tags: master, kube-controller-manager

05-3. 部署高可用 kube-controller-manager 集群

- 05-3. 部署高可用 kube-controller-manager 集群
  - 创建 kube-controller-manager 证书和私钥
  - 创建和分发 kubeconfig 文件
  - 创建 kube-controller-manager systemd unit 模板文件
  - 为各节点创建和分发 kube-controller-mananger systemd unit 文件
  - 启动 kube-controller-manager 服务
  - 检查服务运行状态
  - 查看输出的 metrics
  - 查看当前的 leader
  - 测试 kube-controller-manager 集群的高可用
  - 参考

本文档介绍部署高可用 kube-controller-manager 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 节点不可用时，阻塞的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-controller-manager 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，10252) 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-controller-manager 证书和私钥

创建证书签名请求：

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

```
    },
    "hosts": [
      "127.0.0.1",
      "172.27.138.251",
      "172.27.137.229",
      "172.27.138.239"
    ],
    "names": [
      {
        "C": "CN",
        "ST": "BeiJing",
        "L": "BeiJing",
        "O": "system:kube-controller-manager",
        "OU": "opsnull"
      }
    ]
}
EOF
```

- hosts 列表包含所有 kube-controller-manager 节点 IP；
- CN 和 O 均为 `system:kube-controller-manager`，kubernetes 内置的 ClusterRoleBindings `system:kube-controller-manager` 赋予 kube-controller-manager 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-controller-manager-csr.json | cfssljson -bare kube-controller-manager
ls kube-controller-manager*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 ">>> ${node_ip}"
    scp kube-controller-manager*.pem root@${node_ip}:/etc/kubernetes/cert/
  done
```

创建和分发 kubeconfig 文件

kube-controller-manager 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-controller-manager 证书等信息：

```
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-controller-manager.kubeconfig

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

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

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

- kube-controller-manager 与 kube-apiserver 混布，故直接通过节点 **IP** 访问 kube-apiserver；

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-controller-manager.kubeconfig > kube-
controller-manager-${node_ip}.kubeconfig
    scp kube-controller-manager-${node_ip}.kubeconfig
root@${node_ip}:/etc/kubernetes/kube-controller-manager.kubeconfig
  done
```

创建 kube-controller-manager systemd unit 模板文件

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

[Service]
WorkingDirectory=${K8S_DIR}/kube-controller-manager
```

```
ExecStart=/opt/k8s/bin/kube-controller-manager \\
  --profiling \\
  --cluster-name=kubernetes \\
  --controllers=*,bootstrapsigner,tokencleaner \\
  --kube-api-qps=1000 \\
  --kube-api-burst=2000 \\
  --leader-elect \\
  --use-service-account-credentials\\
  --concurrent-service-syncs=2 \\
  --bind-address=##NODE_IP## \\
  --secure-port=10252 \\
  --tls-cert-file=/etc/kubernetes/cert/kube-controller-manager.pem \\
  --tls-private-key-file=/etc/kubernetes/cert/kube-controller-manager-key.pem \\
  --port=0 \\
  --authentication-kubeconfig=/etc/kubernetes/kube-controller-manager.kubeconfig \\
  --client-ca-file=/etc/kubernetes/cert/ca.pem \\
  --requestheader-allowed-names="aggregator" \\
  --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-controller-manager.kubeconfig \\
  --cluster-signing-cert-file=/etc/kubernetes/cert/ca.pem \\
  --cluster-signing-key-file=/etc/kubernetes/cert/ca-key.pem \\
  --experimental-cluster-signing-duration=876000h \\
  --horizontal-pod-autoscaler-sync-period=10s \\
  --concurrent-deployment-syncs=10 \\
  --concurrent-gc-syncs=30 \\
  --node-cidr-mask-size=24 \\
  --service-cluster-ip-range=${SERVICE_CIDR} \\
  --pod-eviction-timeout=6m \\
  --terminated-pod-gc-threshold=10000 \\
  --root-ca-file=/etc/kubernetes/cert/ca.pem \\
  --service-account-private-key-file=/etc/kubernetes/cert/ca-key.pem \\
  --kubeconfig=/etc/kubernetes/kube-controller-manager.kubeconfig \\
  --logtostderr=true \\
  --v=2
Restart=on-failure
RestartSec=5

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

- `--port=0`：关闭监听非安全端口（http），同时 `--address` 参数无效，`--bind-address` 参数有效；
- `--secure-port=10252`、`--bind-address=0.0.0.0`: 在所有网络接口监听 10252 端口的 https /metrics 请求；
- `--kubeconfig`：指定 kubeconfig 文件路径，kube-controller-manager 使用它连接和验证 kube-apiserver；
- `--authentication-kubeconfig` 和 `--authorization-kubeconfig`：kube-controller-manager 使用它连接 apiserver，对 client 的请求进行认证和授权。`kube-controller-manager` 不再使用 `--tls-ca-file` 对请求 https metrics 的 Client 证书进行校验。如果没有配置这两个 kubeconfig 参数，则 client 连接 kube-controller-

manager https 端口的请求会被拒绝(提示权限不足)。

- `--cluster-signing-*-file`：签名 TLS Bootstrap 创建的证书；
- `--experimental-cluster-signing-duration`：指定 TLS Bootstrap 证书的有效期；
- `--root-ca-file`：放置到容器 ServiceAccount 中的 CA 证书，用来对 kube-apiserver 的证书进行校验；
- `--service-account-private-key-file`：签名 ServiceAccount 中 Token 的私钥文件，必须和 kube-apiserver 的 `--service-account-key-file` 指定的公钥文件配对使用；
- `--service-cluster-ip-range`：指定 Service Cluster IP 网段，必须和 kube-apiserver 中的同名参数一致；
- `--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；
- `--controllers=*,bootstrapsigner,tokencleaner`：启用的控制器列表，tokencleaner 用于自动清理过期的 Bootstrap token；
- `--horizontal-pod-autoscaler-*`：custom metrics 相关参数，支持 autoscaling/v2alpha1；
- `--tls-cert-file`、`--tls-private-key-file`：使用 https 输出 metrics 时使用的 Server 证书和秘钥；
- `--use-service-account-credentials=true`: kube-controller-manager 中各 controller 使用 serviceaccount 访问 kube-apiserver；

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

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 ">>> ${node_ip}"
    scp kube-controller-manager-${node_ip}.service
root@${node_ip}:/etc/systemd/system/kube-controller-manager.service
  done
```

启动 kube-controller-manager 服务

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

确保状态为 active (running)，否则查看日志，确认原因：

```
journalctl -u kube-controller-manager
```

kube-controller-manager 监听 10252 端口，接收 https 请求：

```
$ sudo netstat -lnpt | grep kube-cont
tcp        0      0 172.27.138.251:10252    0.0.0.0:*               LISTEN
 108977/kube-control
```

查看输出的 metrics

注意：以下命令在 kube-controller-manager 节点上执行。

```
$ curl -s --cacert /opt/k8s/work/ca.pem --cert /opt/k8s/work/admin.pem --key
/opt/k8s/work/admin-key.pem https://172.27.138.251:10252/metrics |head
# HELP ClusterRoleAggregator_adds (Deprecated) Total number of adds handled by
workqueue: ClusterRoleAggregator
# TYPE ClusterRoleAggregator_adds counter
ClusterRoleAggregator_adds 3
# HELP ClusterRoleAggregator_depth (Deprecated) Current depth of workqueue:
ClusterRoleAggregator
# TYPE ClusterRoleAggregator_depth gauge
ClusterRoleAggregator_depth 0
# HELP ClusterRoleAggregator_longest_running_processor_microseconds (Deprecated) How
many microseconds has the longest running processor for ClusterRoleAggregator been
running.
# TYPE ClusterRoleAggregator_longest_running_processor_microseconds gauge
ClusterRoleAggregator_longest_running_processor_microseconds 0
# HELP ClusterRoleAggregator_queue_latency (Deprecated) How long an item stays in
workqueueClusterRoleAggregator before being requested.
```

查看当前的 leader

```
$ kubectl get endpoints kube-controller-manager --namespace=kube-system  -o yaml
apiVersion: v1
kind: Endpoints
metadata:
  annotations:
    control-plane.alpha.kubernetes.io/leader: '{"holderIdentity":"zhangjun-k8s-
03_e334e88d-6b52-40e0-b2a1-
a6f7e47593e1","leaseDurationSeconds":15,"acquireTime":"2020-02-
07T07:01:32Z","renewTime":"2020-02-07T07:01:44Z","leaderTransitions":1}'
  creationTimestamp: "2020-02-07T06:59:38Z"
  name: kube-controller-manager
  namespace: kube-system
  resourceVersion: "561"
  selfLink: /api/v1/namespaces/kube-system/endpoints/kube-controller-manager
  uid: e5d52a8c-fe69-4910-a125-d7ec97cead16
```

可见，当前的 leader 为 zhangjun-k8s-03 节点。

测试 kube-controller-manager 集群的高可用

停掉一个或两个节点的 kube-controller-manager 服务，观察其它节点的日志，看是否获取了 leader 权限。

参考

1. 关于 controller 权限和 use-service-account-credentials 参数：https://github.com/kubernetes/kubernetes/issues/48208
2. kubelet 认证和授权：https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authorization