Kube-proxy工作模式

|  |
| --- |
|  |

修改kube-proxy模式为ipvs示例

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 安装ipvs，加载模块      ConfigMap 的核心作用就是 **集中管理配置项**  **配置cube-proxy configmap**        删除kube-proxy所有组件（到此就成功了）       |  |  | | --- | --- | | **片段** | **含义** | | kubectl | 使用 Kubernetes 命令行工具 | | delete | 删除资源对象 | | po | Pod 的缩写（pods） | | -l k8s-app=kube-proxy | 使用 **标签选择器（label selector）** 删除所有标签为 k8s-app=kube-proxy 的 Pod | | -n kube-system | 在命名空间 kube-system 中执行（系统组件都在这里） |   删除了所有 kube-proxy 组件的 Pod。  会重新创建：    以下都是验证：  都是running： |

命令补全

|  |
| --- |
|  执行 source <(kubectl completion bash)   当前 Bash shell 就能识别 kubectl 命令的补全规则 |

证书续签

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 续签示例：  模拟证书过期    续签失败，用于还原  Master查看证书有效期    修改本机时间：（所有节点）    测试：显示过期    续签证书+更新kubeconfig所有配置文件（四个）  其实就是 **重新生成一张新的证书**，并覆盖掉旧的。    kubeadm certs renew all  ubeadm certs renew all的作用是对pki下所有证书增重新生成  新证书的有效期是 从当前系统时间开始重新计算 一年（默认 365 天）， 而不是在旧证书的到期时间基础上“往后延长”。  会又增加一年续签时间    看到根据过期证书时间增加一年：    更新配置文件文件：  删除所有组件conf:      重新生成：    生成4配置文件：    这条命令不是完整初始化集群，而是只执行 init 的 **某个阶段 (phase)**。  kubeadm init 本身分为多个阶段，比如：   * preflight（预检） * certs（生成证书） * kubeconfig（生成 kubeconfig 文件） * control-plane（部署核心组件） * etcd（初始化 etcd） * mark-control-plane 等。   当你加上  phase kubeconfig all  时，意思就是：  “只执行 kubeconfig 阶段中全部子步骤”  **它会生成什么？**  它会在 /etc/kubernetes/ 下生成以下 **4 个 kubeconfig 文件**：   | **文件路径** | **用途** | | --- | --- | | /etc/kubernetes/admin.conf | 集群管理员使用（kubectl 默认用这个） | | /etc/kubernetes/kubelet.conf | kubelet 与 apiserver 通信使用 | | /etc/kubernetes/controller-manager.conf | kube-controller-manager 使用 | | /etc/kubernetes/scheduler.conf | kube-scheduler 使用 |   **🧠 三、这些文件包含什么？**  每个 \*.conf 文件都包含：   * 集群的 **API Server 地址**； * 访问凭证（CA 证书、公私钥）； * 用户身份（如 system:admin, system:kube-controller-manager 等）。   这些 kubeconfig 文件允许不同组件或用户安全地连接 API Server。  拷贝admin.conf到家目录覆盖一下：    **所有核心组件都运行在 kube-system 命名空间**。  具体说明如下：  **1️⃣ 核心组件列表**   |  |  |  |  | | --- | --- | --- | --- | | **组件** | **功能** | **运行方式** | **命名空间** | | kube-apiserver | 提供 Kubernetes API 服务 | 静态 Pod | kube-system | | kube-controller-manager | 管理控制循环，如 Deployment、ReplicaSet | 静态 Pod | kube-system | | kube-scheduler | 调度 Pod 到合适节点 | 静态 Pod | kube-system | | etcd | 存储集群状态（Key-Value 数据库） | 静态 Pod | kube-system | | kube-proxy | 维护节点网络规则 | DaemonSet | kube-system | | CoreDNS / kube-dns | DNS 服务 | Deployment | kube-system |   **静态 Pod 特点**   * 核心组件都是 **静态 Pod**（静态 Pod YAML 文件在 /etc/kubernetes/manifests/） * 由 **kubelet 直接管理**，不是 Deployment/ReplicaSet * 删除 Pod 后，**kubelet 会自动重建**   **重启控制平面组件 Pod**（让它们加载新证书）      删除四个pod，自动创建新的  systemctl restart kubelet #删除有问题就执行这个  最后全部running    更新node证书：      **var/lib/kubelet/pki/kubelet-client-current.pem**   * **类型**：软链接（symlink） * **作用**：   + kubelet 启动和运行时使用的 **客户端证书**；   + 用于与 API Server 进行 TLS 认证；   + 始终指向 kubelet “当前有效”的证书文件   **var/lib/kubelet/pki/kubelet-client-YYYYMMDDHHMMSS.pem**   * **类型**：实际证书文件 * **作用**：   + 具体的 kubelet 客户端证书文件，由 kubelet 自动生成或 API Server 签发后存储；   + 文件名带时间戳，用于区分历史证书和新证书。   kubelet 客户端证书就是 kubelet 的“身份证 + 加密钥匙”，用来安全地向 Kubernetes API Server 认证和通信。  有几类证书，它们 **相互独立**，过期时间也不一样：  **1️⃣ 控制平面证书（Master 证书）**   * 位置：/etc/kubernetes/pki/ * 包括：   + apiserver.crt   + controller-manager.crt   + scheduler.crt   + etcd 相关证书 * **用途**：保护 API Server 与控制平面组件之间的通信 * **过期**：kubeadm certs renew all 管理，过期后可能影响整个控制平面组件之间通信   **2️⃣ kubelet 客户端证书（Node 证书）**   * 位置：/var/lib/kubelet/pki/kubelet-client-current.pem + 历史时间戳文件 * **用途**：证明 kubelet 节点身份，安全地向 API Server 报告状态、获取 Pod 指令 * **过期**：   + 与控制平面证书无关   + kubelet 会在证书快到期时自动生成 CSR 并请求 API Server 签发新证书   + 过期只会影响 kubelet 与 API Server 的通信，不直接影响控制平面内部组件    Master 证书和 kubelet 客户端证书独立管理   Master 证书默认一年，Node 证书默认也是一年，但生效时间取决于 CSR 批准时间   续期顺序建议：先 master 证书 → 再批准 kubelet CSR   CSR 批准只是客户端证书更新的一部分，不影响 Master 证书  更新kubelet.conf证书       * phase kubeconfig all 生成的 kubelet.conf：   + 是 **Master 节点自己的 kubelet** 使用的配置   + 证书 CN 是 Master 节点名   + 不适合直接给远程 Node 使用 * phase kubeconfig kubelet --node-name <remote>：   + 为指定远程 Node 生成 kubelet.conf   + 证书 CN 对应远程节点名   + 拷贝到远程节点后，kubelet 启动即可成功注册集群   核心点：**kubelet.conf 的证书和节点名必须匹配，才能通过 API Server 验证**。  kubelet.conf 让节点“加入集群”， admin.conf 让人“控制集群”。   |  |  |  |  | | --- | --- | --- | --- | | **kubeconfig 文件** | **使用者** | **作用** | **权限** | | **admin.conf** | 人类管理员 / kubectl | 操作集群所有资源 | ✅ 完整权限 | | **controller-manager.conf** | 控制器组件 | 管理控制循环 | 中等权限 | | **scheduler.conf** | 调度器 | 调度 Pod | 中等权限 | | **kubelet.conf** | 各 Node 的 kubelet 进程 | 上报节点状态、运行 Pod | ❌ 无资源管理权限 |   kubeadm init \  --kubernetes-version=v1.23.0 \  phase kubeconfig kubelet \  --node-name k8s-node01 \  --kubeconfig-dir /node01/  🧠 一行一行解释：  参数 作用  kubeadm init 初始化集群的主命令（一般用于创建 master 节点）  phase kubeconfig kubelet 进入 init 的一个“阶段”，只执行生成 kubelet 用的 kubeconfig 文件 这一步，不执行整个初始化流程。  --kubernetes-version=v1.23.0 指定要生成配置的 Kubernetes 版本。  --node-name k8s-node01 指定该配置文件要绑定的节点名称（会写进 kubelet kubeconfig 的 user 或 cluster 信息里）。  --kubeconfig-dir /node01/ 指定生成的 kubeconfig 文件要存放的路径（默认是 /etc/kubernetes/）。  两个node都有：      全部批准了：    Node节点查看证书    到此成功续签证书    续签方式二：  前置：  恢复到原来的2026过期结束  修改系统时间使其过期（全部机器）    此时肯定报过期错误：    执行：      执行脚本：    验证证书时间：  Node和master  可以看到10年，在旧证书时间上  Master：    同时一切正常： |

Kuboard

|  |
| --- |
| master导入镜像    **必须镜像**   1. **kuboard-v3.tar**    * Kuboard 的 **核心服务 + Web 前端**    * 没有它，无法访问 Kuboard 界面，也无法管理集群。 2. **kuboard-agent-v3.tar**    * 部署在集群中，与 Kuboard 核心服务通信    * 没有它，Kuboard 无法获取集群状态，也无法执行操作   **kuboard-agent-v3.tar**   * **作用**：Kuboard 的 **Agent**，部署在 Kubernetes 集群中，用于：   + 收集集群状态（节点、Pod、Service 等）   + 执行 Kuboard 发起的操作（比如创建 Deployment、更新配置） * **特点**：   + 每个节点通常都会运行一个 agent Pod   + 是 Kuboard 与集群通信的桥梁   **kuboard-v3.tar**   * **作用**：Kuboard V3 的 **Web 前端 + 后端服务**   + Web 界面   + 用户认证   + API 服务 * **特点**：   + 这是你访问的核心界面   + 负责管理和展示集群信息   **uestdb-6.0.5.tar**   * **作用**：**时间序列数据库**   + Kuboard 用来存储集群的监控数据、指标（如 CPU/内存使用率）   + QuestDB 性能高，适合高频监控数据写入 * **特点**：   + 可选，但强烈推荐   + 用于监控模块，如果不部署，某些监控功能可能不可用   **etcd-host-3.4.16-2.tar**   * **作用**：   + 如果你希望 Kuboard 自带 **独立的 etcd** 来存储配置和数据   + 在没有外部 etcd 的情况下使用   修改拉取策略：  两处  应用kuboard.yaml  等待全部running：      看到该命名空间下的4个pod    都是启动在master节点上  查看kuboard暴露nodeport：    Nodeip：30080  看到： |