计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(12):268−271 [doi: 10.15888/j.cnki.csa.007682]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Kubernetes 的 CI/CD 平台①

陈　博, 周亦敏

(上海理工大学 光电信息与计算机工程学院, 上海 200093)
通讯作者: 周亦敏, E-mail: Zhouyimin_58@163.com

摘　要: 随着互联网的快速发展以及互联网业务、用户数量的不断增多, 越来越多的传统单体应用为了方便拓展新业务、增加可复用度, 已经选择将业务拆分为多个微服务, 这样可便于后期的管理和拓展. 但若以传统的方式在云平台去部署多个微服务是非常繁琐且消耗人力物力. 为了实现敏捷开发和快速部署, 减少开发与运维之间团队的时间损耗, 在分布式容器编排引擎平台 Kubernetes 的实验环境中, 研究在其中部署 CI/CD 流水线服务, 从而使代码到服务实现自动化构建.

关键词: 云计算; Kubernetes; 持续集成/部署; 平台即服务; 自动化运维

引用格式: 陈博,周亦敏.基于 Kubernetes 的 CI/CD 平台.计算机系统应用,2020,29(12):268–271. http://www.c-s-a.org.cn/1003-3254/7682.html

CI/CD Platform Based on Kubernetes

CHEN Bo, ZHOU Yi-Min

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: With the rapid development of the Internet and the increasing number of Internet businesses and users, more and more traditional monomer applications have chosen to split the business into multiple microservices for the convenience of expanding new business and increasing reusability, which is convenient for later management and expansion. However, it is very cumbersome to deploy multiple microservices on the cloud platform in traditional way, which consumes human and material resources. In order to realize agile development and rapid deployment, and reduce the time loss of team between development and operation and maintenance, this work studies the deployment of CI/CD pipeline service in the experimental environment of Kubernetes, a distributed container arrangement engine platform, so as to realize the automatic construction of code to service.

Key words: cloud computing; Kubernetes; CI/CD; PaaS; automatic operation

随着互联网规模的不断扩大, 越来越多的公司逐渐将业务的重点放在了互联网业务上, 如何合理的开发和管理海量的互联网业务成了当前的热点. 云计算的概念是 2006 年 8 月份由 Google 的 CEO 在搜索引擎大会上提出, 旨在为个人或组织提供虚拟化计算资源, 使得公司可以将自己的互联网业务托管于云服务商, 而不用构建自己的基础设施[1].

因为云的便利和业务量的上升, 越来越多的公司开始采用微服务的架构, 如 B 站推出了自己的微服务框架 Kratos, 阿里巴巴的 Spring Cloud Alibaba 等. 因为在早期传统的单体架构应用中, 往往各个应用之间具有高耦合度、扩展能力弱. 微服务的提出很好地解决了这一痛点[2]. 相比较传统的架构, 微服务架构能够更好地帮助企业将新的功能点快速的迭代插入到现有的

① 收稿时间: 2020-04-07; 修改时间: 2020-05-10; 采用时间: 2020-05-18; csa 在线出版时间: 2020-11-30

生产环境中去, 它减少了开发的复杂性以及部署的复杂性. 同时, 架构本身也降低了资源的消耗[3]. 表 1 是传统单体架构与微服务架构的对比. 微服务因为其拆分的原则, 往往一个业务会被拆分成多个微服务, 无论是开发或是部署都是非常的繁琐. Kubernetes 是一款分布式容器编排引擎[4], 它能够很好的管理各个服务, 可以自动实现容器伸缩, 方便运维人员对容器的管理, 是一个得到生产实践证明的容器编排管理系统. 本文就将基于 Kubernetes, 研究在其上构建持续集成持续部署的自动化流水线平台的最佳实践.

表 1 单体架构与微服务架构的对比

单体架构	微服务架构
耦合度高	耦合度低
可重用性低	各个服务可按需复用, 灵活性高
部署速度慢, 迭代慢	部署速度快, 迭代快
扩展能力弱	扩展能力强
阻碍技术创新	以服务为粒度独立演进, 有更多自主权

1 Kubernetes 架构

在研究基于 Kubernetes 的 CI/CD 平台之前, 我们有必要先了解下它的整体架构及运作方式, 以便于我们更好的针对其架构特点设计出更符合其特性的 CI/CD 流水线. Kubernetes(简称 k8s) 是 Google 使用 go 语言开发的一个自动化容器操作的开源平台. 使用 Kubernetes 可以:

(1) 自动化弹性构建容器;

(2) 自动管理容器;

(3) 提供容器之间的负载均衡;

(4) 方便对容器版本回滚更新;

(5) 易于扩容.

1.1 Kubernetes 组件

Kubernetes 的集群主要由若干个 Master 节点和 Node 节点构成. Master 节点在其上运行相应的 Master 组件和 Node 组件, Node 节点运行 Node 组件, 图 1 是 Kubernetes 集群的架构图.

Master 组件是集群的管理控制中心[4], 如下是 Master 组件:

(1) Kube-Apiserver[5]: 提供 Restful 风格的 API 接口, 通过它我们可以对 k8s 的资源对象进行增删改查, 同样 apiserver 也是 k8s 集群的数据总线和数据中心.

(2) Kube-Scheduler: 负责分配调度 Pod 到集群内的节点上, 它监听 Kube-Apiserver, 查询还未分配 Node 的 Pod, 然后根据调度策略为这些 Pod 分配节点.

(3) ETCD: 是一个高可用的分布式键值数据库, Kubernetes 集群使用其作为它的数据后端.

(4) Kube-Controller-Manager: 集群内部的管理控制中心, 它会及时发现并执行自动化修复流程, 确保集群始终处于预期的工作状态.

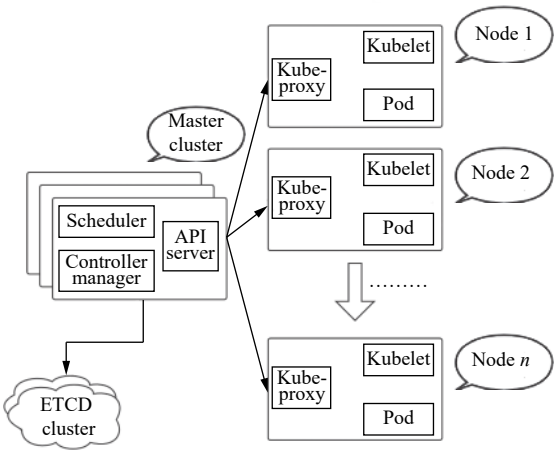

图 1 Kubernetes 集群的架构图

Node 组件在每个节点上运行, 维护运行的 Pod 并提供 Kubernetes 运行时环境.

(1) Kubelet: 是主要的节点代理, 它监测已分配给其节点的 Pod.

(2) Kube-Proxy: 在每个节点上运行网络代理, 并反映每个节点上 Kubernetes API 中定义的服务.

在该分布式系统中, 各个服务运行在 node 节点上, 由 master 节点自动管理. 所以考量将 CI/CD 服务放置集群中运行, 方便自动化运维, 镜像仓库因需要频繁读写操作, 可放置于集群外进行管理, 减少对 Kubernetes 集群的压力. Kubernetes 使用 docker 来进行容器的管理和云上的自动运维, 减少了相应的成本, 也不会再生产相应的环境冲突了, 总而言之是一种非常便利的工具[6].

2 CI/CD 平台设计

CI 即持续集成是指开发将代码提交到 GIT 服务器上, 会触发一次集成服务器的相关功能, 比如编译、测试、输出结果等, 往往一天内会有多次的集成, 确保新增代码能与原先代码正确集成, 这样有利于及时检查代码的缺陷. CD 即持续部署是指通过自动化部署的手段将软件功能频繁的进行交付, 加快了代码的上线速度.

2.1 CI 工具选型

往往持续集成持续部署是相继进行的 CI 的常用工具有 Jenkins、Circle CI、Codeship 等, Jenkins 因其开源和完善的社区丰富的插件被广大公司所采用, 因此本文选型 CI 工具为 Jenkins.

2.2 版本控制

在持续集成中, 版本控制是不可或缺的一部分. 若部署时, 代码发生灾难性缺陷, 通过使用版本控制可及时回溯至上个正常的代码版本. 在常用的版本控制中, GIT 含有的是分布式代码库与文件快照的设计思想, 相对于传统 CVS、SVN 等集中式、文件差异式版本控制工具是一种挑战与颠覆[7]. 所以本文采用 GIT 作为版本控制器, 在实验中将把代码托管给 Github.

2.3 集成测试

在持续集成中, 测试是必不可少的部分, 若未经过测试直接集成于生产环境, 会有重大的隐患. 在构建过程中考虑从仓库下拉代码完毕后执行开发写完的单元测试用例, 并生成 XML 格式的测试报告给 Jenkins 识别. 测试成功则继续后续的构建步骤, 若失败则退出构建.

2.4 CI/CD 应用与 Kubernetes 关键点

Jenkins 集群为 Master 和 Agent 节点, 在 Kubernetes 中, 所有服务均为容器化构建并由 Kubernetes 管理, 所以采用容器化搭建 Jenkins Master 和 Agent 服务, 并托管与 Kubernetes 中. 当用户触发一次 CI 时, Jenkins Master 节点会向 Agent 节点派送相应的 CI 任务. 考虑到在空闲时, Agent 节点并无用处, 故采用动态构建 Agent 节点, 在有 CI 任务时由 Jenkins Master 向 Kubernetes Apiserver 发出构建请求, 可由 Jenkins 中 Kubernetes 该插件完成.

2.5 构建具体流程

我们将 CI/CD 的步骤可分为如下几步[4], 流程如图 2 所示.

(1) 开发将代码提交至 GIT 仓库.

(2) 代码发生变化后触发 GIT HOOK, Jenkins Master 节点请求 Kubernetes Apiserver 生成新的 Jenkins Agent Pod.

(3) Jenkins Slave Pod 生成后开始构建 Jenkins Master 节点下发的任务, 从 GIT 仓库中拉取代码.

(4) 代码拉取成功后, 开始执行单元测试, 单元测试成功则继续执行, 失败则退出构建任务.

(5) CI 服务器根据预先定义的 Pipeline 文件, 将代码进行编译.

(6) 编译完成后, 打包成镜像将镜像推送至镜像仓库, 并打上最新标签.

(7) CI 调用 Kubernetes Cli, 将预先设定好的 Deployment 中的镜像更改为刚刚构建已推送至私用仓库的镜像, 从而完成部署.

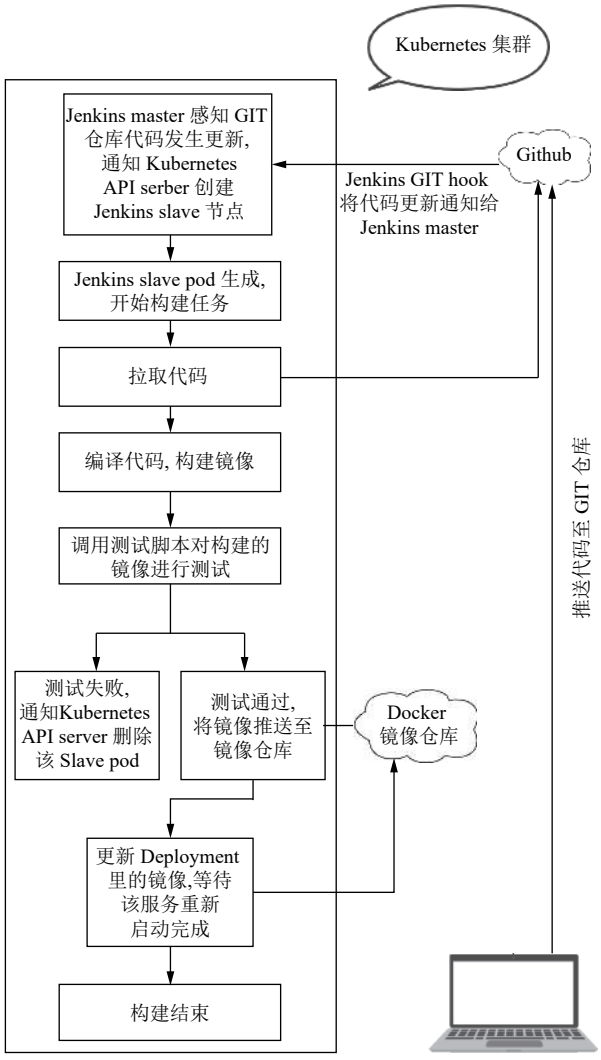

图 2 构建流程

3 实验测试

实验环境为 3 台已组成 Kubernetes 集群的虚拟机. 系统为 Ubuntu19.04 并对外暴露集群 IP 为 192.168.11.31. 已将第一版本的代码, 使用 Flask 框架编写的后端打包部署至该集群 lab 的命名空间里, 并通过 NodePort 的方式对外暴露服务, 端口为 30010, 模拟生产环境的后端接口. 此时我们使用 curl 192.168.11.31:30010 命令,

可返回预先设置好的字符串“This is first version code!”

(1) 修改代码, 设置返回字符串为“This is second version code!”

(2) 提交代码并推送至 GIT 仓库.

(3) 发现 Jenkins Slave Pod 已经自动生成并开始构建任务.

(4) 稍等片刻, Jenkins Slave Pod 显示终止中, 查看 Jenkins console output, 发现已成功完成构建任务.

(5) 再次尝试 curl 192.168.11.31:30010, 返回的字符串是“This is second version code!”, 即服务端代码已成功更新.

实验测试过程如图 3 所示.

图 3 实验测试

4 结论与展望

本文通过实践可以发现, 比起传统的需要人力去打包并手动部署到服务器上从而需要大量的人力物力, 该 CI/CD 流水线仅需开发人员上传代码后即能自动打包部署至相应的环境中, 展现了符合期望的基于 Kubernetes 的 CI/CD 自动化流水线, 可大大提升开发到交付的效率, 并且可实现高并发弹性构自动化建流水线. 对于拥有多个在 Kubernetes 集群中的服务、需要快速迭代频繁修改代码并部署的项目具有省时省力且不容易出错的优势.

参考文献

1 Kovács J, Kacsuk P, Emődi M. Deploying Docker Swarm cluster on hybrid clouds using Occopus. Advances in Engineering Software, 2018, 125: 136–145. [doi: 10.1016/j.advengsoft.2018.08.001]

2 Taherizadeh S, Grobelnik M. Key influencing factors of the Kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications. Advances in Engineering Software, 2020, 140: 102734. [doi: 10.1016/j.advengsoft.2019.102734]

3 苏瑾, 田建斌. 云环境中 Web 应用的微服务架构. 电子技术与软件工程, 2019, (15): 131–132.

4 朱小亮. 基于容器引擎的云平台设计与实现 [硕士学位论文]. 北京: 北京工业大学, 2019.

5 王伟军. 基于 Kubernetes 的容器云平台建设. 电脑知识与技术, 2019, 15(36): 47–48.

6 陈春辉. 基于 Kubernetes 实现云上自动化运维. 科学大众, 2018, (11): 12.

7 杨振华. 软件持续交付平台的研究与实践 [硕士学位论文]. 北京: 北京邮电大学, 2016.