持续集成-最佳实践

Continuous Integration – Best Practice

高磊

15/6/27

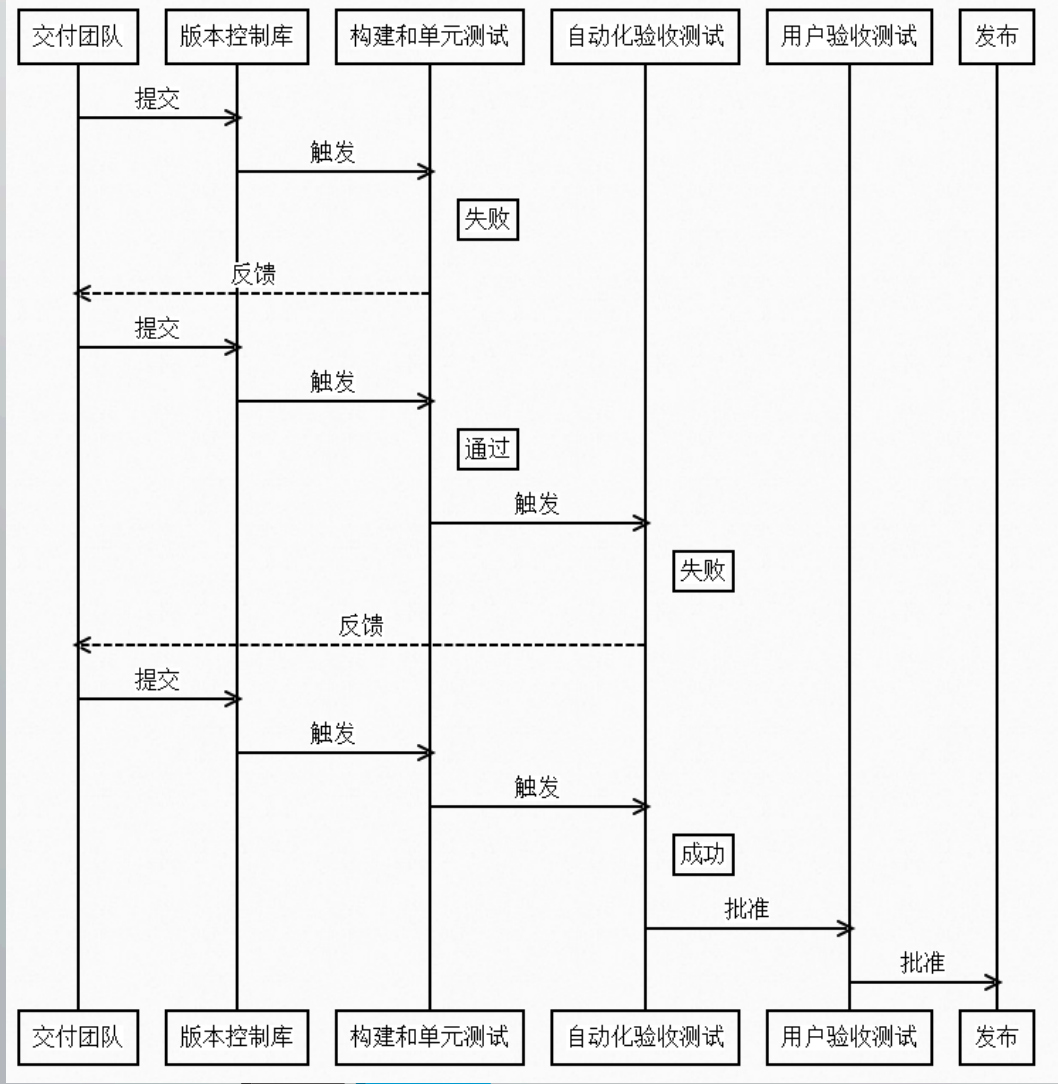
Me

- 高磊
  - gaol@getui.com
- 早年新加坡：全栈工程师
  - 产品，研发，测试，维护
- 个推：平台研发主管
  - 新技术，新方法，新工具
  - 提高技术人员效率

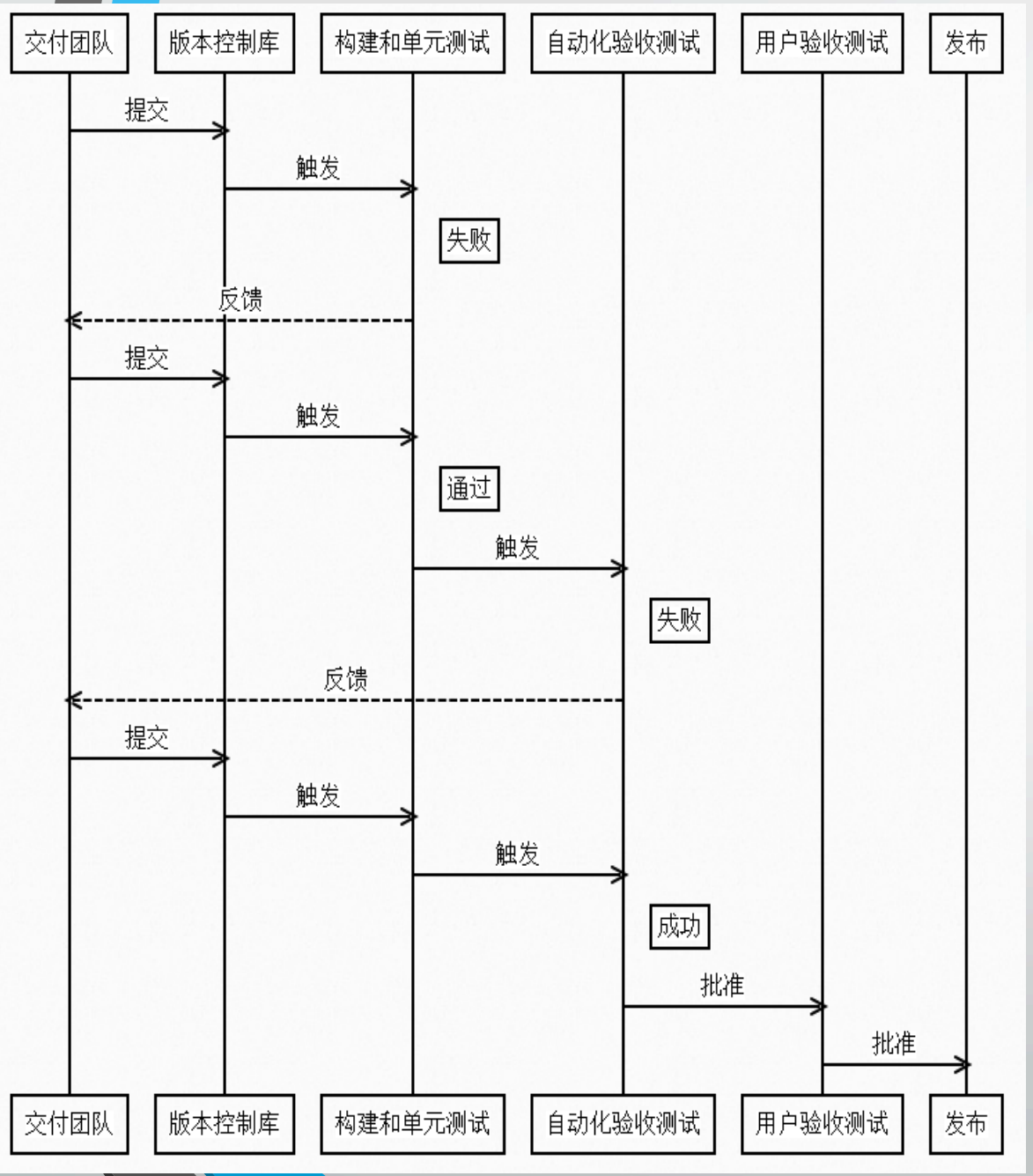
目录

1 持续集成

什么是持续集成



- 开发完成后完成后才布测试环境
- 手工部署
- 手动修改配置
- 手工回归测试



- 软件开发实践
- 统一的代码版本管理库
- **自动化**编译，打包和发布
- **自动化**测试
- 全程对团队**透明**

为什么要开展持续集成

为什么要开展持续集成

为什么要开展持续集成

- 更快地发现软件潜在问题
- 测试自动化
- 软件质量更透明化
- 能在更短的时间里建造整个系统

2 持续集成工具

工具的选择

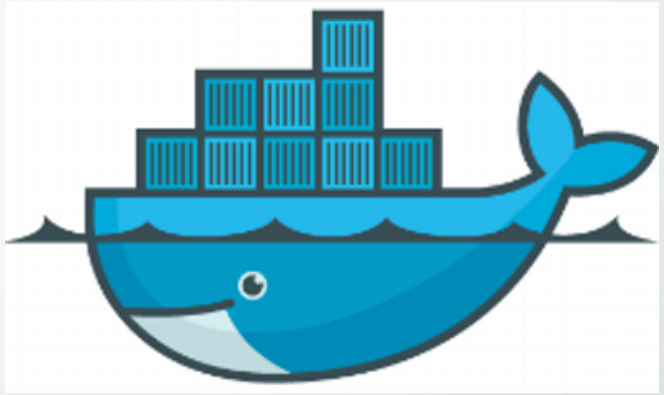
- 版本控制： git svn cvs
- 工程构建工具： maven ant
- 单元测试： Junit, Powermock, Mockito
- 测试环境搭建：docker vagrant
- 验收测试：Junit，pytest，
- 持续集成服务器：jenkins
- 其他工具：
  - codestyle: Checkstyle, Pylint, JSLint
  - 测试覆盖统计: Cobertura

git - 版本控制

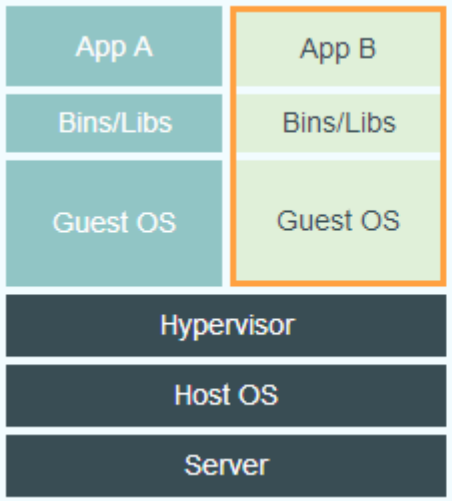
- Fast
- Feature rich
- Excellent for large open source projects
- Branching and merging is what Git does best
- GitHub makes programming a social activity

Docker - 测试环境搭建

- 开源 GO实现
- Linux 容器（LXC）
- 轻量级的操作系统虚拟化
- 应用程序和所需部署环境
- 可移植性

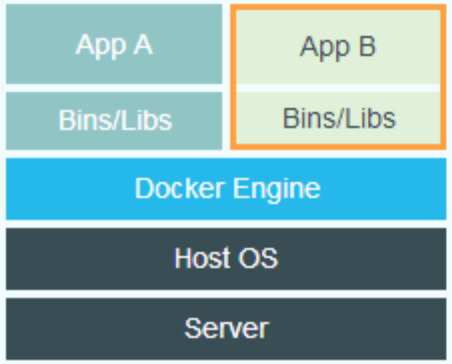


Docker 优点



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.



Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
性能	接近原生	弱于
系统支持量	单机支持上千个容器	一般几十个

Docker基本概念

- 镜像（image)
  - 一张安装光盘
  - 刻盘后不能更改
  - docker images
  - docker build
- 容器（ container)
  - 安装好的系统
  - docker ps
  - docker run
  - docker exec
  - docker logs

Jenkins - 持续集成服务器

- 开源
- 丰富的插件
- 完善的API
- 自由的扩展

Jenkins - 持续集成服务器

- Job
  - Source （对什么做）
  - Trigger（什么情况下）
  - Build（做什么）

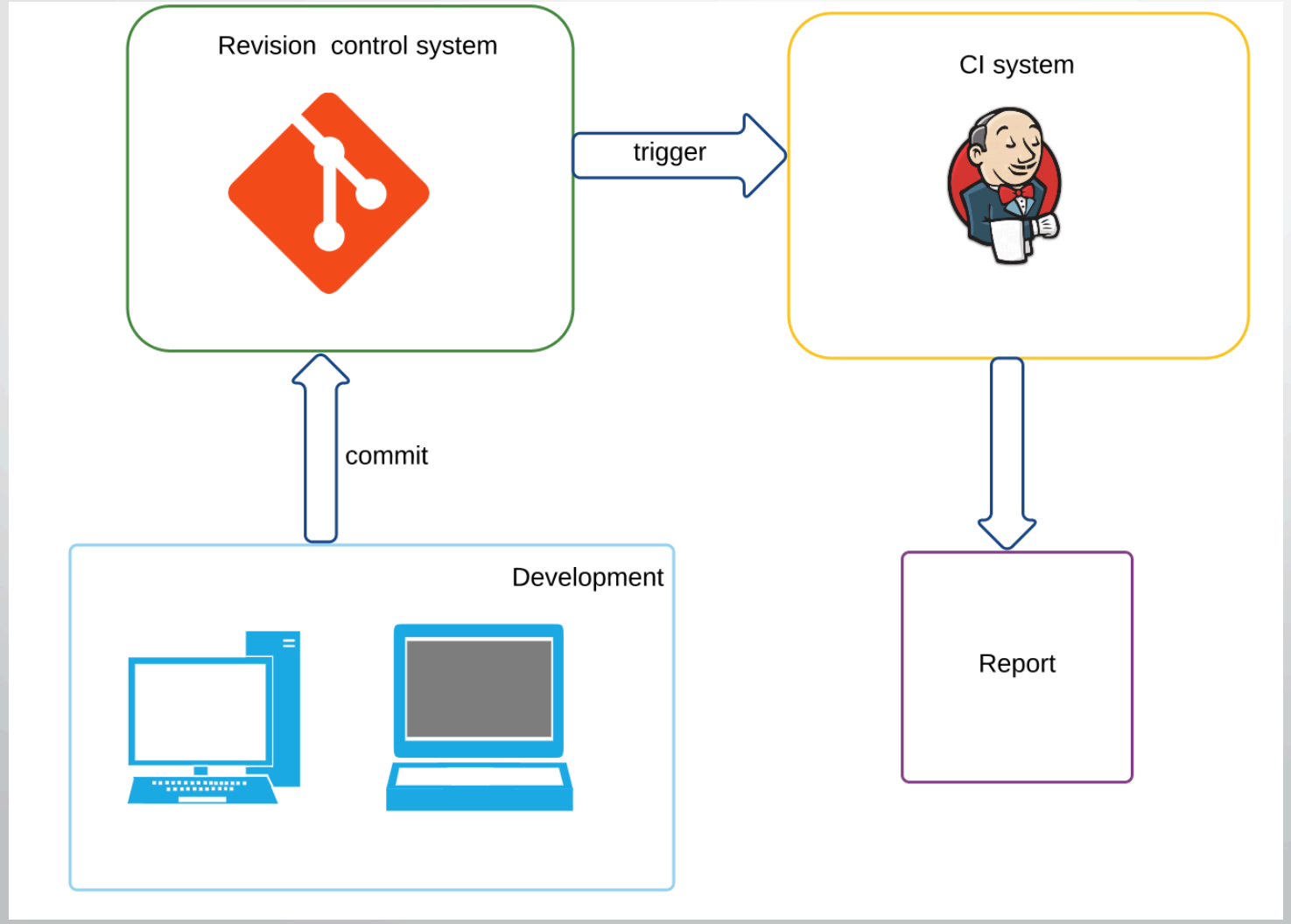
    Pre-build action

    Build action

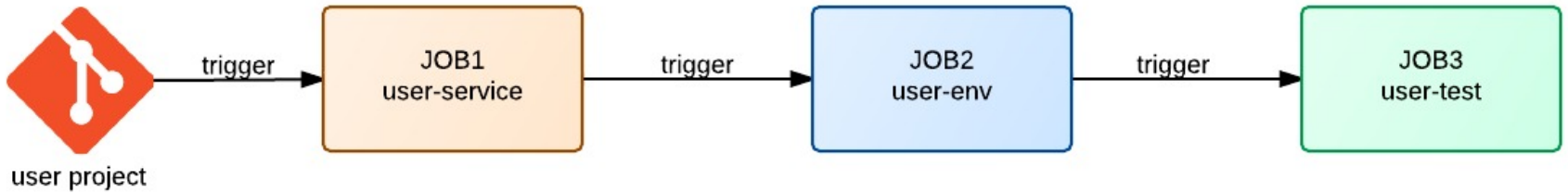
    Post-build action

3 开发和测试的流程

framework

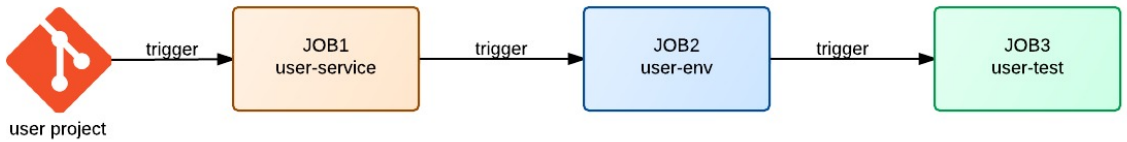

Revision control system
trigger
CI system
commit
Development
Report

jenkins步骤


trigger
JOB1
user-service
trigger
JOB2
user-env
trigger
JOB3
user-test
user project

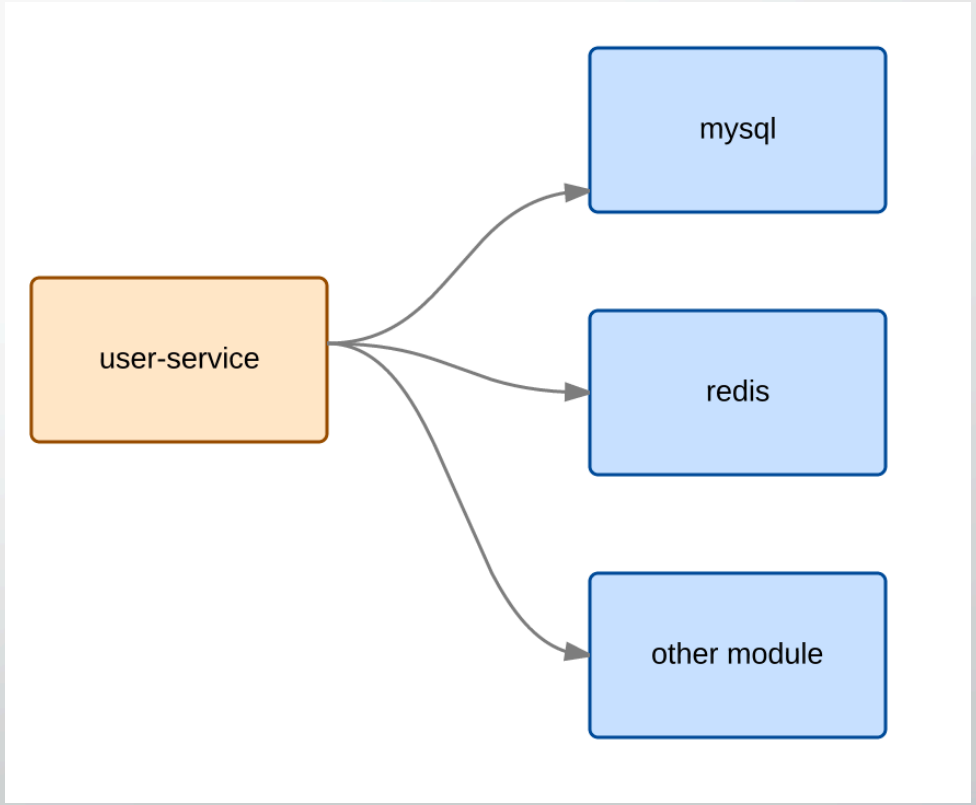
user-service: 检测代码

- Trigger
  - git代码仓库更新
- Build action:
  - mvn compile
  - 单元测试 – 覆盖率
  - Code style
- Post-build Actions:
  - Build user-env



user-env：搭建测试环境

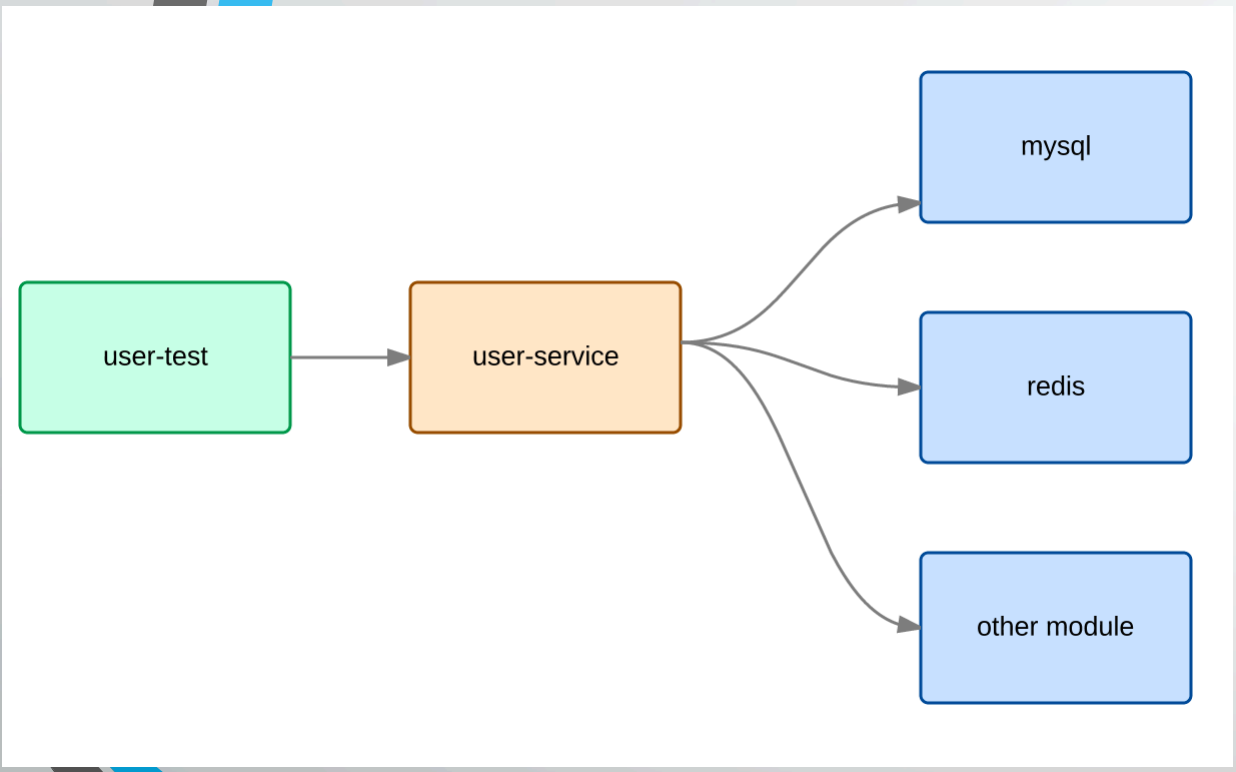
- Trigger
  - user-service
- Build action:
  - stop user docker
  - build user new docker image
  - start user new docker
- Post-build Actions:
  - build user-test



build步骤

- 1. git clone/ git pull
- 2. maven pacakge
- 3. docker run -d -v $PKG_PATH:/home -w /home –p 8080: 8080 192.168.10.211:7030/base:v3 java –jar test.jar

user-test: 验收



- Trigger
  - user-env
- Build action:
  - 验收测试
  - 是否满足业务需求所定义的验收条件


Jenkins
Back to Dashboard
Status
Changes
Modules
Build History
Jenkins
People
Build History
Project Relationship
Check File Fingerprint
all-monitor
-env
#33
2m 32s
service
#21
31s
-test
#29
25s
-env
#17
2m 0s
service
#27
36s
-test
#17
4m 42s

必不可少的实践

- 构建失败之后不要提交代码
- 提交代码之前在本地运行所有的验证测试
- 回家之前，构建必须处于成功状态
- 时刻准备着回滚到前一个版本
- 在回滚之前规定一个修复时间
- 为自己导致的问题负责

回顾

1. 持续集成

   what? why?

2. 持续集成工具

   git, docker, jenkins

3. 开发和测试的流程

   package, env, test

Q/A