# 微服务Devops实战

## 1.Devops是什么？

Devops是让技术团队，运维，测试等团队实现一体式流程自动化；可以实现持续集成和持续交付、版本控制、敏捷的软件开发、基础结构即代码、配置管理、持续监视等功能；

### 1.1：持续集成CI

#### 1.1.1：持续集成的含义

持续集成：从开发—编译—测试—发布的完成自动化流程,持续频繁的多次将本地代码集成到主干分支，并且保证主干分支可用,代码集成到主干之前必须通过编译，代码扫描，安全扫描，自动化测试等操作,如果出现失败,则不能进行集成。

#### 1.1.2：持续集成的好处

通过自动化流程提高集成速度，在传统的研发过程中，一般像代码编译，测试，部署都是通过手动进行操作，自动化活，可以大幅度提高集成频次，同时减少维护手动脚本带来的低级问题。

通过自动化可以将问题前置，在每次提交时候，都会触发编译、测试，能够更快的发现问题。

#### 1.1.3：持续集成触发方式

自动触发，通过自动化的CI服务器或者工具，自动监听代码库Git Push或MR等事件触发；

#### 1.1.4：持续集成的工具

Jenkins、GItlabCi…,这里目前主要以jenkins和gitlabCI为主

#### 1.1.5：gitlab-ci和jenkins的区别

* 分支的可配置性

使用gitlab-ci新创建的分支无需任何进一步配置就可以立即使用ci管道中的自定义作业实现，jenkins实现这个效果需要在jenkins中安装gitlab多分支流水线的插件来实现；

* 定时任务构建

在jenkins可以立即使用,可以在应执行作业或管道的那一刻以cron语法定义；gitlab没有此功能,但是可以通过一种变通的方式进行实现,通过webAPI使用同一台服务器上的cronjob触发作业和管道

* 分支的拉去和请求

jenkins没有与源代码管理系统进一步集成,需要管理员自行写代码或者插件实gitlab与其ci平台紧密集成,可以方便查看每个开关和关闭拉动请求的运行和完成管道

* 权限管理

由于gitlab和gitlab-ci深度整合，权限可以统一管理；jenkins没有内置的存储管理器，因此无法直接进行设置

* 存储库

Jenkins与存储库管理器都是松散耦合的，因此在选择版本控制系统非常灵活

Gtilab-ci是git存储管理器gitlab固定的组件，因此在CI/CD流程和存储库功能之间提供了良好的交互

* 插件管理

Jenkins的本机功能是通过插件完成的，插件的维护，保护和升级成本很高

Gitlab是开放式的，任何人都可以直接响应代码，一旦合并，将自动测试并且维护每个更改;

#### 1.1.6：jenkins和gitlab-ci的特点

Gitlab特点：

* 轻量级,不需要复杂的安装手段
* 配置简单,UIgitlab直接适配
* 实时构建日志十分清晰,UI交互体验很好
* 使用YAML进行配置,任何人都可以方便的使用
* 没有统一的管理界面,无法统筹管理所有的项目
* 配置依赖于代码仓库,耦合度没有jenkins低

Jenkins的特点：

* 编译服务和代码仓库分离,耦合度低
* 2.插件丰富,支持语言众多
* 3.有统一的web管理界面
* 4.插件以及自身安装较为复杂
* 5.体积量较大,不是很适合小型团队

#### 1.1.7：面临的问题和阻碍

* 1. 执行环境异构导致结果差异
  2. 没有严格的测试要求
  3. 非常慢的构建和测试速度

### 1.2：持续交付

#### 1.2.1：持续交付的含义

持续交付：包含持续集成，并且增加将项目部署到对应环境的自动化流程，持续频繁的将软件的新版本交付到预生产环境，交付给测试、产品验收；不管软件如何进行更新，都是随时可以进行交付的，

#### 1.2.2：触发方式

手动触发，通过研发平台手动触发，一般交付的结果是一个二进制包或者镜像

#### 1.2.3：面临的困难

* 1. 很难保证和线上的环境是一致的；
  2. 大量的联调环境问题
  3. 很难保证充分的自动化测试

### 1.3：持续部署

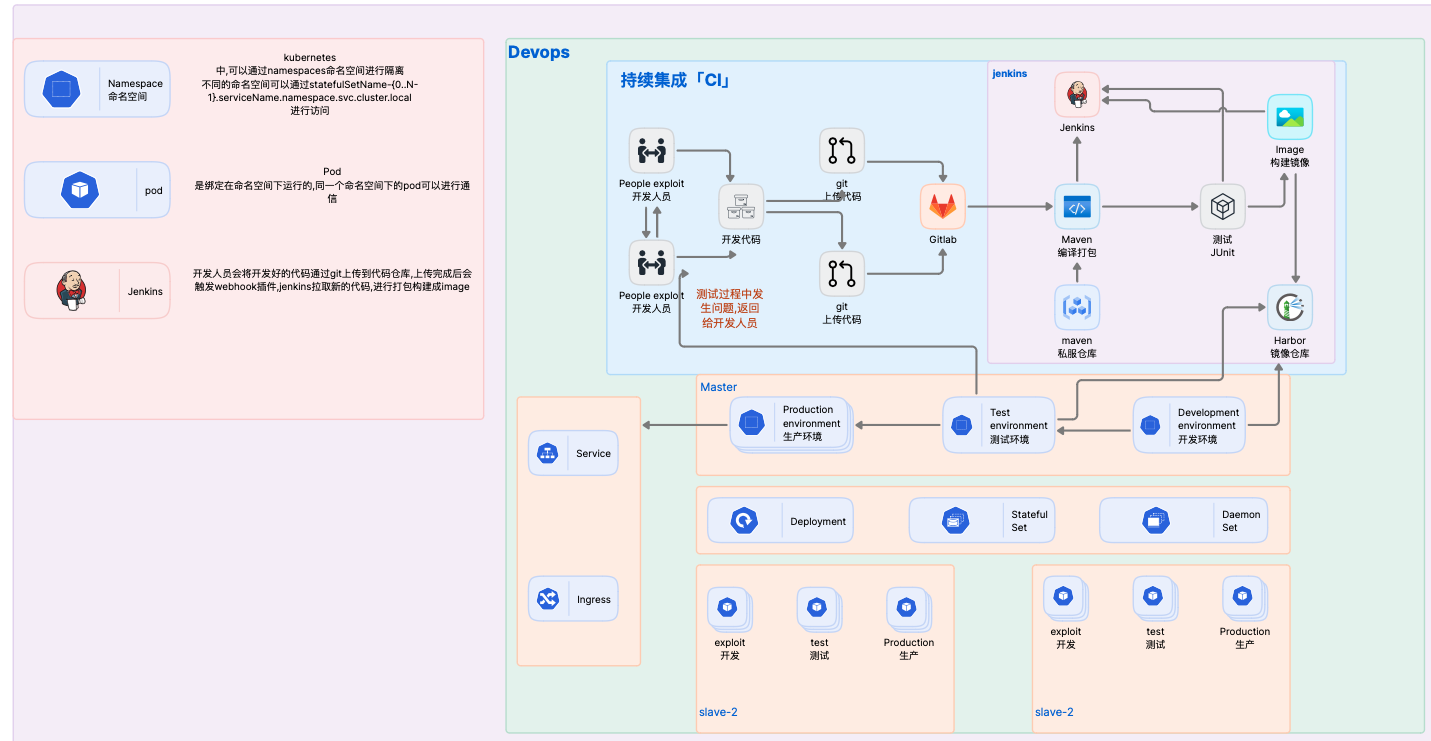
#### 1.3.1：持续部署的含义

持续交付的下一步，自动将代码部署到生产环境

#### 1.3.2：触发方式：

自动触发，通过研发平台配置定时任务，自动获取交付产物进行部署；

### 1.4：CI/CD架构图



#### 1.4.1:架构剖析

上图中主要采用，Kubernetes+gitlab+jenkins+maven+sonarqube+harbor实现的CI/CD流程，开发人员将代码编写完成后，通过git上传到gitlab代码仓库中，会触发webhook插件。Jenkins会将代码拉去到存储库中通过maven进行编译和打包，打包完成后通过sqonarqube进行代码审计，从而生成二进制包或者镜像，如果生成镜像，会将镜像上传到harbor镜像仓库中，kubernete会将镜像拉取到本地，通过代码要求进行创建Kubernetes资源，从而进行测试，再到预生产环境，再到生产环境，在这整个过程中，无论是哪一步出现问题，都可以将代码返回给开发人员进行调整和修改；

### 1.5：环境搭建

### 1.5.1:环境准备

准备一个三台云服务器或者虚拟机搭建的Kubernets集群，版本在1..23.X以上

|  |  |
| --- | --- |
| ECS-1(master) | Gitlab代码仓库 |
| ECS-2(slave-1) | Jenkins+sonarqube+dashboard（K8S集群中部署） |
| ECS-3(slave-2) | Harbor镜像仓库 |

### 15.2：部署过程

参考文档：【有道云笔记】Devops实战.md <https://note.youdao.com/s/NIKK7dxU>

### 1.5.3：演示环境

Dashboard：

url：<https://117.73.9.208:32047/#/login>

token：

eyJhbGciOiJSUzI1NiIsImtpZCI6IjJ5Q0RLZjlnNjMyeTB1TktGMnNnZ2oxRVJCeF9EbHRGSExmRjRHbThEUVUifQ..MOHSPIl9VyP34S5Lluo4aiZi7zJbSGF9wUOeIWxluSiaNFudXWT3FPQ37Smkf0fabdPQcvgS6tG9wiieI1Wah8zHey\_kaqjVn49vxp9AUe9zaRtCCdoKzZCv-ndWZn4RsO0QhbOk949tRe541dYdzILJrH-mS4zZtvvpcfUy\_3qFfr\_dyme-jm\_1CJzTn3vpXGjkCiYkf9P44nZ\_nYrdkDfJ6J1FsFkTKX-CPmHAA3Yu593OXkDcALEbDxD-xUMqlJmKUf9REqYvaPMtCXkgasdxCiydvyl7HZQIaiZVbaMVXn5tXF2LjCfcPNlq0rPAGqYKfFy9TcLEce5iMw5krQ

gitlab：

url: http://117.73.9.208:28080/users/sign\_in

user: root

password: Fan990929

Harbor：

url：http://117.73.12.18:28081/harbor/projects

user: admin

password: Harnor12345

sonarqube：

url: http://117.73.9.208:30830/projects/create

user: admin

password: Fan990929