

使用Git的正确姿势

Aditya Sridhar [优达学城Udacity](#)

编译/ 佑铭

参考/ <https://medium.freecodecamp.org/how-to-use-git-efficiently-54320a236369> (文/ Aditya Sridhar)

昨天代码还好好地今天都不行了 T_T

代码不小心被删了!!!

突然出现了一个奇怪的 bug 但是大家都摸不着头脑???

...

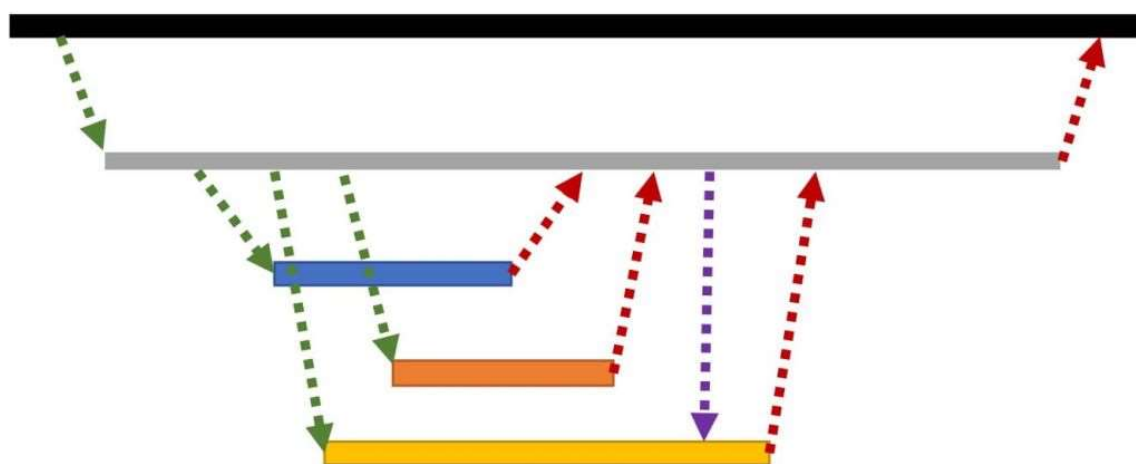
如果你有过上述经历，那这篇文章就是为你而写的!



除了知道git add , git commit , git push之外, Git 还有很多重要的技巧。如果能掌握这些技巧, 将受益无穷。我将会在这篇文章里介绍一些使用 Git 的正确姿势。

Git 工作流

每当项目有多个开发者参与时，正确的 Git 工作流就显得非常重要了。下面我会介绍一个在多人项目中非常有效的工作流。



- Master Branch
- Release Branch / Tech Lead which is you
- Feature Branch 1 / Alice's Branch
- Feature Branch 2 / Bob's Branch
- Feature Branch 3 / John's Branch
- Pull request
- Creating new branch
- git pull or git merge

情景

你一夜之间被任命为一个项目的技术负责人，这个项目旨在打造下一个 Facebook。你的团队有三个开发者：

1. Alice：一年工作经验，熟悉编程
2. Bob：一年工作经验，熟悉编程
3. John：三年工作经验，精通编程

4. 你：技术负责人

Git 开发流程

Master branch（主分支）

1. Master 分支应该始终有生产环境代码(production code) 的副本
2. 任何人（包括技术负责人在内）都不允许直接在 master 分支上写代码，因为它只是生产环境代码的副本
3. 实际代码写在其他分支中

Release branch（发布分支）

1. 项目开始时，首先要为项目创建 release branch。release branch是从 master branch 创建的。
2. 与此项目有关的所有代码都在 release branch 里。release branch 其实只是一个前缀为 release/ 的普通分支。
3. 让我们把这个例子中的 release branch 取名为 release/fb。
4. 由于同一代码库上可能运行着多个项目，所以需要为每一个项目创建一个单独的 release branch。假设还有另一个项目同时运行着，那么这个项目就有一个比如叫 release/messenger 的单独的 release branch。
5. Release branch 的存在是为了让同一代码库能同时运行多个项目而不会相互干扰。

Feature branch（功能分支）

1. 对于构建在应用程序中的每个功能，都会创建一个单独的 feature 分支。这确保了各个功能能被独立构建。
2. Feature 分支只是有着 feature/ 前缀的普通分支。
3. 现在，作为技术负责人的你让 Alice 给你们的 Facebook 写一个登陆界面，于是她创建了一个新的 feature 分支，我们就叫它 feature/login。Alice 将会把所有登陆环节的代码都写在这个 feature 分支上。
4. 这个 feature 分支是从你们的 release 分支创建的。
5. Bob 的任务则是写添加好友页面，所以Bob 创建了一个叫做 feature/friendrequest 的

feature 分支。

6. John 的任务是构建 newsfeed，所以 John 创建了名叫 feature/newsfeed 的 feature 分支。

7. 所有开发者写的代码都在他们各自的 feature 分支，到目前为止一切顺利。

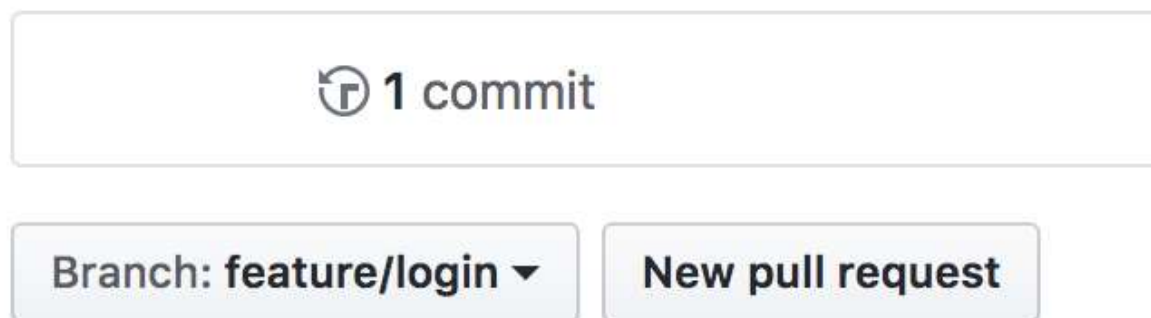
8. 现在，假设 Alice 已经完成了登陆代码，她需要把代码从她的 feature 分支 feature/login 发送到 release 分支 release/fb。这是通过 pull 请求完成的。

Pull 请求 (Pull Request)

首先要指出，pull 请求和 git pull 是两回事。

开发者不能直接把代码推送到 release 分支。feature 分支上的代码需要经过技术负责人的审查，才能推送到 release 分支。这时候需要 pull 请求。


在 Github 中，你可以像下面那样提出 pull 请求：



在分支名的右边有个 "New pull request (新 pull 请求)" 选项，点击它就会打开如下画面：

Comparing changes

Choose two branches to see what's changed or to start a new



上图中：

- compare 分支是 Alice 的 feature 分支 feature/login.
- base 分支是 release 分支 release/fb.

这一步完成之后，Alice 需要为这个 pull 请求输入标题和描述，最终点击" Create Pull Request（创建 Pull 请求）”。Alice 也需要为这个 pull 请求指定一个审查者，因为你是技术负责人，所以她输入了你的名字。

技术负责人审查 pull 请求的代码之后，把代码从 feature 分支合并(merge)到 release 分支。

这样你就成功把 feature/login 分支的代码合并(merge)到 release/fb 分支了，Alice 表示很开心她的代码被合并了。

代码冲突

1. Bob 也写完了代码，提出了一个从 feature/friendrequest 到 release/fb 的 pull 请求。
2. 由于 release 分支已经有登陆功能的代码，引起了代码冲突。审查者有责任解决代码冲突并合并代码，于是作为技术负责人的你出马解决了问题，合并了冲突的代码。
3. 现在 John 也完成了他的代码，想要把代码加入到 release 分支中。但是老鸟 John 非常擅长处理代码冲突。所以 John 从 release/fb 分支抓取了最新的代码到他自己的分支 feature/newsfeed 上（通过 `****git pull` 或 `git merge` 都行）。John 解决了所有的代码冲突，现在 feature/newsfeed 分支也像 release/fb 一样包含了现有的全部代码。
4. 最后，John 提出了 pull 请求。因为他之前已经合并好了代码，这次的 pull 请求没有出现代码冲突。

综上，有两种方法来解决代码冲突：

- 第一种：由pull 请求的审查者解决代码冲突。
- 第二种：开发者确保 release 分支的最新代码已经合并到自己的 feature 分支，自己处理好代码冲突。

又回到 master 分支

当项目完成，release 分支里的代码被合并到 master 分支，然后代码被部署到生产中。因此，生产中的代码和 master 分支中的代码始终保持同步。这也确保了对于任何未来的项目，master 分支中都提供了最新的代码。

恭喜，你现在 get 使用 Git 的正确姿势了！Git 还有一些别的概念如修改提交、重写历史等同样很有用，但 Git 工作流对于大项目的成功来说至关重要。

— 完 —