技术团队代码管理和部署

2015-09-19 1145

主流公司使用svn和git作为代码版本管理，当然也不排除直接copy或者ftp。公司经历了的svn到git的变迁，也深刻体会到不同的版本管理服务，使得技术团队的协作方式变得更为流畅。

简单介绍下背景，有一个项目V5，从版本V1一直演变到现在V5，可见历史之久，想从svn切换到git，其中的代码管理和上线部署迁移，都会是经历很长一段时间的不稳定，尤其是一些开发同学对新的版本管理和部署理解不透彻，很容易引发事故。

在svn的主干开发流程

- 开发同学更新主干代码，提交代码
- 部署测试环境
- 检查每一个要上线的文件版本，是否夹带其它同学的提交，如果有，先回滚再提交，或者是带版本号上线，如：

```
common/request/Base.php
common/config/db.php -r 182993909
```

现在庞大的V5代码库仍然是全公司同学往master里更新代码，开发团队越来越大，这种原始的原始的开发方式，让所有同学都在一个master分支上协作的成本越来越大。

成本大在哪些地方？

团队协作首先是要让成员更方便获得彼此的更新，这样才能更一起完成各个模块，最后联调测试。成员就会把代码加入版本管理，而项目是有时间跨度的，如果时间短的项目和大项目的代码混合了，小项目要上线需要剥离大项目，剥离是否出错先不说，这个时候已经影响到另外一个项目了。如果遇到线上bug，需要紧急修复，需要从线上版本检出，最后和其它成员合并，这工作量都是附加的。

另外，在部署的时候繁重的版本检查，会严重耽误上线时间，改代码和验收花了5分钟，上线检查和处理夹带其它提交也花了5分钟。这时间成本在线上事故面前，那就直接影响着你的KPI了。

切换到git分支开发

这，不是我们想要的开发流程。迁移到git的分支开发是迟早的事情，它首先带来的好处就是团队不同项目的开发是相互隔离的，测试回归可以灵活地选择feature分支验收，通过再合并到主干。

branch和tag

Branch: master 和 development。其中 master 对应目前的发布分支，在这个分支只能增加从 master cherrypick 过来的 commit。development 是当前开发的分支，所有的 pull request 都应该发到这个分支。有些团队会以master作为开发主干，另外推送一个online分支作为发布分支，名字不同而已。

Tag: 对应每个发布版本的 tag。 tag 遵照 tag_[milestone号]_日期 的命名，如 tag_m6_2015-08-12，如果有 bugfix，则在后面增加小写字母，如 tag_m6_2015-08-12后是 tag_m6_2015-08-12a，然后是 tag_m6_2015-08-12b。

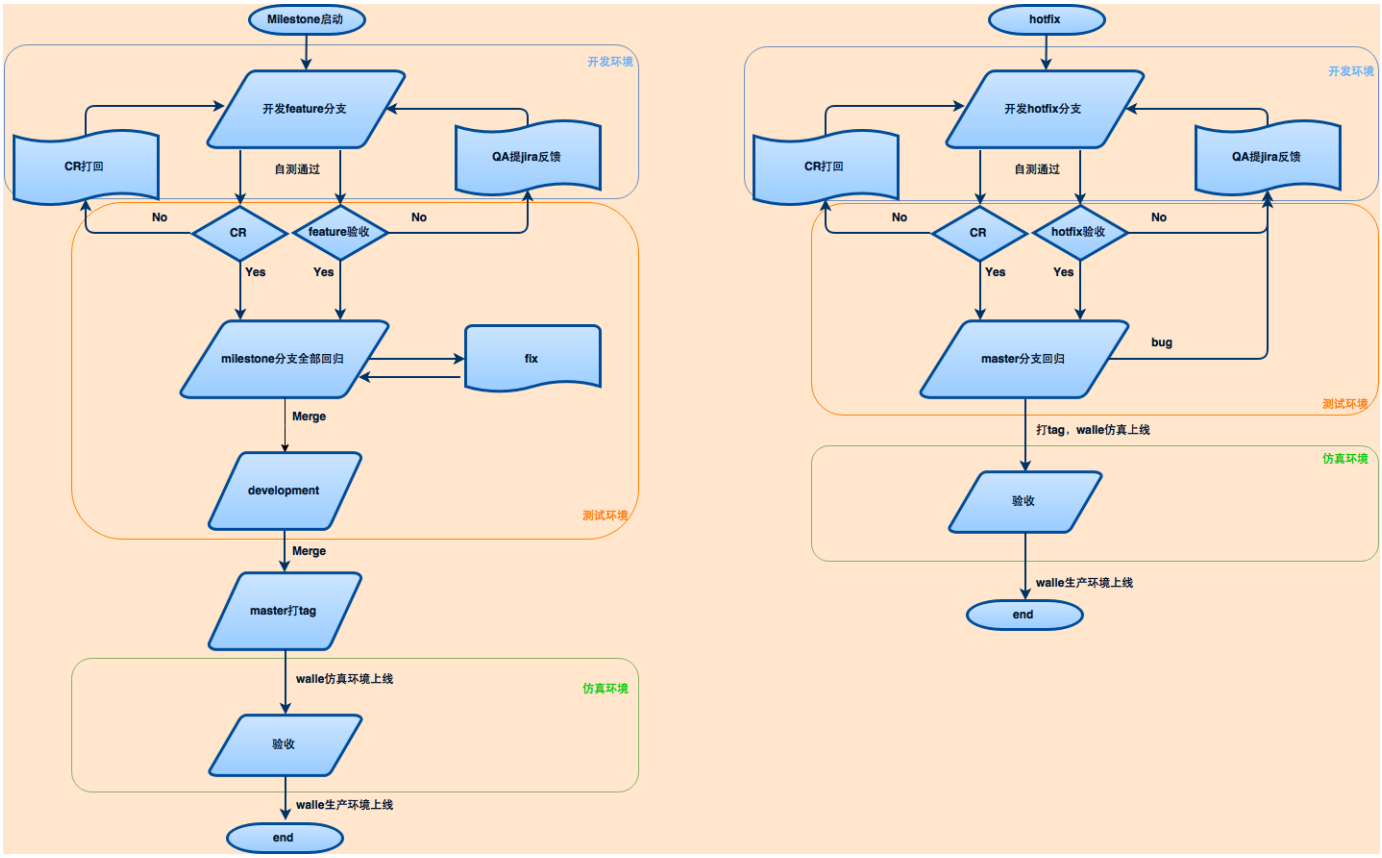
正常开发流程

1. 从master切feature分支开发
2. 自测通过之后，提交 pull request 到 development（如果是大项目milestone的话，可以先 pull request到milestone分支），并通知QA部署feature分支回归
3. CodeReview标Ok，QA验收通过后进行merge到development，如果CodeReview或者QA发现有问题，在feature分支修正再部署到测试环境，直到没有问题
4. merge 验收通过的 feature-fix 到 development
5. 部署development分支到测试环境
6. 测试验收通过，merge development分支到master，打tag
7. 发起仿真环境上线，仿真环境验收
8. 验收通过，发起生产环境上线
9. leader审核上线任务，发起同学进行部署，生产环境验收

hotfix流程

1. 从master切hotfix分支开发
2. merge 确保所有要发布的 pull request 到 master
3. 后面与正常开发流程一致了
4. merge master分支到development

大项目与hotfix开发和上线流程图

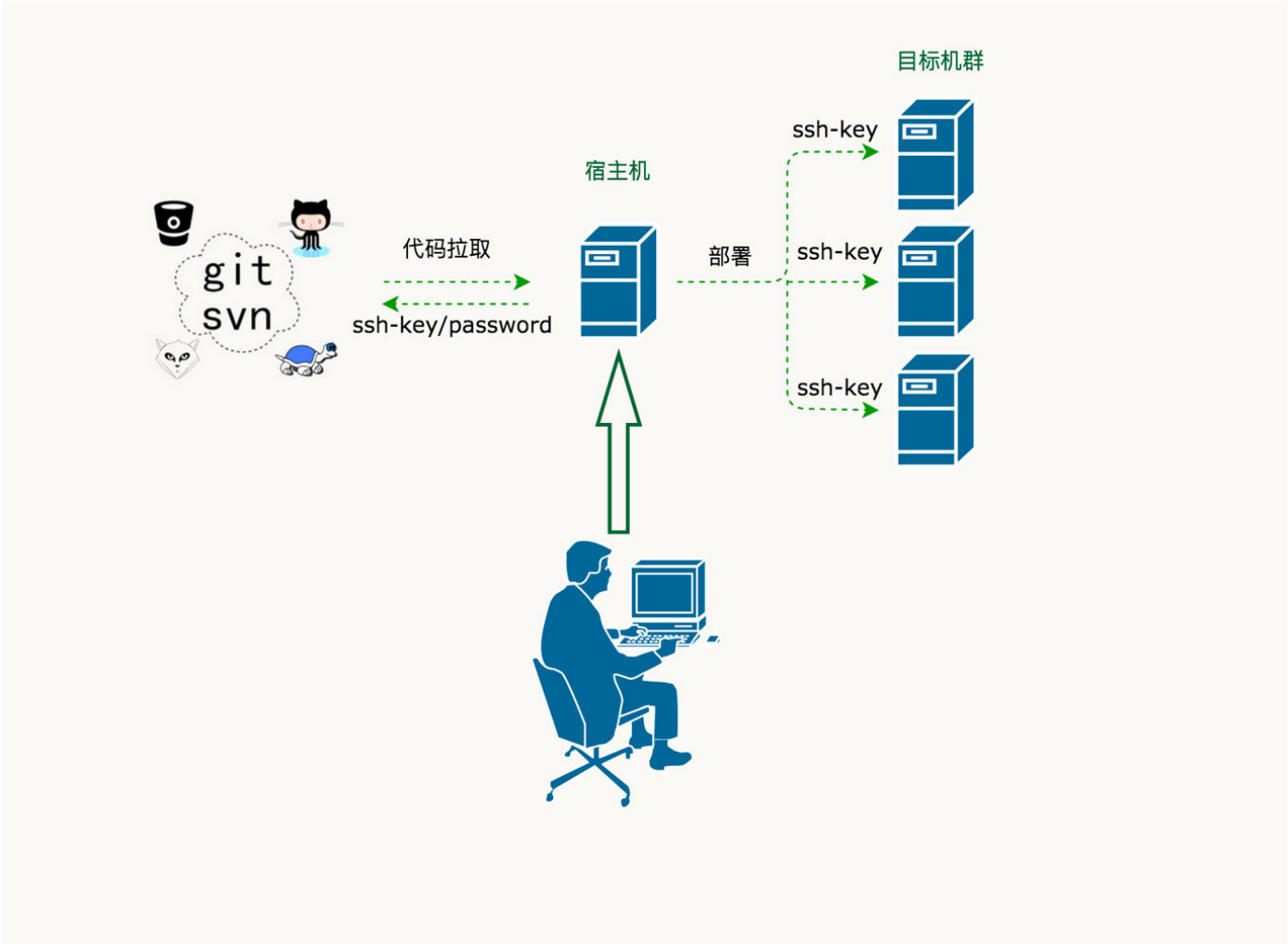


如果不是开发周期有两三个星期的项目，迭代速度快的，一天要上线好几个小功能的，可以在开发完feature分支，跳过milestone分支直接merge到development分支，让QA验收打tag。另外，并不是每个 bug 都有专门发布 bugfix 版的必要，对于不紧急的 bug，可以在 master 里 fix 后随下一个版本发布。

目前试用瓦力上线系统 (https://github.com/meolu/walle-web)：https://github.com/meolu/walle-web，分别部署测试、仿真、生产环境大大节约了脚本上线的成本，开发、测试同学自行可发起部署。部署过程支持 (https://github.com/meolu/walle-web，分别部署测试、仿真、生产环境大大节约了脚本上线的成本，开发、测试同学自行可发起部署。部署过程支持)

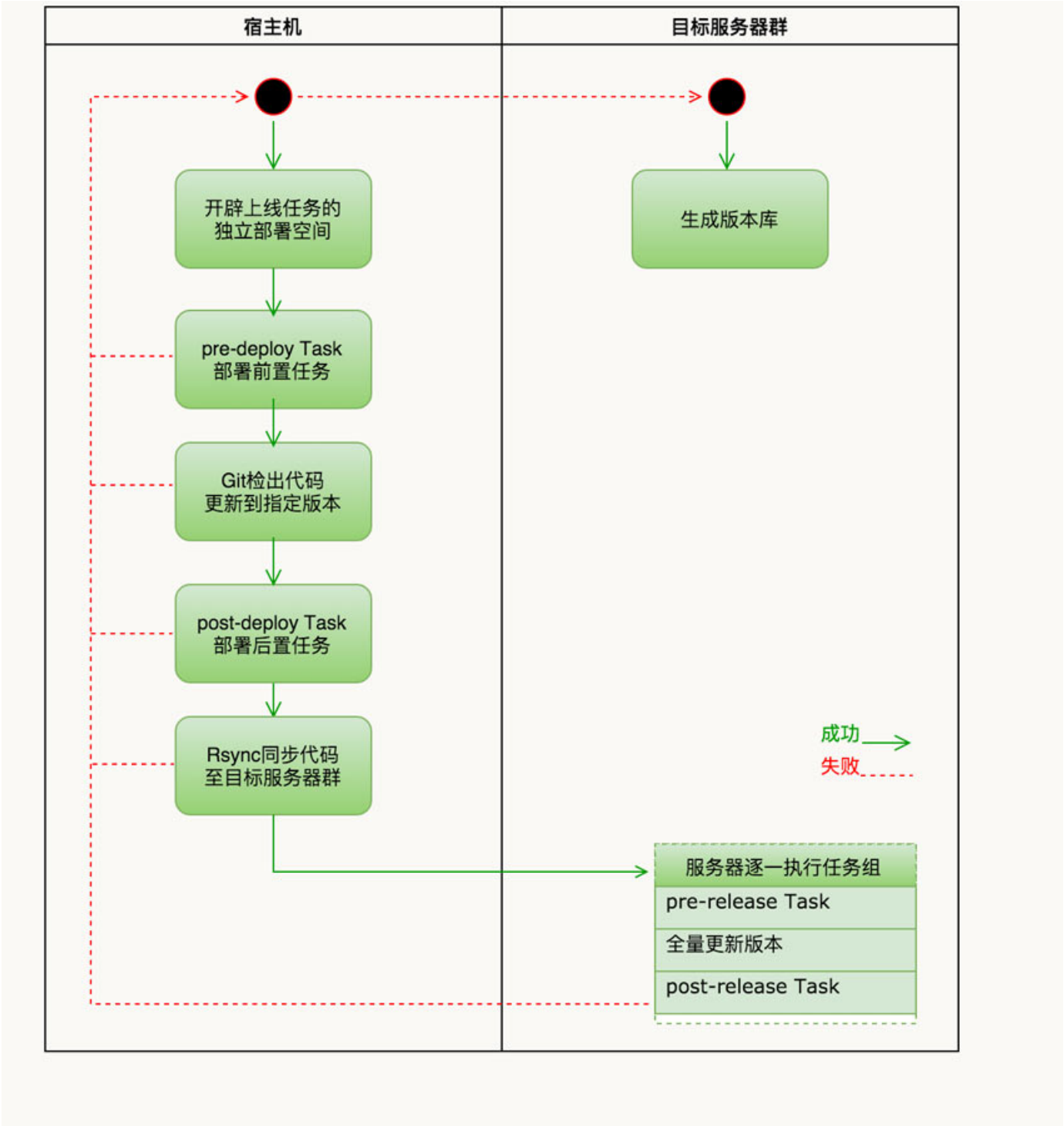
- 用户分身份注册、登录
- 开发者发起上线任务申请
- 管理者审核上线任务
- 支持多项目部署
- 快速回滚
- 部署前准备任务（前置检查）
- 代码检出后处理任务（如vendor，环境配置）
- 同步到各目标机器后收尾任务（如重启）
- 执行sql构建（不要担心忘记测试环境sql同步）
- 线上批量文件指纹检查

其宿主机、目标机群、操作用户关系如下图：



上线流程

部署发布每次都会有版本记录保留，事故一旦发生，回滚可瞬间完成。其上线流程安全可靠，同时支持git、svn，支持多任务辅助，即可对一些前置、后置操作自定义。如vendor更新，java mvn编译。我们看下其部署的原理。



目前花满树的主页 (http://www.huamanshu.com/)、博客 (http://blog.huamanshu.com/)、瓦尔登 (http://walden.huamanshu.com/)、瓦力自己已交由瓦力 (https://github.com/meolu/walle-web)来部署上线，github项目地址：walle - 瓦力 (https://github.com/meolu/walle-web)，欢迎star、fork试用，有任何问题请反馈：)

← walden瓦尔登是怎么被开源出来的 (/?date=2015-09-23)

其实我请假去玩了 → (/?date=2015-09-14)

名字不重要

发表评论