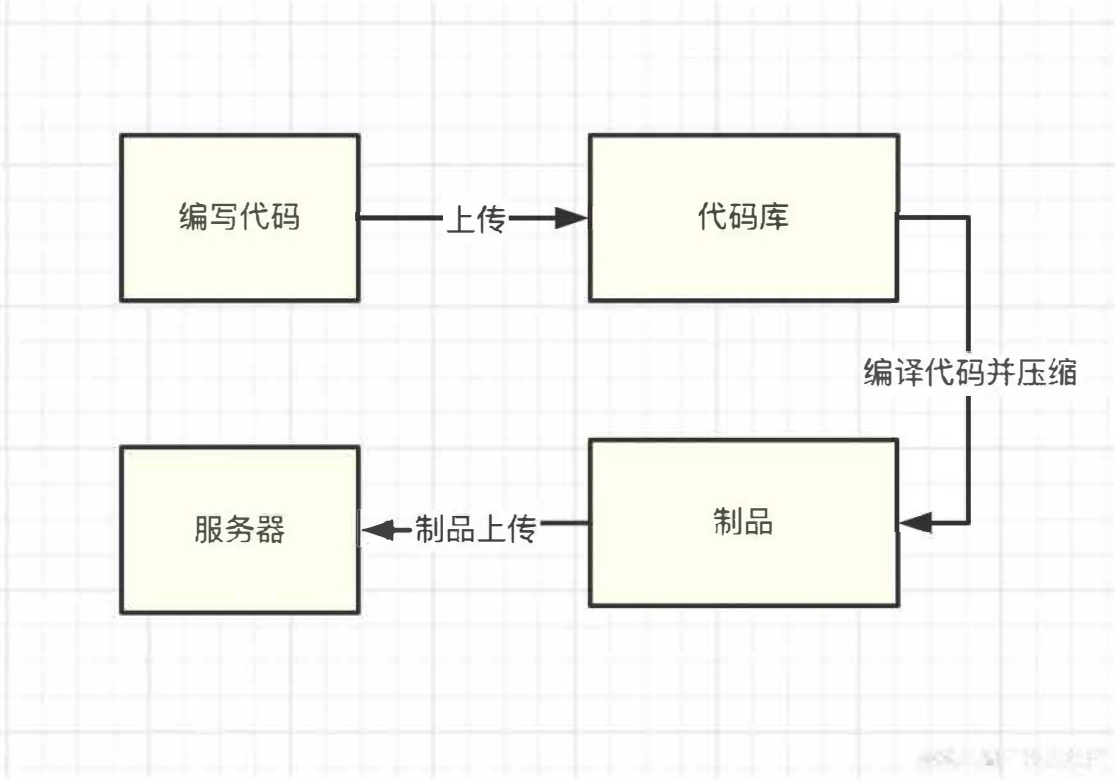
什么是 CI/CD

在开发阶段，许多编译工具会将我们的源码编译可使用的文件。例如 `vue-cli` 的项目会被 `webpack` 打包编译为浏览器的文件， `Java` 项目会被编译为 `.class/jar`文件以供服务器使用。

但是，开发人员过多关注构建和部署过程是很浪费时间的。以之前古老的的构建部署流程为例子，需要经历以下步骤：

1. 开发人员将源代码，经过编译、压缩等一系列流程打包为**制品**（意思为打包后的成品）
2. 将制品上传到服务器。
3. 在服务器将编译后的文件，手动可用的容器服务内（例如 `Nginx`，`Tomcat`，`Apache` 等服务）



显而易见，这种流程不仅繁琐，且容易出错，是非常影响开发效率的。开发人员要花一些时间浪费在这上面。那么有没有高效率，简单便捷一些的方式呢?

制品库里面。常用工具有 Gitlab CI，Github CI，Jenkins 等。这个环节不参与部署，只负责构建代码，然后保存构建物。构建物被称为 制品，保存制品的地方被称为 “制品库”

CD 则有2层含义： 持续部署（Continuous Deployment） 和 持续交付（Continuous Delivery） 。 持续交付 的概念是：将制品库的制品拿出后，部署在测试环境 / 交付给客户提前测试。 持续部署 则是将制品部署在生产环境。可以进行持续部署的工具也有很多： Ansible 批量部署， Docker 直接推拉镜像等等。当然也包括我们后面要写到的 Kubernetes 集群部署。

为什么要学 CI/CD

相信大家在了解它们的用途后，会有几点以下疑问：

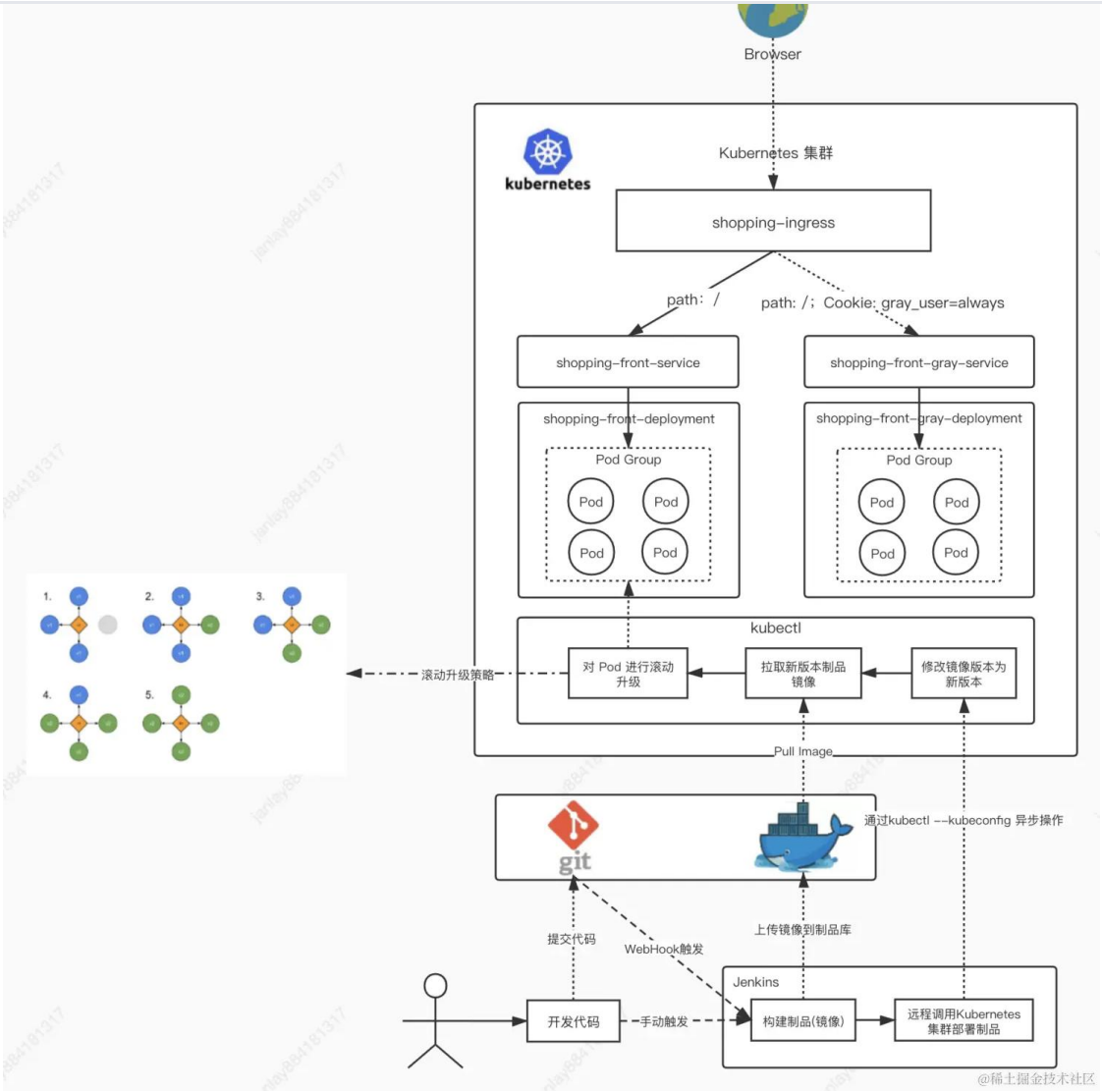
- 这不是运维干的活吗?
- 好像和业务代码不相关，那我了解它有何意义?
- 全是服务器知识，我不了解相关知识怎么学习?

相信这是许多前后端同学一致的疑问。的确，对于曾经的我，也有过这些疑问。门槛高，和工作内容不相关。那他的意义在哪里?

但是当我通过学习这些知识和在团队中实践这些流程后，我在知识面上得到了很大的扩展。对操作系统，对实际的构建部署，甚至对工程化拥有了全新的认识。甚至可以提出建议，如何更好的优化这些流程。这些都是你可以获得成长和学习的地方。你也可以选择将这部分知识点写入你的简历，作为面试和筛选的加分项。从更高的角度看整个项目的全貌，往往产生思考的维度是和一般的角度不同的。你会成长更快，渐渐地突破思维天花板。

当然，如果你对 Linux 操作系统不是很熟悉，建议先补习下基础的系统安装，操作命令，基础概念等知识（系统推荐 CentOS / Ubuntu ），在小册中将不会对基础 Linux 命令有过多解释。当然，如果遇到部分不懂的现场搜索也可以，相信你学起来这部分知识可以更加得心应手。

小册整体架构设计



上面是一张全景架构图，小册内容和章节将围绕该图展开编写内容。其中不包含单元测试和代码扫描环节，只关注构建和部署环节。

换成文字叙述就是这样的：

1. 你写完了代码，提交到了 `Git` 代码库
2. 随后，代码库配置的 `WebHook` 钩子或人工手动启动了 `Jenkins` 的构建流程
3. `Jenkins` 启动构建流程。按照你之前配置好的构建脚本，将代码编译成功。
4. 编译成功后，将编译后的文件打包为 `docker` 镜像，并将镜像上传到私有镜像库。
5. 随后，使用 `kubectl` 指定远程的k8s集群，发送镜像版本更新指令
6. 远程的k8s集群接收到指令后，去镜像库拉取新镜像
7. 镜像拉取成功，按照升级策略（滚动升级）进行升级，此时不会停机。
8. 升级完毕。

服务器搭配方案

系统选用 CentOS 7：mirrors.aliyun.com/centos/7.9....

1. 全本地虚拟机 / 全上云

这里所有主机都必须为云服务器/本地虚拟机。要保持统一

配置	技术栈	类型	标签
2核4G	Jenkins + Nexus + Docker	本地虚拟机 / Cloud	构建机
2核4G	Docker + Kubernetes	本地虚拟机 / Cloud	Kubernetes Master
1核1G	Docker + Kubernetes	本地虚拟机 / Cloud	Kubernetes Node

2. 半云半本地虚拟机

构建机器放本地，要部署的机器放云上面。否则的话构建机找不到要部署的机器
缺点：无法使用 Git 的 Webhook

配置	技术栈	类型	标签
2核4G	Jenkins + Nexus + Docker	本地虚拟机	构建机
2核4G	Docker + Kubernetes	Cloud	Kubernetes Master
1核1G	Docker + Kubernetes	Cloud	Kubernetes Node

上一章 下一章

留言