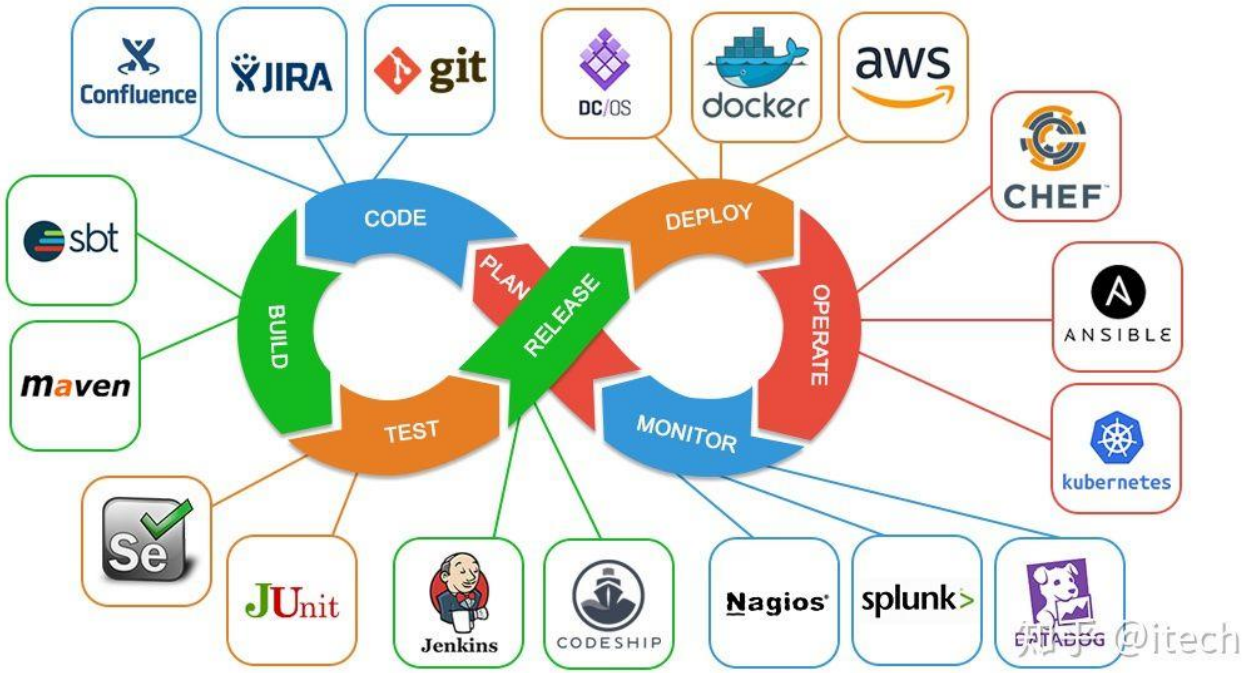
浅谈 DevOps

汇报人：sunling

目录

CONTENT

浅谈 DevOps



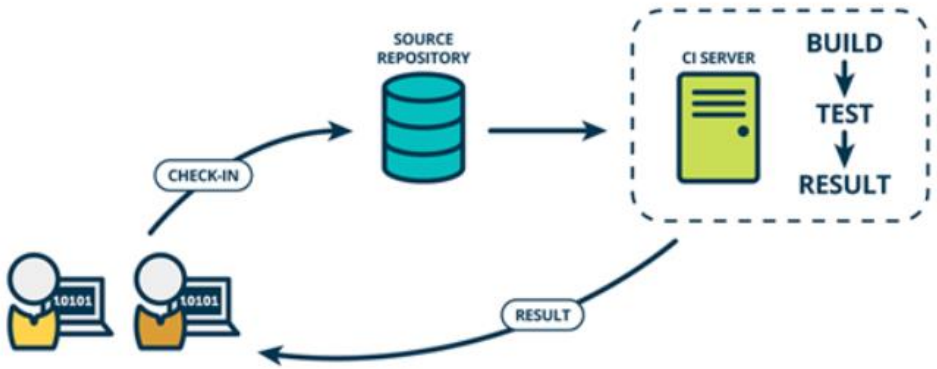
从字面上来看，“DevOps”一词是由英文 Development（开发）和 Operations （运维）组合而成，是一种方法论，是一组过程、方法与系统的统称，用于促进应用开发、应用运维和质量保障（QA）部门之间的沟通、协作与整合。

CI/CD 是一种通过在应用开发阶段引入自动化来频繁向客户交付应用的方法。CI/CD 的核心概念是持续集成、持续交付和持续部署。作为一个面向开发和运营团队的解决方案，CI/CD 主要针对在集成新代码时所引发的问题。

认识 CI/CD

CI 持续集成（Continuous Integration）

开发人员将会频繁地向主干提交代码，这些新提交的代码在最终合并到主干前，需要经过编译和自动化测试流进行验证。



CD 持续交付（Continuous Delivery）

完成 CI 中构建及单元测试和集成测试的自动化流程后，持续交付可自动将已验证的代码发布到存储库。

CD 持续部署（Continuous Deployment）

对于一个成熟的 CI/CD 管道来说，最后的阶段是持续部署，即自动部署到生产环境，降低人为的风险。

Jenkins 介绍

当 SVN/Git 提交完成后，触发钩子，进行编译工作；（另有每晚凌晨定时）除编译工作以外，Jenkins还会承担各种有自动化需求的工作，如测试环境的部署。 它是开发、运维连接的核心桥梁。

Jenkins 练习环境：

http://120.26.178.242:8080/

Username: xdlr208
Password: #03130313#

GitHub Actions

大家知道，持续集成由很多操作组成，比如抓取代码、运行测试、登录远程服务器，发布到第三方服务等等。

GitHub 把这些操作就称为 actions。使用 GitHub Actions 来创建自定义的持续集成 (CI) 工作流程，以构建并测试以不同编程语言编写的项目。

官方文档： https://docs.github.com/cn/free-pro-team@latest/actions/guides

Travis CI 介绍

Travis CI是在软件开发领域中的一个在线的，分布式的持续集成服务，用来构建及测试在GitHub托管的代码。这个软件的代码同时也是开源的，可以在GitHub上下载到，尽管开发者当前并不推荐在闭源项目中单独使用它。

WorkShop 项目

使用 Jenkins + Hugo + GitHub 持续集成静态博客

使用 GitHub Actions 自动同步码云仓库

使用 GitHub Actions 持续集成代码规范

使用 GitHub Actions 自动同步码云仓库

适用场景：访问 GitHub 网络不便的时候，使用访问较快的码云仓库作为备用仓库，提升开发效率和使用体验。

在开源贡献的代码托管的过程中，我们有时候有需要将Github的代码同步到其他远端仓库的需求。具体的，对于我们目前参与的项目来说核心诉求是：**以Github社区作为主仓，并且定期自动同步到Gitee作为镜像仓库**。

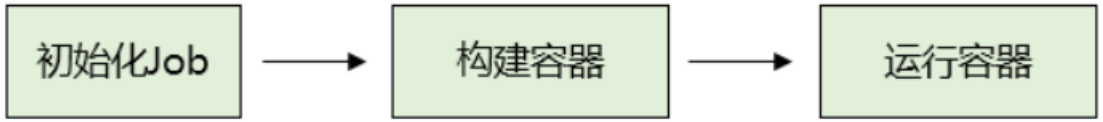
手动同步：繁琐、增加人力、不够智能

自动同步：写脚本自动同步、应用了持续集成（CI）的思想

使用 GitHub Actions 自动同步码云仓库

Github Actions 提供了2种方式去实现Actions：

1, Docker container：这种方式相当于在Github提供的计算资源起个container，在container里面把功能实现。



2, JavaScript：这种方式相当于在Github提供的计算资源上，直接用JS脚本去实现功能。

使用 GitHub Actions 自动同步码云仓库

在这里我们使用 Docker 镜像来作为自动同步的方法。

这是脚本的实现原理：

1.通过Github APIs读取Repo列表。
2.下载或者更新Github repo的代码
3.设置远端分支(码云)
4.将最新同步的commit、branch、tag推送到Gitee。

使用 GitHub Actions 自动同步码云仓库

步骤：（2，3，4，5 步骤都是为了调用 GitHub APIs 使两者实现通信）

0，在系统生成本地的公私钥文件，等待后续使用。

1，新建一个 repo，在根目录下新建 .github/workflows/ 目录，用来存放脚本文件（.yaml）

2，在 GitHub 新建一个 GITEE_PRIVATE_KEY 变量用来存储私钥

3，同时将对应的公钥保存在码云的 GITEE 变量中

4，在码云上创建一个 Token，记录它的值

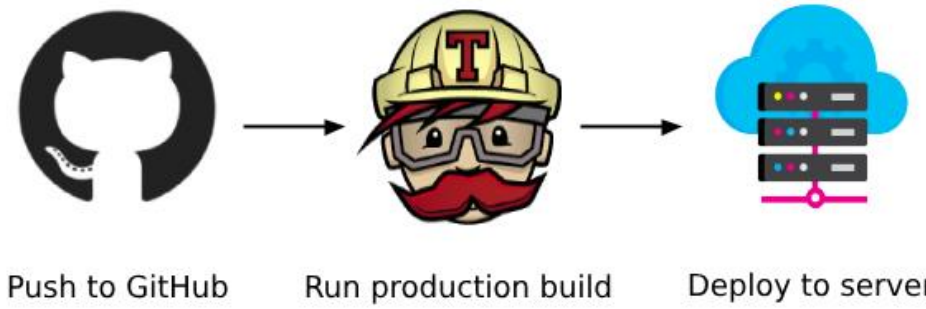
5，在 GitHub 新建一个 GITEE_TOKEN 变量用来存储 Token

6，在 .github/workflows 下新建 auto.yml 脚本文件

7，推送至主仓库（GitHub）

8，等待 GitHub Action 构建脚本，让备用仓库（Gitee）同步更新上游仓库

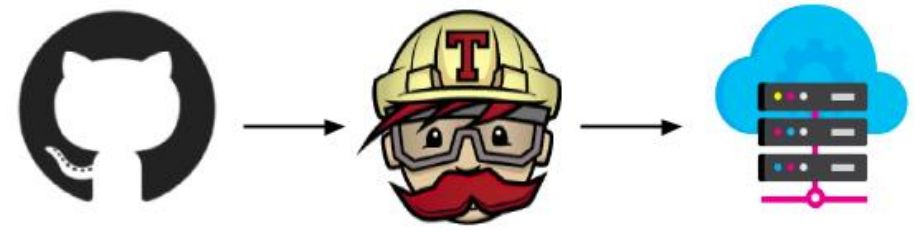
使用 Travis CI 持续集成代码规范



使用背景：在多人开发协作中，良好的代码规范有助于提升项目的可维护性、代码的可读性。常见的代码规范包括但不限于 commit 规范、语法规范、文档规范等。

工作原理：利用 Travis CI 持续集成的特性，将仓库的每次 commit 和 push 都集成到 CI 服务器上，根据脚本文件（.travis.yml）构建对应的代码规范规则。若符合规范，则显示通过的选项；若不符合规范，则显示不通过的选项。

使用 Travis CI 持续集成代码规范



以持续集成 Markdown 语法规范为例：

1，新建一个 repo，然后在根目录新建一个 .travis.yml 脚本文件

2，在文件写入以下内容

3，push

4，到 Travis CI 面板查看构建信息

16 lines (12 sloc) | 229 Bytes

```
language: go

go:
  - "1.10.x"

install: true

stages:
  - markdownlint

jobs:
  include:
    - stage: markdownlint
      script:
        - gem install mdl --version 0.9.0 ## mdl 0.1.0 有 bug，指定版本 0.9.0
        - mdl .
```

谢谢

汇报人：sunling