tags: master, kube-apiserver

05-2. 部署 kube-apiserver 集群

本文档讲解部署一个三实例 kube-apiserver 集群的步骤.

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 kubernetes-master 证书和私钥

创建证书签名请求：

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > kubernetes-csr.json <<EOF
{
  "CN": "kubernetes-master",
  "hosts": [
    "127.0.0.1",
    "172.27.138.251",
    "172.27.137.229",
    "172.27.138.239",
    "${CLUSTER_KUBERNETES_SVC_IP}",
    "kubernetes",
    "kubernetes.default",
    "kubernetes.default.svc",
    "kubernetes.default.svc.cluster",
    "kubernetes.default.svc.cluster.local.",
    "kubernetes.default.svc.${CLUSTER_DNS_DOMAIN}."
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "opsnull"
    }
  ]
}
EOF
```

- hosts 字段指定授权使用该证书的 **IP 和域名列表**，这里列出了 master 节点 IP、kubernetes 服务的 IP 和域名；

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

将生成的证书和私钥文件拷贝到所有 master 节点：

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

创建加密配置文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > encryption-config.yaml <<EOF
kind: EncryptionConfig
apiVersion: v1
resources:
  - resources:
      - secrets
```

```
    providers:
      - aescbc:
          keys:
            - name: key1
              secret: ${ENCRYPTION_KEY}
      - identity: {}
EOF
```

将加密配置文件拷贝到 master 节点的 `/etc/kubernetes` 目录下：

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

创建审计策略文件

```
cd /opt/k8s/work
source /opt/k8s/bin/environment.sh
cat > audit-policy.yaml <<EOF
apiVersion: audit.k8s.io/v1beta1
kind: Policy
rules:
  # The following requests were manually identified as high-volume and low-risk, so
drop them.
  - level: None
    resources:
      - group: ""
        resources:
          - endpoints
          - services
          - services/status
    users:
      - 'system:kube-proxy'
    verbs:
      - watch

  - level: None
    resources:
      - group: ""
        resources:
          - nodes
          - nodes/status
    userGroups:
      - 'system:nodes'
```

```yaml
    verbs:
      - get

  - level: None
    namespaces:
      - kube-system
    resources:
      - group: ""
        resources:
          - endpoints
    users:
      - 'system:kube-controller-manager'
      - 'system:kube-scheduler'
      - 'system:serviceaccount:kube-system:endpoint-controller'
    verbs:
      - get
      - update

  - level: None
    resources:
      - group: ""
        resources:
          - namespaces
          - namespaces/status
          - namespaces/finalize
    users:
      - 'system:apiserver'
    verbs:
      - get

  # Don't log HPA fetching metrics.
  - level: None
    resources:
      - group: metrics.k8s.io
    users:
      - 'system:kube-controller-manager'
    verbs:
      - get
      - list

  # Don't log these read-only URLs.
  - level: None
    nonResourceURLs:
      - '/healthz*'
      - /version
      - '/swagger*'

  # Don't log events requests.
  - level: None
    resources:
      - group: ""
        resources:
          - events

  # node and pod status calls from nodes are high-volume and can be large, don't log
responses
  # for expected updates from nodes
  - level: Request
    omitStages:
      - RequestReceived
    resources:
      - group: ""
        resources:
          - nodes/status
          - pods/status
    users:
      - kubelet
      - 'system:node-problem-detector'
      - 'system:serviceaccount:kube-system:node-problem-detector'
    verbs:
      - update
      - patch

  - level: Request
    omitStages:
      - RequestReceived
    resources:
      - group: ""
        resources:
          - nodes/status
          - pods/status
    userGroups:
      - 'system:nodes'
    verbs:
      - update
      - patch

  # deletecollection calls can be large, don't log responses for expected namespace
deletions
  - level: Request
    omitStages:
      - RequestReceived
    users:
      - 'system:serviceaccount:kube-system:namespace-controller'
    verbs:
      - deletecollection

  # Secrets, ConfigMaps, and TokenReviews can contain sensitive & binary data,
  # so only log at the Metadata level.
  - level: Metadata
    omitStages:
      - RequestReceived
    resources:
      - group: ""
```

```
          resources:
            - secrets
            - configmaps
        - group: authentication.k8s.io
          resources:
            - tokenreviews
    # Get repsonses can be large; skip them.
    - level: Request
      omitStages:
        - RequestReceived
      resources:
        - group: ""
        - group: admissionregistration.k8s.io
        - group: apiextensions.k8s.io
        - group: apiregistration.k8s.io
        - group: apps
        - group: authentication.k8s.io
        - group: authorization.k8s.io
        - group: autoscaling
        - group: batch
        - group: certificates.k8s.io
        - group: extensions
        - group: metrics.k8s.io
        - group: networking.k8s.io
        - group: policy
        - group: rbac.authorization.k8s.io
        - group: scheduling.k8s.io
        - group: settings.k8s.io
        - group: storage.k8s.io
      verbs:
        - get
        - list
        - watch

    # Default level for known APIs
    - level: RequestResponse
      omitStages:
        - RequestReceived
      resources:
        - group: ""
        - group: admissionregistration.k8s.io
        - group: apiextensions.k8s.io
        - group: apiregistration.k8s.io
        - group: apps
        - group: authentication.k8s.io
        - group: authorization.k8s.io
        - group: autoscaling
        - group: batch
        - group: certificates.k8s.io
        - group: extensions
        - group: metrics.k8s.io
        - group: networking.k8s.io
        - group: policy
        - group: rbac.authorization.k8s.io
        - group: scheduling.k8s.io
        - group: settings.k8s.io
        - group: storage.k8s.io

  # Default level for all other requests.
  - level: Metadata
    omitStages:
      - RequestReceived
EOF
```

分发审计策略文件:

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

创建后续访问 metrics-server 或 kube-prometheus 使用的证书

创建证书签名请求:

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

- CN 名称需要位于 kube-apiserver 的 `--requestheader-allowed-names` 参数中，否则后续访问 metrics 时会

提示权限不足。

生成证书和私钥：

```
cfssl gencert -ca=/etc/kubernetes/cert/ca.pem \
  -ca-key=/etc/kubernetes/cert/ca-key.pem  \
  -config=/etc/kubernetes/cert/ca-config.json  \
  -profile=kubernetes proxy-client-csr.json | cfssljson -bare proxy-client
ls proxy-client*.pem
```

将生成的证书和私钥文件拷贝到所有 master 节点：

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

创建 kube-apiserver systemd unit 模板文件

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

[Service]
WorkingDirectory=${K8S_DIR}/kube-apiserver
ExecStart=/opt/k8s/bin/kube-apiserver \\
  --advertise-address=##NODE_IP## \\
  --default-not-ready-toleration-seconds=360 \\
  --default-unreachable-toleration-seconds=360 \\
  --feature-gates=DynamicAuditing=true \\
  --max-mutating-requests-inflight=2000 \\
  --max-requests-inflight=4000 \\
  --default-watch-cache-size=200 \\
  --delete-collection-workers=2 \\
  --encryption-provider-config=/etc/kubernetes/encryption-config.yaml \\
  --etcd-cafile=/etc/kubernetes/cert/ca.pem \\
  --etcd-certfile=/etc/kubernetes/cert/kubernetes.pem \\
  --etcd-keyfile=/etc/kubernetes/cert/kubernetes-key.pem \\
  --etcd-servers=${ETCD_ENDPOINTS} \\
  --bind-address=##NODE_IP## \\
  --secure-port=6443 \\
  --tls-cert-file=/etc/kubernetes/cert/kubernetes.pem \\
  --tls-private-key-file=/etc/kubernetes/cert/kubernetes-key.pem \\
  --insecure-port=0 \\
  --audit-dynamic-configuration \\
  --audit-log-maxage=15 \\
  --audit-log-maxbackup=3 \\
  --audit-log-maxsize=100 \\
  --audit-log-truncate-enabled \\
  --audit-log-path=${K8S_DIR}/kube-apiserver/audit.log \\
  --audit-policy-file=/etc/kubernetes/audit-policy.yaml \\
  --profiling \\
  --anonymous-auth=false \\
  --client-ca-file=/etc/kubernetes/cert/ca.pem \\
  --enable-bootstrap-token-auth \\
  --requestheader-allowed-names="aggregator" \\
  --requestheader-client-ca-file=/etc/kubernetes/cert/ca.pem \\
  --requestheader-extra-headers-prefix="X-Remote-Extra-" \\
  --requestheader-group-headers=X-Remote-Group \\
  --requestheader-username-headers=X-Remote-User \\
  --service-account-key-file=/etc/kubernetes/cert/ca.pem \\
  --authorization-mode=Node,RBAC \\
  --runtime-config=api/all=true \\
  --enable-admission-plugins=NodeRestriction \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --event-ttl=168h \\
  --kubelet-certificate-authority=/etc/kubernetes/cert/ca.pem \\
  --kubelet-client-certificate=/etc/kubernetes/cert/kubernetes.pem \\
  --kubelet-client-key=/etc/kubernetes/cert/kubernetes-key.pem \\
  --kubelet-https=true \\
  --kubelet-timeout=10s \\
  --proxy-client-cert-file=/etc/kubernetes/cert/proxy-client.pem \\
  --proxy-client-key-file=/etc/kubernetes/cert/proxy-client-key.pem \\
  --service-cluster-ip-range=${SERVICE_CIDR} \\
  --service-node-port-range=${NODE_PORT_RANGE} \\
  --logtostderr=true \\
  --v=2
Restart=on-failure
RestartSec=10
Type=notify
LimitNOFILE=65536

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

- `--advertise-address`：apiserver 对外通告的 IP（kubernetes 服务后端节点 IP）；
- `--default-*-toleration-seconds`：设置节点异常相关的阈值；
- `--max-*-requests-inflight`：请求相关的最大阈值；
- `--etcd-*`：访问 etcd 的证书和 etcd 服务器地址；
- `--bind-address`： https 监听的 IP，不能为 `127.0.0.1`，否则外界不能访问它的安全端口 6443；
- `--secret-port`：https 监听端口；

- `--insecure-port=0`：关闭监听 http 非安全端口(8080)；
- `--tls-*-file`：指定 apiserver 使用的证书、私钥和 CA 文件；
- `--audit-*`：配置审计策略和审计日志文件相关的参数；
- `--client-ca-file`：验证 client (kue-controller-manager、kube-scheduler、kubelet、kube-proxy 等)请求所带的证书；
- `--enable-bootstrap-token-auth`：启用 kubelet bootstrap 的 token 认证；
- `--requestheader-*`：kube-apiserver 的 aggregator layer 相关的配置参数，proxy-client & HPA 需要使用；
- `--requestheader-client-ca-file`：用于签名 `--proxy-client-cert-file` 和 `--proxy-client-key-file` 指定的证书；在启用了 metric aggregator 时使用；
- `--requestheader-allowed-names`：不能为空，值为逗号分割的 `--proxy-client-cert-file` 证书的 CN 名称，这里设置为 "aggregator"；
- `--service-account-key-file`：签名 ServiceAccount Token 的公钥文件，kube-controller-manager 的 `--service-account-private-key-file` 指定私钥文件，两者配对使用；
- `--runtime-config=api/all=true`： 启用所有版本的 APIs，如 autoscaling/v2alpha1；
- `--authorization-mode=Node,RBAC`、`--anonymous-auth=false`： 开启 Node 和 RBAC 授权模式，拒绝未授权的请求；
- `--enable-admission-plugins`：启用一些默认关闭的 plugins；
- `--allow-privileged`：运行执行 privileged 权限的容器；
- `--apiserver-count=3`：指定 apiserver 实例的数量；
- `--event-ttl`：指定 events 的保存时间；
- `--kubelet-*`：如果指定，则使用 https 访问 kubelet APIs；需要为证书对应的用户(上面 kubernetes*.pem 证书的用户为 kubernetes) 用户定义 RBAC 规则，否则访问 kubelet API 时提示未授权；
- `--proxy-client-*`：apiserver 访问 metrics-server 使用的证书；
- `--service-cluster-ip-range`： 指定 Service Cluster IP 地址段；
- `--service-node-port-range`： 指定 NodePort 的端口范围；

如果 kube-apiserver 机器没有运行 kube-proxy，则还需要添加 `--enable-aggregator-routing=true` 参数；

关于 `--requestheader-XXX` 相关参数，参考：

- https://github.com/kubernetes-incubator/apiserver-builder/blob/master/docs/concepts/auth.md
- https://docs.bitnami.com/kubernetes/how-to/configure-autoscaling-custom-metrics/

注意：

1. `--requestheader-client-ca-file` 指定的 CA 证书，必须具有 `client auth and server auth`；
2. 如果 `--requestheader-allowed-names` 不为空,且 `--proxy-client-cert-file` 证书的 CN 名称不在 allowed-names 中，则后续查看 node 或 pods 的 metrics 失败，提示：

```
  $ kubectl top nodes
  Error from server (Forbidden): nodes.metrics.k8s.io is forbidden: User "aggregator"
cannot list resource "nodes" in API group "metrics.k8s.io" at the cluster scope
```

为各节点创建和分发 kube-apiserver systemd unit 文件

替换模板文件中的变量，为各节点生成 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-apiserver.service.template > kube-apiserver-${NODE_IPS[i]}.service
  done
ls kube-apiserver*.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 ">>> ${node_ip}"
    scp kube-apiserver-${node_ip}.service root@${node_ip}:/etc/systemd/system/kube-
apiserver.service
  done
```

启动 kube-apiserver 服务

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

检查 kube-apiserver 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-apiserver |grep 'Active:'"
  done
```

确保状态为 `active (running)`，否则查看日志，确认原因：

```
journalctl -u kube-apiserver
```

检查集群状态

```
$ kubectl cluster-info
Kubernetes master is running at https://172.27.138.251:6443

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

$ kubectl get all --all-namespaces
NAMESPACE   NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
default     service/kubernetes   ClusterIP   10.254.0.1   <none>        443/TCP
3m53s

$ kubectl get componentstatuses
NAME                 AGE
controller-manager   <unknown>
scheduler            <unknown>
etcd-0               <unknown>
etcd-2               <unknown>
etcd-1               <unknown>
```

- Kubernetes 1.16.6 存在 Bugs 导致返回结果一直为 `<unknown>`，但 `kubectl get cs -o yaml` 可以返回正确结果；

检查 kube-apiserver 监听的端口

```
$ sudo netstat -lnpt|grep kube
tcp        0      0 172.27.138.251:6443     0.0.0.0:*               LISTEN
 101442/kube-apiserv
```

- 6443: 接收 https 请求的安全端口，对所有请求做认证和授权；
- 由于关闭了非安全端口，故没有监听 8080；