tags: worker, kube-proxy

06-5. 部署 kube-proxy 组件

kube-proxy 运行在所有 worker 节点上，它监听 apiserver 中 service 和 endpoint 的变化情况，创建路由规则以提供服务 IP 和负载均衡功能。

本文档讲解部署 ipvs 模式的 kube-proxy 过程。

注意：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，然后远程分发文件和执行命令。

下载和分发 kube-proxy 二进制文件

参考 05-1.部署master节点.md。

创建 kube-proxy 证书

创建证书签名请求：

```
cd /opt/k8s/work
cat > kube-proxy-csr.json <<EOF
{
  "CN": "system:kube-proxy",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
```

```
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "opsnull"
    }
  ]
}
EOF
```

- CN：指定该证书的 User 为 system:kube-proxy；
- 预定义的 RoleBinding system:node-proxier 将User system:kube-proxy 与 Role system:node-proxier 绑定，该 Role 授予了调用 kube-apiserver Proxy 相关 API 的权限；
- 该证书只会被 kube-proxy 当做 client 证书使用，所以 hosts 字段为空；

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  kube-proxy-csr.json | cfssljson -bare kube-proxy
ls kube-proxy*
```

创建和分发 kubeconfig 文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
kubectl config set-cluster kubernetes \
  --certificate-authority=/opt/k8s/work/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=kube-proxy.kubeconfig

kubectl config set-credentials kube-proxy \
  --client-certificate=kube-proxy.pem \
  --client-key=kube-proxy-key.pem \
  --embed-certs=true \
  --kubeconfig=kube-proxy.kubeconfig

kubectl config set-context default \
  --cluster=kubernetes \
  --user=kube-proxy \
  --kubeconfig=kube-proxy.kubeconfig

kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig
```

分发 kubeconfig 文件：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_name in ${NODE_NAMES[@]}
  do
    echo ">>> ${node_name}"
    scp kube-proxy.kubeconfig root@${node_name}:/etc/kubernetes/
  done
```

创建 kube-proxy 配置文件

从 v1.10 开始，kube-proxy 部分参数可以配置文件中配置。可以使用 `--write-config-to` 选项生成该配置文件，或者参考 源代码的注释。

创建 kube-proxy config 文件模板：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > kube-proxy-config.yaml.template <<EOF
kind: KubeProxyConfiguration
apiVersion: kubeproxy.config.k8s.io/v1alpha1
clientConnection:
  burst: 200
  kubeconfig: "/etc/kubernetes/kube-proxy.kubeconfig"
  qps: 100
bindAddress: ##NODE_IP##
healthzBindAddress: ##NODE_IP##:10256
metricsBindAddress: ##NODE_IP##:10249
enableProfiling: true
clusterCIDR: ${CLUSTER_CIDR}
hostnameOverride: ##NODE_NAME##
mode: "ipvs"
portRange: ""
iptables:
  masqueradeAll: false
ipvs:
  scheduler: rr
  excludeCIDRs: []
EOF
```

- `bindAddress`: 监听地址；
- `clientConnection.kubeconfig`: 连接 apiserver 的 kubeconfig 文件；
- `clusterCIDR`: kube-proxy 根据 `--cluster-cidr` 判断集群内部和外部流量，指定 `--cluster-cidr` 或 `--masquerade-all` 选项后 kube-proxy 才会对访问 Service IP 的请求做 SNAT；
- `hostnameOverride`: 参数值必须与 kubelet 的值一致，否则 kube-proxy 启动后会找不到该 Node，从而不会创建任何 ipvs 规则；
- `mode`: 使用 ipvs 模式；

为各节点创建和分发 kube-proxy 配置文件：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for (( i=0; i < 3; i++ ))
  do
    echo ">>> ${NODE_NAMES[i]}"
    sed -e "s/##NODE_NAME##/${NODE_NAMES[i]}/" -e "s/##NODE_IP##/${NODE_IPS[i]}/"
kube-proxy-config.yaml.template > kube-proxy-config-${NODE_NAMES[i]}.yaml.template
    scp kube-proxy-config-${NODE_NAMES[i]}.yaml.template
root@${NODE_NAMES[i]}:/etc/kubernetes/kube-proxy-config.yaml
  done
```

创建和分发 kube-proxy systemd unit 文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > kube-proxy.service <<EOF
[Unit]
Description=Kubernetes Kube-Proxy Server
Documentation=https://github.com/GoogleCloudPlatform/kubernetes
After=network.target

[Service]
WorkingDirectory=${K8S_DIR}/kube-proxy
ExecStart=/opt/k8s/bin/kube-proxy \\
  --config=/etc/kubernetes/kube-proxy-config.yaml \\
  --logtostderr=true \\
  --v=2
Restart=on-failure
RestartSec=5
LimitNOFILE=65536

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

分发 kube-proxy systemd unit 文件：

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

启动 kube-proxy 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 ${K8S_DIR}/kube-proxy"
    ssh root@${node_ip} "modprobe ip_vs_rr"
    ssh root@${node_ip} "systemctl daemon-reload && systemctl enable kube-proxy &&
systemctl restart kube-proxy"
  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 kube-proxy|grep Active"
  done
```

确保状态为 `active (running)`，否则查看日志，确认原因：

```
journalctl -u kube-proxy
```

查看监听端口

```
$ sudo netstat -lnpt|grep kube-prox
tcp        0      0 172.27.138.251:10256    0.0.0.0:*               LISTEN
 30590/kube-proxy
tcp        0      0 172.27.138.251:10249    0.0.0.0:*               LISTEN
 30590/kube-proxy
```

- 10249：http prometheus metrics port;
- 10256：http healthz port;

查看 ipvs 路由规则

```
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh root@${node_ip} "/usr/sbin/ipvsadm -ln"
  done
```

预期输出：

```
>>> 172.27.138.251
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.254.0.1:443 rr
  -> 172.27.137.229:6443          Masq    1      0          0
  -> 172.27.138.239:6443          Masq    1      0          0
  -> 172.27.138.251:6443          Masq    1      0          0
>>> 172.27.137.229
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.254.0.1:443 rr
  -> 172.27.137.229:6443          Masq    1      0          0
  -> 172.27.138.239:6443          Masq    1      0          0
  -> 172.27.138.251:6443          Masq    1      0          0
>>> 172.27.138.239
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.254.0.1:443 rr
  -> 172.27.137.229:6443          Masq    1      0          0
  -> 172.27.138.239:6443          Masq    1      0          0
  -> 172.27.138.251:6443          Masq    1      0          0
```

可见所有通过 https 访问 K8S SVC kubernetes 的请求都转发到 kube-apiserver 节点的 6443 端口；