理解Git的工作流程 - 博客 - 伯乐在线

笔记本： Git
创建时间： 2012/7/30 21:35
URL： http://blog.jobbole.com/24379/

理解Git的工作流程

发布时间：2012-07-27 14:39　来源：伯乐在线　分类： IT技术, 程序员 都等你发言 :)

如果你不理解Git的设计动机，那你就会处处碰壁。知道足够多的命令和参数后，你就会强行让Git按你想的来工作，而不是按Git自己的方式来。这就像把螺丝刀当锤子用；也能把活干完，但肯定干的差极了，花费很长时间，还会弄坏螺丝刀。

想想常见的Git工作流程是怎么失效的吧。

> 从Master创建一个分支，写代码，然后把这个分支合并回Master。

多数时候这样做的效果会如你所愿，因为从你创建分支到合并回去之间，Master一般都会有些变动。然后，有一天当你想把一个功能（feature）分支合并进Master的时候，而Master并没有像以往那样有变动，问题来了：这时Git不会进行合并commit，而是将Master指向功能分支上的最新commit。（看图）

不幸的是，你的功能分支有用来备份代码的commit（作者称之为checkpoint commit），这些经常进行的commit对应的代码可能处于不稳定状态！而这些commit现在没法和Master上那些稳定的commit区分开来了。当你想回滚的时候，很容易发生灾难性后果。

于是你就记住了：“当合并功能分支的时候，加上 -no-ff 选项强制进行一次全新的commit。”嗯，这么做好像解决问题了，那么继续。

然后一天你在线上环境中发现了一个严重bug，这时你需要追溯下这个bug是什么时候引入的。你运行了bisect命令，但却总是追溯到一些不稳定的commit。因此你不得不放弃，改用人肉检查。

最后你将bug范围缩小到一个文件。你运行blame命令查看这个文件在过去48小时里的变动。然后blame告诉你这个文件已经好几周没有被修改过了——你知道根本不可能没有变动。哦，原来是因为blame计算变动是从第一次commit算起，而不是merge的时候。你在几周前的一次commit中改动了这个文件，但这个变动今天才被merge回来。

用no-ff来救急，bisect又临时失效，blame的运作机制又那么模糊，所有这些现象都说明一件事儿，那就是你正在把螺丝刀当锤子用。

反思版本控制

版本控制的存在是因为两个原因。

首先，版本控制是用来辅助写代码的。因为你要和同事同步代码，并经常备份自己的代码。当然了，把文件压缩后发邮件也行，不过工程大了大概就不好办了。

其次，就是辅助配置管理工作。其中就包括并行开发的管理，比如一边给线上版本修复bug，一边开发下一个版本。配置管理也可以帮助弄清楚变动发生的具体时间，在追溯bug中是一个很好的工具。

一般说来，这两个原因是冲突的。

在开发一个功能的时候，你应该经常做备份性的commit。然而，这些commit经常会让软件没法编译。

理想情况是，你的版本更新历史中的每一次变化都是明确且稳定的，不会有备份性commit带来的噪声，也不会有超过一万行代码变动的超大commit。一个清晰的版本历史让回滚和选择性merge都变得相当容易，而且也方便以后的检查和分析。然而，要维护这样一个干净的历史版本库，也许意味着总是要等到代码完善之后才能提交变动。

那么，经常性的commit和干净的历史，你选择哪一个？

如果你是在刚起步的创业公司中，干净的历史没有太大帮助。你可以放心地把所有东西都往Master中提交，感觉不错的时候随时发布。

如果团队规模变大或是用户规模扩大了，你就需要些工具和技巧来做约束，包括自动化测试，代码检查，以及干净的版本历史。

功能分支貌似是一个不错的折中选择，能够基本的并行开发问题。当你写代码时候，可以不用怎么在意集成的问题，但它总有烦到你的时候。

当你的项目规模足够大的时候，简单的branch/commit/merge工作流程就出问题了。缝缝补补已经不行了。这时你需要一个干净的版本历史库。

Git之所以是革命性的，就是因为它能同时给你这两方面的好处。你可以在原型开发过程中经常备份变动，而搞定后只需要交付一个干净的版本历史。

工作流程

考虑两种分支：公共的和私有的。

公共分支是项目的权威性历史库。在公共分支中，每一个commit都应该确保简洁、原子性，并且有完善的提交信息。此分支应该尽可能线性，且不能更改。公共分支包括Master和发行版的分支。

私有分支是供你自己使用的，就像解决问题时的草稿纸。

安全起见，把私有分支只保存在本地。如果你确实需要push到服务器的话（比如要同步你在家和办公室的电脑），最好告诉同事这是私有的，不要基于这个分支展开工作。

绝不要直接用merge命令把私有分支合并到公共分支中。要先用reset、rebase、squash merges、commit amending等工具把你的分支清理一下。

把你自己看做一个作者，每一次的commit视为书中的一章。作者不会出版最初的草稿，就像Michael Crichton说的，“伟大的书都不是写出来——而是改出来的”。

如果你没接触过Git，那么修改历史对你来说好像是种禁忌。你习惯于认为提交过的所有东西都应该像刻在石头上一样不能抹去。但如果按这种逻辑，我们在文本处理软件器中也不应该使用“撤销”功能了。

实用主义者们直到变化变为噪音的时候才关注变化。对于配置管理来说，我们关注宏观的变化。日常commit（checkpoint commits）只是备份于云端的用于“撤销”的缓冲。

如果你保持公共历史版本库的简洁，那么所谓的fast-forward merge就不仅安全而且可取了，它能保证版本变更历史的线性和易于追溯。

关于 -no-ff 仅剩的争论就只剩“文档证明”了。人们可能会先merge再commit，以此代表最新的线上部署版本。不过，这是反模式的。用tag吧。

规则和例子

根据改变的多少、持续工作时间的长短，以及分支分叉了多远，我使用三种基本的方法。

1）短期工作

绝大多数时间，我做清理时只用squash merge命令。

假设我创建了一个功能分支，并且在接下来一个小时里进行了一系列的checkpoint commit。

```
git checkout -b private_feature_branch
touch file1.txt
git add file1.txt
git commit -am "WIP"
```

完成开发后，我不是直接执行git merge命令，而是这样：

```
git checkout master
git merge --squash private_feature_branch
git commit -v
```

然后我会花一分钟时间写个详细的commit日志。

2）较大的工作

有时候一个功能可以延续好几天，伴有大量的小的commit。

我认为这些改变应该被分解为一些更小粒度的变更，所以squash作为工具来说就有点儿太糙了。（根据经验我一般会问，“这样能让阅读代码更容易吗？”）

如果我的checkpoint commits之后有合理的更新，我可以使用rebase的交互模式。

交互模式很强大。你可以用它来编辑、分解、重新排序、合并以前的commit。

在我的功能分支上：

```
git rebase --interactive master
```

然后会打开一个编辑器，里边是commit列表。每一行上依次是，要执行的操作、commit的SHA1值、当前commit的注释。并且提供了包含所有可用命令列表的图例。

默认情况下，每个commit的操作都是“pick”，即不会修改commit。

```
pick ccd6e62 Work on back button
pick 1c83feb Bug fixes
pick f9d0c33 Start work on toolbar
```

我把第二行修改为“squash”，这样第二个commit就会合并到第一个上去。

```
pick ccd6e62 Work on back button
squash 1c83feb Bug fixes
pick f9d0c33 Start work on toolbar
```

保存并退出，会弹出一个新的编辑器窗口，让我为本次合并commit做注释。就这样。

舍弃分支

也许我的功能分支已经存在了很久很久，我不得不将好几个分支合并进这个功能分支中，以便当我写代码时这个分支是足够新的的。版本历史让人费解。最简单的办法是创建一个新的分支。

```
git checkout master
git checkout -b cleaned_up_branch
git merge --squash private_feature_branch
git reset
```

现在，我就有了一个包含我所有修改且不含之前分支历史的工作目录。这样我就可以手动添加和commit我的变更了。

总结

如果你在与Git的默认设置背道而驰，先问问为什么。

将公共分支历史看做不可变的、原子性的、容易追溯的。将私有分支历史看做一次性的、可编辑的。

推荐的工作流程是：

基于公共分支创建一个私有分支。
经常向这个私有分支`commit`代码。
一旦你的代码完善了，就清理掉下私有分支的历史。
将干净的私有分支`merge`到公共分支中。

英文原文：~~Understanding the Git workflow~~ 编译： 张重骐

本文由张重骐（@candyhorse）投稿于伯乐在线，也欢迎其他朋友投稿。提示：投稿时记得留下微博账号哦