代码发布系统实现

关于项目开源

由于挺多同学请求开源此项目，在这里说明一下:其实本人是想开源的，由于是本人写的第一个运维方面的系统，且写这个项目的时间时间紧，只达到了可以使用的程度，完全没有达到开源的要求，希望理解！

日常运维问题

在我日常运维工作中，代码发布可能是最普遍的一项工作之一，尤其是网页代码的更新，碎片化发布需求非常频繁。在前期开发人员比较少时，还可以由自己 来上服务器通过脚本来发布代码。但随着公司项目的增多，更多的开发人员加入到公司，发布代码需求开始增多，这就占用了我大部分时间，经常的被打断其它工作 来发布代码，非常地不爽，然后开始想解决方法。

尝试解决问题

当然，发布代码肯定是运维的职责之一了，但频繁的发布导致运维大部分时间浪费在重复的操作上，非常的不值得。基于此，开始限制代码发布频率，要求把 不是很紧急的更新延后到一周中的几个时间点。但实施起来效果不理想，治标不治本，原因是你不能强制把需要立即上线的更改延后。实施这样的定时发布，有可能 影响项目的快速迭代。

最终解决方案

想到这样子下去也不是办法，会造成工作很被动，于是开始着手建立以 Web 操作方式，结合 git,rsync 来实现自动代码发布。公司代码管理目前用 的是 svn，开发人员在发布前也没有打 Tag 的习惯，所以想到分布式的 git 来完成版本的管理，rsync 当然是用来同步代码到其它服务器了。附上几张代 码发布系统的截图：

开源技术使用

- rsync:用来同步代码到服务器;
- git: 用来标记版本，回滚版本;
- tornado: python 的一个 web 构架，提供后台服务;
- angularjs: 前端的一个 mvc 框架，用来实现浏览器与后端的交互，使得后端不需要关心前端网页的渲染，专注后端逻辑的开发。前端和后端通过 json 数据来通信;
- bootstrap: 让运维人员写的网站后台 UI 也可以很专业。

代码发布流程

从流程图可以看到，我们只需要把审核发布的权限交给开发组负责人，运维只需要维护系统的稳定，之后代码发布就不需要运维来参与了。

以上是整体的流程，现在来说详细说下具体的逻辑实现：

- 1、开发人员提交代码更新，主要提交的字段包括“更新理由”，“svn 代码路径”;
- 2、后端收到请求后，把此数据插入到数据库，标记此更新单为“等待预发布环境更新”的状态;
- 3、后台进程定时查询是否有等待预发布环境更新的更新单，如果有，读取 svn 路径，执行 svn up 更新代码操作,并标记此更新单为“预发布环境已更新，等待完成测试”;
- 4、开发人员或者测试人员通过预发布环境的域名来测试功能是否正常，如果不正常，作代码修改后提交 svn，再到 web 发布后台点击“返回修改”， 对 svn 路径或者不做任何修改再点击“重新提交”，然后更新单又一次回到”等待预发布环境更新“状态。循环 3、4 步骤，直至预发布环境测试通过为止;
- 5、在确认测试通过后，开发人员点击”测试通过“，这时更新单进入”等待审核状态“;
- 6、负责人确认可以发布后，点击”审批“按钮，这时更新单进入”审核通过，等待执行发布操作“的状态。这时,开发人员得到发布代码的授权;
- 7、开发人员点击”发布代码“按钮，更新单进入”已执行发布，等待系统完成发布“状态;
- 8、后台进程查询状态为”已执行发布，等待系统完成发布“的更新单，执行 git 发布命令。git 命令大概为，进入预发布代码目录，执行 git add .;git commit -m “更新原因”;git tag 上一次版本号+1，再进入已发布代码的目录,执行 git pull 同步预发布代码目录的更改。最后调用 rsync 命令同步代码到生产环境。

下面是回滚流程：

- 1、进入 web 代码发布系统，选择已发布的版本，点击“申请回滚”;
- 2、负责人审核此次回滚;
- 3、开发人员执行回滚操作;
- 4、后台查询“等待回滚”的记录，假如回滚的版本号为 18，进入已发布代码的目录,执行 git checkout -b 18 18;git checkout 18(这两条 git 命令作用为,以 tag 18 创建分支号为 18 的分支，并切换当前分支为 18)，然后再通过 rsync 命令来同步代码到生产环境，这样就实现了版本的回滚。

最后想说的话

最后想说的是，运维工作可以是枯燥的，也可以是有趣的。枯燥是因为没有意识或者懒得把重复的操作通过制定流程来使其自动化，在不断地把各种在运维工 作中占用时间比较多的重复操作通过技术来使得自动化时，我们既高效完成了工作，节省了时间，又能提高编程和解决问题的能力，只有这样，我们才能让运维工作 变得既有趣又有挑战性。

你现在所遭遇的每一个不幸，都来自一个不肯努力的曾经