GIT 特性分支开发流程

陶文
problem solver

从原博客导入：我的GIT工作流程 - taowen - 博客园

Friendbuy是一家互联网创业公司。产品的源代码是托管在GITHUB上的。在EC2上有三套环境：生产环境，测试环境和持续集成环境。基本上每天都有大量的代码被提交，测试和部署。一年多的磨合下来，逐渐理顺了GIT的使用流程。但是，最开始并不是这样的，所有的开发人员都没有使用过GIT，基本上都是SVN的背景。

最开始的使用方式

只有一个GIT分支，就是MASTER。开发团队直接向MASTER提交新的改动，部署其实就是在生产环境下执行

```
git pull
```

开发人员的日常工作也很简单

```
git pull --rebase
git commit -a -m "xxxx"
git push
```

基本上是把Git当作SVN来使用。

生产环境不稳定

很快就出现了问题。在一次给客户的演示的过程中，掉链子了。

老板很不高兴，对于生产环境部署的质量产生了怀疑。

于是为了使得生产环境稳定，团队决定牺牲时效性，建立更加正规的流程，添加了测试环境和自动集成环境

每次提交的代码都会被自动集成环境自动进行测试

不定期的会人工部署到测试环境中测试

当测试环境测得差不多的时候才会决定部署到生产环境

另外每次部署到测试环境和产品环境的时候都会打一个标签

```
git tag -a xxx -m xxx
```

速度就是生命

最要命的是，网站为了尝试不同的风格，还经常全面改版。每次完整的改版都要协调各方面的资源，特别是有一个很长的UI调整过程。

问题是在网站局部改版的情况下，客户的其他特性仍然要响应，产品的线上BUG仍然要FIX。

于是就有了分支。

```
git branch new-retailer-site master
git checkout new-retailer-site
#change some thing
git commit -a -m "xxx"
git push origin new-retailer-site
```

分支用完了之后，要合并到master中

```
git checkout master
git merge new-retailer-site
#fix conflit
git commit -a -m "xxx"
git push
```

最后就可以把分支删除了

```
git push origin :new-retailer-site
```

千万不能搞乱master

分支在很短的时间就冲到了两位数。对于分支的管理一开始也并不在意。

直到出了这么一个事情:

开发团队一致决定new-retailer-site已经差不多了，可以“准备”发布了。然后new-retailer-site被合并到了master中。

然后在master上有开发了一些其他特性。

但是new-retailer-site的UI迟迟不能够让人满意。直到有一天开发团队被要求先把master中的“有用”的feature发布，样式改版工作延后发布。

这可难办了，所有的改动已经混杂在同一个分支中了。

最后的解决办法是用

git log

把一条条的改动给找出来，由于比对实在太麻烦了，还写了点代码来干这事

```
def list_commits(branch):
    commits = local('git log ' + branch + ' ^master --no-merges --format=format:%s,%H'
    commits = commits.split('\n')
```

```
print('==> %s <==' % commit.split(',')[0])
print('https://github.com/friendbuy/apps/commit/%s' % commit.split(',')[1])
```

然后把找出来的commit，一条条cherry pick出来

```
git cherry-pick <commit id>
```

接下来很长一段时间都是搞不清楚，到底是哪个分支是产品部署的分支。

直到某一天，团队决定以后再也不能随便把无关分支合并到master了。

正常的工作流程应该是，选定要候选发布的branch，比如说new-retailer-site

```
git checkout new-retailer-site
git merge master
# fix conflict
git commit -a -m "merge"
git push
```

然后在测试环境下

```
git checkout new-retailer-site
git pull
```

测试通过了之后，确定可以发布到产品环境了

```
git checkout master
git merge new-retailer-site
git push
```

然后把剩余的分支，逐个更新

```
git checkout feature-branch-1
git merge master
# ...
git checkout feature-branch-2
git merge master
# ...
```

总结

使用feature branch的方式，可以同时进行很多项特性的开发，并有产品的需要决定什么时候发布哪些特性，比如在圣诞节期间，基本上就没有feature branch被发布，但是开发并不会因此停滞

在使用多分支开发的时候要保持一个master分支始终对应“一定会被发布到产品环境的代码”，以master为中心保持一致的合并方向，不然分支之间乱合并就很难管理了。

目前有一个缺陷是持续集成环境只对master进行测试，理想的情况下应该建立一个staging的分支，持续集成环境也要对staging分支进行测试。

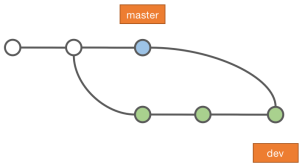
编辑于 2017-02-21

Git

文章被以下专栏收录

taowen 进入专栏

推荐阅读



Git在rebase时如何保留merge commit

颜海镜 发表于颜海镜的博...

Git 核心概念

Tony 发表于FIM

这些GIT经验够你用一年了

知一

1 条评论 切换为时间排序

写下你的评论...

知乎用户 2018-01-18

测试环境和开发环境中有不同的配置该如何配置呢?

赞