tags: flanneld

E. 部署 flannel 网络

kubernetes 要求集群内各节点(包括 master 节点)能通过 Pod 网段互联互通。flannel 使用 vxlan 技术为各节点创建一个可以互通的 Pod 网络，使用的端口为 UDP 8472（需要开放该端口，如公有云 AWS 等）。

flanneld 第一次启动时，从 etcd 获取配置的 Pod 网段信息，为本节点分配一个未使用的地址段，然后创建 `flannedl.1` 网络接口（也可能是其它名称，如 flannel1 等）。

flannel 将分配给自己的 Pod 网段信息写入 `/run/flannel/docker` 文件，docker 后续使用这个文件中的环境变量设置 `docker0` 网桥，从而从这个地址段为本节点的所有 Pod 容器分配 IP。

注意：

1. 如果没有特殊指明，本文档的所有操作**均在 zhangjun-k8s01 节点上执行**，然后远程分发文件和执行命令；
2. flanneld 与本文档部署的 etcd v3.4.x 不兼容，需要将 etcd 降级到 v3.3.x；
3. flanneld 与 docker 结合使用；

下载和分发 flanneld 二进制文件

从 flannel 的 release 页面 下载最新版本的安装包：

```
cd /opt/k8s/work
mkdir flannel
wget https://github.com/coreos/flannel/releases/download/v0.11.0/flannel-v0.11.0-linux-amd64.tar.gz
tar -xzvf flannel-v0.11.0-linux-amd64.tar.gz -C flannel
```

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 ">>> ${node_ip}"
    scp flannel/{flanneld,mk-docker-opts.sh} root@${node_ip}:/opt/k8s/bin/
    ssh root@${node_ip} "chmod +x /opt/k8s/bin/*"
  done
```

创建 flannel 证书和私钥

flanneld 从 etcd 集群存取网段分配信息，而 etcd 集群启用了双向 x509 证书认证，所以需要为 flanneld 生成证书和私钥。

创建证书签名请求：

```
cd /opt/k8s/work
cat > flanneld-csr.json <<EOF
{
  "CN": "flanneld",
  "hosts": [],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "4Paradigm"
    }
  ]
}
EOF
```

- 该证书只会被 kubectl 当做 client 证书使用，所以 hosts 字段为空；

生成证书和私钥：

```
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 flanneld-csr.json | cfssljson -bare flanneld
ls flanneld*pem
```

将生成的证书和私钥分发到所有节点（master 和 worker）：

```
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/flanneld/cert"
    scp flanneld*.pem root@${node_ip}:/etc/flanneld/cert
  done
```

向 etcd 写入集群 Pod 网段信息

注意：本步骤只需执行一次。

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
etcdctl \
  --endpoints=${ETCD_ENDPOINTS} \
  --ca-file=/opt/k8s/work/ca.pem \
  --cert-file=/opt/k8s/work/flanneld.pem \
  --key-file=/opt/k8s/work/flanneld-key.pem \
  mk ${FLANNEL_ETCD_PREFIX}/config '{"Network":"'${CLUSTER_CIDR}'", "SubnetLen": 21,
"Backend": {"Type": "vxlan"}}'
```

- flanneld 当前版本 **(v0.11.0)** 不支持 **etcd v3**，故使用 etcd v2 API 写入配置 key 和网段数据；
- 写入的 Pod 网段 `${CLUSTER_CIDR}` 地址段（如 /16）必须小于 `SubnetLen`，必须与 `kube-controller-manager` 的 `--cluster-cidr` 参数值一致；

创建 flanneld 的 systemd unit 文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > flanneld.service << EOF
[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
ExecStart=/opt/k8s/bin/flanneld \\
  -etcd-cafile=/etc/kubernetes/cert/ca.pem \\
  -etcd-certfile=/etc/flanneld/cert/flanneld.pem \\
```

```
  -etcd-keyfile=/etc/flanneld/cert/flanneld-key.pem \\
  -etcd-endpoints=${ETCD_ENDPOINTS} \\
  -etcd-prefix=${FLANNEL_ETCD_PREFIX} \\
  -iface=${IFACE} \\
  -ip-masq
ExecStartPost=/opt/k8s/bin/mk-docker-opts.sh -k DOCKER_NETWORK_OPTIONS -d
/run/flannel/docker
Restart=always
RestartSec=5
StartLimitInterval=0

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

- `mk-docker-opts.sh` 脚本将分配给 flanneld 的 Pod 子网段信息写入 `/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；
- flanneld 运行时需要 root 权限；
- `-ip-masq`: flanneld 为访问 Pod 网络外的流量设置 SNAT 规则，同时将传递给 Docker 的变量 `--ip-masq`（`/run/flannel/docker` 文件中）设置为 false，这样 Docker 将不再创建 SNAT 规则；
  Docker 的 `--ip-masq` 为 true 时，创建的 SNAT 规则比较“暴力”：将所有本节点 Pod 发起的、访问非 docker0 接口的请求做 SNAT，这样访问其他节点 Pod 的请求来源 IP 会被设置为 flannel.1 接口的 IP，导致目的 Pod 看不到真实的来源 Pod IP。
  flanneld 创建的 SNAT 规则比较温和，只对访问非 Pod 网段的请求做 SNAT。

分发 flanneld systemd unit 文件到所有节点

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

启动 flanneld 服务

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

检查启动结果

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

确保状态为 active (running)，否则查看日志，确认原因：

```
journalctl -u flanneld
```

检查分配给各 flanneld 的 Pod 网段信息

查看集群 Pod 网段(/16)：

```
source /opt/k8s/bin/environment.sh
etcdctl \
  --endpoints=${ETCD_ENDPOINTS} \
  --ca-file=/etc/kubernetes/cert/ca.pem \
  --cert-file=/etc/flanneld/cert/flanneld.pem \
  --key-file=/etc/flanneld/cert/flanneld-key.pem \
  get ${FLANNEL_ETCD_PREFIX}/config
```

输出：

```
{"Network":"172.30.0.0/16", "SubnetLen": 24, "Backend": {"Type": "vxlan"}}
```

查看已分配的 Pod 子网段列表(/24)：

```
source /opt/k8s/bin/environment.sh
etcdctl \
  --endpoints=${ETCD_ENDPOINTS} \
  --ca-file=/etc/kubernetes/cert/ca.pem \
  --cert-file=/etc/flanneld/cert/flanneld.pem \
  --key-file=/etc/flanneld/cert/flanneld-key.pem \
  ls ${FLANNEL_ETCD_PREFIX}/subnets
```

输出（结果视部署情况而定）：

```
/kubernetes/network/subnets/172.30.80.0-24
/kubernetes/network/subnets/172.30.32.0-24
/kubernetes/network/subnets/172.30.184.0-24
```

查看某一 Pod 网段对应的节点 IP 和 flannel 接口地址:

```
source /opt/k8s/bin/environment.sh
etcdctl \
  --endpoints=${ETCD_ENDPOINTS} \
  --ca-file=/etc/kubernetes/cert/ca.pem \
  --cert-file=/etc/flanneld/cert/flanneld.pem \
  --key-file=/etc/flanneld/cert/flanneld-key.pem \
  get ${FLANNEL_ETCD_PREFIX}/subnets/172.30.80.0-24
```

输出（结果视部署情况而定）：

```
{"PublicIP":"172.27.137.240","BackendType":"vxlan","BackendData":
{"VtepMAC":"ce:9c:a9:08:50:03"}}
```

- 172.30.80.0/21 被分配给节点 zhangjun-k8s01（172.27.137.240）；
- VtepMAC 为 zhangjun-k8s01 节点的 flannel.1 网卡 MAC 地址；

检查节点 flannel 网络信息

```
[root@zhangjun-k8s01 work]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    link/ether 00:22:0d:33:89:75 brd ff:ff:ff:ff:ff:ff
    inet 172.27.137.240/20 brd 172.27.143.255 scope global dynamic eth0
       valid_lft 100647283sec preferred_lft 100647283sec
3: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN
group default
    link/ether ce:9c:a9:08:50:03 brd ff:ff:ff:ff:ff:ff
    inet 172.30.80.0/32 scope global flannel.1
       valid_lft forever preferred_lft forever
```

- flannel.1 网卡的地址为分配的 Pod 子网段的第一个 IP（.0），且是 /32 的地址；

```
[root@zhangjun-k8s01 work]# ip route show |grep flannel.1
172.30.32.0/24 via 172.30.32.0 dev flannel.1 onlink
172.30.184.0/24 via 172.30.184.0 dev flannel.1 onlink
```

- 到其它节点 Pod 网段请求都被转发到 flannel.1 网卡；
- flanneld 根据 etcd 中子网段的信息，如 `${FLANNEL_ETCD_PREFIX}/subnets/172.30.80.0-24` ，来决定进请求发送给哪个节点的互联 IP；

验证各节点能通过 Pod 网段互通

在各节点上部署 flannel 后，检查是否创建了 flannel 接口(名称可能为 flannel0、flannel.0、flannel.1 等)：

```
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh ${node_ip} "/usr/sbin/ip addr show flannel.1|grep -w inet"
  done
```

输出：

```
>>> 172.27.137.240
    inet 172.30.80.0/32 scope global flannel.1
>>> 172.27.137.239
    inet 172.30.32.0/32 scope global flannel.1
>>> 172.27.137.238
    inet 172.30.184.0/32 scope global flannel.1
```

在各节点上 ping 所有 flannel 接口 IP，确保能通：

```
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh ${node_ip} "ping -c 1 172.30.80.0"
    ssh ${node_ip} "ping -c 1 172.30.32.0"
    ssh ${node_ip} "ping -c 1 172.30.184.0"
  done
```