GIT+jenkins+docker持续自动部署方案

| 版本号 | 修订人 | 修订说明 |
| --- | --- | --- |
| V0.1.1 | 欧阳智杰 | V1版本 |
|  |  |  |
|  |  |  |

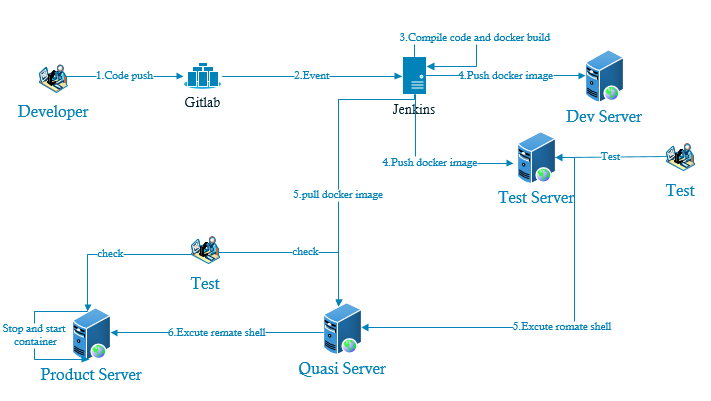
# 前言

随着业务的增长，需求不断增加，需求大小开发周期，发布时间都不一致，基于微服务的系统架构，功能叠加，对应的服务的数量不断增加，大小功能的快速迭代，更新要求部署的快速化，智能化，统一化。因此，传统的人工部署已经心有余而力不足。

持续集成，持续部署，持续交互对于微服务开发团队来说，是提高团队整体效率不可缺少的一环，合理的使用CI（Continuous integration），CD(Continuous deployment/Continuous delivery)

# GIT+Jenkins+Docker

持续集成网络结构图：



结构内容：

1.开发人员代码推送到gitlab；

2.gitlab中的事件推送到jenkins；

3.jenkins获取源码，编译，打包，构建镜像

4.jenkins push镜像到开发，测试环境

5.准生产执行脚本拉取测试环境代码或直接pull docker 镜像

6.执行脚本推送代码到生产服务器，停止老版本容器，启动新版本容器

# 安装持续集成环境

系统环境：Centos7.2、JDK SE 1.8 、Maven3.1.1、Jenkins2.4.1、GIT、Docker、docker-compose、gitlab

环境说明：

|  |  |  |
| --- | --- | --- |
| 机器IP | 用途 | 软件包 |
| 172.16.13.100 | 本地docker仓库 | Docker-registry |
| 172.16.13.101 | 代码库，maven打包war，jar | Gitlab、maven、docker、jdk、git |
| 172.16.13.102 | 持续集成工具 | Jenkins、maven、jdk、git |
| 172.16.13.103 | 业务部署测试环境 | Docker、Docker-compose |

# GITLAB部署及配置

# Jenkins部署及配置

插件下载：

maven integration plugin,docker-build-step,docker plugin,gitlab hook plugin,gitlab plugin

PHP：

Java：

基于maven构建镜像，上传到私有仓库

maven 构建镜像配置

pom.xml 文件添加docker插件

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>0.4.11</version>

<configuration>

<imageName>${docker.image.prefix}/${project.artifactId}</imageName>

<imageTags>

<imageTag>${project.version}</imageTag>

<imageTag>latest</imageTag>

</imageTags>

<dockerDirectory>src/main/docker</dockerDirectory>

<resources>

<resource>

<targetPath>/</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

</configuration>

</plugin>

${docker.image.prefix} 是镜像的前缀

${project.artifactId} 是镜像名字

${project.version} 版本号，此处也用来当做镜像的版本号

docker-maven-plugin使用

src/main/docker 目录下新增文件 Dockerfile

FROM frolvlad/alpine-oraclejdk8:slim

VOLUME /tmp

ADD demo-service-ver-0.0.1.jar app.jar

RUN sh -c 'touch /app.jar'

ENV JAVA\_OPTS=""

ENTRYPOINT [ "sh", "-c", "java $JAVA\_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ]

以上的配置可以把一个服务打包成镜像，只需要执行

$ sudo mvn package docker:build