像大多数新手一样，我一开始是在 StackOverflow 上搜索 Git 命令，然后把答案复制粘贴，并没有真正理解它们究竟做了什么。



Image credit: XKCD[1]

我曾经想过：“如果有一个最常见的 Git 命令的列表，以及它们的功能是什么，这不是极好的吗？”

多年之后，我编制了这样一个列表，并且给出了一些最佳实践，让新手们甚至中高级开发人员都能从中发现有用的东西。

为了保持实用性，我将这个列表与我过去一周实际使用的 Git 命令进行了比较。

几乎每个开发人员都在使用 Git，当然很可能是 GitHub。但大多数开发者大概有 99％ 的时间只是使用这三个命令：

git add --all

git commit -am "<message>"

git push origin master

如果你只是单枪匹马，或者参加一场黑客马拉松或开发一次性的应用时，它工作得很好，但是当稳定性和可维护性开始成为一个优先考虑的事情后，清理提交、坚持分支策略和提交信息的规范性就变得很重要。

我将从常用命令的列表开始，使新手更容易了解 Git 能做什么，然后进入更高级的功能和最佳实践。

**经常使用的命令**

要想在仓库repo中初始化 Git，你只需输入以下命令即可。如果你没有初始化 Git，则不能在该仓库内运行任何其他的 Git 命令。

git init

如果你在使用 GitHub，而且正在将代码推送到在线存储的 GitHub 仓库中，那么你正在使用的就是远程remote仓库。该远程仓库的默认名称（也称为别名）为 origin。如果你已经从 Github 复制了一个项目，它就有了一个 origin。你可以使用命令 git remote -v 查看该 origin，该命令将列出远程仓库的 URL。

如果你初始化了自己的 Git 仓库，并希望将其与 GitHub 仓库相关联，则必须在 GitHub 上创建一个，复制新仓库提供的 URL，并使用 git remote add origin <URL> 命令，这里使用 GitHub 提供的 URL 替换 <URL>。这样，你就可以添加、提交和推送更改到你的远程仓库了。

最后一条命令用在当你需要更改远程仓库时。如果你从其他人那里复制了一个仓库，并希望将远程仓库从原始所有者更改为你自己的 GitHub 帐户。除了改用 set-url 来更改远程仓库外，流程与 git remote add origin 相同。

git remote -v

git remote add origin <url>

git remote set-url origin <url>

复制仓库最常见的方式是使用 git clone，后跟仓库的 URL。

请记住，远程仓库将连接到克隆仓库原属于的帐户。所以，如果你克隆了一个属于别人的仓库，你将无法推送到 GitHub，除非你使用上面的命令改变了 origin。

git clone <url>

你很快就会发现自己正在使用分支。如果你还不理解什么是分支，有许多其他更深入的教程，你应该先阅读它们，再继续下面的操作。（这里是一个教程[2]）

命令 git branch 列出了本地机器上的所有分支。如果要创建一个新的分支，可以使用命令 git branch <name>，其中 <name> 表示分支的名字，比如说 master。

git checkout <name> 命令可以切换到现有的分支。你也可以使用 git checkout -b 命令创建一个新的分支并立即切换到它。大多数人都使用此命令而不是单独的 branch 和 checkout 命令。

git branch

git branch <name>

git checkout <name>

git checkout -b <name>

如果你对一个分支进行了一系列的更改，假如说此分支名为 develop，如果想要将该分支合并回主分支（master）上，则使用 git merge <branch> 命令。你需要先检出（checkout）主分支，然后运行 git merge develop 将 develop 合并到主分支中。

git merge <branch>

如果你正在与多个人进行协作，你会发现有时 GitHub 的仓库上已经更新了，但你的本地却没有做相应的更改。如果是这样，你可以使用 git pull origin <branch> 命令从远程分支中拉取最新的更改。

git pull origin <branch>

如果您好奇地想看到哪些文件已被更改以及哪些内存正在被跟踪，可以使用 git status 命令。如果要查看每个文件的更改，可以使用 git diff 来查看每个文件中更改的行。

git status

git diff --stat

**高级命令和最佳实践**

很快你会到达一个阶段，这时你希望你的提交看起来整洁一致。你可能还需要调整你的提交记录，使得提交更容易理解或者能还原一个意外的有破坏性的更改。

git log 命令可以输出提交的历史记录。你将使用它来查看提交的历史记录。

你的提交会附带消息和一个哈希值，哈希值是一串包含数字和字母的随机序列。一个哈希值示例如下：c3d882aa1aa4e3d5f18b3890132670fbeac912f7。

git log

假设你推送了一些可能破坏了你应用程序的东西。你最好回退一个提交然后再提交一次正确的，而不是修复它和推送新的东西。

如果你希望及时回退并从之前的提交中检出（checkout）你的应用程序，则可以使用该哈希作为分支名直接执行此操作。这将使你的应用程序与当前版本分离（因为你正在编辑历史记录的版本，而不是当前版本）。

git checkout c3d88eaa1aa4e4d5f

然后，如果你在那个历史分支中做了更改，并且想要再次推送，你必须使用强制推送。

注意：强制推送是危险的，只有在绝对必要的时候才能执行它。它将覆盖你的应用程序的历史记录，你将失去之后版本的任何信息。

git push -f origin master

在其他时候，将所有内容保留在一个提交中是不现实的。也行你想在尝试有潜在风险的操作之前保存当前进度，或者也许你犯了一个错误，但希望在你的版本历史中避免尴尬地留着这个错误。对此，我们有 git rebase。

假设你在本地历史记录上有 4 个提交（没有推送到 GitHub），你要回退这是个提交。你的提交记录看起来很乱很拖拉。这时你可以使用 rebase 将所有这些提交合并到一个简单的提交中。

git rebase -i HEAD~4

上面的命令会打开你计算机的默认编辑器（默认为 Vim，除非你将默认修改为其他的），提供了几个你准备如何修改你的提交的选项。它看起来就像下面的代码：

pick 130deo9 oldest commit message

pick 4209fei second oldest commit message

pick 4390gne third oldest commit message

pick bmo0dne newest commit message

为了合并这些提交，我们需要将 pick 选项修改为 fixup（如代码下面的文档所示），以将该提交合并并丢弃该提交消息。请注意，在 Vim 中，你需要按下 a 或 i 才能编辑文本，要保存退出，你需要按下 Esc 键，然后按 shift + z + z。不要问我为什么，它就是这样。

pick 130deo9 oldest commit message

fixup 4209fei second oldest commit message

fixup 4390gne third oldest commit message

fixup bmo0dne newest commit message

这将把你的所有提交合并到一个提交中，提交消息为 oldest commit message。

下一步是重命名你的提交消息。这完全是一个建议的操作，但只要你一直遵循一致的模式，都可以做得很好。这里我建议使用 Google 为 Angular.js 提供的提交指南[3]。

为了更改提交消息，请使用 amend 标志。

git commit --amend

这也会打开 Vim，文本编辑和保存规则如上所示。为了给出一个良好的提交消息的例子，下面是遵循该指南中规则的提交消息：

feat: add stripe checkout button to payments page

- add stripe checkout button

- write tests for checkout

保持指南中列出的类型type的一个优点是它使编写更改日志更加容易。你还可以在页脚footer（再次，在指南中规定的）中包含信息来引用问题issue。

注意：如果你正在协作一个项目，并将代码推送到了 GitHub，你应该避免重新引用（rebase）并压缩（squash）你的提交。如果你开始在人们的眼皮子底下更改版本历史，那么你可能会遇到难以追踪的错误，从而给每个人都带来麻烦。

Git 有无数的命令，但这里介绍的命令可能是您最初几年编程所需要知道的所有。

Sam Corcos 是 Sightline Maps[4] 的首席开发工程师和联合创始人，Sightline Maps 是最直观的 3D 打印地形图的平台，以及用于构建 Phoenix 和 React 的可扩展生产应用程序的中级高级教程网站 LearnPhoenix.io[5]。使用优惠码：freecodecamp 取得 LearnPhoenix 的20美元。

（题图：GitHub Octodex[6]）

via: https://medium.freecodecamp.org/git-cheat-sheet-and-best-practices-c6ce5321f52

作者：Sam Corcos[7] 译者：firmianay 校对：wxy

本文由 LCTT 原创编译，Linux中国 荣誉推出

* [1]: XKCD - https://xkcd.com/1597/
* [2]: 这里是一个教程 - https://guides.github.com/introduction/flow/
* [3]: Google 为 Angular.js 提供的提交指南 - https://github.com/angular/angular.js/blob/master/CONTRIBUTING.md#-git-commit-guidelines
* [4]: Sightline Maps - //sightlinemaps.com/
* [5]: LearnPhoenix.io - //learnphoenix.io/
* [6]: GitHub Octodex - https://octodex.github.com/
* [7]: Sam Corcos - https://medium.freecodecamp.org/@SamCorcos?source=post\_header\_lockup