tags: master, kube-scheduler

05-4. 部署高可用 kube-scheduler 集群

本文档介绍部署高可用 kube-scheduler 集群的步骤。

该集群包含 3 个节点，启动后将通过竞争选举机制产生一个 leader 节点，其它节点为阻塞状态。当 leader 节点不可用后，剩余节点将再次进行选举产生新的 leader 节点，从而保证服务的可用性。

为保证通信安全，本文档先生成 x509 证书和私钥，kube-scheduler 在如下两种情况下使用该证书：

1. 与 kube-apiserver 的安全端口通信;
2. 在安全端口(https，10251) 输出 prometheus 格式的 metrics；

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。

创建 kube-scheduler 证书和私钥

创建证书签名请求：

```
cd /opt/k8s/work
cat > kube-scheduler-csr.json <<EOF
{
    "CN": "system:kube-scheduler",
    "hosts": [
      "127.0.0.1",
      "172.27.138.239",
      "172.27.137.229",
      "172.27.138.251"
    ],
```

```
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
      {
        "C": "CN",
        "ST": "BeiJing",
        "L": "BeiJing",
        "O": "system:kube-scheduler",
        "OU": "opsnull"
      }
    ]
}
EOF
```

- hosts 列表包含所有 kube-scheduler 节点 IP；
- CN 和 O 均为 `system:kube-scheduler`，kubernetes 内置的 ClusterRoleBindings `system:kube-scheduler` 将赋予 kube-scheduler 工作所需的权限；

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-scheduler-csr.json | cfssljson -bare kube-scheduler
ls kube-scheduler*pem
```

将生成的证书和私钥分发到所有 master 节点：

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

创建和分发 kubeconfig 文件

kube-scheduler 使用 kubeconfig 文件访问 apiserver，该文件提供了 apiserver 地址、嵌入的 CA 证书和 kube-scheduler 证书：

```
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="https://##NODE_IP##:6443" \
  --kubeconfig=kube-scheduler.kubeconfig

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

kubectl config set-context system:kube-scheduler \
  --cluster=kubernetes \
  --user=system:kube-scheduler \
  --kubeconfig=kube-scheduler.kubeconfig

kubectl config use-context system:kube-scheduler --kubeconfig=kube-
scheduler.kubeconfig
```

分发 kubeconfig 到所有 master 节点：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    sed -e "s/##NODE_IP##/${node_ip}/" kube-scheduler.kubeconfig > kube-scheduler-
${node_ip}.kubeconfig
    scp kube-scheduler-${node_ip}.kubeconfig root@${node_ip}:/etc/kubernetes/kube-
scheduler.kubeconfig
  done
```

创建 kube-scheduler 配置文件

```
cd /opt/k8s/work
cat >kube-scheduler.yaml.template <<EOF
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
bindTimeoutSeconds: 600
clientConnection:
  burst: 200
  kubeconfig: "/etc/kubernetes/kube-scheduler.kubeconfig"
  qps: 100
enableContentionProfiling: false
enableProfiling: true
hardPodAffinitySymmetricWeight: 1
healthzBindAddress: ##NODE_IP##:10251
leaderElection:
```

```
  leaderElect: true
metricsBindAddress: ##NODE_IP##:10251
EOF
```

- `--kubeconfig`：指定 kubeconfig 文件路径，kube-scheduler 使用它连接和验证 kube-apiserver；
- `--leader-elect=true`：集群运行模式，启用选举功能；被选为 leader 的节点负责处理工作，其它节点为阻塞状态；

替换模板文件中的变量：

```
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]}/"
kube-scheduler.yaml.template > kube-scheduler-${NODE_IPS[i]}.yaml
  done
ls kube-scheduler*.yaml
```

- 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 kube-scheduler 配置文件到所有 master 节点：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    scp kube-scheduler-${node_ip}.yaml root@${node_ip}:/etc/kubernetes/kube-
scheduler.yaml
  done
```

- 重命名为 kube-scheduler.yaml;

创建 kube-scheduler systemd unit 模板文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > kube-scheduler.service.template <<EOF
[Unit]
Description=Kubernetes Scheduler
Documentation=https://github.com/GoogleCloudPlatform/kubernetes

[Service]
WorkingDirectory=${K8S_DIR}/kube-scheduler
ExecStart=/opt/k8s/bin/kube-scheduler \\
  --config=/etc/kubernetes/kube-scheduler.yaml \\
```

```
  --bind-address=##NODE_IP## \\
  --secure-port=10259 \\
  --port=0 \\
  --tls-cert-file=/etc/kubernetes/cert/kube-scheduler.pem \\
  --tls-private-key-file=/etc/kubernetes/cert/kube-scheduler-key.pem \\
  --authentication-kubeconfig=/etc/kubernetes/kube-scheduler.kubeconfig \\
  --client-ca-file=/etc/kubernetes/cert/ca.pem \\
  --requestheader-allowed-names="" \\
  --requestheader-client-ca-file=/etc/kubernetes/cert/ca.pem \\
  --requestheader-extra-headers-prefix="X-Remote-Extra-" \\
  --requestheader-group-headers=X-Remote-Group \\
  --requestheader-username-headers=X-Remote-User \\
  --authorization-kubeconfig=/etc/kubernetes/kube-scheduler.kubeconfig \\
  --logtostderr=true \\
  --v=2
Restart=always
RestartSec=5
StartLimitInterval=0

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

为各节点创建和分发 kube-scheduler 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]}/"
kube-scheduler.service.template > kube-scheduler-${NODE_IPS[i]}.service
  done
ls kube-scheduler*.service
```

分发 systemd unit 文件到所有 master 节点：

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

启动 kube-scheduler 服务

```
source /opt/k8s/bin/environment.sh
for node_ip in ${NODE_IPS[@]}
  do
    echo ">>> ${node_ip}"
    ssh root@${node_ip} "mkdir -p ${K8S_DIR}/kube-scheduler"
    ssh root@${node_ip} "systemctl daemon-reload && systemctl enable kube-scheduler
&& systemctl restart kube-scheduler"
  done
```

检查服务运行状态

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

确保状态为 active (running)，否则查看日志，确认原因：

```
journalctl -u kube-scheduler
```

查看输出的 metrics

注意：以下命令在 kube-scheduler 节点上执行。

kube-scheduler 监听 10251 和 10259 端口：

- 10251：接收 http 请求，非安全端口，不需要认证授权；
- 10259：接收 https 请求，安全端口，需要认证授权；

两个接口都对外提供 /metrics 和 /healthz 的访问。

```
$ sudo netstat -lnpt |grep kube-sch
tcp        0      0 172.27.138.251:10251    0.0.0.0:*               LISTEN
114702/kube-schedul
tcp        0      0 172.27.138.251:10259    0.0.0.0:*               LISTEN
114702/kube-schedul
```

```
$ curl -s http://172.27.138.251:10251/metrics |head
# HELP apiserver_audit_event_total Counter of audit events generated and sent to the
audit backend.
# TYPE apiserver_audit_event_total counter
apiserver_audit_event_total 0
# HELP apiserver_audit_requests_rejected_total Counter of apiserver requests rejected
due to an error in audit logging backend.
# TYPE apiserver_audit_requests_rejected_total counter
apiserver_audit_requests_rejected_total 0
# HELP apiserver_client_certificate_expiration_seconds Distribution of the remaining
lifetime on the certificate used to authenticate a request.
# TYPE apiserver_client_certificate_expiration_seconds histogram
apiserver_client_certificate_expiration_seconds_bucket{le="0"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="1800"} 0
```

```
$ curl -s --cacert /opt/k8s/work/ca.pem --cert /opt/k8s/work/admin.pem --key
/opt/k8s/work/admin-key.pem https://172.27.138.251:10259/metrics |head
# HELP apiserver_audit_event_total Counter of audit events generated and sent to the
audit backend.
# TYPE apiserver_audit_event_total counter
apiserver_audit_event_total 0
# HELP apiserver_audit_requests_rejected_total Counter of apiserver requests rejected
due to an error in audit logging backend.
# TYPE apiserver_audit_requests_rejected_total counter
apiserver_audit_requests_rejected_total 0
# HELP apiserver_client_certificate_expiration_seconds Distribution of the remaining
lifetime on the certificate used to authenticate a request.
# TYPE apiserver_client_certificate_expiration_seconds histogram
apiserver_client_certificate_expiration_seconds_bucket{le="0"} 0
apiserver_client_certificate_expiration_seconds_bucket{le="1800"} 0
```

查看当前的 leader

```
$ kubectl get endpoints kube-scheduler --namespace=kube-system  -o yaml
apiVersion: v1
kind: Endpoints
metadata:
  annotations:
    control-plane.alpha.kubernetes.io/leader: '{"holderIdentity":"zhangjun-k8s-
01_ce04632e-64e4-477e-b8f0-
4e69020cd996","leaseDurationSeconds":15,"acquireTime":"2020-02-
07T07:05:00Z","renewTime":"2020-02-07T07:05:28Z","leaderTransitions":0}'
  creationTimestamp: "2020-02-07T07:05:00Z"
  name: kube-scheduler
  namespace: kube-system
  resourceVersion: "756"
  selfLink: /api/v1/namespaces/kube-system/endpoints/kube-scheduler
  uid: 1b687724-a6e2-4404-9efb-a1f0e201fecc
```

可见，当前的 leader 为 zhangjun-k8s-01 节点。

测试 kube-scheduler 集群的高可用

随便找一个或两个 master 节点，停掉 kube-scheduler 服务，看其它节点是否获取了 leader 权限。