**Kubernetes学习笔记**

**第一章**

1. k8s主要是进行容器化集群管理，Docker主要是单机管理，虽然Docker也可以做到集群管理但是没有k8s方便、且k8s利于应用扩展、让部署容器化应用更加简洁和高效
2. k8s的8个优势：
3. 自动装箱
4. 自我修复(自愈能力)
5. 水平扩展
6. 服务发现（负载均衡）
7. 滚动更新
8. 版本回退
9. 密钥和配置管理
10. 存储编排
11. 批处理
12. master组件：
13. APIserver：集群统一入口，以restful方式，交给etcd存储
14. scheduler：节点调度，选择node节点应用部署
15. controller-manager：处理集群中常规后台任务，一个资源对应一个控制器
16. etcd：存储系统，用于保存集群相关的数据
17. node组件：
18. kubelet：master派到node节点代表，管理本机容器
19. kube-proxy：提供网络代理、负载均衡等操作

5k8s的核心概念：

1. pod：
2. 最小部署单元
3. 一组容器的集合
4. 共享网络（一组内容器共享）
5. 生命周期是短暂的，关闭后需重新部署
6. controller
7. 确保预期的pod的数量
8. 无状态的应用部署（直接拿过来随便用）
9. 有状态的应用部署（有条件才可以使用）
10. 确保所有的node运行同一个pod
11. 一次性任务和定时任务
12. service

定义一组pod的访问规则

（通过service统一入口访问然后执行controller去创建pod进行部署）

**搭建**

1. 搭建集群要求硬件较高，目前先学理论知识不做实践
2. 搭建k8s环境平台规划：
   1. 方式一：单master集群（一master对应多node）
   2. 方式二：单master集群->多master集群（多master对应多node；master和node中间有负载均衡）
3. 目前生产部署 Kubernetes 集群主要有两种方式：Kubeadm和二进制包

*Kubeadm 是一个 K8s 部署工具，提供 kubeadm init 和 kubeadm join，用于快速部署 Kubernetes 集群。而二进制包则需要手动部署每个组件，组成 Kubernetes 集群。 Kubeadm 降低部署门槛，但屏蔽了很多细节，遇到问题很难排查。使用二进制包部署可以学习很多工作原理，也利于后期维护。*

1. kubeadm方式

*创建三台虚拟机（master，node1，node2）->按照文档和视频进行安装->在master节点执行kubeadm init命令初始化->在node节点上执行kubeadm join 命令把node节点添加到当前集群里。*

1. 二进制方式

*创建虚拟机->操作系统初始化->为etcd和apiserver自签证书->部署etcd集群->部署master集群->部署node组件->部署集群网络*

Etcd 是一个分布式键值存储系统，Kubernetes 使用 Etcd 进行数据存储，所以先准备

一个 Etcd 数据库，为解决 Etcd 单点故障，应采用集群方式部署，这里使用 3 台组建集

群，可容忍 1 台机器故障，当然，你也可以使用 5 台组建集群，可容忍 2 台机器故障。

cfssl 是一个开源的证书管理工具，使用 json 文件生成证书，相比 openssl 更方便使用。

找任意一台服务器操作

TLS Bootstraping：Master apiserver 启用 TLS 认证后，Node 节点 kubelet 和kube-proxy 要与 kube-apiserver 进行通信，必须使用 CA 签发的有效证书才可以，当 Node节点很多时，这种客户端证书颁发需要大量工作，同样也会增加集群扩展复杂度。为了简化流程，Kubernetes 引入了 TLS bootstraping 机制来自动颁发客户端证书，kubelet会以一个低权限用户自动向 apiserver 申请证书，kubelet 的证书由 apiserver 动态签署。所以强烈建议在 Node 上使用这种方式，目前主要用于 kubelet，kube-proxy 还是由我们统一颁发一个证书。

**核心技术**

1. kubectl [command] [type] [name] [flages] --k8s的命令行工具语法

建议使用kubectl ... --help查看具体语法使用规则

1. 资源清单文件yaml（配置文件）可以对资源对象进行编排部署和管理。两种快速创建方式：
   1. 使用kubectl create命令生成yaml文件
   2. 使用kubectl get命令导出已经写过的yaml文件
2. Pod
3. Pod是K8S系统中可以创建和部署的最小单元，由一个或多个container组成。其它的资源对象都是用来支撑或扩展Pod对象功能的。
4. 每一个 Pod 都有一个根容器（Pause）和一个或多个紧密相关的用户业务容器。
5. 一个Pod中容器共享网络命名空间
6. Pod是短暂的（IP不唯一，重启就发生变化）
7. Pod存在的意义
8. 容器由Docker创建，一个容器包含一个应用（该设计适用于单进程）。
9. Pod是多进程设计，它有多个容器，可以运行多个应用程序。
10. Pod存在也是为了亲密性应用
    1. 两个应用之间进行交互
    2. 网络之间调用
    3. 两个应用需要频繁调用
11. Pod实现机制
12. 共享网络

容器之间通过Linux的命名空间（namespace）和组（group）进行隔离。先创建pause容器，然后把其它业务容器加入到pause容器，这样所有的业务容器都在同一个命名空间中，他们的IP、MAC、Port是一样的，由此实现了网络共享

1. 共享存储

引入数据卷概念volumn，使用数据卷进行持久化存储

1. Pod镜像拉取策略

IfNotPresent：不存在时拉取（默认）；Always：每次创建Pod都会拉取；Never：Pod不会主动拉取

1. Pod资源限制

通过requests和limits命令实现

1. Pod重启机制
2. Always：容器终止后总是重启容器（默认）
3. OnFailure：容器异常退出时才重启容器（状态码非0）
4. Never：容器终止退出，从不重启容器
5. Pod健康检查

检查策略： （1）存活检查（livenessProbe），检查失败就杀死容器

（2）就绪检查（readinessProbe），检查失败就把Pod移除

检查方法： （1）发送HTTP请求，返回200-400范围状态码为成功

（2）执行shell命令返回状态码是0为成功

（3）发起TCP Socket建立即为成功

1. 创建Pod基本流程：

第一部分在master节点上：：createPod（创建Pod）->apiserver（进行创建操作）->etcd（将在apiserver里的操作进行存储）

第二部分在master节点上：scheduler->apiserver（scheduler通过apiserver监听有没有新的Pod创建）->etcd（如果监听到把Pod调度到某个node节点上）

第三部分，在node上：kubelet->apiserver（读取etcd拿到分配给当前节点的Pod）->通过Docker创建容器

1. 影响Pod调度的属性：
   1. Pod资源限制对Pod调度产生影响
   2. 节点选择器标签影响Pod调度
   3. 节点亲和性影响Pod调度。硬亲和性：约束条件必须满足；软亲和性：尝试满足
   4. 污点和污点容忍
2. Controller是什么

Cotroller是在集群上管理和运行容器的对象，Pod和Controller通过label标签建立关系，然后通过Controller实现应用的运维，比如伸缩和滚动升级等

1. Pod的作用
2. 确保预期的Pod副本数量
3. 无状态应用部署
4. 有状态应用部署
5. 确保所有的node运行同一个Pod
6. 一次性任务和定时任务
7. Deployment控制器应用场景：
   1. 部署无状态应用
   2. 管理Pod和ReplicaSet
   3. 部署和滚动升级等功能
8. 使用deployment控制器部署应用的步骤
   1. 导出yaml文件
   2. 使用yaml部署应用
   3. 对外暴露端口号

**Service**

1. Service的作用
2. 防止Pod失联（服务发现）
3. 定义一组Pod的访问策略（负载均衡）
4. 常用的Service类型
   1. ClusterIP：在集群内部使用
   2. NodePort：对外给访问的应用使用
   3. LoadBalancer：对外给访问的应用使用，公有云
5. 无状态应用：
   1. 认为Pod都是一样的
   2. 没有顺序要求
   3. 不用考虑在哪个node运行
   4. 随意进行伸缩扩展
6. 有状态应用：
   1. 上面有状态的因素都要考虑到
   2. 让每个Pod独立，保持Pod启动顺序和唯一性。（有序：比如MySQL主从；唯一的网络标识符；持久存储）
7. 部署有状态应用（无头Service：ClusterIP为none）
   1. 通过SatefulSet部署有状态应用

...................

**Secret**

1. Secret

Secret可以加密数据存在etcd里，让Pod容器以挂载Volume的方式访问

1. Secret使用步骤：
   1. 创建加密数据（一般会用base64编码）
   2. 以变量形式挂载到Pod容器中
   3. 以volume形式挂载到Pod容器中
2. ConfigMap

存储不加密数据到etcd，让Pod变量或者Volume挂载到容器中，步骤：

1. 创建配置文件
2. 创建configmap
3. 以volume形式挂载到pod容器中
4. K8S集群安全机制
5. 访问K8S集群时，需要经过三个步骤完成具体操作：认证->鉴权（授权）->准入控制
6. 进行访问时，过程中都需要apiserver做统一协调（访问过程中需要token或者用户和密码，如果访问Pod需要serviceAccount）
7. 传输安全

对外不暴露8080端口，只能内部访问，对外使用6443端口

1. 认证方式：
   1. https证书认证，基于ca证书
   2. http token认证，通过token识别用户
   3. http基本认证，用户名+密码
2. 鉴权（授权）方式：RBAC（基于角色的访问控制）

RBAC 引入了 4 个新的顶级资源对象：Role、ClusterRole、RoleBinding、ClusterRoleBinding。用户可以使用 kubectl 或者 API 调用等方式操作这些资源对象。

1. 准入控制

就是准入控制器的列表，如果列表有请求内容则通过，没有则拒绝

1. 对某一用户对特定命名空间的访问权限设置的步骤：
   1. 创建命名空间
   2. 在新创建的命名空间里创建Pod
   3. 创建角色
   4. 创建角色绑定
   5. 使用证书识别身份

**Ingress**

1. Ingress

之前：使用Service里面的NodePort实现端口号对外暴露，然后通过IP+端口号进行访问。NodePort缺陷：首先NodePort会在每个节点上都会起到端口，在访问时候通过任何节点的IP+端口号的方式实现访问，这样做意味着每个端口只能使用一次且一个端口对应一个应用，但实际访问中都是用域名，根据不同域名跳转到不同端口服务中。

1. Ingress和Pod关系

Ingress作为统一入口，由Service关联一组Pod

1. 使用Ingress的步骤：
   1. 部署Ingress Controller
   2. 创建Ingress规则（配置）

**Helm**

1. Helm的作用

之前：部署应用过程：编写yaml文件（Deployment->Service->Ingress）;部署单一应用只有少量服务的话比较合适；如果部署的是微服务项目，有几十个服务每个服务都有一头yaml文件，不方便维护

Helm：把上述这些yaml作为一个整体管理，实现yaml的高效复用,也能实现应用级别的版本管理

1. Helm的三个重要的概念
   1. helm：命令行工具端，主要用于K8S应用Chart的创建、打包、发布和管理
   2. Chart：应用描述，一系列用于描述K8S资源相关文件的集合
   3. Release：基于Chart部署的实体，一个Chart被Helm运行后将会生成一个对应的Release；将在K8S中创建出真实运行的资源对象
2. Helm V3版本的变化：
   1. 删除Tiller
   2. Release可以在不同命名空间重用
   3. 支持将Chart推送到Docker仓库中
3. Helm的安装（没实践，需要安装时需看视频复习）
   1. 下载Helm压缩包上传到Linux系统，解压后放到user/bin目录下
   2. 配置Helm仓库：添加仓库->更新仓库地址
4. 使用Helm快速部署应用
   1. 使用命令搜索应用（helm search...）
   2. 根据搜索内容选择安装然后查看安装状态 （helm list和helm status...）
5. 持久存储

数据卷是本地存储也是临时存储，当Pod重启后，数据就不存在了，需要持久化。有两种持久化存储

1. nfs网络存储
   1. 找一台服务器作为安装nfs作为nfs服务端并设置挂载路径（挂载路径需要先手动创建）
   2. 在K8S集群node节点安装nfs
   3. 在nfs服务器启动nfs服务
   4. 在K8S集群部署应用使用nfs持久网络存储
2. PV和PVC
   1. PV：持久化存储，对存储进行抽象，对外提供可以调用的地方（生产者）
   2. PVC：用于调用，不需要关心内部实现细节（消费者）
3. 集群资源监控
   1. 监控指标（集群监控）
      1. 节点资源利用率
      2. 节点数
      3. 运行pods
   2. 监控指标（pod监控）
      1. 容器指标
      2. 应用程序
   3. 监控平台（prometheus+Grafana）--需要自行搭建
      1. prometheus是开源的，可以监控、报警；他以HTTP协议周期性抓取被监控组件状态且不需要复杂的集成过程，使用http接口接入就可以了
      2. Grafana是开源的数据分析和可视化根据，支持多种数据源。prometheus对数据进行抓取然后交由Grafana显示。
4. *搭建监控平台（还没实践）*
   1. *部署prometheus*
      1. *部署守护进程*
      2. *部署其它yaml文件*
   2. *部署Grafana*
   3. *打开Granfana，配置数据源，导入显示模板*
      1. *通过查看端口号访问*
      2. *默认用户名和密码admin*
      3. *配置数据源，使用prometheus*
      4. *设置数据显示模板*

**搭建高可用集群**

**背景在单master集群中，当master节点宕机后，集群不能正常工作，所以需要搭建多master集群，实现高可用性**

1. 搭建过程

多node需要连接到load balancer再由它连接到多个master。load balancer起到负载均衡和检查master节点的作用

load balancer中需要有VIP（虚拟IP）

master节点中需部署keepalived（检查作用）和haproxy（负载作用）

***具体过程看视频和文档《使用kubeadm搭建高可用的K8S集群》***

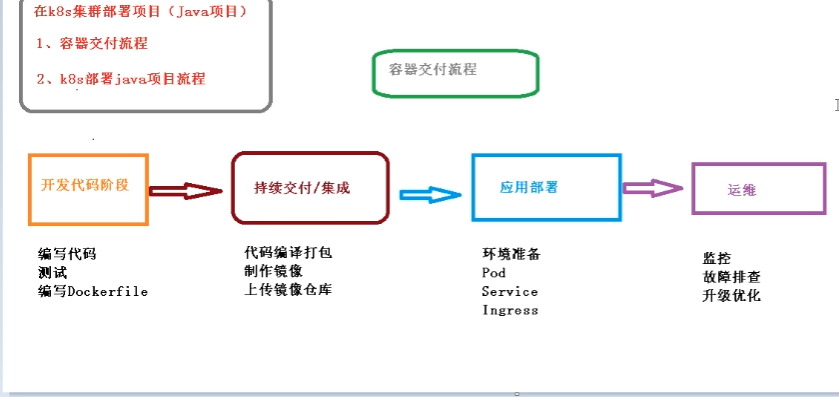


1. VIP（虚拟IP）

**在K8S上部署实际项目**

1. 部署基本过程

开发阶段->持续交付/集成->应用部署->运行维护



1. K8S部署项目细节流程

项目编译打包后通过Dockerfile打包成镜像->推送到镜像仓库（阿里云/网易）->控制器部署镜像（Deployment）->对外暴露应用（Service、Ingress）->运行维护

1. **后续需要搜索学习用k8s部署Golang项目的过程**