkdir -p /opt/platform/etcd/data && chown etcd:etcd -R /opt/platform/etcd &&
chmod 755 /opt/platform/etcd/data
systemctl enable etcd.service && systemctl start etcd.service
```

etcd 3.4注意事项

ETCD3.4版本ETCDCTL_API=3 etcdctl 和 etcd --enable-v2=false 成为了默认配置，如要使用v2版本，执行etcdctl时候需要设置ETCDCTL_API环境变量，例如：ETCDCTL_API=2 etcdctl
ETCD3.4版本会自动读取环境变量的参数，所以EnvironmentFile文件中有的参数，不需要再次在ExecStart启动参数中添加，二选一，如同时配置，会触发以下类似报错“etcd: conflicting environment variable "ETCD_NAME" is shadowed by corresponding command-line flag (either unset environment variable or disable flag)”
flannel操作etcd使用的是v2的API，而kubernetes操作etcd使用的v3的API

验证 etcd 集群状态

```
[root@localhost data]# etcdctl --endpoints=https://192.168.110.100:2379 --
cacert="/opt/ssl/ca.pem" --cert="/opt/ssl/etcd.pem" --key="/opt/ssl/etcd-
key.pem" endpoint health
https://192.168.110.100:2379 is healthy: successfully committed proposal: took =
16.533873ms
[root@localhost data]# etcdctl --endpoints=https://192.168.110.101:2379 --
cacert="/opt/ssl/ca.pem" --cert="/opt/ssl/etcd.pem" --key="/opt/ssl/etcd-
key.pem" endpoint health
https://192.168.110.101:2379 is healthy: successfully committed proposal: took =
24.19221ms
[root@localhost data]# etcdctl --endpoints=https://192.168.110.102:2379 --
cacert="/opt/ssl/ca.pem" --cert="/opt/ssl/etcd.pem" --key="/opt/ssl/etcd-
key.pem" endpoint health
https://192.168.110.102:2379 is healthy: successfully committed proposal: took =
20.12311ms
```

查看 etcd 集群成员

```
etcdctl --endpoints=https://192.168.110.100:2379 --cacert="/opt/ssl/ca.pem" --
cert="/opt/ssl/etcd.pem" --key="/opt/ssl/etcd-key.pem" member list
3e49d4614347212d, started, etcd1, https://192.168.110.100:2380,
https://192.168.110.100:2379, false
8d2730b8d7d6bbb3, started, etcd3, https://192.168.110.102:2380,
https://192.168.110.102:2379, false
f7edfa2ae51ff5e8, started, etcd2, https://192.168.110.101:2380,
https://192.168.110.101:2379, false
```

原文：https://www.cnblogs.com/xiexun/p/14620133.html