Git 和 GitHub 入门指南（教程）

2015 年 10 月 1 日 / 作者 MEGHAN NELSON

.

· 分享

.

8 月，我们在 HubSpot 主持了一次《女性代码见面会》，并开展了针对初学者的使用 git 和 GitHub 的研讨会。我首先浏览了有关 git 的 基础知识 和背景的``幻灯片演示"，然后我们分成几个小组来运行我创建的教程来模拟大型协作项目的工作。活动结束后我们得到了反馈，它是一个实用的动手介绍。因此，如果您也是 git 的新手，请按照以下步骤操作以轻松更改代码库，打开拉取请求（PR）并将代码合并到 master 分支中。任何重要的 git 和 GitHub 术语均以粗体显示，并带有官方 git 参考资料的链接。

*准备工作：*安装 git 并创建一个 GitHub 帐户

安装 git 并创建一个 GitHub 帐户

请按照 此处的 说明安装 git（如果没有安装的话）***请注意，对于本教程，我们将仅在命令行上使用 git。***尽管有一些很棒的 git GUI（图形用户界面），但我认为先使用 git 特定的命令学习 git，然后在更熟悉该命令后尝试 git GUI 会更容易。

完成此操作后，在此处创建一个 GitHub 的账户.（公共存储库是免费的，但私人存储库是收费的。）

安装 git 并创建一个 GitHub 帐户

请按照 此处的 说明安装 git（如果没有安装的话）***请注意，对于本教程，我们将仅在命令行上使用 git。***尽管有一些很棒的 git GUI（图形用户界面），但我认为先使用 git 特定的命令学习 git，然后在更熟悉该命令后尝试 git GUI 会更容易。

完成此操作后，在此处创建一个 GitHub 的账户.（公共存储库是免费的，但私人存储库是收费的。）

步骤 1*创建本地 git 存储库*

*使用 git 在本地计算机上创建新项目时，首先要创建一个新项目*的仓库* (短时间的, '*仓库*',).

要使用 git，我们将使用终端。如果您对终端和基本命令没有太多经验，请查看这份教程 （尤其是“导航文件系统”和“移动”部分）。

首先，打开一个终端，然后使用cd（更改目录）命令移动到要在本地计算机上放置项目的位置。例如，如果您的桌面上有一个'项目'文件夹，则可以执行以下操作：

mnelson:Desktop mnelson$ cd ~/Desktop

mnelson:Desktop mnelson$ mkdir myproject

mnelson:Desktop mnelson$ cd myproject/

*view raw**terminalcd.md* hosted with ❤ by *GitHub*

mnelson:Desktop mnelson$ cd ~/Desktop

mnelson:Desktop mnelson$ mkdir myproject

mnelson:Desktop mnelson$ cd myproject/

*view raw**terminalcd.md* hosted with ❤ by *GitHub*

*要在文件夹的根目录中初始化 git 存储库，请运行**git init* 命令:

mnelson:myproject mnelson$ git init

Initialized empty Git repository in /Users/mnelson/Desktop/myproject/.git/

*view raw**gitinit.md* hosted with ❤ by *GitHub*

mnelson:myproject mnelson$ git init

Initialized empty Git repository in /Users/mnelson/Desktop/myproject/.git/

*view raw**gitinit.md* hosted with ❤ by [***GitHub***]

步骤 2：在仓库里创建一个新文件夹

命令。继续，使用您喜欢的任何文本编辑器或运行命令将新文件添加到项目中。

在包含 git repo 的文件夹中添加或修改文件后，git 会注意到该 repo 内部已进行更改。但是，除非您明确告诉它，否则 git 不会正式跟踪该文件（即将其放入提交中-接下来我们将进一步讨论提交）。

```
mnelson:myproject mnelson$ touch mnelson.txt
```

```
mnelson:myproject mnelson$ ls

mnelson.txt
```

view rawaddfile.md hosted with ❤ by GitHub

```
mnelson:myproject mnelson$ touch mnelson.txt

mnelson:myproject mnelson$ ls

mnelson.txt
```

view rawaddfile.md hosted with ❤ by GitHub

创建新文件后，您可以使用"status"'命令查看 git 知道存在哪些文件。

```
mnelson:myproject mnelson$ git status

在分支机构主管



初次提交



未跟踪的文件：

   （使用“ git add <文件> ...”包括将要提交的内容）

                mnelson.txt

没有添加任何内容提交但存在未跟踪的文件（使用“ git add”进行跟踪）
```

view rawgitstatus.md hosted with ❤ by GitHub

```
mnelson:myproject mnelson$ git status

在分支机构主管

初次提交

未跟踪的文件：

   （使用“ git add <文件> ...”包括 in 将要提交的内容）

                mnelson.txt

没有添加任何内容提交但存在未跟踪的文件（使用“ git add”进行跟踪）
```

view rawgitstatus.md hosted with ❤ by GitHub

这基本上是说：“嘿，我们注意到您创建了一个名为 mnelson.txt 的新文件，但是除非您使用'gitadd'``命令，否则我们将不会对其进行任何处理。``”

插曲：登台环境，提交和您

初学 git 时最容易混淆的部分之一是暂存环境的概念及其与提交的关系。

“提交”记录了自上次提交以来您更改了哪些文件。本质上，您可以对存储库进行更改（例如，添加文件或修改文件），然后告诉 git 将这些文件放入提交中。

提交构成了项目的本质，并允许您随时回到项目状态。

那么，如何告诉 git 提交哪些文件呢？**这是“**临时环境”或“索引”** 进入的地方。**如步骤 2 所示，当您对存储库进行更改时，git 会注意到文件已更改，但不会对其进行任何操作（例如将其添加到提交中）。

要将文件添加到提交中，首先需要将其添加到登台环境中。为此，您可以使用 gitadd**<****文件名>** 命令（请参阅下面的步骤 3）。

一旦使用了 git add 命令将所需的所有文件添加到暂存环境中，就可以使用 gitcommit 命令告诉 git 将它们打包为一个提交。

注意：**暂存环境（也称为``暂存"）是此的新的首选术语，但您也可以将其称为”索引"。**

初学 git 时最容易混淆的部分之一是暂存环境的概念及其与提交的关系。

“提交”记录了自上次提交以来您更改了哪些文件。本质上，您可以对存储库进行更改（例如，添加文件或修改文件），然后告诉 git 将这些文件放入提交中。

提交构成了项目的本质，并允许您随时回到项目状态。

那么，如何告诉 git 提交哪些文件呢？**这是“**临时环境”或“索引”** 进入的地方。**如步骤 2 所示，当您对存储库进行更改时，git 会注意到文件已更改，但不会对其进行任何操作（例如将其添加到提交中）。

要将文件添加到提交中，首先需要将其添加到登台环境中。为此，您可以使用 gitadd**<****文件名>** 命令（请参阅下面的步骤 3）。

一旦使用了 git add 命令将所需的所有文件添加到暂存环境中，就可以使用 gitcommit 命令告诉 git 将它们打包为一个提交。

注意：**暂存环境（也称为``暂存"）是此的新的首选术语，但您也可以将其称为”索引"。**

步骤 3:将文件添加到暂存环境

使用 **gitadd** 命令将文件添加到暂存环境。

如果重新运行 git status 命令，则会看到 git 已将文件添加到登台环境中（请注意“要提交的更改“行）。

```
mnelson:myproject mnelson$ git status


在分支机构主管
```

初次提交

所做更改：

（使用“ git rm --cached <file> ...”取消登台）

新文件：mnelson.txt

查看由 GitHub 托管于❤的 rawaddtostaging.md

mnelson:myproject mnelson$ git status

在分支机构主管

初次提交

所做更改：

（使用“ git rm --cached <file> ...”取消登台）

```
                新文件：mnelson.txt
```

查看由 GitHub 托管于❤的 rawaddtostaging.md

重申一下，该文件**尚未**添加到提交中，但是已经添加了。

步骤 4：创建提交

是时候创建您的第一次提交了！

运行命令 `git commit -m“```关于提交的消息```”`

```
mnelson:myproject mnelson$ git commit -m "This is my first commit!"

[master (root-commit) b345d9a] This is my first commit!

 1 file changed, 1 insertion(+)

 create mode 100644 mnelson.txt
```

view rawcommit.md hosted with ❤ by GitHub


```
mnelson:myproject mnelson$ git commit -m "This is my first commit!"

[master (root-commit) b345d9a] This is my first commit!

 1 file changed, 1 insertion(+)

 create mode 100644 mnelson.txt
```

view rawcommit.md hosted with ❤ by GitHub

提交末尾的消息应该与提交所包含的内容有关-可能是一项新功能，可能是一个错误修复，也许只是在修复输入错误。不要放置“ asdfadsf”或“ foobar”之类的消息。这使看到您的承诺的其他人感到悲伤。非常非常伤心。

步骤 5：创建新分支

现在您进行了新的提交，让我们尝试一些更高级的东西。

假设您要制作新功能，但担心在开发功能时对主项目进行更改。这是 **git 分支** 出现的地方

分支允许您在项目的“状态”之间来回移动。例如，如果您想在网站上添加新页面，则可以仅为该页面创建一个新分支，而不会影响项目的主要部分。完成页面后，您可以将更改从分支合并 到 master 分支。当您创建一个新分支时，Git 会跟踪分支提交的分支，因此它知道所有文件的历史记录。

假设您在 master 分支上，并且想要创建一个新分支来开发您的网页。以下是你需要做的：运行 **git checkout -b <我的分支名称>**.该命令将自动创建一个新分支，然后在其上“签出”，这意味着 git 会将您移至该分支，脱离 master 分支。

在运行上述命令之后，您可以使用 git 分支 命令来确定您的分支已创建：

```
mnelson:myproject mnelson$ git branch


  master* my-new-branch
```

view rawgitbranch.md hosted with ❤ by GitHub


```
mnelson:myproject mnelson$ git branch


  master* my-new-branch
```

view rawgitbranch.md hosted with ❤ by GitHub

分支名称旁边带有星号表示在给定时间指向的分支。

现在，如果您切换回 master 分支并进行更多提交，则在您将这些更改合并 到新分支之前，新分支将看不到任何更改。

步骤 6：建立新仓库

如果您只想在本地保存密码足迹，那么您不必使用 Github 但如果您想团队合作，那么您可以用 Github 合作修改项目密码

如果您想在 Github 创立新仓库，请先登录 Github 点击首页您将看到一个绿色的“+新仓库”按钮

点击该按钮，Github 将会让您命名新仓库并给您提供一个简介

当您完成信息填写后，请点击“创立仓库”按钮，创立新仓库

Github 将询问您是否想重新创建一个新仓库，还是想添加一个本地创立的仓库在本例中，由于我们已经在本地创建了一个新的回购，因此我们希望将其推送到 Gihub 上，以便遵循**.或从命令行部分推送**现有存储库：

```
mnelson:myproject mnelson$ git remote add origin
https://github.com/cubeton/mynewrepository.git

mnelson:myproject mnelson$ git push -u origin master

Counting objects:3, done.

Writing objects:100% (3/3), 263 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/cubeton/mynewrepository.git

 * [new branch]          master -> master

Branch master set up to track remote branch master from origin.
```

view rawaddgithub.md hosted with ❤ by GitHub

```
mnelson:myproject mnelson$ git remote add origin
https://github.com/cubeton/mynewrepository.git

mnelson:myproject mnelson$ git push -u origin master

Counting objects:3, done.

Writing objects:100% (3/3), 263 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/cubeton/mynewrepository.git

 * [new branch]          master -> master

Branch master set up to track remote branch master from origin.
```

view rawaddgithub.md hosted with ❤ by GitHub

(您需要将第一个命令行中的 URL 更改为本节中 GitHub 列出的 URL，因为您的 GitHub 用户名和 repo 名称不同。)

步骤 7:将分支推送到 GitHub

现在，我们将把您分支中的提交到您的新 GitHub 回购 **推送** 到您的新 GitHub 回购 这样，其他人就可以看到您所做的更改如果它们得到存储库所有者的批准，则可以将更改合并到主分支中

要将更改推送到 GitHub 上的新分支，您需要运行***git push*** **origin yourbranchname**``**.**``GitHub 会自动在远程存储库上为您创建分支：

```
mnelson:myproject mnelson$ git push origin my-new-branch

Counting objects:3, done.

Delta compression using up to 8 threads.

Counting objects:100%(2/2), done.

Writing objects:100% (3/3), 313 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/cubeton/mynewrepository.git

 * [new branch]          my-new-branch -> my-new-branch
```

view rawaddgithub.md hosted with ❤ by GitHub

```
mnelson:myproject mnelson$ git push origin my-new-branch

Counting objects:3, done.

Delta compression using up to 8 threads.

Counting objects:100%(2/2), done.

Writing objects:100% (3/3), 313 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/cubeton/mynewrepository.git

 * [new branch]          my-new-branch -> my-new-branch
```

view rawaddgithub.md hosted with ❤ by GitHub

您可能想知道上面命令中的“起源”这个词是什么意思发生的情况是，当您将远程存储库克隆到本地计算机时，Git 会创建一个 **别名**为您在几乎所有情况下，此别名都称为"原点."它本质上是远程存储库的 URL 的简写**因此，要将更改推送到远程存储库，您可以使用以下命令**:** `git push git@github.com:git/git.git yourbranchname` 或者 `git push origin yourbranchname`

(如果这是您第一次在本地使用 GitHub，可能会提示您使用 GitHub 用户名和密码登录。)

如果您刷新 GitHub 页面，您将看到一条注释，上面写着使用您的名字的分支刚刚被推入存储库中您也可以单击“分支机构”链接，查看那里列出的分支机构

现在单击上面屏幕截图中的绿色按钮我们要提出！ **撤退请求**！

步骤 8：创建请（PR）

拉取请求（或 PR）是一种提醒存储库所有者要更改其代码的方式。它允许他们检查代码，并确保代码看起来不错，然后再将更改放入主分支。

这是提交 PR 页之前的样子：

提交 PR 请求后的样子：

您可能会在底部看到一个绿色的大按钮，上面显示“合并拉取请求”。点击这意味着您将您的更改合并到主分支中。

请注意，此按钮并非总是绿色。在某些情况下，它会是灰色的，这意味着您将面临“合并冲突”。 这是当一个文件中的更改与另一个文件中的更改冲突并且 git 无法确定要使用哪个版本时。您必须手动输入并告诉 git 使用哪个版本。

有时，您将成为回购的共同所有者或唯一所有者，在这种情况下，您可能无需创建 PR 来合并您的更改。但是，创建一个分支仍然是一个好主意，这样您可以保留更完整的更新历史记录并确保在进行更改时始终创建一个新分支。

步骤 9：合并成一个 PR

继续并单击绿色的“合并拉取请求”按钮。这会将您的更改合并到主分支中。

完成之后，建议您删除分支（太多分支会变得凌乱），因此也请点击灰色的“删除分支”按钮。

您可以通过单击新存储库首页上的“提交”链接来仔细检查提交是否已合并。

这将显示该分支中所有提交的列表。您可以看到我刚刚合并的那个（合并请求 2）。

您还可以看到在右侧提交的哈希代码 。哈希码是该特定提交的唯一标识符。**这对于引用特定的提交以及撤消更改很有用（使用“ git revert”**<**``**哈希码编号**``**>**``**命令回溯）。**

步骤 10：将 GitHub 上的更改回馈到您的计算机

目前，GitHub 上的资料库看起来与您本地计算机上的略有不同。例如，您在分支中做出并合并到主要分支中的提交在本地计算机上的主分支中不存在。

为了获得您或其他人在 giHub 上合并的最新更改，请使用 **git Pull Origin Master****和**命令(在主分支上工作时)。

```
mnelson：myproject mnelson$git Pull Origin Master

远程：对对象进行计数：1，完成。

远程：总计 1(增量 0)、重用 0(增量 0)、包重用 0

解包对象：100%(1/1)，完成。

来自 https://github.com/cubeton/mynewrepository
```

```
 *分支  主管  ->快点，快点。

    b345d9a..5381b7c 主机->原点/母版

合并由递归的 / 循环的 策略。

 minelsons.txt | 1 +

 1 个文件已更改，1 个插入(+)
```

视图原始 pulloriginmaster.md 由❤托管 GitHub

```
mnelson：myproject mnelson$git Pull Origin Master

远程：对对象进行计数：1，完成。

远程：总计 1(增量 0)、重用 0(增量 0)、包重用 0

解包对象：100%(1/1)，完成。

来自 https://github.com/cubeton/mynewrepository

 *分支  主管  ->快点，快点。

    b345d9a..5381b7c 主机->原点/母版

合并由递归的 / 循环的 策略。

 minelsons.txt | 1 +

 1 个文件已更改，1 个插入(+)
```

原始视图 pulloriginmaster.md 由❤托管 GitHub

这将显示所有已更改的文件以及它们是如何更改的。

现在我们可以使用 git log 命令再次查看所有新提交。

(您可能需要将分支切换回主分支。您可以使用来执行此操作。) **git 签出主机**命令)

```
mnelson:myproject mnelson$ git log

commit3e270876db0e5ffd3e9bfc5edede89b64b83812c
```

合并：4f1cb175381b7c

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Fri Sep1117:48:112015-0400

　　出现分支'主管' of https://github.com/cubeton/mynewrepository

commit4f1cb1798b6e6890da797f98383e6337df577c2a

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Fri Sep 11 17:48:00 2015 -0400

　　添加新文件夹

commit5381b7c53212ca92151c743b4ed7dde07d9be3ce

合并：:b345d9a1e8dc08

作者 Meghan Nelson <meghan@meghan.net>

日期：　Fri Sep 11 17:43:22 2015 -0400

　　Merge 拉取请求 #2 from cubeton/my-newbranch

　　为文件夹添加更多文本

commit1e8dc0830b4db8c93efd80479ea886264768520c

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Fri Sep 11 17:06:05 2015 -0400

　　为文件夹添加更多文本

commit b345d9a25353037afdeaa9fcaf9f330effd157f1

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Thu Sep 10 17:42:15 2015 -0400

　　这是我第一次提交!

原始视图 pulloriginmaster.md 由❤托管 GitHub

mnelson:myproject mnelson$ git log

commit3e270876db0e5ffd3e9bfc5edede89b64b83812c

合并：4f1cb175381b7c

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Fri Sep1117:48:112015-0400

　　出现分支'主管' of https://github.com/cubeton/mynewrepository

commit4f1cb1798b6e6890da797f98383e6337df577c2a

作者 Meghan Nelson <mnelson@hubspot.com>

日期：　Fri Sep 11 17:48:00 2015 -0400

　　添加新文件夹

commit5381b7c53212ca92151c743b4ed7dde07d9be3ce

合并：:b345d9a1e8dc08

作者 Meghan Nelson <meghan@meghan.net>

```
日期：       Fri Sep 11 17:43:22 2015 -0400

    Merge 拉取请求 #2 from cubeton/my-newbranch

    为文件夹添加更多文本

commit1e8dc0830b4db8c93efd80479ea886264768520c
作者 Meghan Nelson <mnelson@hubspot.com>
日期：       Fri Sep 11 17:06:05 2015 -0400

    为文件夹添加更多文本

commit b345d9a25353037afdeaa9fcaf9f330effd157f1
作者 Meghan Nelson <mnelson@hubspot.com>
日期：       Thu Sep 10 17:42:15 2015 -0400

    这是我第一次提交!
```

view rawgitlogaftermerge.md hosted with ❤ by GitHub

步骤 11：徜徉在你的 git 中

您已经成功地进行了公关，并将代码合并到主分支。恭喜！如果你想更深入了解，请点击 这个 Git101 文件夹 更多使用方法 git and GitHub.

我还建议找出一些时间与您的团队一起模拟一个较小的团队项目，就像我们在这里所做的那样。让您的团队使用您的团队名称创建一个新文件夹，并向其中添加一些带文本的文件。然后，尝试将这些更改推送到此远程回购。这样，您的团队就可以开始对他们最初没有创建的文件进行更改，并练习使用公关功能。而且，

使用 GitHub 上的 git 指责和 git 历史工具来熟悉跟踪文件中进行了哪些更改以及是谁进行了这些更改。

你用的 GIT 越多，你就会越舒服.。 别管它了。(我无法抗拒。)

创作由[Meghan Nelson]