CI/CD Pipeline 原理、使用工具与案例

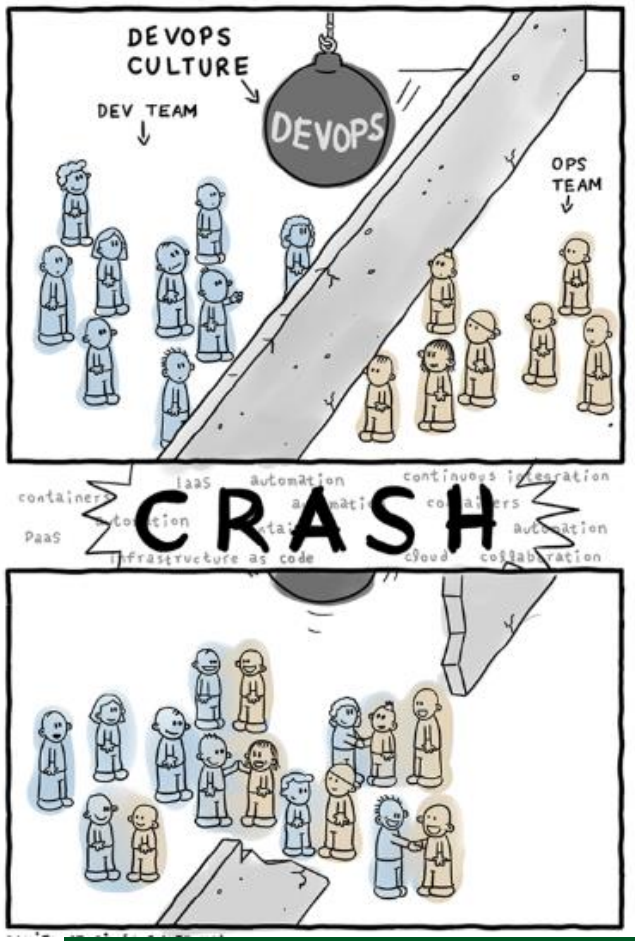
19215062　　钟展辉

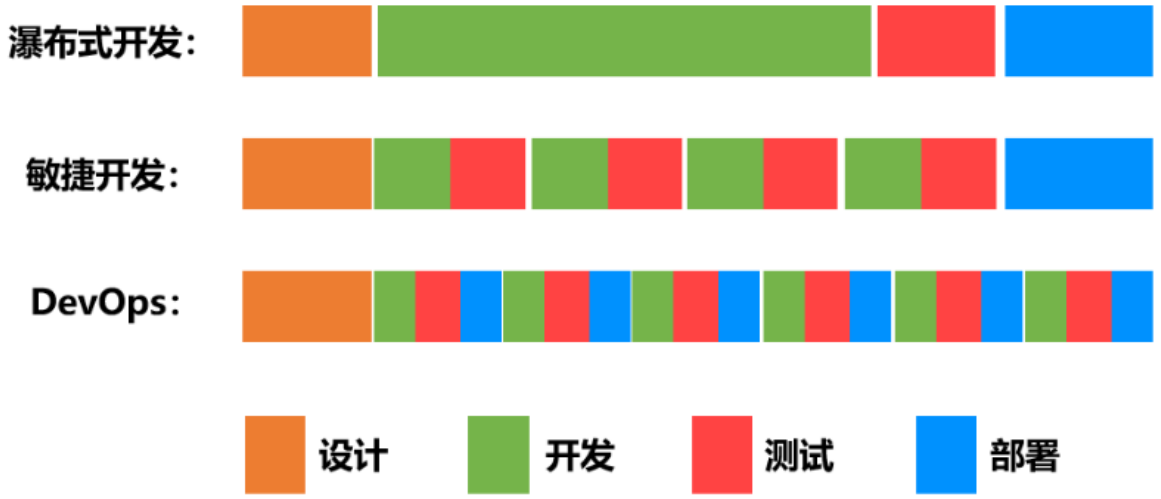
目录

1.DevOps简介

DevOps（开发与运维 – Development and Operations）

突出重视软件开发人员和运维人员的沟通合作，通过自动化流程来使得软件构建、测试、发布更加快捷、频繁和可靠。

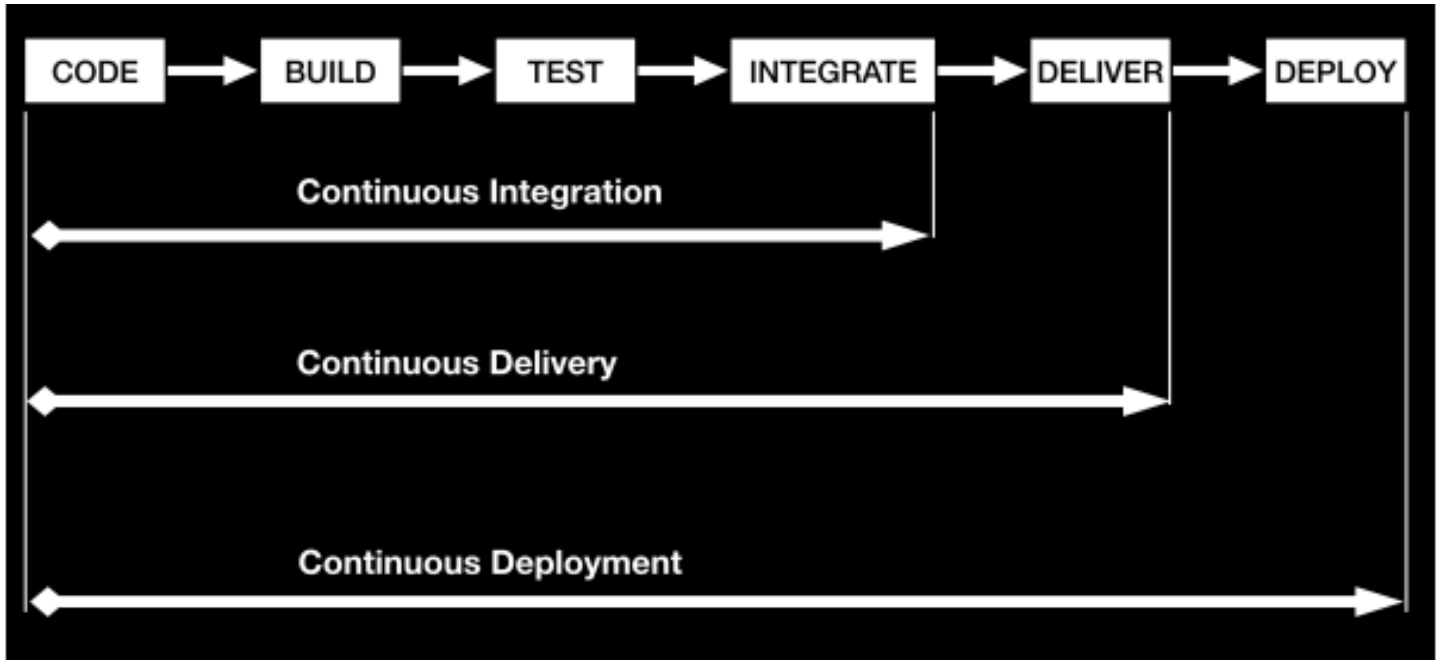




2. CI/CD Pipeline原理

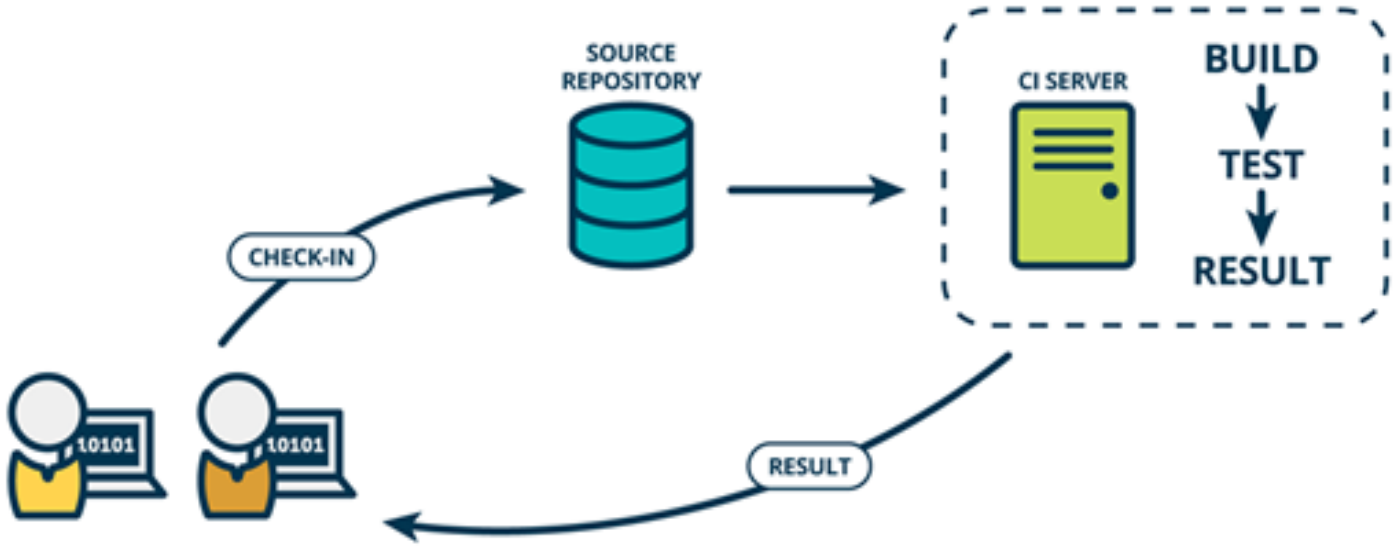
CI/CD 流水线

有两个词经常会伴随着DevOps出现，那就是CI和CD。
Continuous Integration（持续集成）
Continuous Delivery（持续交付）
Continuous Deployment（持续部署）



持续集成（CI – Continuous Integration）

不同的开发人员各自编写自己负责部分的代码，然后上传到源代码库中合并， CI 服务器负责构建软件并测试是否能正常运行，将测试结果反馈给开发人员。

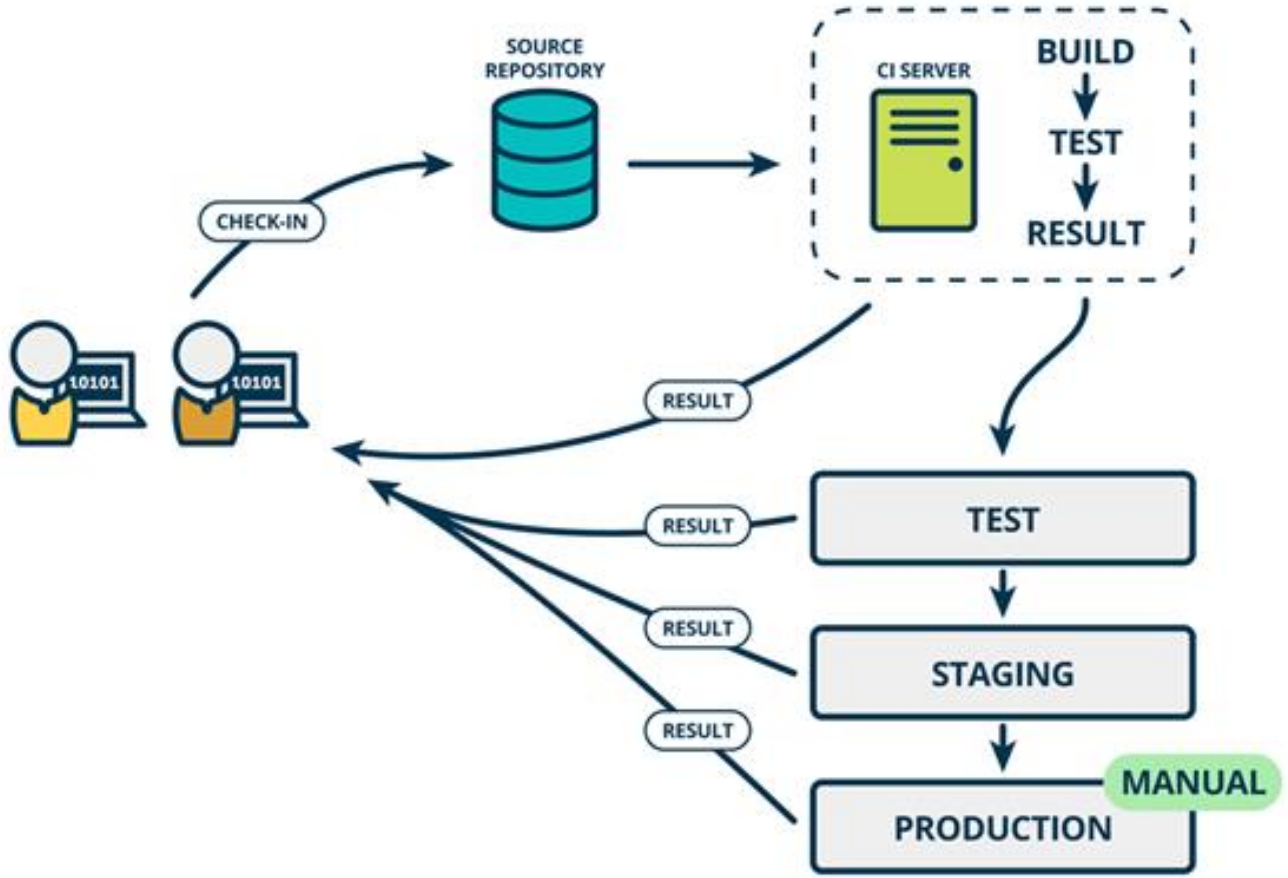


持续交付（Continuous Delivery ）

持续交付是在持续集成的基础上，将集成后的代码部署到更贴近真实运行的环境（类生产环境，production-like environments）中。比如，我们完成单元测试后，可以把代码部署到连接数据库的Staging环境中更多的测试。

1、手动部署

2、有部署的能力，但不一定部署

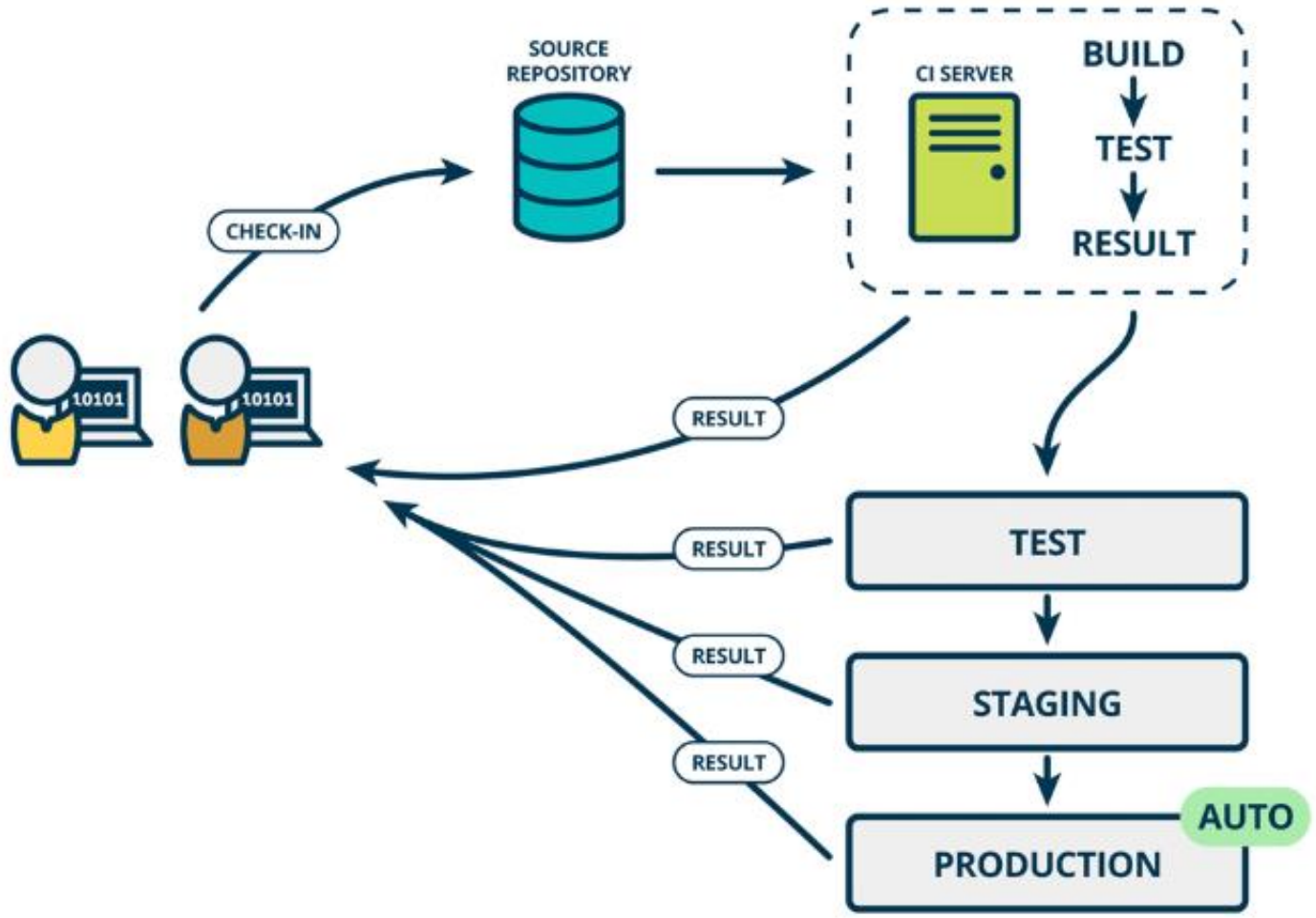


持续部署（Continuous Deployment）

持续部署是在持续交付的基础上，把部署到生产环境的这个过程自动化。

1、持续部署是自动的

2、持续部署是持续交付的最高阶段



3.使用工具

工具

代码管理（SCM）：GitHub、GitLab、SubVersion、码云
构建工具：Ant、Gradle、maven
持续集成（CI）：Travis、Jenkins、Drone
容器：Docker、LXC、第三方厂商如AWS
编排：Kubernetes、Core、Apache Mesos、DC/OS
服务注册与发现：Zookeeper、etcd、Consul
日志管理：ELK、Logentries
应用服务器：Tomcat、JBoss
Web服务器：Apache、Nginx、IIS
数据库：MySQL、Oracle、PostgreSQL等关系型数据库；cassandra、mongoDB、redis等NoSQL数据库

工具、CI/CD、DevOps的关系：通过技术工具链完成持续集成、持续交付、持续部署、用户反馈和系统优化的整合，实现跨团队的无缝协作(DevOps).

根据实际需要，选择合适的工具组合起来构建CI/CD流水线
Docker + Gogs + Jenkins + MYSQL + Nginx
Docker + GitLab + Drone + Nginx
.....

4.案例

使用Jenkins、Docker构建CI/CD流水线

目标：将软件开发生命周期的整个过程都自动化，即从开发人员向代码库中提交代码开始，到将此代码投入生产环境中使用为止，全部自动化完成。

使用码云+jenkins+docker来部署springboot项目

- **码云**：基于Git的代码托管和研发协作平台，是一个供开发人员提交代码的仓库。
- **Jenkins**：持续集成、交付、部署（软件/代码的编译、打包、部署）的基于web界面的平台，可安装各种插件处理 任何类型的构建或持续集成。
- **docker**：保证开发环境和生产环境一致，方便部署。

1、springboot项目

首先写一个最简单的只有响应"hello spring boot"的springboot项目， 编写Dockerfile：

```
#FROM openjdk:8-jdk-alpine   #基与jdk1.8
FROM hub.c.163.com/dwyane/openjdk:8
VOLUME /tmp      #挂载容器的位置，容易找到
ADD docker-springboot-1.0-SNAPSHOT.jar app.jar #宿主机器的jar文件 移动到 容器的jar文件
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
#容器执行的命令
```

上传此项目到码云上

2、安装和配置Jenkins

安装

1. Jenkins是基于Java语言开发的，所以需要先安装JDK〔Java development kit〕Java开发组件。
2. 这里通过tomcat来启动Jenkins，因此先安装tomcat。
3. 到官网https://jenkins.io/download/ 下载Jenkins的war包，war包需要配置Tomcat服务器才能运行访问，因此复制war包到tomcat的webapps目录下，启动tomcat

解锁

打开Jenkins后会提示解锁Jenkins，复制 /Users/Shared/Jenkins/Home/secrets/initialAdminPassword文件内容到页面的Administrator password栏中解锁

安装默认插件

然后提示安装插件，选择默认的安装，等待安装完成。

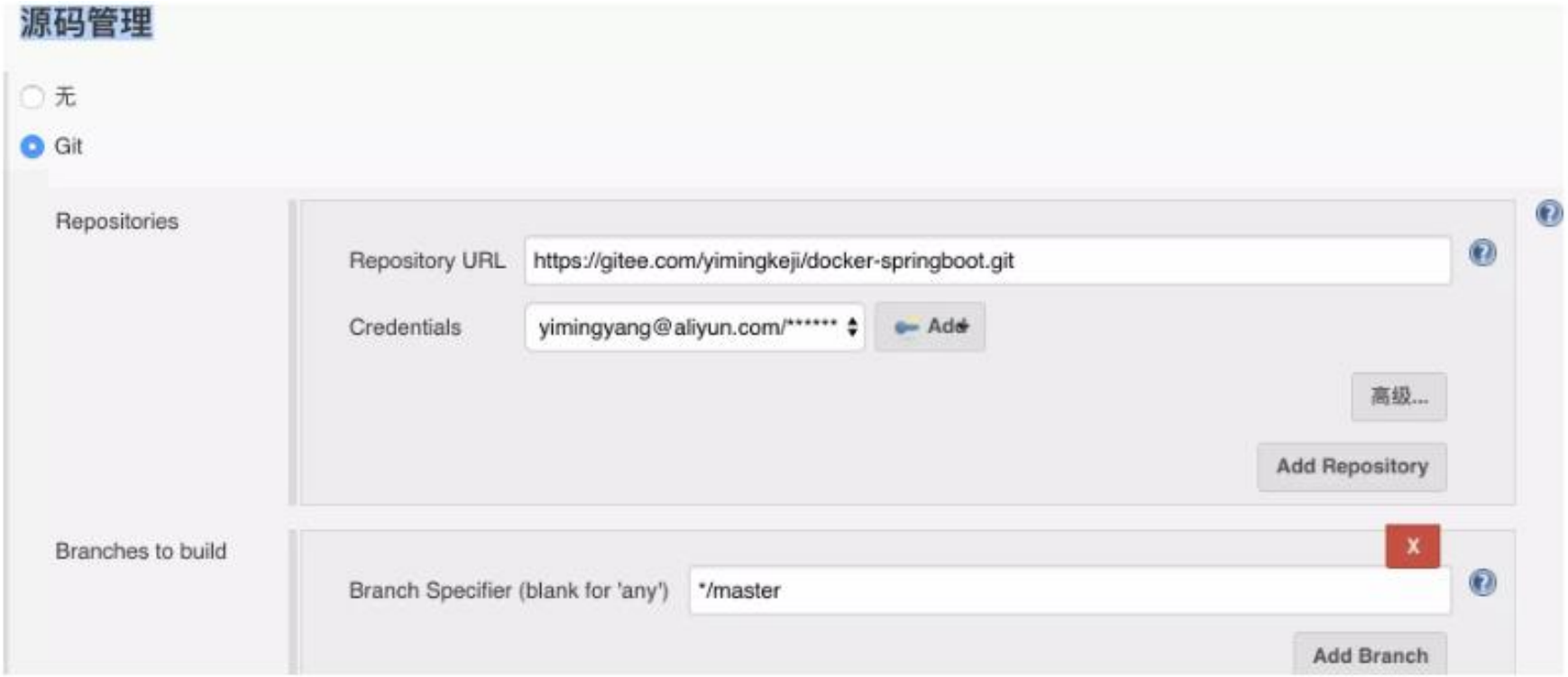
全局配置

点击 >系统设置>全局工具配置，配置maven、JDK、Docker、git的路径（事先安装好这些软件）

3、使用Jenkins部署项目到docker

在jenkins页面新建自由风格的软件项目。

（1）在源码管理中，添加git仓库和用户名、密码配置，并且选择代码分支（这里是master）



（2）在构建步骤中，添加2个步骤：

1.顶级maven 选择maven版本，
添加maven打包命令

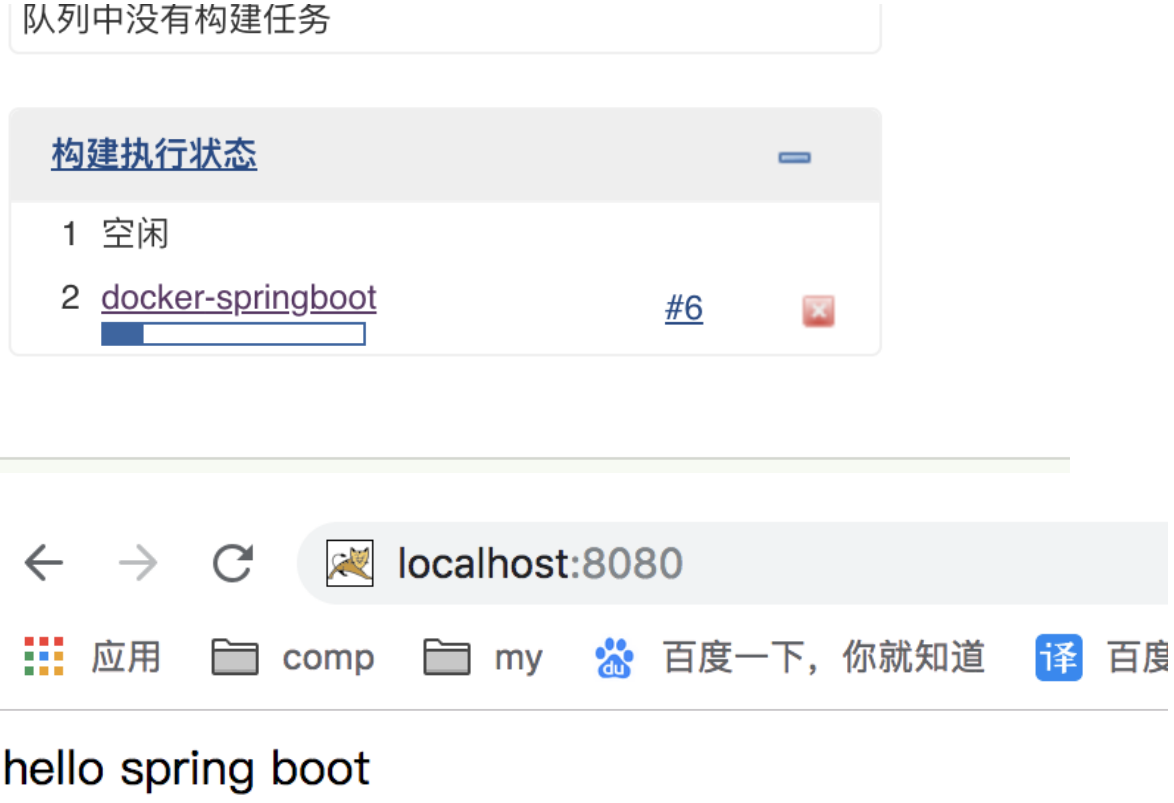
clean install -Dmaven.test.skip=true

2.执行shell 添加shell：

mvn docker:build #构建docker镜像
echo "当前docker 镜像： "
docker images | grep dockerspringboot
echo "启动容器----->"
docker run -p 8080:8080 -d dockerspringboot
echo "启动服务成功！ "

4、执行构建并启动服务

上面配置完成后，到Jenkins的Web主页，选择配置好的项目，菜单中点击立即构建，看到左边菜单里有执行的进度条，点进去后可以看到执行日志，如果启动服务成功，则可以到浏览器访问localhost:8080



5、改造shell脚本

如果下次构建该项目的时候，docker镜像和服务已存在，需要先删除镜像和服务

```
# 先删除之前的容器
echo "remobe old container"
docker ps -a | grep dockerspringboot | awk '{print $1}'| xargs docker rm -f
# 删除之前的镜像
echo "romove old image"
docker rmi dockerspringboot

# 构建镜像
mvn docker:build
# 打印当前镜像
echo "current docker images"
docker images | grep dockerspringboot
# 启动容器
echo "start container"
docker run -p 8001:8001 -d dockerspringboot
# 打印当前容器
echo "current container"
docker ps -a | grep dockerspringboot
echo "star service success!"
```

谢谢