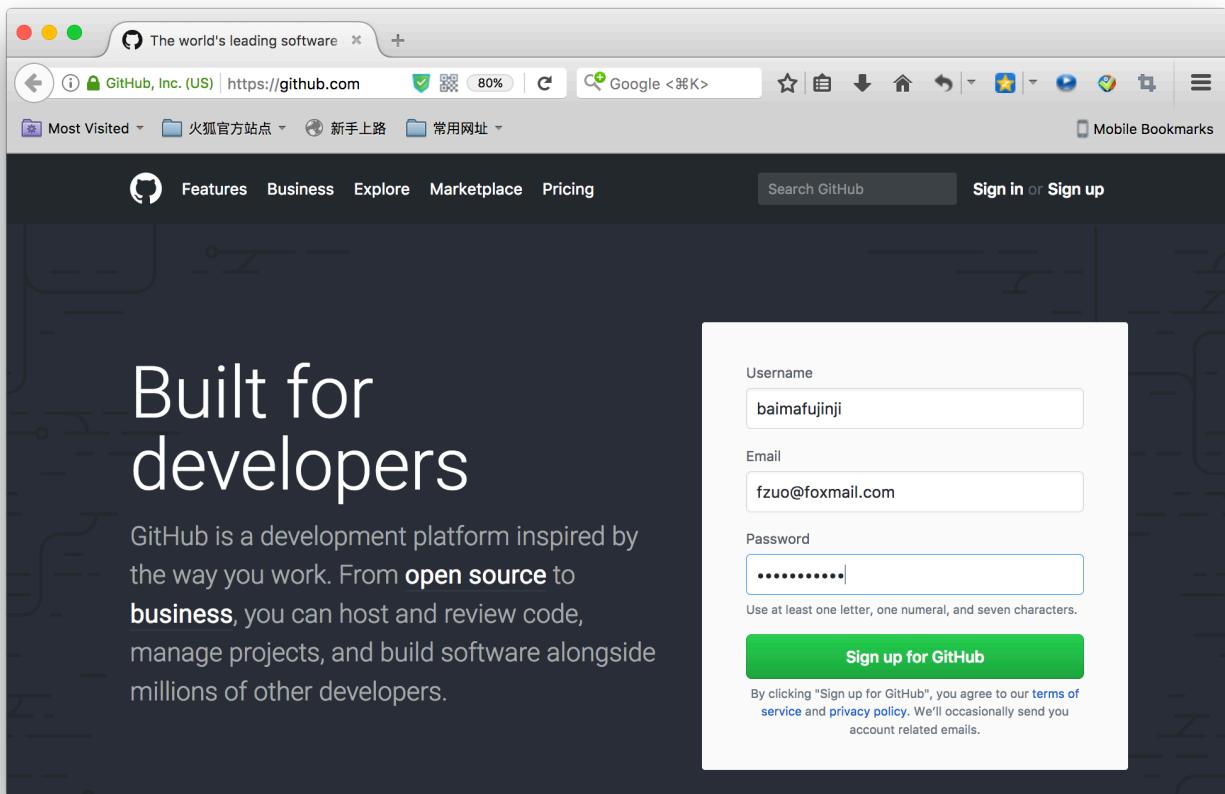


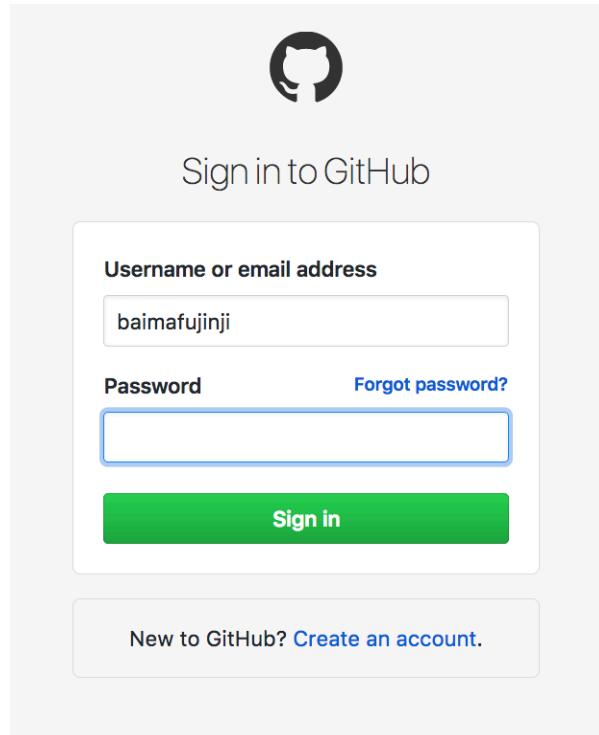
GitHub是一个面向开源及私有软件项目的托管平台、开源代码库以及版本控制系统，因为只支持 Git 作为唯一的版本库格式进行托管，故名 GitHub。通常在 Windows 下使用 GitHub 的教程是非常多的，因此也无需主页君在此多费唇舌。本文主要讨论在 Mac OS X 系统上使用 GitHub 的方法。其实，在 Mac OS X 系统上使用 GitHub 的教程网上也有，但是大部分都过于陈旧，加之系统或用户界面更新的缘故，有些内容已经非常不准确了，这往往给初学者带来很大困扰。本文所讨论之内容均基于最新版本的软件，对于初涉 GitHub 的用户来说更加参考价值。

## 1、注册 GitHub

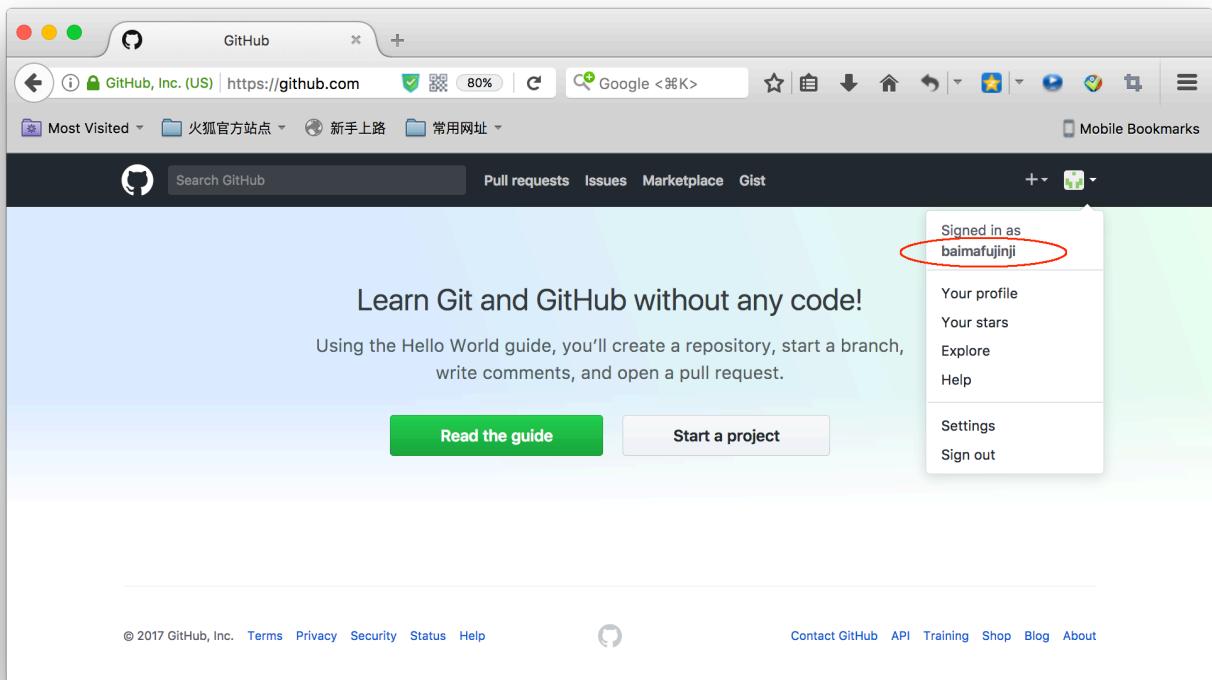
GitHub首先是一个网站，为了把你的代码托管到这个网站上，你应该首先注册为该网站的用户。为此，请先登录 GitHub 的网站 (<https://github.com/>)，然后填写用户名、邮件地址等信息以完成注册，如下图所示。



注册过程没有太多需要解释的，一步一步按提示操作直到注册完成，你就可以用这个账户来登录（Sign In）你的GitHub账户了！登录界面如下。



登录成功后的界面如下（可见红色圆圈里显示了我的登录名）。由于是新注册用户，所以现在GitHub上还没有任何托管的软件项目。按图所示，现在你可以阅读GitHub的用户指引（Read the guide），这会指导你如何使用GitHub（当然，这也是接下来我们要演示的事情）。或者，你也可以直接开始一个新的项目（Start a project）。



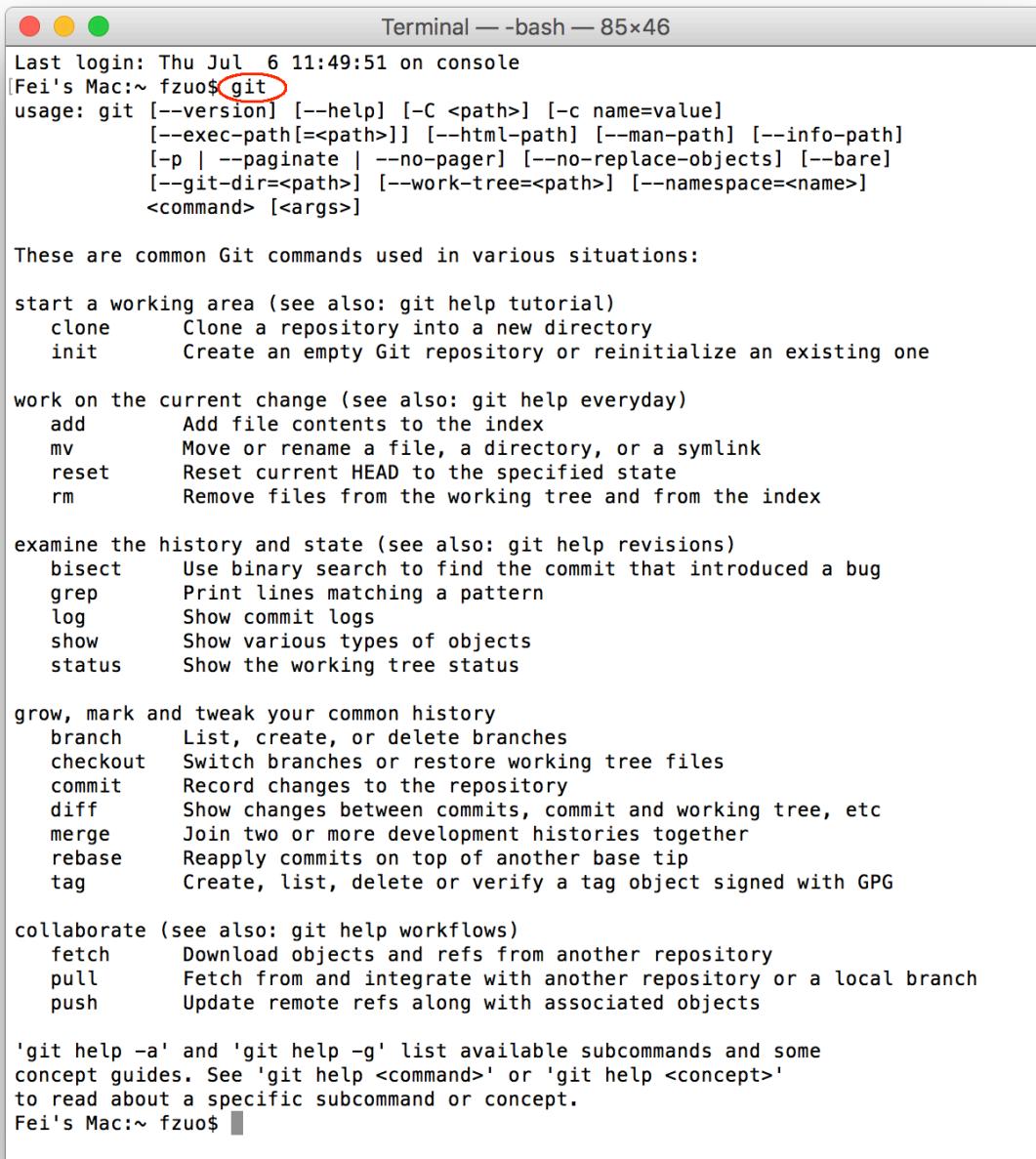
## 2、安装或更新Git

在很多教程里，你会发现很多开发人员是通过命令行界面来操作和使用GitHub的。从专业人士角度来说，这也才真正是打开GitHub的正确方式。在Mac OS X中，命令行也就是指Terminal。要在Terminal中通过键入指令的方法来操作GitHub那么正确地安装Git就是一个先决条件。

当然本文要重点介绍的方法比这个（也就是在命令行界面通过键入指令的方法来操作GitHub）会更简单一些，后面你就会看到我们这里的方法是使用具有图形用户界面的客户端来对GitHub进行操作的。但是，安装Git仍然是必要的。“If you don't already have Git installed, you must configure it before using GitHub Desktop.” 这也是GitHub Desktop的用户手册里明确指出的。

通常，Mac OS X中都已经安装了Git。但是，Git的版本未必是最新的。我们仍然建议你安装使用最新版本的Git。下面来检查一下你的Mac电脑上是否已经安

装了Git。打开你的Terminal，然后在提示符后面输入git，如下图所示，如果git命令可以被系统识别，那么就表示你的电脑上已经安装了Git，系统会列出Git的一些使用方法（因为你现在输入的git指令是不完整的）。



The screenshot shows a terminal window titled "Terminal — bash — 85x46". The command "git" was typed at the prompt, and the system responded with a detailed list of Git commands and their descriptions. The output is organized into sections: common commands, working area, current change, history and state, common history, workflows, and a note about available subcommands and concept guides.

```
Last login: Thu Jul 6 11:49:51 on console
[Fei's Mac:~ fzuo$ git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset     Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

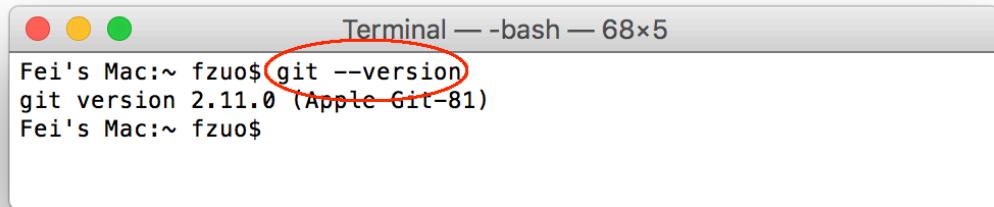
examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status

grow, mark and tweak your common history
  branch   List, create, or delete branches
  checkout Switch branches or restore working tree files
  commit   Record changes to the repository
  diff     Show changes between commits, commit and working tree, etc
  merge   Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  tag     Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch   Download objects and refs from another repository
  pull    Fetch from and integrate with another repository or a local branch
  push    Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
Fei's Mac:~ fzuo$
```

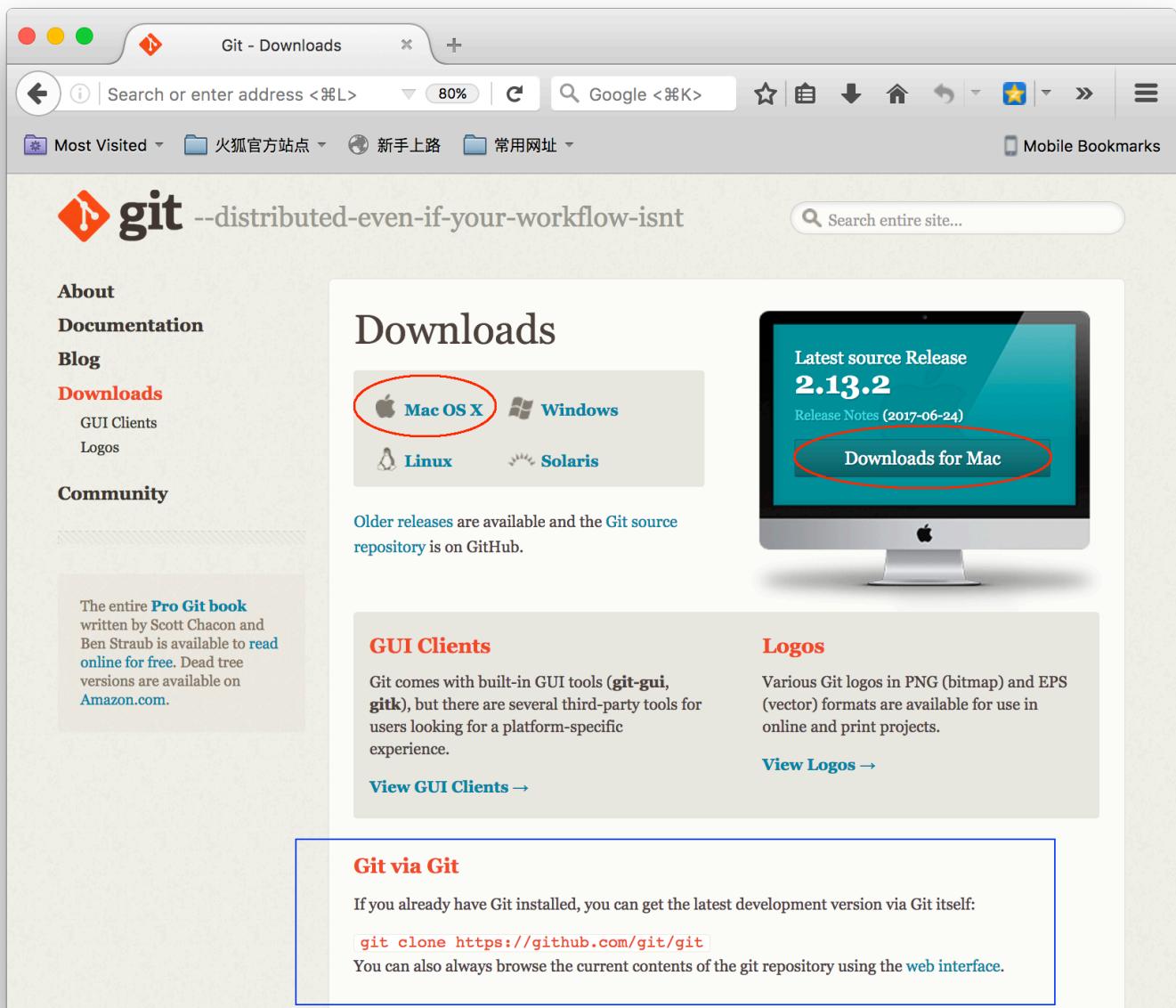
当然，从上面列出Git的一些使用方法中你或许也能看到 通过 git --version 命令可以查看当前安装的Git的版本，让我们来实践一下。



A screenshot of a Mac OS X Terminal window titled "Terminal — -bash — 68x5". The window shows the command "git --version" being run in a terminal session. The output is "git version 2.11.0 (Apple Git-81)". The command "git --version" is highlighted with a red oval.

```
Fei's Mac:~ fzuos$ git --version
git version 2.11.0 (Apple Git-81)
Fei's Mac:~ fzuos$
```

如果你发现你的系统上没有安装Git，那么你要到Git的网站 (<https://git-scm.com/downloads>) 下载并安装最新版本的Git，如下图所示。



不管怎样，（即使你发现你的Mac OS X上已经安装了Git）我们仍然建议你及时更新自己的Git。在上图所示的Git下载界面中也显示（最下方蓝色方框所标识的地方）如果你的电脑上已经安装了Git，那么你可以通过在Terminal中输入下面的指令来自动更新你电脑上的Git。

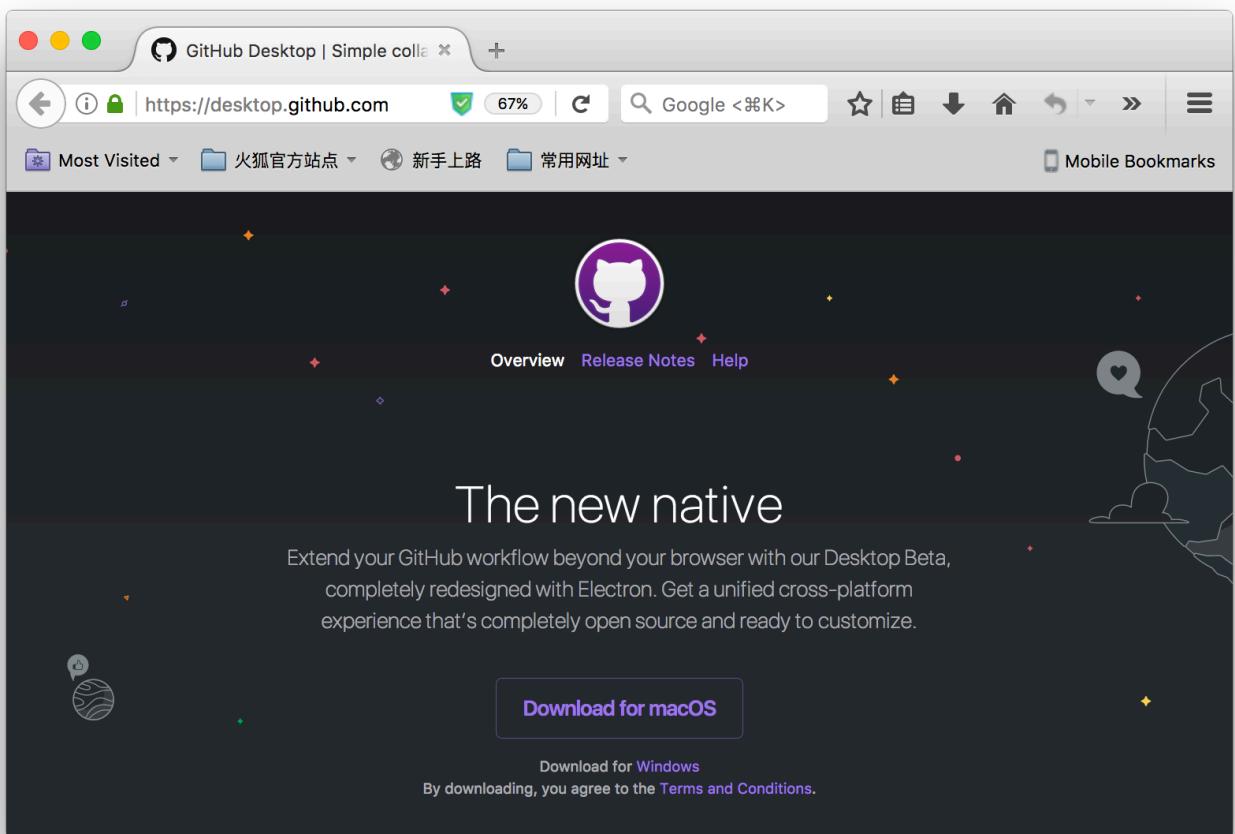
[plain] [view plain](#) [copy](#)

```
1. git clone https://github.com/git/git
```

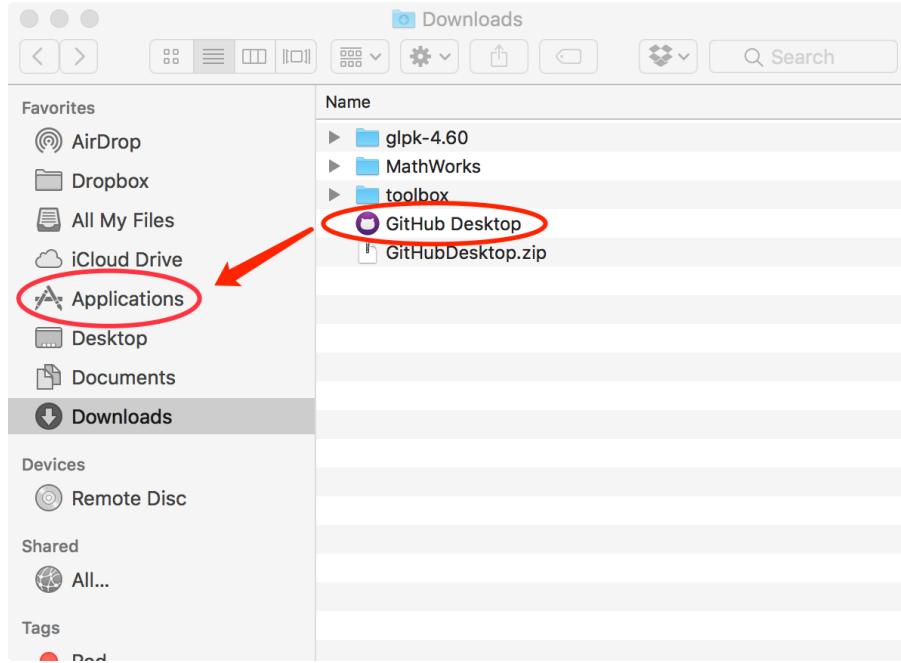
为了在电脑上可以正常使用Git和GitHub，这一步请务必完成。

### 3、安装GitHub Desktop

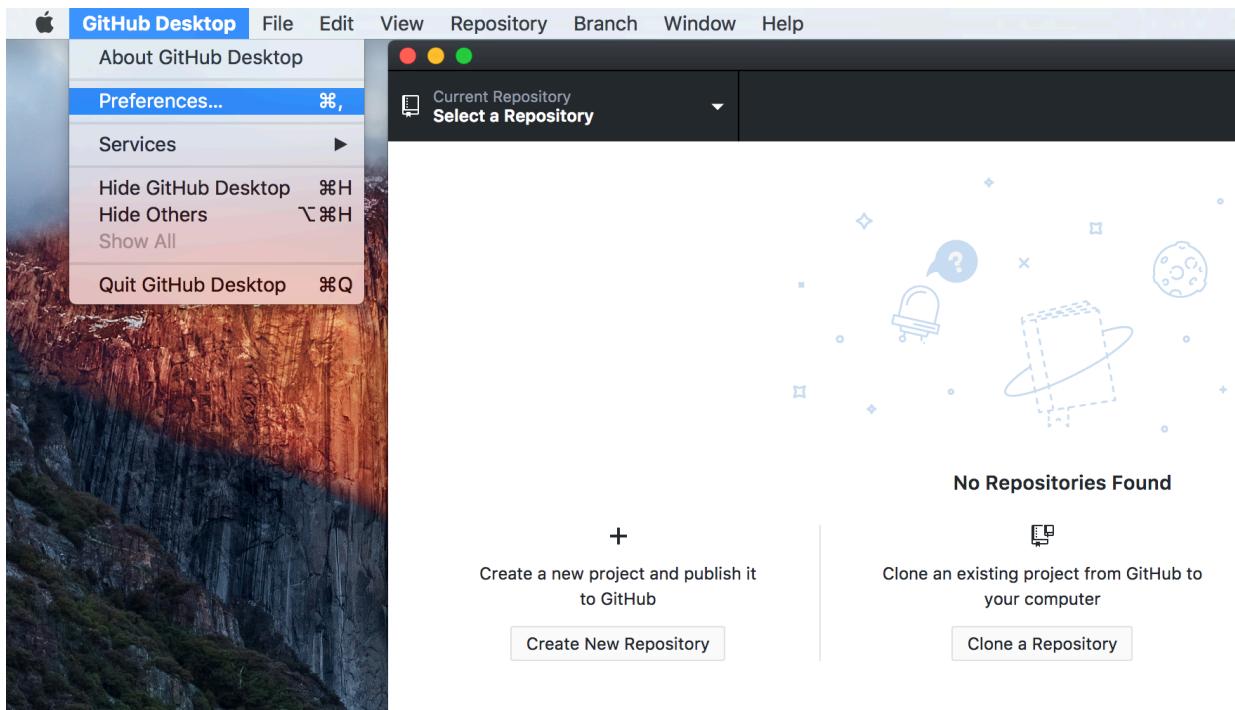
在Mac OS X上通过GitHub Desktop（即GitHub的桌面版或客户端）来操作GitHub是非常方便的。为此请登录到GitHub Desktop的网站（<https://desktop.github.com/>），然后选择【Download for MacOS】，如下图所示。



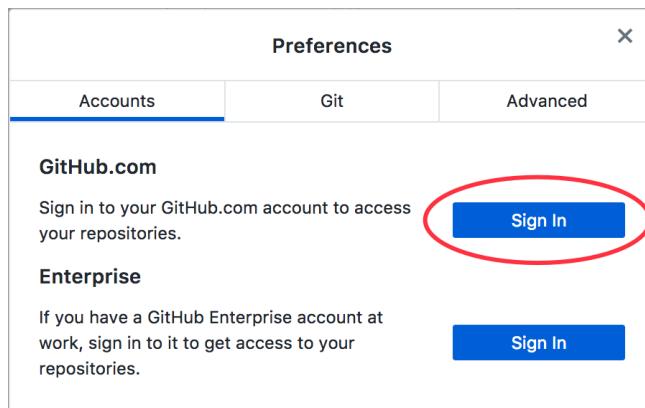
你下载下来的文件应该是一个zip的压缩包，解压后得到一个名为GitHub Desktop的程序文件，然后请把这个文件拖拽到Applications中，如下图所示。这样在Launchpad中就会出现GitHub Desktop的图标了。



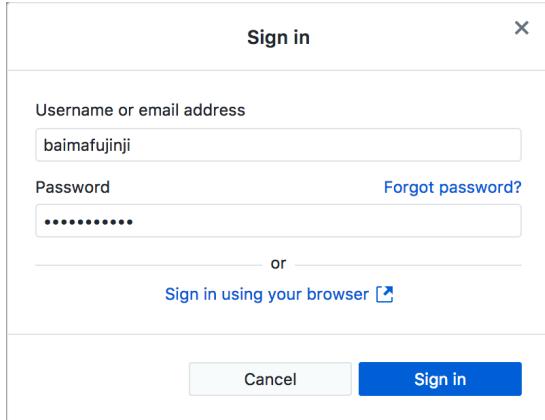
下面用你在GitHub网站上注册的用户名来配置GitHub Desktop。从Launchpad中启动GitHub Desktop，如下图所示，然后从屏幕左上方选择【GitHub Desktop】→【Preferences】。



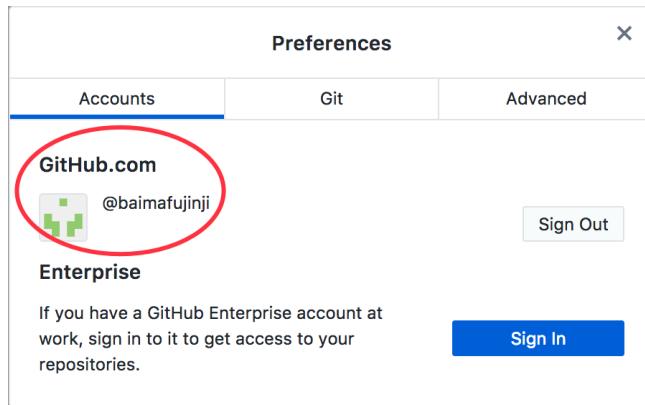
如下图所示，在弹出的对话框中选择【Sign In】。你会看到两个【Sing In】，一个是登录到"GitHub.com"，另外一个是登录到"Enterprise"。注意"Enterprise"是付费版的，对于普通个人用户而言，请登录到"GitHub.com"。



然后在登录界面中输入你的用户名和密码，如下图所示，并按【Sign In】按钮以完成操作。

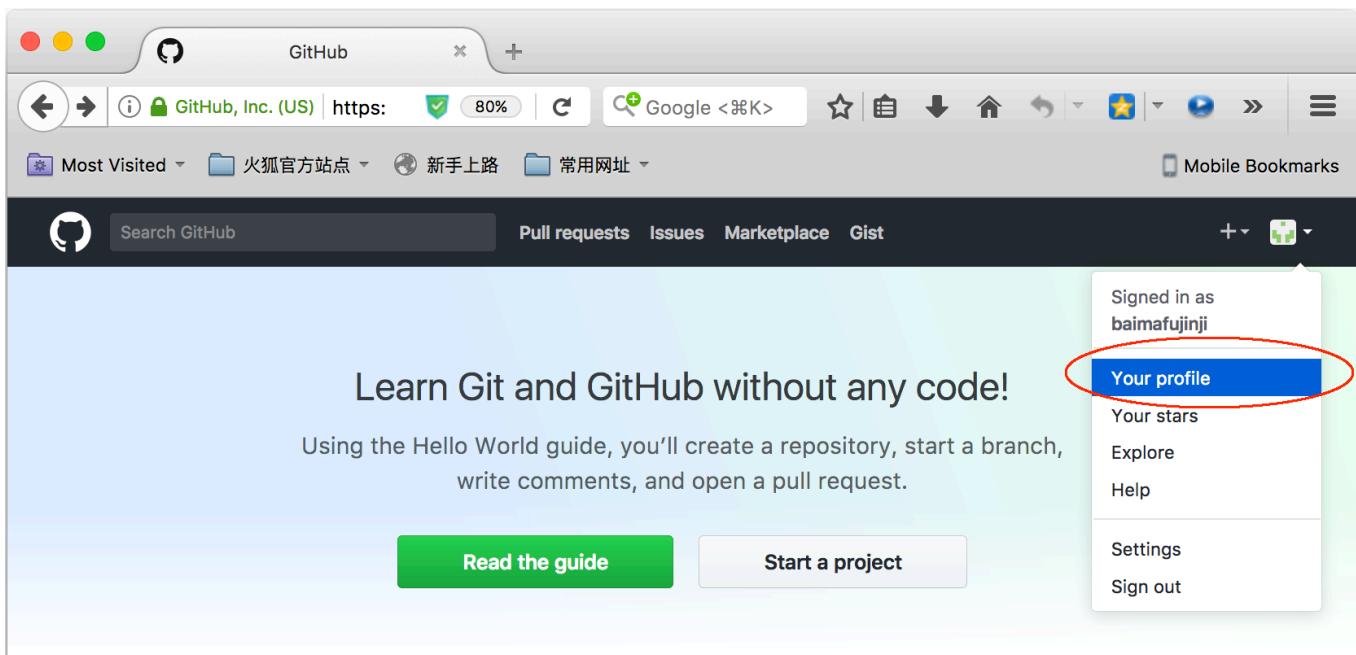


如果你登录成功（用户名和密码无误），Accounts界面里机会显示你当前登录的用户名，如下图所示。同时，在Git选项卡中，用户名和邮箱地址也会被更新。注意，尽管Git选项卡里已经更新了内容，我们仍然建议在这一步里你应该在Git选项卡界面上单击【Save】按钮以保存账号信息。

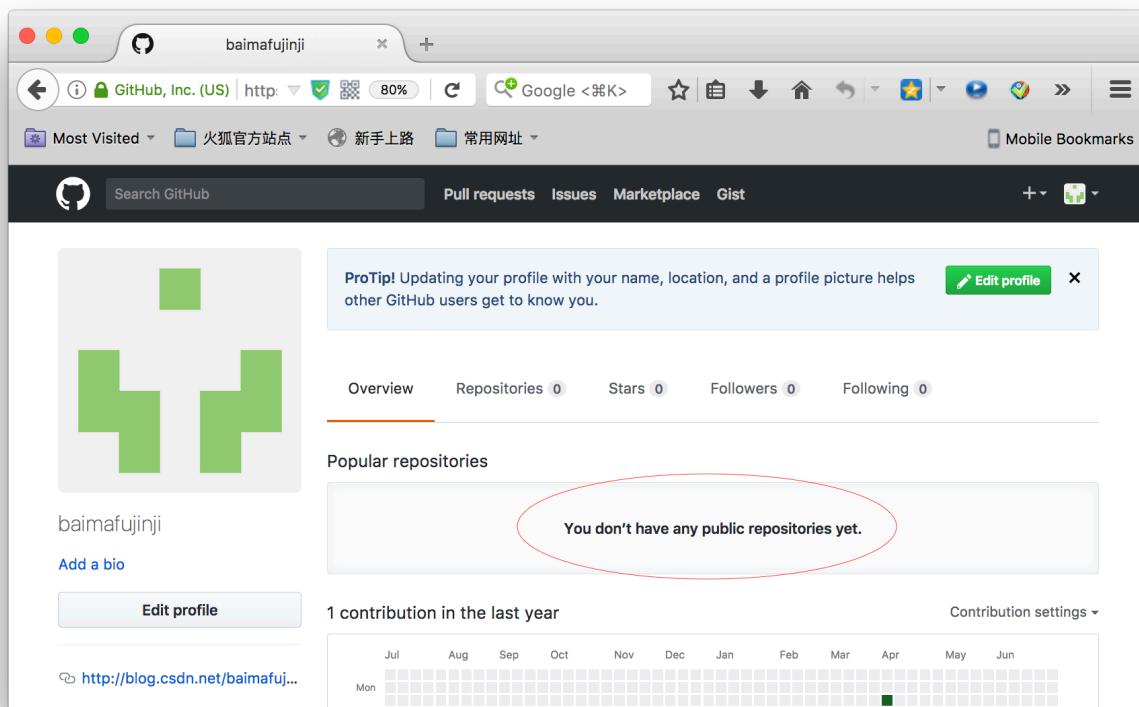


#### 4、使用GitHub Desktop来将项目托管到GitHub

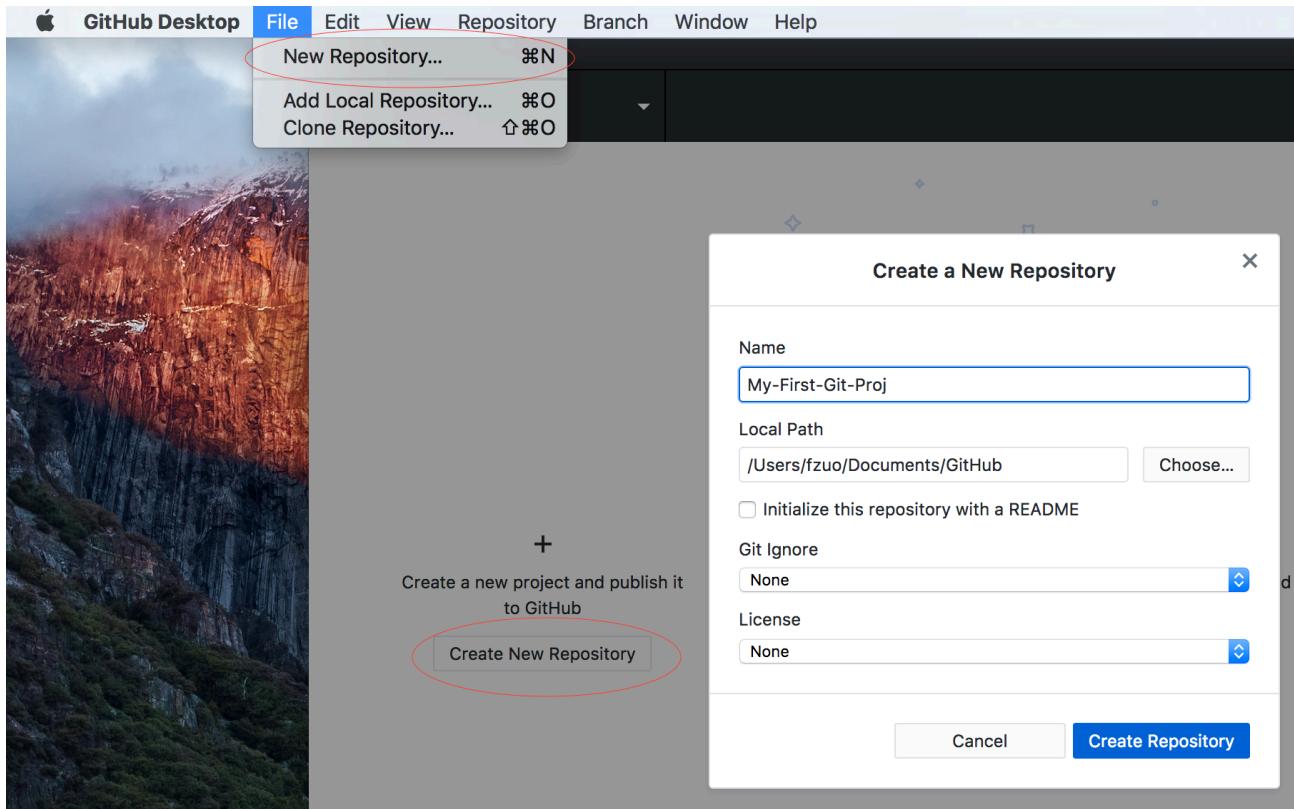
要利用GitHub来对你的开发项目进行版本管理，你就要创建【Repositories】，然后把你的项目代码即其他文件上传到相应的Repositories中。在你登录到GitHub后，单击右上角的图标，并从下拉菜单中选择【Your Profile】，如下图所示。



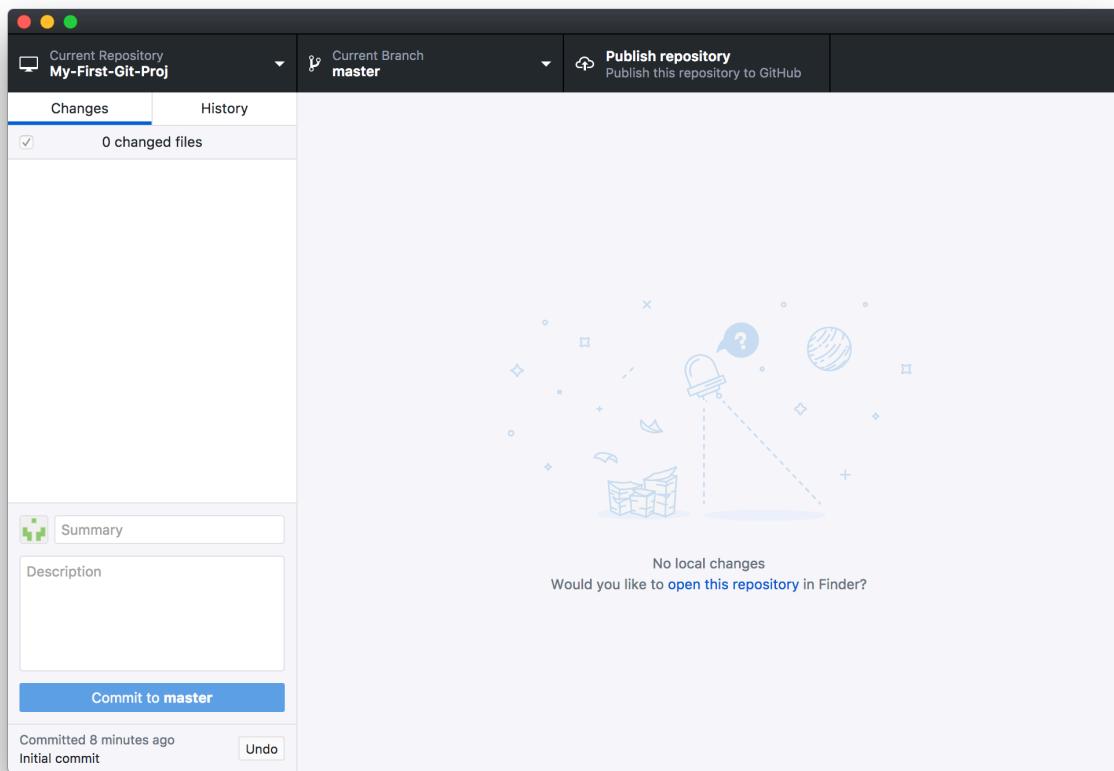
如下图所示，会发现你还没有创建任何Repositories。下面我们就通过GitHub Desktop来创建Repositories并上传项目文件至GitHub。



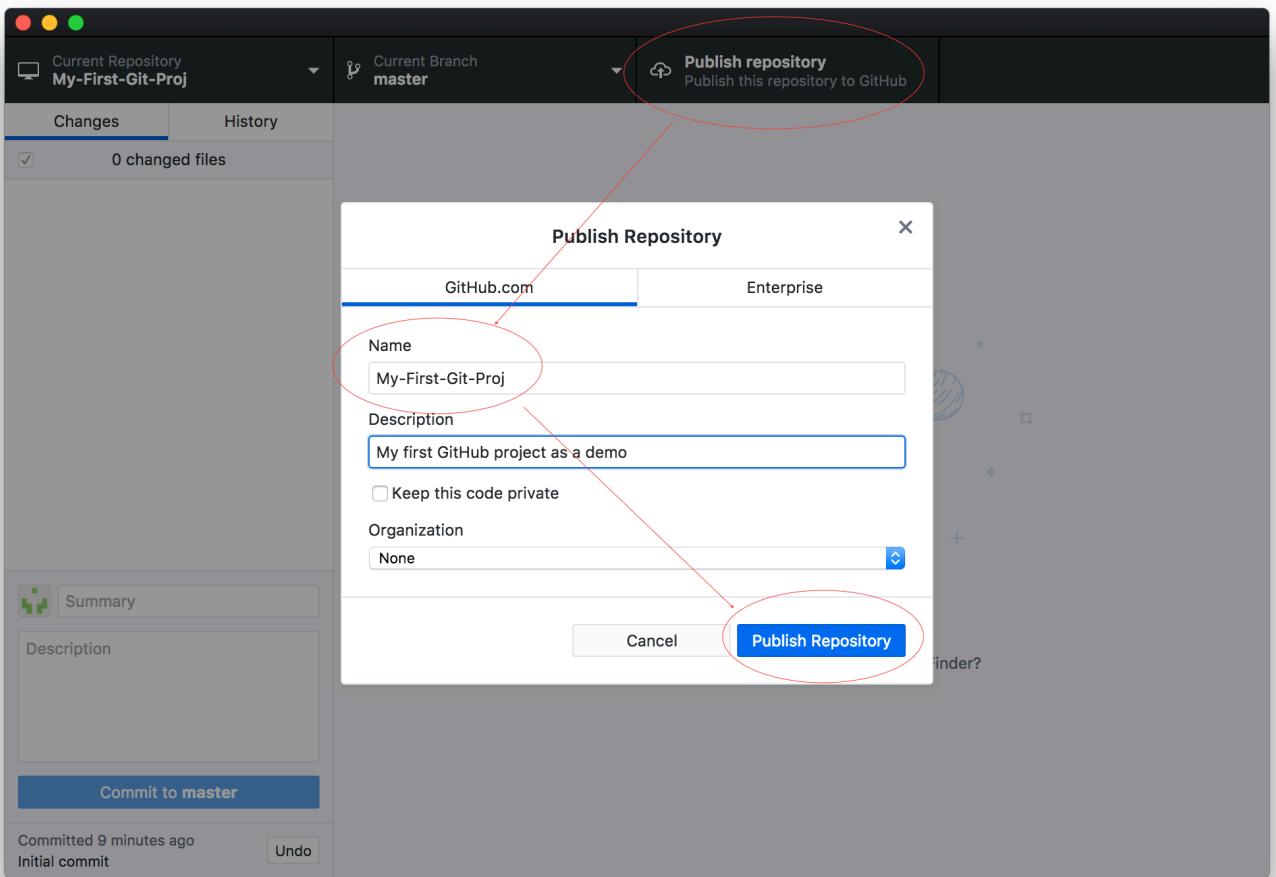
下面在本地创建一个新的Repository，如下图所示你可以从屏幕上方的【File】菜单中点选【New Repository】，也可在未创建任何Repository的GitHub Desktop操作界面上直接单击【Create New Repository】。接下来，在弹出的对话框中输入Repository的名字（系统会在本地创建一个该名字的文件夹）。单击【Create Repository】按钮完成操作。



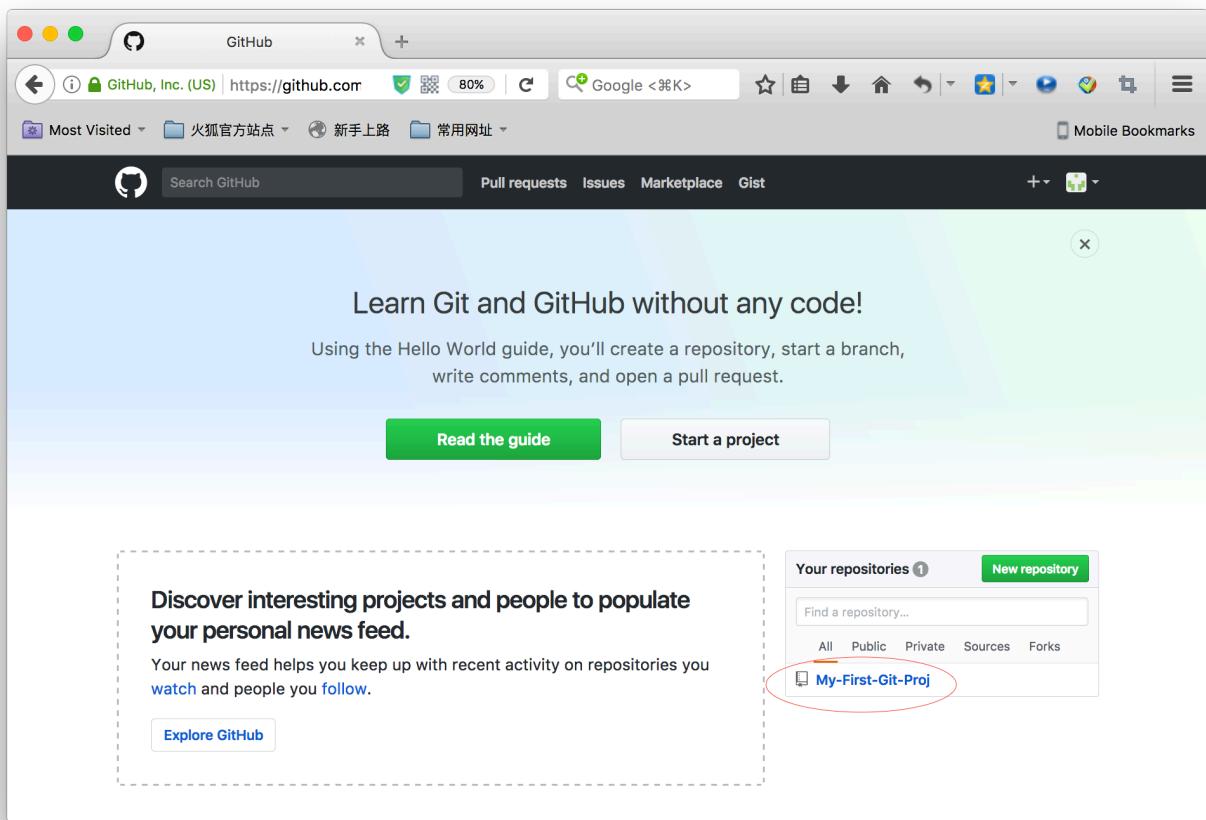
成功创建一个新的Repository之后，如下图左上角所示可以看到“Current Repository”就是刚刚创建的Repository，由于现在这个Repository是空的，所以里面并没有任何文件。此外，还应该注意到此时My-First-Git-Proj的图标是一个电脑形状，这是因为该Repository仅仅存在你的电脑上，还没有上传到GitHub网站上。



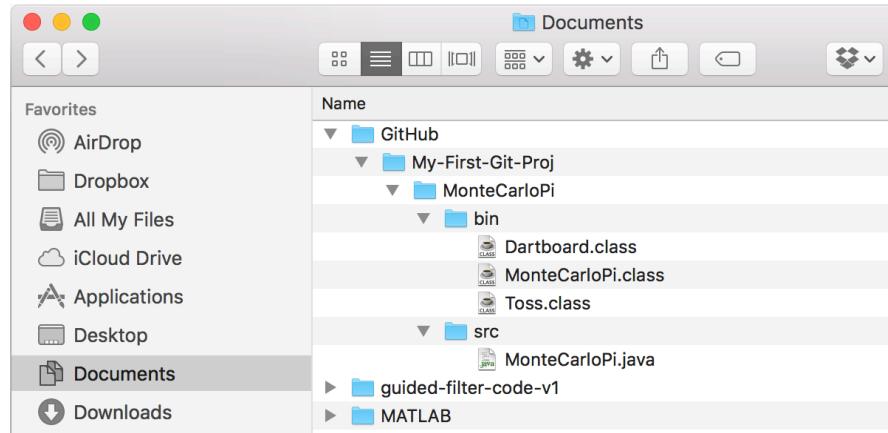
记下来就实现GitHub网站的同步更新。如下图所示，单击【Publish repository】，然后在弹出的对话框中填入必要的信息，并单击【Publish Repository】以完成操作。



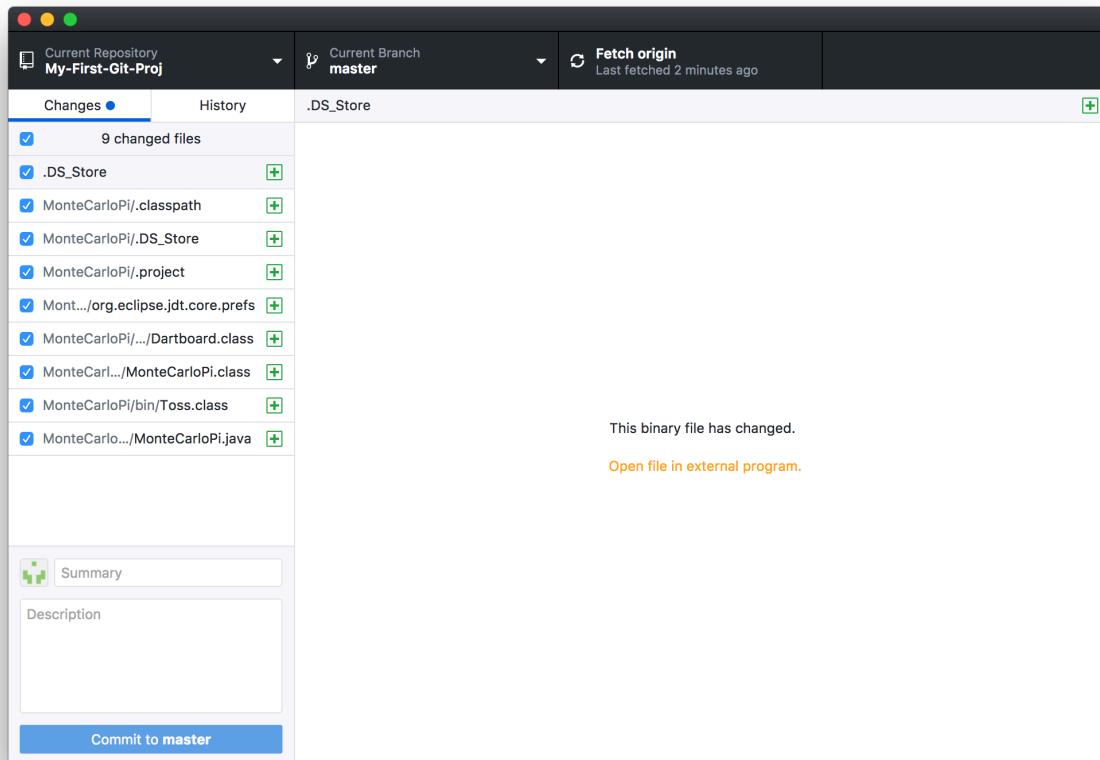
完成上述步骤后，Repository的图标就会发生改变，即不再是一个电脑形状的图标。此外，如果你登录到GitHub网站上，如下图所示，你也会发现My-First-Git-Proj也已经出现在你的Repositories中了。当然，如果你单击My-First-Git-Proj进入该Repository，也会发现这是一个空的项目。



下面我们来为刚刚建好的Repository文件夹加入一些文件（也就是你在开发项目时的工程更新内容）。然后再试着将这些内容同步更新到GitHub网站上。如下图所示，我们在My-First-Git-Proj文件夹里放入了一个Java的程序文件（其实你可以随便放入任何文件）。



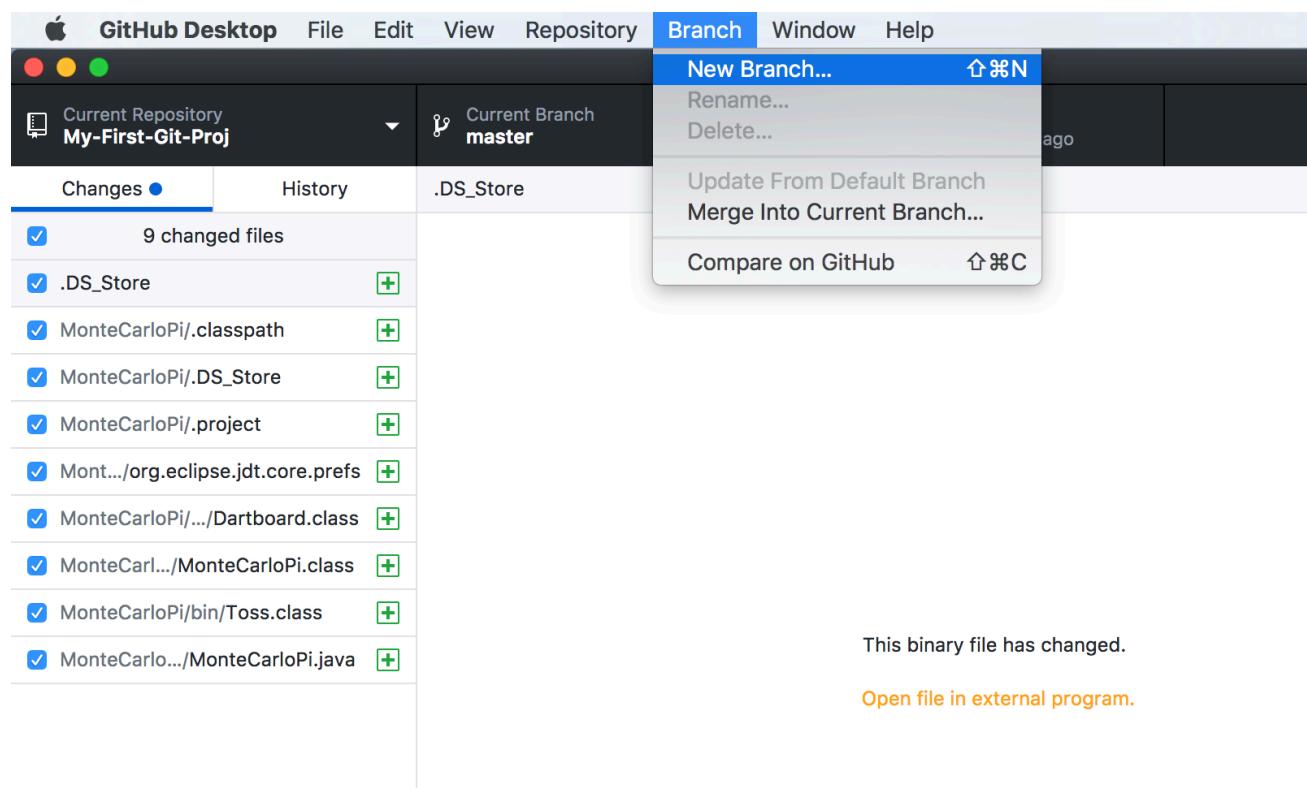
随机你就会发现GitHub Desktop界面中的My-First-Git-Proj Repository已经同步更新了最新加入的文件。如下图所示。



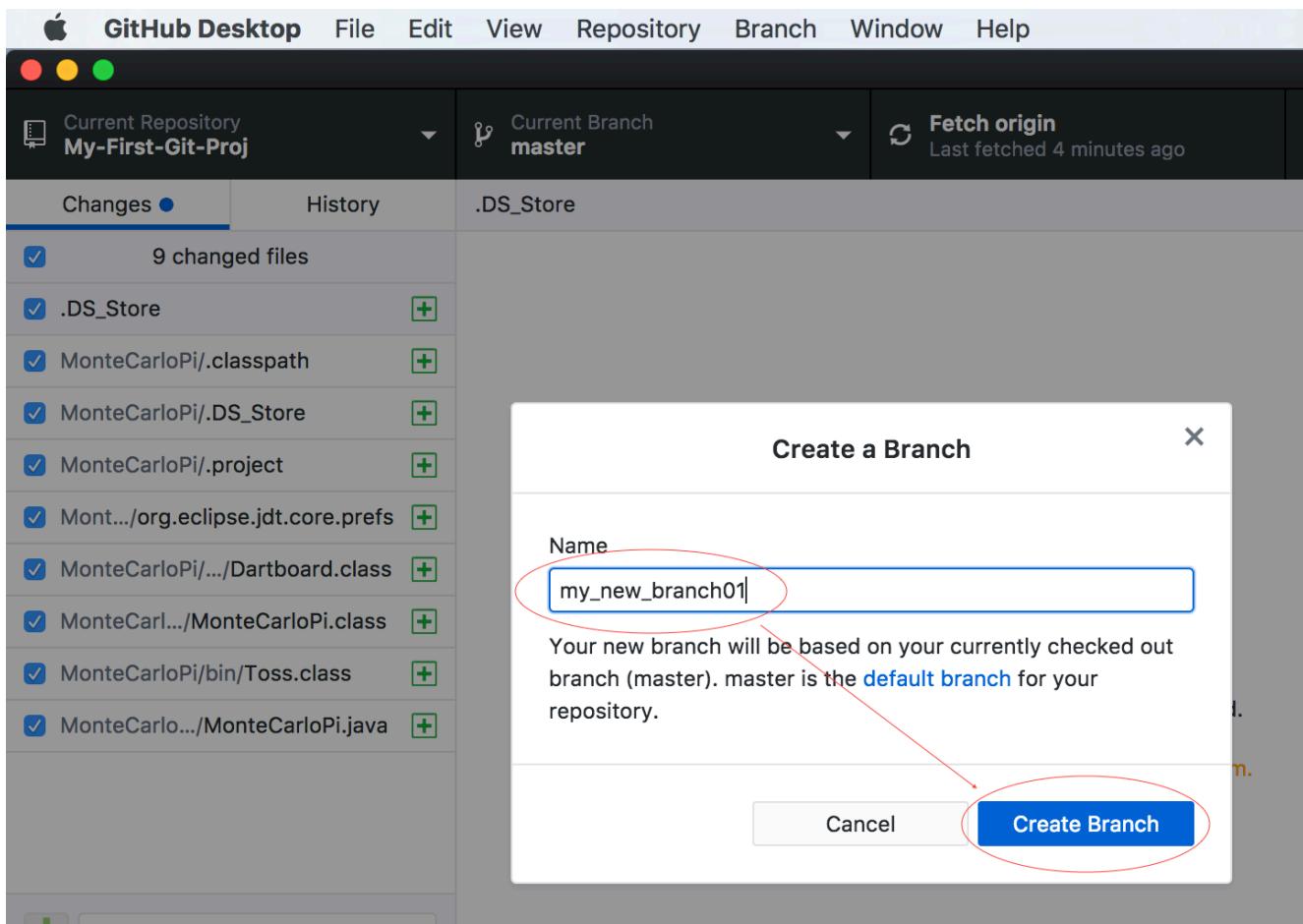
现在设法把项目更新同步上传到GitHub网站上。现在就涉及到一个概念Branch，这是用于版本控制的一种设置。默认情况下所有的项目内容都位于master Br

anch下面。但是有些时候，你对程序文件进行了修改之后，可能又想找回原来的版本。为此，就应该新建一个Branch，把新的更新放在新的Branch里面。如果要找回原来更改前的文件，只要到原来的Branch里面就会得到修改前的程序文件。下面来演示一下，在新的Branch里面同步更新程序文件的方法。

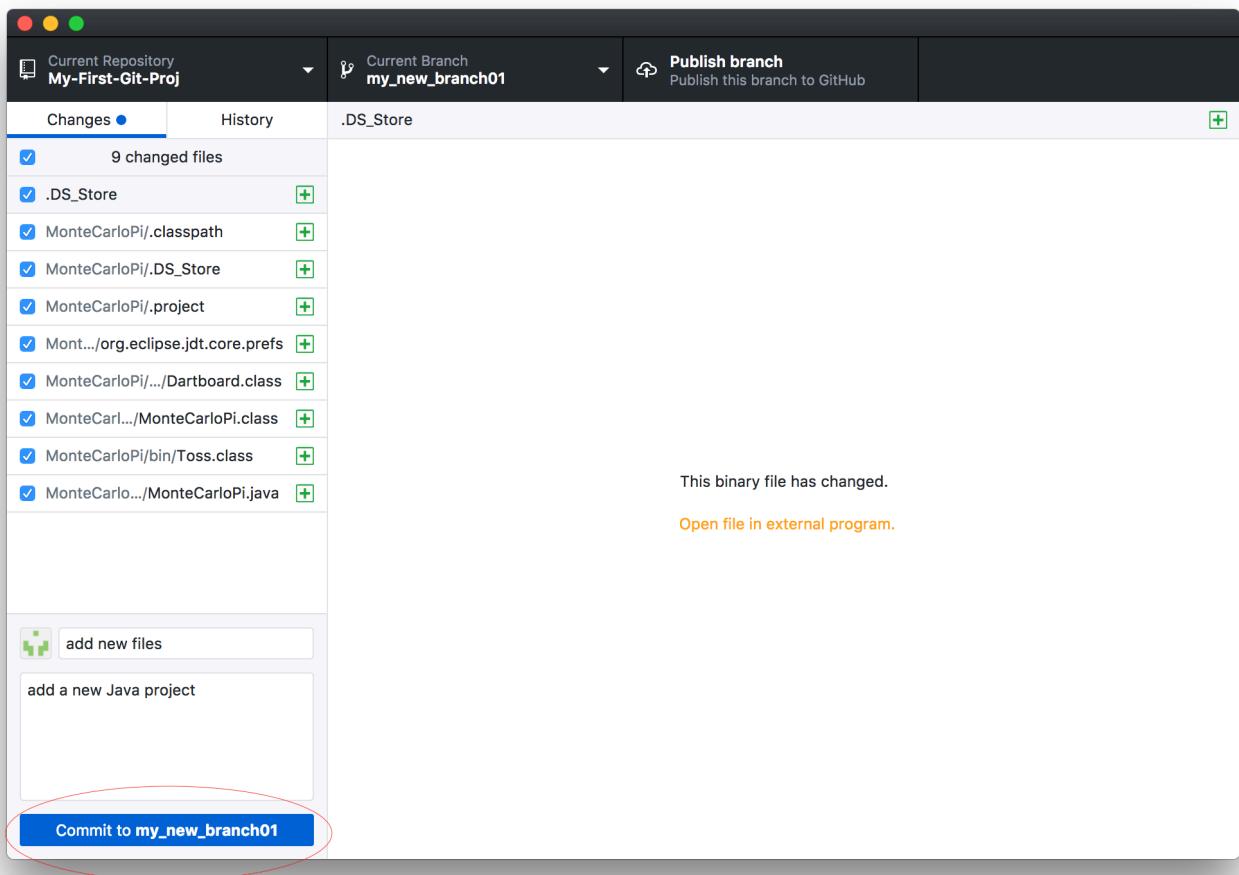
首先从屏幕上方的菜单栏中寻找【Branch】，然后从下拉菜单中选择【New Branch...】，如下图所示。



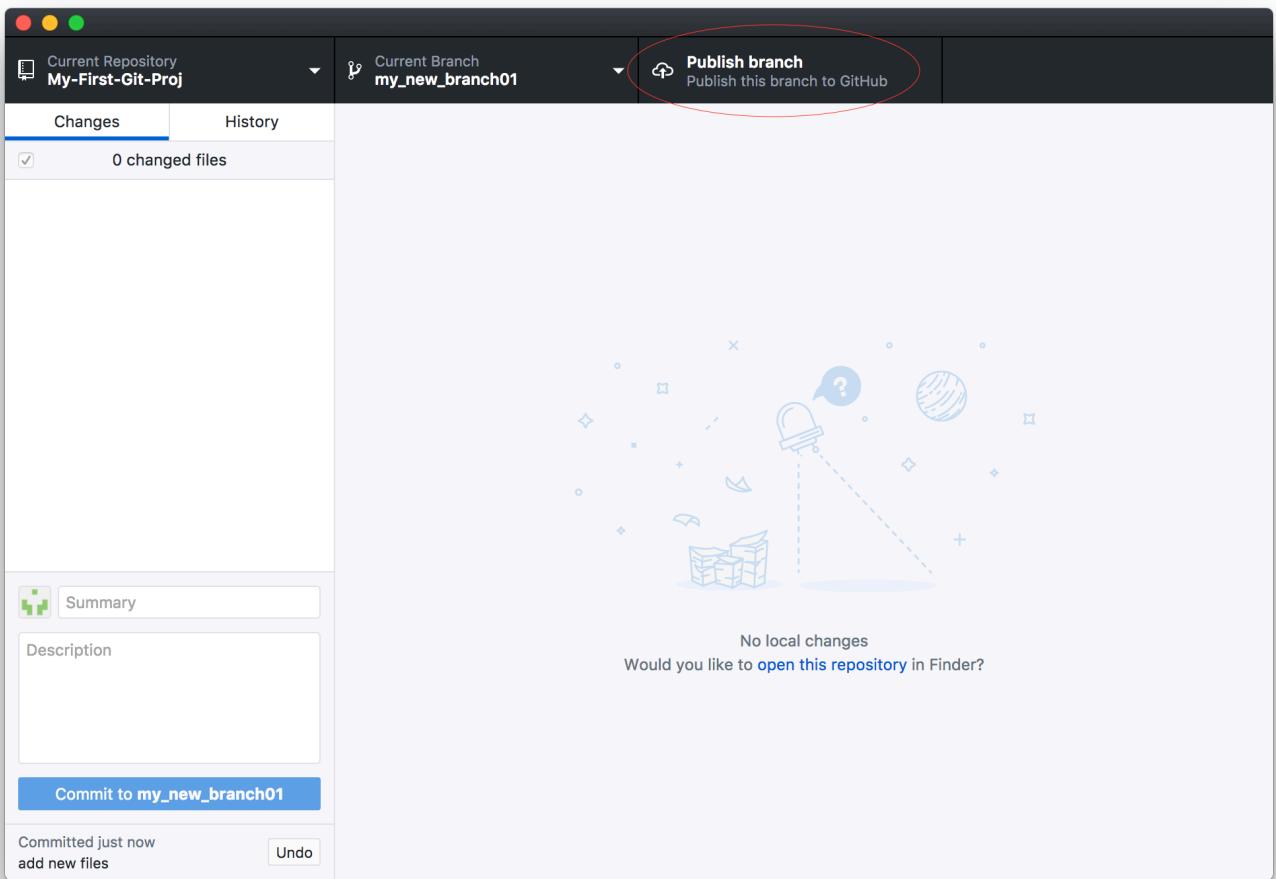
在弹出的对话框中，编辑新的branch的名字，并单击【Create Branch】按钮，完成操作，如下图所示。



创建Branch成功后，编辑Summary和Description（也就是描述此次项目文件更新的一些内容），然后单击【Commit to ...】按钮（意思就是把最新的更新提交到新建的Branch中），如下图所。



然后点击【Publish branch】，GitHub Desktop便会把更新的内容同步上传到GitHub网站上，如下图所示。



此时，再登录你的GitHub网站，便会发现，在Repository中已经新建了一个Branch，而且刚刚更新的项目文件也已经同步上传到GitHub网站上了。如下图所示。

Branches · baimafujinji/My-First-Git-Pro

GitHub, Inc. (US) | https | 80% | Google <36K>

Most Visited | 火狐官方站点 | 新手上路 | 常用网址 | Mobile Bookmarks

This repository | Search | Pull requests | Issues | Marketplace | Gist

baimafujinji / My-First-Git-Pro

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

Overview | Yours | Active | Stale | All branches | Search branches...

Default branch

master Updated an hour ago by baimafujinji Default Change default branch

Your branches

my\_new\_branch01 Updated 2 minutes ago by baimafujinji 0|1 New pull request ⚡

Active branches

my\_new\_branch01 Updated 2 minutes ago by baimafujinji 0|1 New pull request ⚡

© 2017 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About

The screenshot shows a GitHub repository page for 'baimafujinji/My-First-Git-Proj'. The repository has 2 commits, 2 branches, 0 releases, and 1 contributor. A commit from 'MonteCarloPi' is highlighted with a red oval. The commit details are as follows:

File	Message	Time
MonteCarloPi	add new files	2 minutes ago
.DS_Store	add new files	2 minutes ago
.gitattributes	Initial commit	an hour ago

At the bottom, there is a note to 'Add a README'.