tags: etcd

04. 部署 etcd 集群

etcd 是基于 Raft 的分布式 KV 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 集群持久化存储所有 API 对象、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 集群之间的通信；
- 创建 etcd 的 systemd unit 文件，配置服务参数；
- 检查集群工作状态；

etcd 集群节点名称和 IP 如下：

- zhangjun-k8s-01：172.27.138.251
- zhangjun-k8s-02：172.27.137.229
- zhangjun-k8s-03：172.27.138.239

注意：

1. 如果没有特殊指明，本文档的所有操作**均在 zhangjun-k8s-01 节点上执行**；
2. flanneld 与本文档安装的 etcd v3.4.x 不兼容，如果要安装 flanneld（本文档使用 calio），则需要将 etcd 降级到 **v3.3.x** 版本；

下载和分发 etcd 二进制文件

到 etcd 的 release 页面 下载最新版本的发布包：

```
cd /opt/k8s/work
wget https://github.com/coreos/etcd/releases/download/v3.4.3/etcd-v3.4.3-linux-amd64.tar.gz
tar -xvf etcd-v3.4.3-linux-amd64.tar.gz
```

分发二进制文件到集群所有节点：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    scp etcd-v3.4.3-linux-amd64/etcd* root@${node_ip}:/opt/k8s/bin
    ssh root@${node_ip} "chmod +x /opt/k8s/bin/*"
  done
```

创建 etcd 证书和私钥

创建证书签名请求：

```
cd /opt/k8s/work
cat > etcd-csr.json <<EOF
{
  "CN": "etcd",
  "hosts": [
    "127.0.0.1",
    "172.27.138.251",
    "172.27.137.229",
    "172.27.138.239"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "opsnull"
    }
  ]
}
EOF
```

- `hosts`：指定授权使用该证书的 etcd 节点 IP 列表，需要将 **etcd** 集群所有节点 **IP** 都列在其中；

生成证书和私钥：

```
cd /opt/k8s/work
cfssl gencert -ca=/opt/k8s/work/ca.pem \
    -ca-key=/opt/k8s/work/ca-key.pem \
    -config=/opt/k8s/work/ca-config.json \
    -profile=kubernetes etcd-csr.json | cfssljson -bare etcd
ls etcd*pem
```

分发生成的证书和私钥到各 etcd 节点：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh root@${node_ip} "mkdir -p /etc/etcd/cert"
    scp etcd*.pem root@${node_ip}:/etc/etcd/cert/
  done
```

创建 etcd 的 systemd unit 模板文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > etcd.service.template <<EOF
[Unit]
Description=Etcd Server
After=network.target
After=network-online.target
Wants=network-online.target
Documentation=https://github.com/coreos

[Service]
Type=notify
WorkingDirectory=${ETCD_DATA_DIR}
ExecStart=/opt/k8s/bin/etcd \\
  --data-dir=${ETCD_DATA_DIR} \\
  --wal-dir=${ETCD_WAL_DIR} \\
  --name=##NODE_NAME## \\
  --cert-file=/etc/etcd/cert/etcd.pem \\
  --key-file=/etc/etcd/cert/etcd-key.pem \\
  --trusted-ca-file=/etc/kubernetes/cert/ca.pem \\
  --peer-cert-file=/etc/etcd/cert/etcd.pem \\
  --peer-key-file=/etc/etcd/cert/etcd-key.pem \\
  --peer-trusted-ca-file=/etc/kubernetes/cert/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --listen-peer-urls=https://##NODE_IP##:2380 \\
```

```
  --initial-advertise-peer-urls=https://##NODE_IP##:2380 \\
  --listen-client-urls=https://##NODE_IP##:2379,http://127.0.0.1:2379 \\
  --advertise-client-urls=https://##NODE_IP##:2379 \\
  --initial-cluster-token=etcd-cluster-0 \\
  --initial-cluster=${ETCD_NODES} \\
  --initial-cluster-state=new \\
  --auto-compaction-mode=periodic \\
  --auto-compaction-retention=1 \\
  --max-request-bytes=33554432 \\
  --quota-backend-bytes=6442450944 \\
  --heartbeat-interval=250 \\
  --election-timeout=2000
Restart=on-failure
RestartSec=5
LimitNOFILE=65536

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

- WorkingDirectory、--data-dir：指定工作目录和数据目录为 ${ETCD_DATA_DIR}，需在启动服务前创建这个目录；
- --wal-dir：指定 wal 目录，为了提高性能，一般使用 SSD 或者和 --data-dir 不同的磁盘；
- --name：指定节点名称，当 --initial-cluster-state 值为 new 时，--name 的参数值必须位于 --initial-cluster 列表中；
- --cert-file、--key-file：etcd server 与 client 通信时使用的证书和私钥；
- --trusted-ca-file：签名 client 证书的 CA 证书，用于验证 client 证书；
- --peer-cert-file、--peer-key-file：etcd 与 peer 通信使用的证书和私钥；
- --peer-trusted-ca-file：签名 peer 证书的 CA 证书，用于验证 peer 证书；

为各节点创建和分发 etcd systemd unit 文件

替换模板文件中的变量，为各节点创建 systemd unit 文件：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for (( i=0; i < 3; i++ ))
  do
    sed -e "s/##NODE_NAME##/${NODE_NAMES[i]}/" -e "s/##NODE_IP##/${NODE_IPS[i]}/"
etcd.service.template > etcd-${NODE_IPS[i]}.service
  done
ls *.service
```

- NODE_NAMES 和 NODE_IPS 为相同长度的 bash 数组，分别为节点名称和对应的 IP；

分发生成的 systemd unit 文件：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    scp etcd-${node_ip}.service root@${node_ip}:/etc/systemd/system/etcd.service
  done
```

启动 etcd 服务

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh root@${node_ip} "mkdir -p ${ETCD_DATA_DIR} ${ETCD_WAL_DIR}"
    ssh root@${node_ip} "systemctl daemon-reload && systemctl enable etcd &&
systemctl restart etcd " &
  done
```

- 必须先创建 etcd 数据目录和工作目录;
- etcd 进程首次启动时会等待其它节点的 etcd 加入集群，命令 `systemctl start etcd` 会卡住一段时间，为正常现象;

检查启动结果

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh root@${node_ip} "systemctl status etcd|grep Active"
  done
```

确保状态为 `active (running)`，否则查看日志，确认原因:

```
journalctl -u etcd
```

验证服务状态

部署完 etcd 集群后，在任一 etcd 节点上执行如下命令:

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    /opt/k8s/bin/etcdctl \
    --endpoints=https://${node_ip}:2379 \
    --cacert=/etc/kubernetes/cert/ca.pem \
    --cert=/etc/etcd/cert/etcd.pem \
    --key=/etc/etcd/cert/etcd-key.pem endpoint health
  done
```

- 3.4.3 版本的 etcd/etcdctl 默认启用了 V3 API，所以执行 etcdctl 命令时不需要再指定环境变量 `ETCDCTL_API=3`；
- 从 K8S 1.13 开始，不再支持 v2 版本的 etcd；

预期输出：

```
>>> 172.27.138.251
https://172.27.138.251:2379 is healthy: successfully committed proposal: took =
2.756451ms
>>> 172.27.137.229
https://172.27.137.229:2379 is healthy: successfully committed proposal: took =
2.025018ms
>>> 172.27.138.239
https://172.27.138.239:2379 is healthy: successfully committed proposal: took =
2.335097ms
```

输出均为 `healthy` 时表示集群服务正常。

查看当前的 leader

```
source /opt/k8s/bin/environment.sh
/opt/k8s/bin/etcdctl \
  -w table --cacert=/etc/kubernetes/cert/ca.pem \
  --cert=/etc/etcd/cert/etcd.pem \
  --key=/etc/etcd/cert/etcd-key.pem \
  --endpoints=${ETCD_ENDPOINTS} endpoint status
```

输出：

```
+----------------------------+------------------+---------+---------+-----------+---
---------+-----------+------------+--------------------+--------+
|          ENDPOINT          |        ID        | VERSION | DB SIZE | IS LEADER | IS
LEARNER | RAFT TERM | RAFT INDEX | RAFT APPLIED INDEX | ERRORS |
+----------------------------+------------------+---------+---------+-----------+---
---------+-----------+------------+--------------------+--------+
| https://172.27.138.251:2379 | 4250b255e93e0076 |   3.4.3 |   20 kB |     false |
   false |         2 |          8 |                  8 |        |
| https://172.27.137.229:2379 | b3d912e6166f1213 |   3.4.3 |   20 kB |      true |
   false |         2 |          8 |                  8 |        |
| https://172.27.138.239:2379 | 8a4d4a2904de8446 |   3.4.3 |   20 kB |     false |
   false |         2 |          8 |                  8 |        |
+----------------------------+------------------+---------+---------+-----------+---
---------+-----------+------------+--------------------+--------+
```

- 可见，当前的 leader 为 172.27.138.229。