Kubernetes cluster with Vagrant and Virtualbox

节点网络IP： 192.168.99.91 ~ 192.168.99.93
容器IP范围： 172.33.0.0/16
Kubernetes service IP范围： 10.254.0.0/16

证书

生成的 CA 证书和秘钥文件如下：

```
ca.pem              ca-key.pem
kubernetes.pem      kubernetes-key.pem
kube-proxy.pem      kube-proxy-key.pem
admin.pem           admin-key.pem
```

使用证书的组件如下：

```
etcd：                    使用 ca.pem、kubernetes-key.pem、kubernetes.pem；
kube-apiserver：          使用 ca.pem、kubernetes-key.pem、kubernetes.pem；
kubelet：                 使用 ca.pem；
kube-proxy：              使用 ca.pem、kube-proxy-key.pem、kube-proxy.pem；
kubectl：                 使用 ca.pem、admin-key.pem、admin.pem；
kube-controller-manager：使用 ca-key.pem、ca.pem
```

主要环境变量

```
# TLS Bootstrapping 使用的 Token，可以使用命令 head -c 16 /dev/urandom | od -An -t x | tr -d ' ' 生成
BOOTSTRAP_TOKEN="9c64d78dbd5afd42316e32d922e2da47"

# 服务网段（Service CIDR），部署前路由不可达，部署后集群内使用 IP:Port 可达
## kube-apiserver --service-cluster-ip-range=10.254.0.0/16
## kube-controller-manager --service-cluster-ip-range=10.254.0.0/16
SERVICE_CIDR="10.254.0.0/16"

# POD 网段（Cluster CIDR），部署前路由不可达，部署后路由可达（flanneld 保证）（容器 IP）
CLUSTER_CIDR="172.33.0.0/16"

# 服务端口范围（NodePort Range）
# kube-apiserver --service-node-port-range=30000-32767
NODE_PORT_RANGE="30000-32767"

# etcd 集群服务地址列表
ETCD_ENDPOINTS="https://192.168.99.91:2379"
# ETCD_ENDPOINTS="https://192.168.99.91:2379,https://192.168.99.92:2379,https://192.168.99.93:2379"

# flanneld 网络配置前缀
FLANNEL_ETCD_PREFIX="/kube-centos/network"

# kubernetes 服务 IP（预分配，一般是 SERVICE_CIDR 中第一个IP）
CLUSTER_KUBERNETES_SVC_IP="10.254.0.1"

# 集群 DNS 服务 IP（从 SERVICE_CIDR 中预分配）
CLUSTER_DNS_SVC_IP="10.254.0.2"

# 集群 DNS 域名
CLUSTER_DNS_DOMAIN="cluster.local."
```

etcd 启动参数

```
--name "node1"                              # 成员名字
--data-dir=/var/lib/etcd/default.etcd       # 数据目录路径

# 用于监听客户端通讯的 client URL列表。
--listen-client-urls "http://192.168.99.91:2379,http://localhost:2379"
# 列出这个成员的 client URL，通告给集群中的其他成员。
--advertise-client-urls "http://192.168.99.91:2379"
```

flanneld 启动参数

```
-etcd-endpoints=http://192.168.99.91:2379      # etcd 的地址
-etcd-prefix=/kube-centos/network              # 在 etcd 中配置的网络参数的 key
-iface=eth2                                    # 监听的网卡
```

向 etcd 写入集群 Pod 网段信息

在etcd中创建网络配置，docker分配IP地址段。（子网IP范围：172.33.0.0）
本步骤只需在第一次部署 Flannel 网络时执行，后续在其它节点上部署 Flannel 时无需再写入该信息！

provision-etcd.sh

```
echo 'create kubernetes ip range for flannel on 172.33.0.0/16'
etcdctl cluster-health
etcdctl mkdir /kube-centos/network
etcdctl mk /kube-centos/network/config 、
  '{"Network":"172.33.0.0/16","SubnetLen":24,"Backend":{"Type":"host-gw"}}'
```

查询 flannel 网络信息

```
etcdctl --endpoints=${ETCD_ENDPOINTS} ls ${FLANNEL_ETCD_PREFIX}/subnets
etcdctl ls /kube-centos/network/subnets
```

可在各节点查询子网网关，确认能ping通。

kube-apiserver 启动参数

```
## 必须项 ------------
--service-cluster-ip-range=10.254.0.0/16    # service 要使用的网段，使用 CIDR 格式，参考 service 的定义
--etcd-servers=http://192.168.99.91:2379    # 以逗号分隔的 etcd 服务列表，与 `--etcd-config` 互斥

## 可选项 -------------
## HTTP/HTTPS 监听的IP与端口
--apiserver-count=3                         # apiservers 数量 (默认1)
--advertise-address=192.168.99.91           # 通过该 ip 地址向集群其他节点公布 api server 的信息
--bind-address=192.168.99.91                # HTTPS 安全端口监听的IP (默认 0.0.0.0)
--secure-port=6443                          # HTTPS 安全端口 (默认 6443)
--insecure-bind-address=192.168.99.91       # HTTP 非安全端口监听的IP (默认 127.0.0.1)
--insecure-port=8080                        # HTTP 非安全端口监听的端口 (默认 8080)
--service-node-port-range=30000-32767       # Service 的 NodePort 所能使用的主机端口号范围
--runtime-config=rbac.authorization.k8s.io/v1beta1  # 打开或关闭针对某个api版本支持

## 证书
# HTTPS密钥与证书
--tls-private-key-file=/etc/kubernetes/ssl/kubernetes-key.pem
--tls-cert-file=/etc/kubernetes/ssl/kubernetes.pem
# 认证: 证书认证 + Token 认证
--client-ca-file=/etc/kubernetes/ssl/ca.pem # 证书认证: client证书文件
--token-auth-file=/etc/kubernetes/token.csv # tocken 认证: token文件
# 授权模式： 安全接口上的授权
--authorization-mode=Node,RBAC
# 准入控制： 一串用逗号连接的有序的准入模块列表
--admission-control=ServiceAccount,NamespaceLifecycle,NamespaceExists,LimitRanger,ResourceQuota

--service-account-key-file=/etc/kubernetes/ssl/ca-key.pem
--enable-bootstrap-token-auth               # 启动引导令牌认证（Bootstrap Tokens）
--allow-privileged=true                     # 是否允许 privileged 容器运行
--kubelet-https=true                        # 指定 kubelet 是否使用 HTTPS 连接
--enable-swagger-ui=true                    # 开启 Swagger UI

## 日志
--logtostderr=true                          # 输出到 `stderr`,不输到日志文件。
--v=0                                       # 日志级别
--event-ttl=1h                              # 各种事件在系统中的保存时间
--audit-log-path=/var/lib/audit.log         # 审计日志路径
--audit-log-maxage=30                       # 旧日志最长保留天数
--audit-log-maxbackup=3                     # 旧日志文件最多保留个数
--audit-log-maxsize=100                     # 日志文件最大大小（单位MB）
```

kube-controller-manager 启动参数

```
--logtostderr=true                          # 输出到 `stderr`,不输到日志文件。
--v=0                                       # 日志级别
--leader-elect=true                         # 启动选举

--master=http://192.168.99.91:8080          # Kubernetes master apiserver 地址
--address=127.0.0.1                         # 绑定主机 IP 地址，apiserver 与 controller-manager在同一主机
--service-cluster-ip-range=10.254.0.0/16    # service 要使用的网段，使用 CIDR 格式，参考 service 的定义
--cluster-name=kubernetes                   # Kubernetes 集群名，也表现为实例化的前缀
--root-ca-file=/etc/kubernetes/ssl/ca.pem   # 用来对 kube-apiserver 证书进行校验，被用于 Service Account。

# 用于给 Service Account Token 签名的 PEM 编码的 RSA 或 ECDSA 私钥文件。
--service-account-private-key-file=/etc/kubernetes/ssl/ca-key.pem

# 指定的证书和私钥文件用来签名为 TLS BootStrap 创建的证书和私钥；
--cluster-signing-cert-file=/etc/kubernetes/ssl/ca.pem
--cluster-signing-key-file=/etc/kubernetes/ssl/ca-key.pem
```

kube-scheduler 启动参数

```
--logtostderr=true                          # 输出到 `stderr`,不输到日志文件。
--v=0                                       # 日志级别
--leader-elect=true                         # 启动选举
--master=http://192.168.99.91:8080          # Kubernetes master apiserver 地址
--address=127.0.0.1                         # 绑定主机 IP 地址，apiserver 与 controller-manager在同一主机
--kubeconfig=/etc/kubernetes/scheduler.conf # kubeconfig 配置文件，包含 master 地址信息和必要的认证信息
```

kube-proxy 启动参数

```
--logtostderr=true                          # 输出到 `stderr`,不输到日志文件。
--v=0                                       # 日志级别
--master=http://192.168.99.91:8080          # Kubernetes master apiserver 地址
--bind-address=192.168.99.91                # 主机绑定的IP地址。
--cluster-cidr=10.254.0.0/16                # kube-proxy 根据此判断集群内部和外部流量
--hostname-override=192.168.99.91           # 值须与kubelet的值一致，否则kube-proxy启动后会找不到该Node
--hostname-override=node1                   # 值须与kubelet的值一致，否则kube-proxy启动后会找不到该Node
# kubeconfig 配置文件
--kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig
```

kubelet 启动参数

```
--logtostderr=true                          # 输出到 `stderr`,不输到日志文件。
--v=0                                       # 日志级别
--allow-privileged=true                     # 是否允许容器运行在 privileged 模式
--address=192.168.99.91                     # 绑定主机 IP 地址
--hostname-override=node1                   #
--pod-infra-container-image=docker.io/openshift/origin-pod   # 基础镜像容器
--runtime-cgroups=/systemd/system.slice     # 如果使用systemd方式启动，增加此参数
--kubelet-cgroups=/systemd/system.slice     # 如果使用systemd方式启动，增加此参数
--cgroup-driver=systemd                     # 配置成 systemd，不要使用 cgroup
--cluster-dns=10.254.0.2                    # 指定kubedns的Service IP，--cluster-domain指定域名后缀
                                            # 这两个参数同时指定后才会生效；
--cluster-domain=cluster.local              # 指定 pod 启动时 /etc/resolve.conf 文件中的 search domain

# kubelet 使用该文件中的用户名和 token 向 kube-apiserver 发送 TLS Bootstrapping 请求；
--bootstrap-kubeconfig=/etc/kubernetes/bootstrap.kubeconfig
--require-kubeconfig                        # 如未指定--apiservers，则须指定此选项后
                                            # 才从配置文件读取 kube-apiserver 地址
# kubeconfig 配置文件，在配置文件中包含 master 地址信息和必要的认证信息
--kubeconfig=/etc/kubernetes/kubelet.kubeconfig

--cert-dir=/etc/kubernetes/ssl              # TLS证书所在的目录。
--hairpin-mode promiscuous-bridge           # kubelet应该如何设置 hairpin NAT。
--serialize-image-pulls=false               # 一次拉出一个镜像。
--allow-privileged=true                     # 是否允许 privileged 容器运行

## 未使用
# $KUBELET_API_SERVER="--api-servers=http://172.20.0.113:8080"
# $KUBELET_PORT="--port=10250"
```