# 掌握 Git 之美

## 内容

使用 7 条简单的 Git 命令开始你的软件开发之旅

你是否曾经想知道如何学好 Git？你长期以来都是跌跌撞撞地在使用 Git。最终，你总需要掌握它的窍门。这就是我写这篇文章的原因，我将带你去启蒙之旅。这儿是我关于如何加快 Git 学习过程的基本指南。我将介绍 Git 的实际情况以及我使用最多的 7 条 Git 命令。本文主要针对有兴趣的开发人员和大学新生，他们需要关于 Git 的介绍以及如何掌握基础知识。

---

你可以往前继续阅读整篇文章，或者只读 TLDR; 部分，尽管这将使我很受伤。

### TLDR;

在学习 Git 的过程中，请养成下面这些步骤的习惯：

1. 随时使用 git status！
2. 只更改那些你真正想更改的文件。
3. git add -A 会是你的朋友。
4. 随时使用命令 git commit -m "meaningful messages"。
5. 做任何推送（push）之前先使用命令 git pull，但是这需要在你提交过一些更改之后。
6. 最后，git push 推送提交的更改。

---

### 良宵莫辜负

对任何开发人员来说，通常第一步都是选择一个广泛使用的地方托管他或她的代码库。那就是，GitHub。它是一切有关代码的聚集地。要理解 GitHub 的概念，你先需要知道什么是 Git。

Git 是一款基于命令行的版本控制软件，在 Windows 和 Mac 系统上也有几款可用的桌面应用。 Git 由 Linux 之父 Linus Torvalds 开发，Linus Torvalds 还是是计算机科学中最有影响力的人物之一。因为这一优势，Git 已经成为绝大多数软件开发人员关于共享和维护代码的标准。这一大段话，让我们将其细细道来。正如它的名字所说，版本控制软件 Git 让你可以预览你写过的代码的所有版本。从字面上来说， 开发人员的每个代码库都将永远存储在其各自的仓库中，仓库可以叫做任何名字，从 *pineapple* 到 *express* 都行。在此仓库开发代码的过程中，你将进行出无数次的更改，直到第一次正式发布。这就是版本控制软件如此重要的核心原因所在。它让作为开发人员的你可以清楚地了解对代码库进行的所有更改、修订和改进。从另外一个方面说，它使协同合作更容易，下载代码进行编辑，然后将更改上传到仓库。然而，尽管有了这么多好处，然而还有一件事可以锦上添花。你可以下载并使用这些文件，即使你在整个开发过程中什么事也没有做。

让我们回到文章的 GitHub 部分。它只是所有仓库的枢纽（hub），这些仓库可以存储在其中并在线浏览。它是一个让有着共同兴趣的人相聚的地方。

### 千里之行始于足下

OK，记住，Git 是一款软件，像任何其他软件一样，你首先需要安装它：

Git - 安装 Git，如果你希望从源代码安装 Git，你需要安装这些 Git 的依赖库： autotools —— 来自 git-scm.com
*Tips：请点击上面的链接，然后按照说明开始。*
完成了安装过程，很好。现在你需要在你的浏览器地址栏输入 github.com 访问该网站。如果你还没有帐号的话需要新创建一个帐号，这就是你的起舞之处。登录并创建一个新仓库，命名为 Steve ，没有什么理由，只是想要一个名为史蒂夫的仓库好玩而已。选中 “Initialize this repository with a README” 复选框并点击创建按钮。现在你有了一个叫做 Steve 的仓库。我相信你会为你自己感到自豪。

## 现在开始在使用 Git

现在是比较有趣的部分。你将把 Steve 克隆到你本地的机器上。可以把这个过程看作从 Github 上复制仓库到你的电脑上。点击 “clone or download” 按钮，你将看到一个类似下面这样的 URL:

https://github.com/yourGithubAccountName/Steve.git
复制这个 URL 并打开命令提示符窗口。现在输入并运行条命令：

git clone https://github.com/yourGithubAccountName/Steve.git
Abrakadabra！Steve 仓库已经被自动克隆到了你的电脑上。查看你克隆这个仓库的目录，你会看到一个叫做 Steve 的文件夹。这个本地的文件夹现在已经链接到了它的 “origin” ，也就是 GitHub 上的远程仓库。

记住这个过程，在你的软件开发工程人员的职业生涯中你一定会重复这个过程很多次的。完成所有这些准备工作之后，你就可以开始使用最普通且常用的 Git 命令了。

## 你现在已经开始在真实场景使用 Git 了

找到 Steve 目录并在该目录中打开命令提示符窗口，运行下面的命令：

git status
这会输出你的工作目录的状态，让你知道所有你编辑过的文件。这意味着它显示了远程库中和本地工作目录中之间的文件差异。status 命令被用来作为 commit 的模版，我将在这篇教程后面进一步谈论 commit 。简单的说，[git status][1] 告诉你你编辑过哪些文件，以给你一个你想要上传到远程库的概述。
但是，在你做任何上传之前，你首先需要做的是选择你需要发送回远程库的文件。使用下面命令完成：

git add
接着在 Steve 目录新建一个文本文件，可以取一个好玩的名字 pineapple.txt。在这个文件里面随便写些你想写的内容，返回命令提示符，然后再次输入 git status。现在，你将看到这个文件以红色出现在标记 “untracked files” 下面。

On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
(use "git add <file>..." to include in what will be commited)

pineapple.txt
下一步就是将它添加到暂存区（staging）。暂存区可以看作是这样的一个环境：你做过的所有更改在提交时都将捆绑为一个更改而被提交。现在，你可以将这个文件加入暂存区：

git add -A
-A 选项意味着所有你更改过的文件都会被加到暂存区等待提交。然而， git add 非常灵活，它也可以像这样一个文件一个文件的添加：
git add pineapple.txt
这种方法让你有能力选择你想要暂存的每一个文件，而不用担心加入那些你不想改变的东西。

再次运行 git status，你会看到如下输出：
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

new file: pineapple.txt
准备好提交更改了吗？开始吧。

git commit -m "Write your message here"
Git commit 命令会将存储在暂存区中的文件和来自用户的用于描述更改的日志信息一起存储在一个新的地方。-m 选项加入了写在双引号内的信息。
再次检查状态，你会看到：

On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working directory clean
所有的更改现在都被加入到一次提交当中了，同时会有一条与你所做相关的信息。现在你可以用 git push 将这次提交推送到远程库 “origin”了。这条命令就像字面意义所说，它会把你提交的更改从本地机器上传到 GitHub 的远程仓库中。返回到命令提示符，然后运行：
git push
你会被要求输入你的 GitHub 帐号和密码，之后你会看到类似下面的这些内容：

Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/yourGithubUserName/Steve.git
c77a97c..08bb95a master -> master

就是这样。你已经成功上传了你本地的更改。看看你在 GitHub 上的仓库，你会看到它现在包含了一叫做 pineapple.txt 的文件。

如果你是一个开发小组的一员呢？如果他们都推送提交到 “origin”，将会发生什么？这就是 Git 真正开始发挥它的魔力的时候。你可以使用一条简单的命令轻松地将最新版本的代码库 pull 到你本地的机器上：

```
git pull
```

但是 Git 也有限制：你需要有相匹配的版本才能推送到 “origin”。这意味着你本地的版本需要和 origin 的版本大致一样。当你从 “origin” 拉取（pull）文件时，在你的工作目录中不能有文件，因为它们将会在这个过程中被覆盖。因此我给出了这条简单的建议。在学习 Git 的过程中，请养成下面这些步骤的习惯：

随时使用 git status！
只更改那些你真正想更改的文件。
git add -A 会是你的朋友。
随时使用命令 git commit -m "meaningful messages"。
做任何推送（push）之前先使用命令 git pull，但是这需要在你提交过一些更改之后。
最后，git push 推送提交的更改。

---

嘿！你还在看这篇文章吗？你已经看了很久了，休息一下吧！

休息好了吗？好的！让我们来处理一些错误。如果你不小心更改了一些你本不应该更改的文件后怎么办呢？不需要担心，只需要使用 git checkout。让我们在文件 pineapple.txt 里更改一些内容：在文件中加入一行，比方说，“Steve is mega-awesome!” 。然后保存更改并用 git status 检查一下：

```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

modified: pineapple.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

正如预料的那样，它已经被记录为一次更改了。但是，假如 Steve 实际上并不是很优秀呢？假如 Steve 很差劲呢？不用担心！最简单的还原更改的方式是运行命令：

```
git checkout -- pineapple.txt
```

现在你会看到文件已经恢复到了先前的状态。

但是假如你玩砸了呢？我是说，事情已经变得混乱，并且需要把所有东西重置到与 “origin” 一样的状态。也不需要担心，在这种紧急情况下我们可以享受 Git 的美妙之处：

```
git reset --hard
```

git reset 命令和 --hard 选项一起可以抛弃自上次提交以来的所有更改，有些时候真的很好用。

---

最后，我想鼓励你尽可能多地使用 Git。这是能够熟练使用它的最好学习方式。除此之外，养成阅读 Git 文档的习惯。一开始可能会有些云里雾里，但是过段时间后你就会明白它的窍门了。

*希望你们（小伙子和姑娘们）读这篇文章的时候会和我写它时一样的开心。如果你认为这篇文章对别人有用，你尽可以与别人分享它。或者，如果你喜欢这篇文章，你可以在下面点下赞以便让更多的人看到这篇文章。*