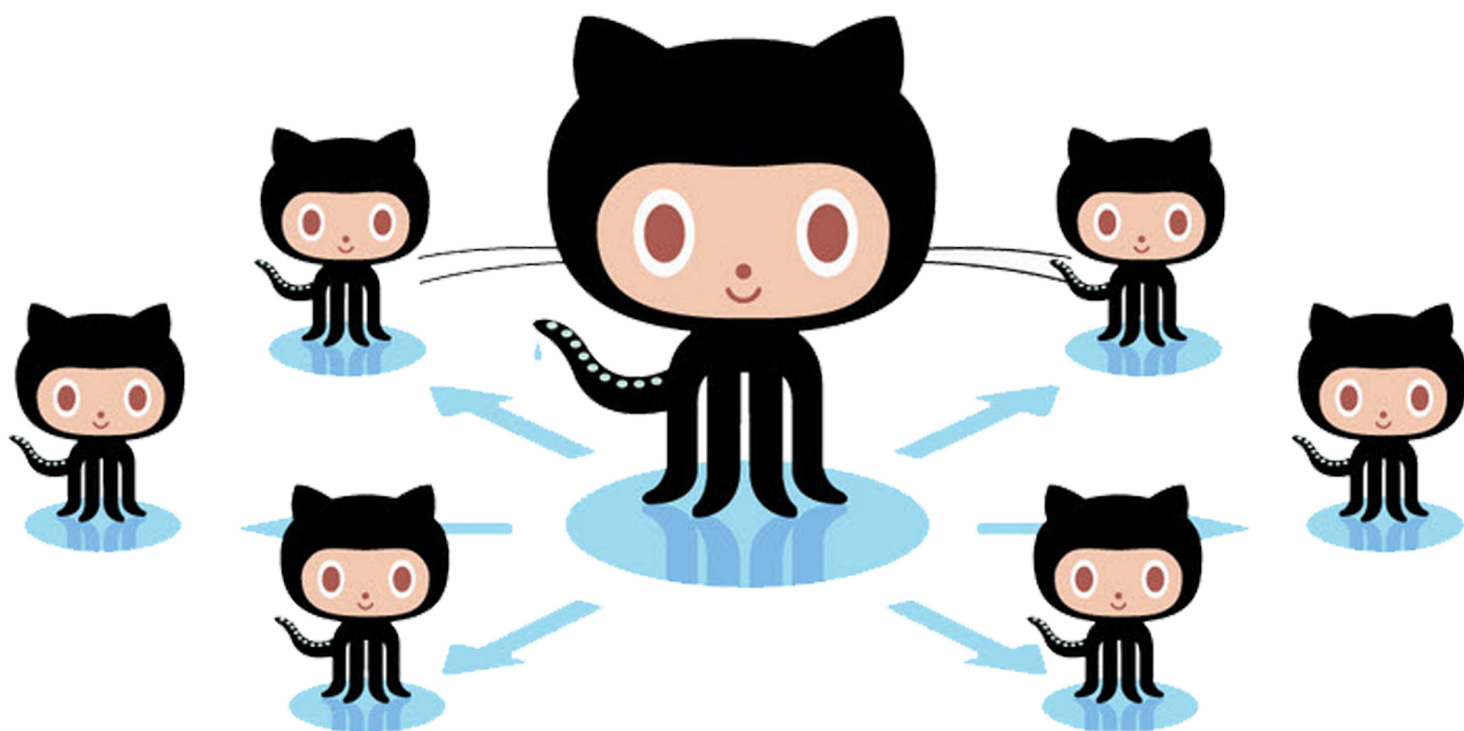


基于 Git 的开源及私有软件项目托管平台



GitHub

使用手册 – 基础篇

极客学院出版

前言

缘起

Git已经成为程序员必备技能之一，而GitHub做为流行的Git仓库托管平台，其不仅提供Git仓库托管，还是一个非常棒的技术人员社交平台，可以通过开源的项目进行协作、交流，是现在优秀的工程师必须娴熟运用的。

本套教程从 GitHub 的历史入手，介绍 Git 安装、创建仓库、Fork、社会化、命令行开发，到最后的图形化工具的使用。

学习完本教程，将不仅掌握 GitHub 命令行使用方法，也会学会图形化使用方法。

目录

- GitHub 初识
 - [GitHub 简介 \(#86097bc65e24ba33d1694edc3c96c8e7\)](#)
 - [GitHub 优势 \(#23a92d0c09c2ec23fbb2097dfe77587d\)](#)
 - [GitHub 注册 \(#d101bc2f052de81725b140ccdc303e89\)](#)
- 安装 Git
 - [安装 Git \(#d9787e5db93ce630c69e495d3e9a1d95\)](#)
 - [通过 Git 验证 GitHub \(#c9c3b8ede99b3ceb5289580efdfaa42d\)](#)
- 创建仓库
 - [在 GitHub 上创建新仓库 \(#209b25d01b1e6dd04b1139e1d4977e1e\)](#)
 - [提交你的第一个修改 \(#dca9ac73dc7f36b1c81e020c6ee1aed9\)](#)
- Fork 一个仓库
 - [Fork 一个示例仓库 \(#e657eade6e91ffde53b5aa94221d242c\)](#)
 - [同步你的 Fork 仓库 \(#64cc6be0c703395163fb2a6bcc136ea3\)](#)
 - [检索其他仓库来 Fork \(#aafa4d21e588591041457bd7161b6744\)](#)

- 社会化
 - [Follow 一个人 \(#db9c0d69a288bd78baa9b65ba1291408\)](#)
 - [Watch 一个项目 \(#718ce9dd4f61aab8f4b52e9758269df6\)](#)
 - [其他你可以做的事 \(#d39458c86ae4affd3727f63dd93ccb85\)](#)
- 图形化工具
 - [GitHub for Windows \(#a4ccb83653d20a23ebdb07b04c05aa3e\)](#)
 - [GitHub for Mac \(#648d95ee60abc7e48c2f8434422eb724\)](#)

参与其中

如果你觉得这篇教程对你有帮助，欢迎通过如下方式参与其中

- 纠错：对教程中有错误的或者欠妥当的地方进行纠错，我们会第一时间完善和修订；
- 传播：看到好的文章可以顺手分享给更多需要的人，让本套教程能服务更多人；
- 贡献：如果你想加入编撰者行列，请加入QQ群（ 288936172 ），和我们一起创建更有价值的wiki~

目录

前言	1
第 1 章 GitHub 初识	5
GitHub 简介	6
GitHub 优势	7
GitHub 注册	8
第 2 章 安装 Git	12
安装 Git	13
通过 Git 验证 GitHub	14
通过 HTTPS 建立连接（推荐）	15
通过 SSH 建立连接	16
第 3 章 创建仓库	17
在 GitHub 上创建一个新仓库	18
提交你的第一个更改	21
第 4 章 Fork 一个仓库	25
Fork 一个示例仓库	26
同步你的 Fork 仓库	27
第一步：安装 Git	28
第二步：为你 fork 的仓库创建一个本地克隆	29
第三步：通过配置 Git 来同步你 fork 的原始 Spoon-Knife 仓库	30
检索其他仓库来 Fork	32
第 5 章 社会化	33
Follow 一个人	34
Watch 一个项目	35

	其他你可以做的事	36
	关注	37
	发现 / 交流	38
第 6 章	图形化工具	40
	GitHub for Windows	41
	Fork	43
	提交到本地	44
	同步远程仓库	45
	GitHub for Mac	46
	Fork	43
	提交到本地	44
	同步远程仓库	45



GitHub 初识



GitHub 简介

Git 是一个优秀的分布版本控制系统。版本控制系统可以保留一个文件集合的历史记录，并能回滚文件集合到另一个状态（历史记录状态）。另一个状态可以是不同的文件，也可以是不同的文件内容。在一个分布版本控制系统中，每个人都有一份完整的源代码（包括源代码所有的历史记录信息），而且可以对这个本地的数据进行操作。分布版本控制系统不需要一个集中式的代码仓库。

GitHub 是一个面向开源及私有软件项目的托管平台，因为只支持 Git 作为唯一的版本库格式进行托管，故名 Git Hub。

GitHub 于 2008 年 4 月 10 日正式上线，除了 Git 代码仓库托管及基本的 Web 管理界面以外，还提供了订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。目前，其注册用户已经超过百万，托管版本数量也是非常之多，其中不乏知名开源项目 Ruby on Rails、jQuery 等。

GitHub 优势

GitHub 之所以如此受欢迎，它的优势是不容忽视的：

1. GitHub 只支持 Git 格式的版本库托管，而不像其他开源项目托管平台还对 CVS、SVN、Hg 等格式的版本库进行托管。GitHub 的哲学很简单，既然 Git 是最好的版本控制系统之一（对于很多喜欢 Git 和 GitHub 的人没有之一），没有必要为兼顾其他版本控制系统而牺牲 Git 某些独有特性。因此没有支持其他版本控制系统的历史负担，是 GitHub 成功的要素之一。
2. GitHub 对 Git 版本库提供了完整的协议支持，支持 HTTP 智能协议、Git-daemon、SSH 协议。
3. GitHub 提供在线编辑文件的功能，不熟悉 Git 的用户也可以直接通过浏览器修改版本库里的文件。
4. 将社交网络引入项目托管平台是 GitHub 的创举。用户可以关注项目、关注其他用户进而了解项目和开发者动态。
5. 项目的 Fork 和 Pull Request 构成 GitHub 最独具一格的工作模式。对提交代码的逐行评注及 Pull Request 构成 GitHub 特色的代码审核。
6. GitHub 通过私有版本库托管、面向企业的版本库托管和项目管理平台、人才招聘等付费服务获得了商业上的成功，这种成功使得 GitHub 不必以页面中嵌入广告的方式维持运营，最大的受益者还是用户。
7. GitHub 网站采用 Ruby on Rails 架构，在 Web 设计中运用了大量的 JavaScript、AJAX、HTML5 等技术，支持对使用 Markdown 等标记语言的内容进行渲染和显示等。关注细节使得 GitHub 成为了项目托管领域的后起之秀。

GitHub 注册

要想使用 GitHub 第一步当然是注册 GitHub 账号：

- 1、首先打开 <https://github.com/pricing> 进行注册。
- 2、在打开的页面中点击「Sign up now」注册，如图 1.1 所示。

Plans and pricing

GitHub is free to use for public projects. Collaborate on private repositories with any of our paid plans.

Sign up now



Personal plans

For individuals looking to share their own projects and collaborate with others.

图片 1.1 Sign up now


图 1.1


- 3、在接下来的页面中创建用户名，填写 email 和设定密码，点击「Create an account」按钮创建账户，如图 1.2。


GitHub Explore Features Enterprise Blog Sign in

Join GitHub

The best way to design, build, and ship software.

 **Step 1:**
Set up a personal account

 **Step 2:**
Choose your plan

 **Step 3:**
Go to your dashboard

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

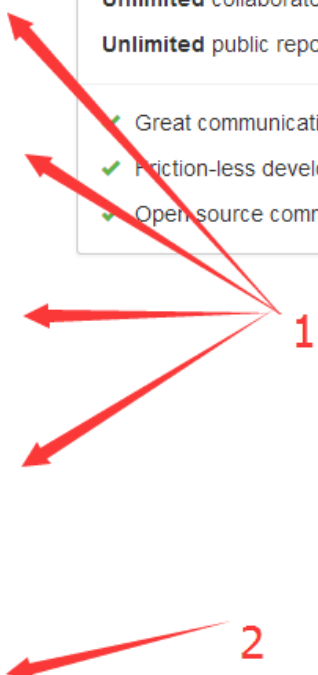
Confirm your password

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

You'll love GitHub

- Unlimited** collaborators
- Unlimited** public repositories
- ✓ Great communication
- ✓ Friction-less development
- ✓ Open source community



1

2

图片 1.2 Sign up now

图 1.2

4、选择账户类型，这里我们默认选择“Free”类型，点击「Finish sign up」按钮完成注册，如图 1.3。

Welcome to GitHub

You've taken your first step into a larger world, @.

Completed
Set up a personal account

Step 2:
Choose your plan

Step 3:
Go to your dashboard

Choose your personal plan

Plan	Cost (view in CNY)	Private repos	
Large	\$50/month	50	Choose
Medium	\$22/month	20	Choose
Small	\$12/month	10	Choose
Micro	\$7/month	5	Choose
Free	\$0/month	0	Chosen

Each plan includes:

Unlimited collaborators
Unlimited public repositories

- ✓ Free setup
- ✓ SSL Protection
- ✓ Email support
- ✓ Wikis, Issues, Pages, & more

Charges to your account will be made in **US Dollars**.
Converted prices are provided as a convenience and are only an *estimate* based on *current* exchange rates. Local prices will change as the exchange rate fluctuates.
Don't worry, you can cancel or upgrade at any time.

- ☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations.](#)

Finish sign up

图片 1.3 Sign up now

图 1.3

注：不同类型的选择根据我们的需要，如果存放开源项目，则免费托管；存放私有库，则需要付费。费用如下表：

私有库空间	费用	私有库数量
大	\$50/月	50
中	\$20/月	20

私有库空间	费用	私有库数量
小	\$10/月	10
微小	\$5/月	5
免费	\$0/月	0



安装 Git



安装Git

- 1、[下载并安装 Git 最新版本 \(http://git-scm.com/downloads\)](http://git-scm.com/downloads) 。
- 2、安装完成后，打开 Terminal 命令（针对苹果系统用户）或者命令提示行（针对 Windows 和 Linux 用户）。
- 3、告诉 Git 你的姓名，以便你的提交能被正确地标记。在 `$` 后输入下面的内容：

```
$ git config --global user.name "YOUR NAME"
```

- 4、告诉 Git 邮箱地址，以便与你的 Git 提交进行关联。你指定的邮箱要和[邮箱设置 \(https://help.github.com/articles/adding-an-email-address-to-your-github-account/\)](https://help.github.com/articles/adding-an-email-address-to-your-github-account/) 里的是同一个。如何保持你的邮箱地址隐藏，请参考：[保持你的邮箱地址私有 \(https://help.github.com/articles/keeping-your-email-address-private\)](https://help.github.com/articles/keeping-your-email-address-private) 。

```
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

通过 Git 验证 GitHub

当你通过 Git 连接到一个 GitHub 仓库后，我们需要验证 GitHub，这里有两种验证方法：

- 通过 HTTPS 验证
- 通过 SSH 进行验证

通过 HTTPS 建立连接（推荐）

如果选择 HTTPS 方式 (<https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-https-recommended>)，我们可以用一个证书小帮手把 GitHub 密码缓存在 Git (<https://help.github.com/articles/caching-your-github-password-in-git>)。

通过 SSH 建立连接

如果选择 SSH 方式 (<https://help.github.com/articles/which-remote-url-should-i-use#cloning-with-ssh>)，我们需要在电脑中生成 SSH keys (<https://help.github.com/articles/generating-ssh-keys>)，用来从 GitHub 中 push 或 pull。

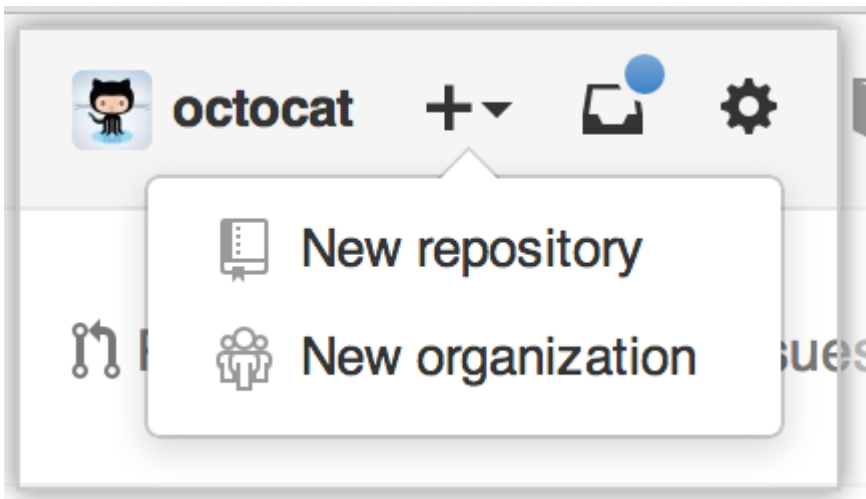


创建仓库



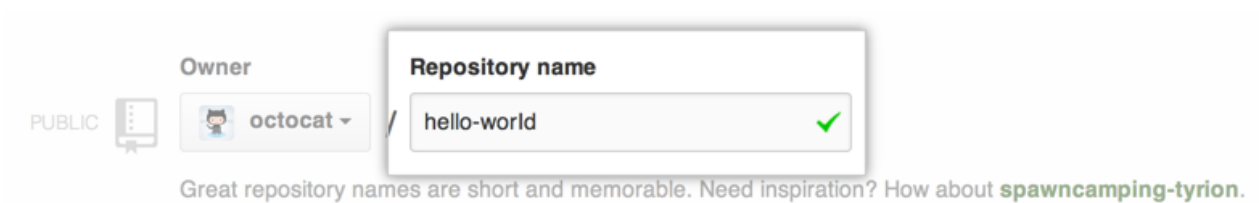
在 GitHub 上创建一个新仓库

1、在任意的页面右上角点击 +，然后点击新建仓库 New repository。



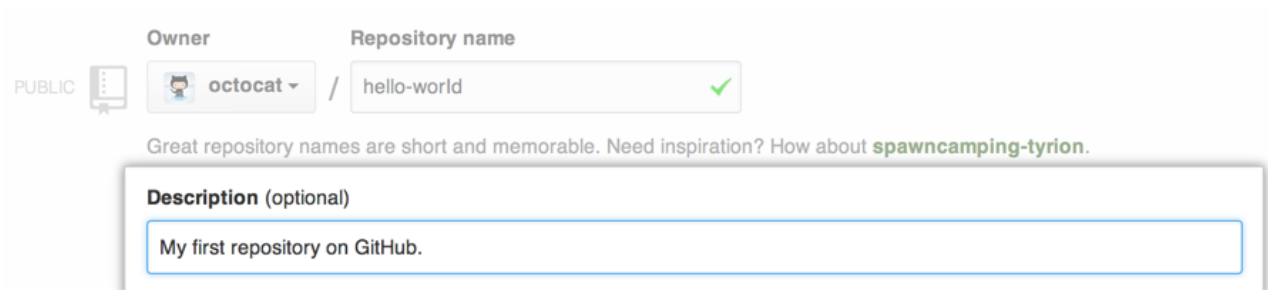
图片 3.1 新建仓库

2、为你的仓库创建一个简短便于记忆的名字。例如 “hello-world”。



图片 3.2 创建一个简短便于记忆的名字

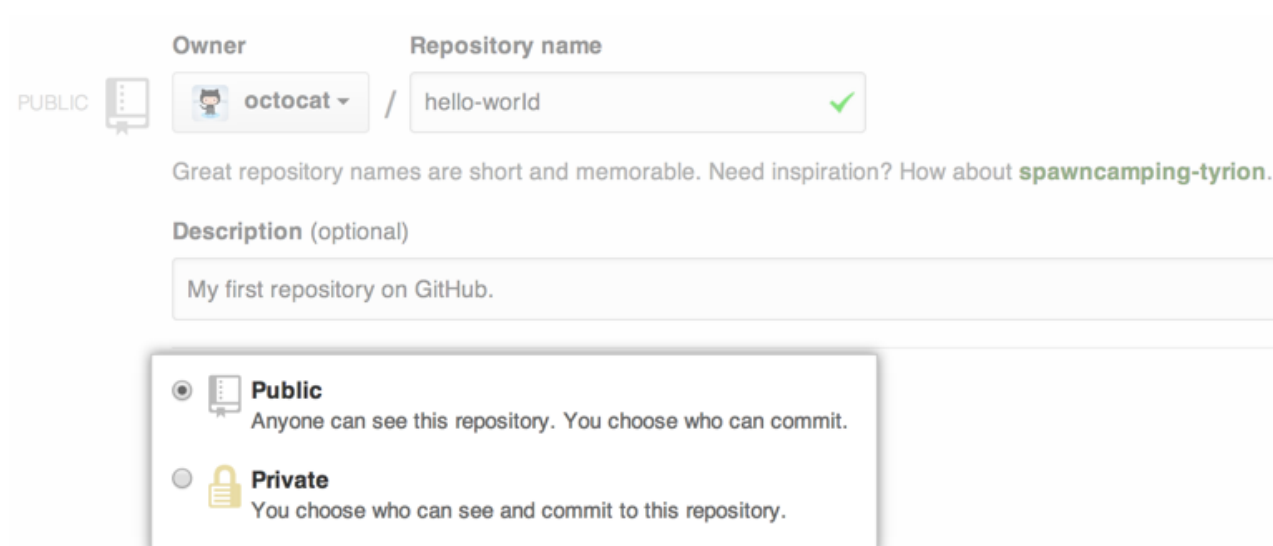
3、为你的仓库添加一个描述（非必须的）。例如 “My first repository on GitHub”。



图片 3.3 为你的仓库添加一个描述

4、选择你的仓库类型为公有或者私有：

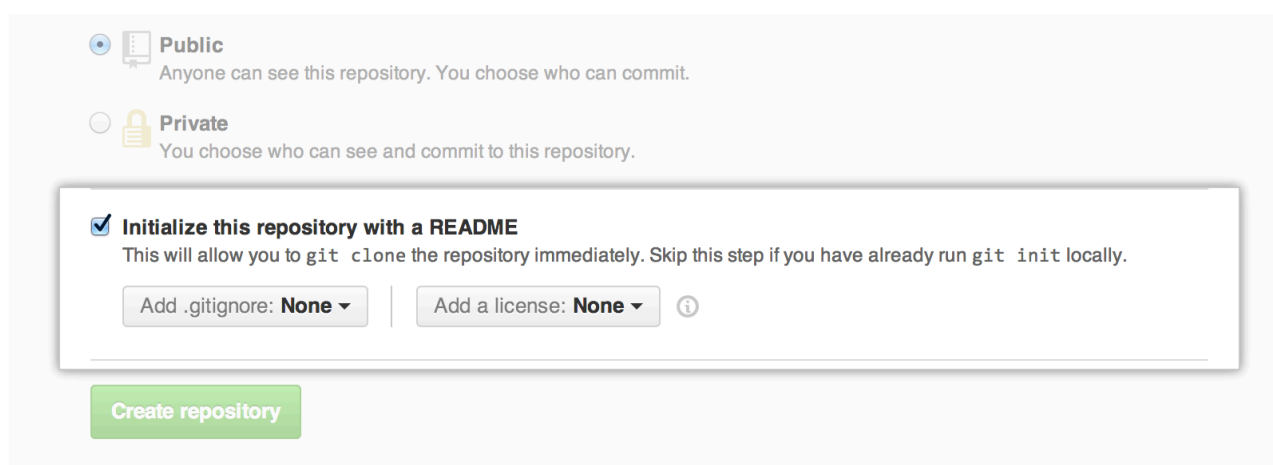
- **Public**: 公有仓库对于一个刚入门的新手来说是一个不错的选择。这些仓库在 GitHub 上对于每个人是可见，你可以从协作型社区中受益。
- **Private**: 私有仓库需要更多地步骤。它们只对于你来说是可用的，这个仓库的所有者属于你和你所指定要分享的合作者。私有仓库仅仅对付费账户可用。更多的信息请参照 "[What plan should I choose? \(https://help.github.com/articles/what-plan-should-i-choose\)](https://help.github.com/articles/what-plan-should-i-choose)"。



The screenshot shows the GitHub repository creation interface. At the top, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'octocat'. The 'Repository name' field contains 'hello-world' and has a green checkmark. Below these fields, there is a tip: 'Great repository names are short and memorable. Need inspiration? How about **spawncamping-tyrion**.' Underneath is a 'Description (optional)' text area with the placeholder text 'My first repository on GitHub.' At the bottom, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.'

图片 3.4 选择你的仓库类型为公有或者私有

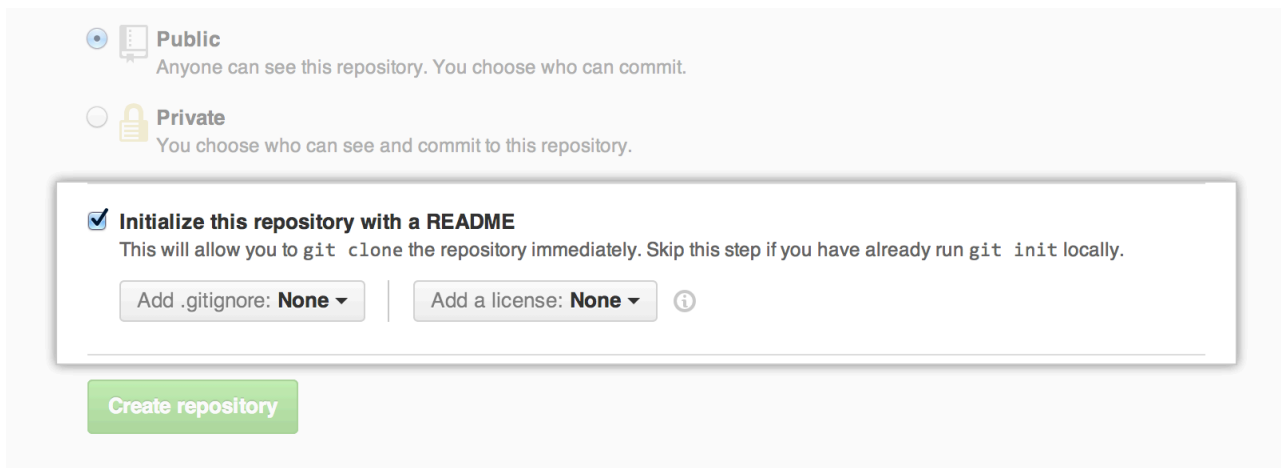
5、选择Initialize this repository with a README。



The screenshot shows the next step in the GitHub repository creation process. It features two radio button options: 'Public' (selected) and 'Private'. Below these, there is a section for 'Initialize this repository with a README'. This section has a checked checkbox and the text: 'This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.' Below this text are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', followed by an information icon. At the bottom of this section is a green button labeled 'Create repository'.

图片 3.5 Initialize this repository with a README

6、点击Create repository。



☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▼ | Add a license: **None** ▼ ⓘ

Create repository

图片 3.6 Create repository

恭喜！你已经成功地创建你的第一个仓库，并且通过 README 文件初始化了它。

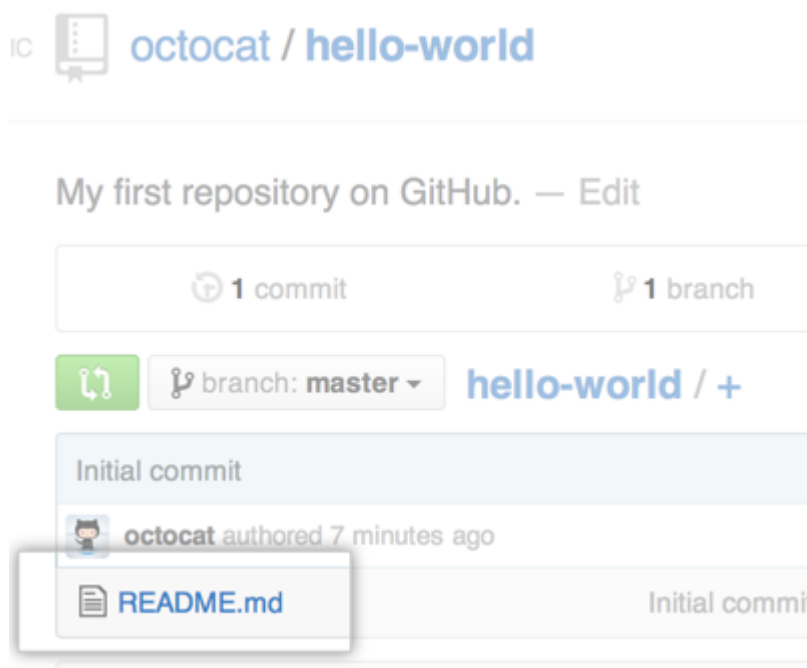
提交你的第一个更改

一个提交 (<https://help.github.com/articles/github-glossary#commit>) 就像你项目里的文件在一个特定时间点上的快照一样。

当你创建了一个新仓库，你通过 README 文件初始化它。README 文件里有关于你这个项目详细的解释，或者添加一些关于如何安装或者使用该项目的文档。README 文件的内容会自动地显示在你仓库的首页。

让我们提交一个对 README 文件的修改。

- 1、在你仓库的文件列表，点击 README.md。

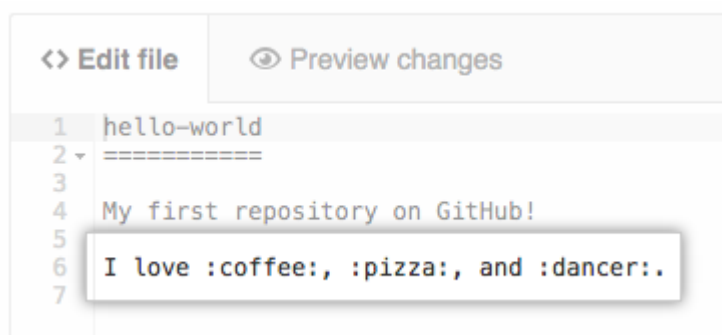


图片 3.7 New repository

- 2、在文件内容的上方，点击编辑按钮。



- 3、在 Edit file 标签上，输入一些关于你自己的信息。



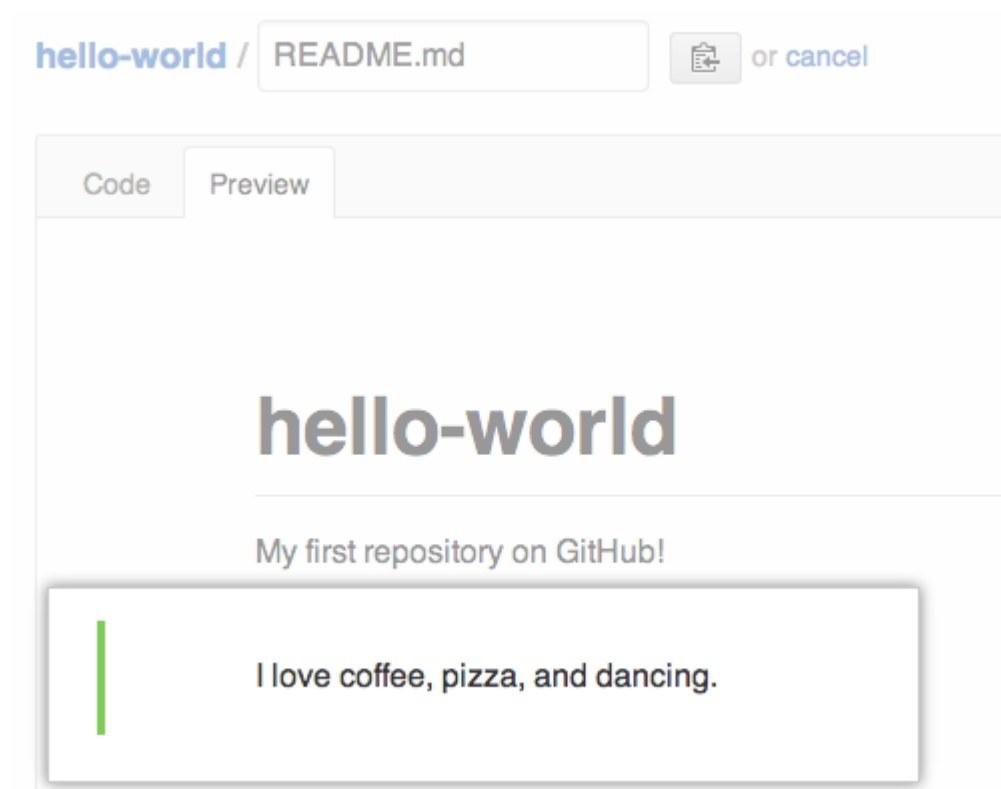
图片 3.9 你的仓库添加一个描述

4、在新内容的上方，点击 Preview changes。



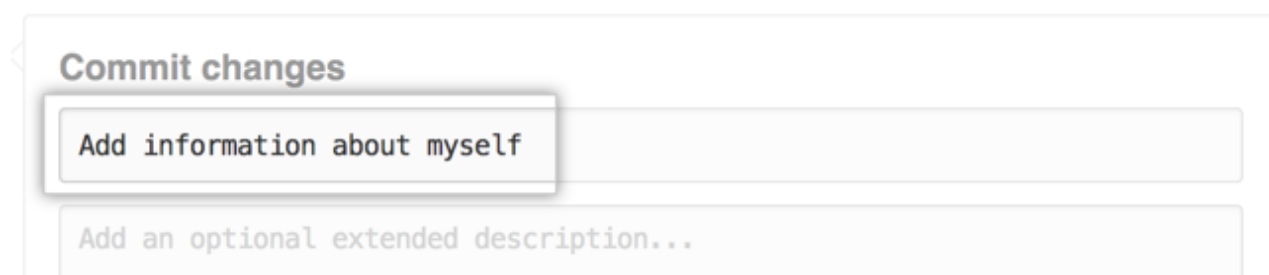
图片 3.10 Preview changes

5、检查一下你对这个文件进行的更改，你会看到新的内容被绿色标记。



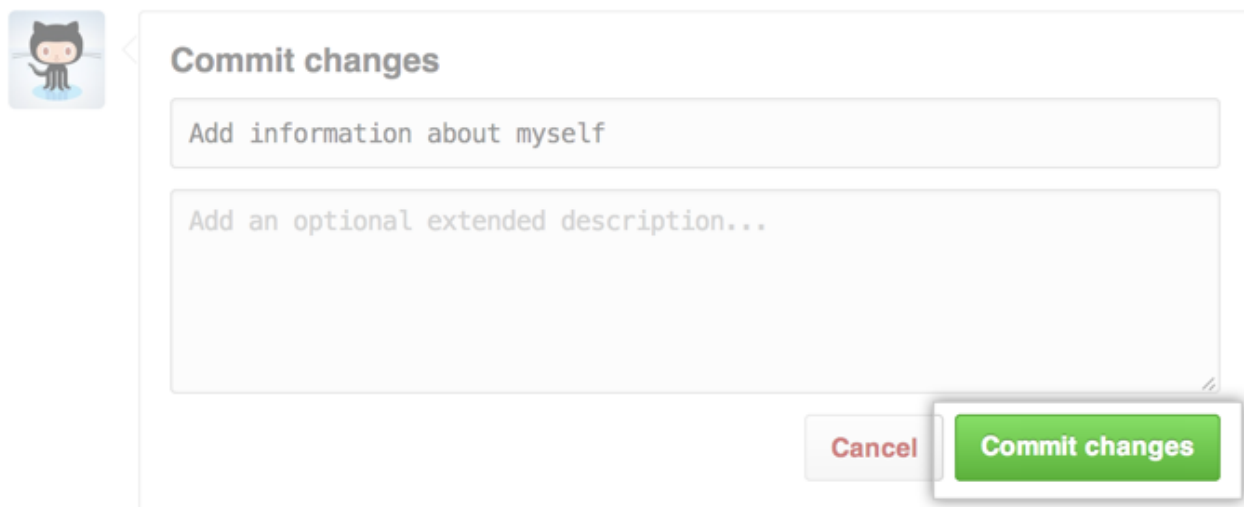
图片 3.11 新的内容被绿色标记

6、在页面的底部，即 "Commit changes" 下方，输入一些简短、有意义的提交信息来解释你对这个文件所进行的修改。



图片 3.12 Commit changes

7、点击 commit changes。



图片 3.13 Commit changes



Fork 一个仓库



Fork 一个示例仓库

Fork 是对一个仓库的克隆。克隆一个仓库允许你自由试验各种改变，而不影响原始的项目。

一般来说，forks 被用于去更改别人的项目（贡献代码给已经开源的项目）或者使用别人的项目作为你自己想法的初始开发点。

提出更改别人的项目

使用 forks 提出改变的一个很好的例子是漏洞修复。与其记录一个你发现的问题，不如：

- Fork 这个仓库
- 进行修复
- 向这个项目的拥有者提交一个 pull request

如果这个项目的拥有者认同你的成果，他们可能会将你的修复更新到原始的仓库中！

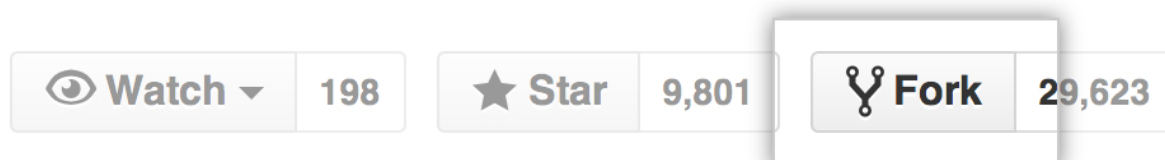
使用别人的项目作为你自己想法的初始开发点

[开源的核心](http://opensource.org/about) (<http://opensource.org/about>) 是共享代码，我们可以制作更好、更可靠的软件。

事实上，当你在 GitHub 上创建一个仓库时，你可以选择自动包含一个[许可文件](http://choosealicense.com/) (<http://choosealicense.com/>)，这个文件决定你是否希望将你的项目分享给其他人。

Fork 一个仓库分为简单的两步。我们已经创建了一个仓库让你用于练习！

1. 在 GitHub 上，定位到 [octocat/Spoon-Knife](https://github.com/octocat/Spoon-Knife) (<https://github.com/octocat/Spoon-Knife>) 仓库。
2. 在页面右上角，点击 Fork 按钮。



图片 4.1 Fork

就这样！现在你已经 fork 这个原始的 octocat/Spoon-Knife 仓库。

同步你的 Fork 仓库

你或许已经 fork 一个项目为了提交更改向 upstream 或原始仓库。这种情况下，很好的实现了将 upstream 仓库定期同步到你的 fork。要做到这一点，你需要在命令行中使用 Git。你可以使用你刚刚 fork 的 [octocat/Spoon-Knife](https://github.com/octocat/Spoon-Knife) (<https://github.com/octocat/Spoon-Knife>) 仓库去练习设置 upstream 仓库。

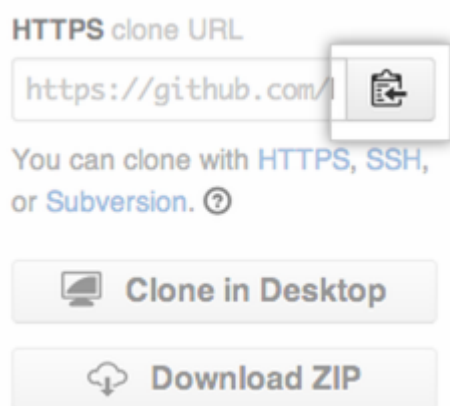
第一步：安装 Git

你首先应该 [安装 Git \(https://help.github.com/articles/set-up-git\)](https://help.github.com/articles/set-up-git)，如果你还没有。也不要忘记 [通过 Git 验证 GitHub \(https://help.github.com/articles/set-up-git#next-steps-authenticating-with-github-from-git\)](https://help.github.com/articles/set-up-git#next-steps-authenticating-with-github-from-git)。

第二步：为你 fork 的仓库创建一个本地克隆

现在，你已经成功 fork Spoon-Knife 仓库，但在你自己的计算机上并没有这个仓库的文件。让我们克隆你 Fork 的代码到你本地的计算机上。

- 1、在 GitHub 上，定位到你 fork 的 Spoon-Knife 仓库。
- 2、在你 fork 的仓库页面的右侧边栏，点击复制图标复制你 fork 的 URL。



图片 4.2 复制图标复制你 fork 的 URL

- 3、打开 Terminal 命令（针对苹果系统用户）或者命令提示行（针对 Windows 和 Linux 用户）。
- 4、输入 `git clone`，然后粘贴在步骤 2 复制的 URL。它看起来就像如下所示，用你 GitHub 的用户名代替 `YOUR-USERNAME`：

```
$ git clone https://github.com/YOUR-USERNAME/Spoon-Knife
```

- 5、按下 回车键，你的本地克隆就创建了。

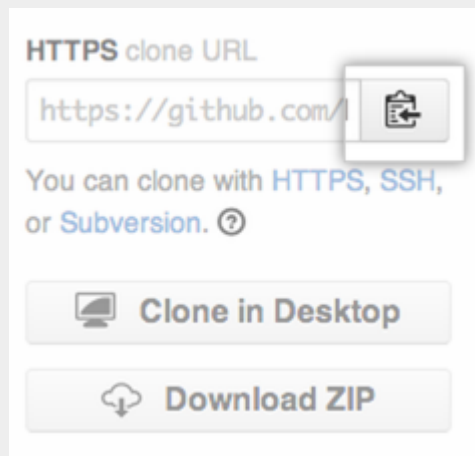
```
`` $ git clone https://github.com/YOUR-USERNAME/Spoon-Knife Cloning into 'Spoon-Knife' ... remote: Counting objects: 10, done. remote: Compressing objects: 100% (8/8), done. remote: Total 10 (delta 1), reused 10 (delta 1) Unpacking objects: 100% (10/10), done.
```

现在，你已经有了针对你 fork 的 Spoon-Knife 仓库的本地克隆代码！

第三步：通过配置 Git 来同步你 fork 的原始 Spoon-Knife 仓库

当你 fork 一个项目是为了提出更改这个原始的仓库，你可以配置 Git 将原始的或者 upstream 的变化更改到你本地。

- 1、在 GitHub 上，定位到 [octocat/Spoon-Knife](https://github.com/octocat/Spoon-Knife) (<https://github.com/octocat/Spoon-Knife>) 仓库。
- 2、在这个仓库页面的右侧边栏，点击复制图标复制这个仓库的 URL。



图片 4.3 复制图标复制你 fork 的 URL

- 3、打开 Terminal 命令(针对 Mac 用户)或提示命令行(Windows 和 Linux 用户)。
- 4、更改到你在步骤 2（创建一个本地）创建的你的 fork 的本地的目录。
 1. 回到根目录，只输入 `cd`。
 2. 输入 `ls`，列出当前目录的文件和文件夹。
 3. 输入 `cd 目录名` 进入你输入的目录下。
 4. 输入 `cd ..` 回到上一目录。

5、输入 `git remote -v`，按下回车键，你将会看到你的 fork 当前配置的远程仓库：

...

```
$ git remote -v
origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch)
origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (push)
```

6、输入 `git remote add upstream`，然后粘贴你在步骤 2 复制的 URL 并按下回车键。它看起来如下所示：

```
$ git remote add upstream https://github.com/octocat/Spoon-Knife.git
```

7、验证你 fork 里新指明的这个 upstream 仓库，再次输入 `git remote -v`。你将会看到你 fork 的 URL 作为原始的地址，而原始的仓库的 URL 作为 upstream。

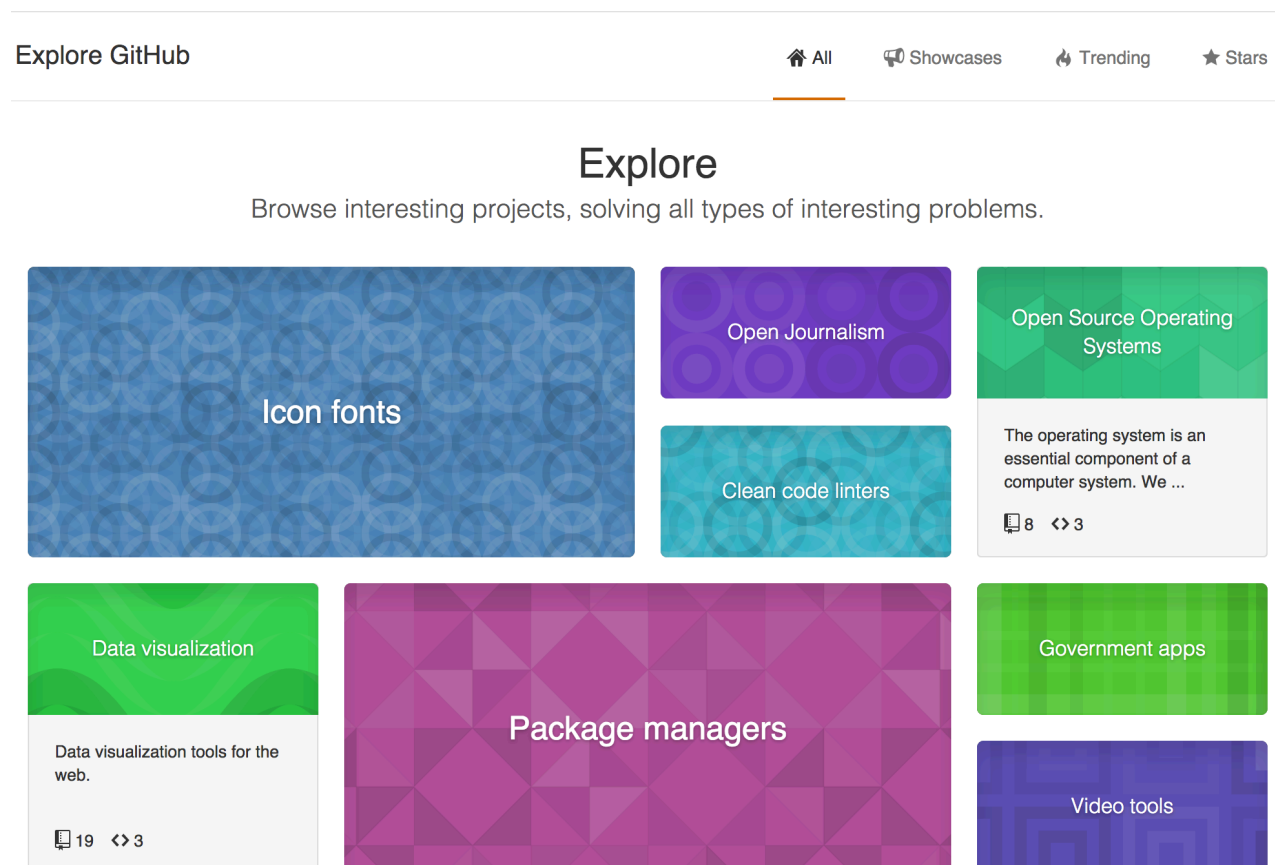
```
$ git remote -v origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (fetch) origin https://github.com/YOUR_USERNAME/YOUR_FORK.git (push) upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (fetch) upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git (push)
```

现在，你可以保持你的 fork 与 upstream 的仓库的同步，利用几个 Git 命令。想知道更多信息，请参阅 [Syncing a fork \(https://help.github.com/articles/syncing-a-fork/\)](https://help.github.com/articles/syncing-a-fork/)。

检索其他仓库来 Fork

每个公开的仓库都可以被 fork，所以你可以搜索你感兴趣的项目并 fork 它！

[Explore GitHub \(https://github.com/explore\)](https://github.com/explore) 是一个大的平台，可以让你找到感兴趣的项目。经常访问这个页面去关注最新和最酷的东西。



图片 4.4 Explore GitHub



社会化



Follow 一个人

GitHub 一个很强大的特性就是可以看到其他人在从事或者与什么相关的工作。

当你在 GitHub 上跟踪了某些人之后，你就会在你的面板里面收到他们活动的动态通知。你可以在他们的页面上，点击 Follow 按钮。



Watch 一个项目

在某些情况下，你可以需要实时跟踪一个特别项目的动态，这和跟踪一个用户比较类似，只是关注点仅仅在于该项目的事件。你可以给这个项目发送电子邮件订阅或者在页面上配置通知设置。比较典型的通知比如对补丁或者问题的评论，或者仅仅是项目的一个评论。

我们的朋友 Octocat 有一个叫做 [Hello World \(https://github.com/octocat/Hello-World\)](https://github.com/octocat/Hello-World) 的项目，我们想去查看。

当你在项目页面的时候，你会发现页面顶部有一个 watch 按钮，点击它。



图片 5.2 watch

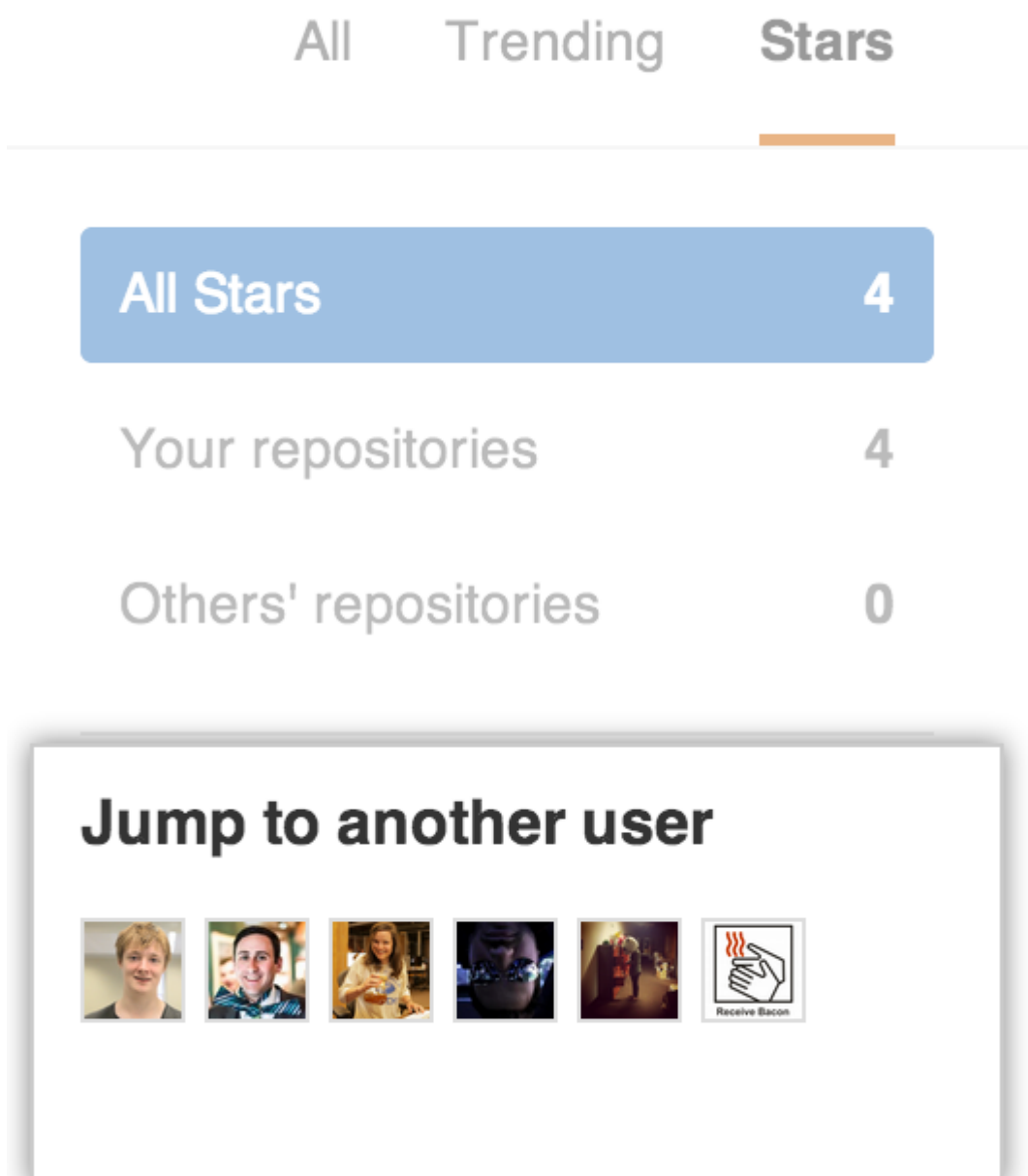
恭喜，你关注了这个 Hello World 项目。如果 Octocat 更新了这个项目，你将会在你的控制面板收到改动的通知。

其他你可以做的事

你已经使用了一些 GitHub 提供的最基本的交互功能，但不要仅仅限制于那些！看看其他的一些交互特征吧！

关注

如果你对你的朋友以前关注的一些项目感兴趣，但是你没有在你的控制面板看到这些，鼠标指向 [start page \(https://github.com/stars\)](https://github.com/stars) 然后 jump to another user。

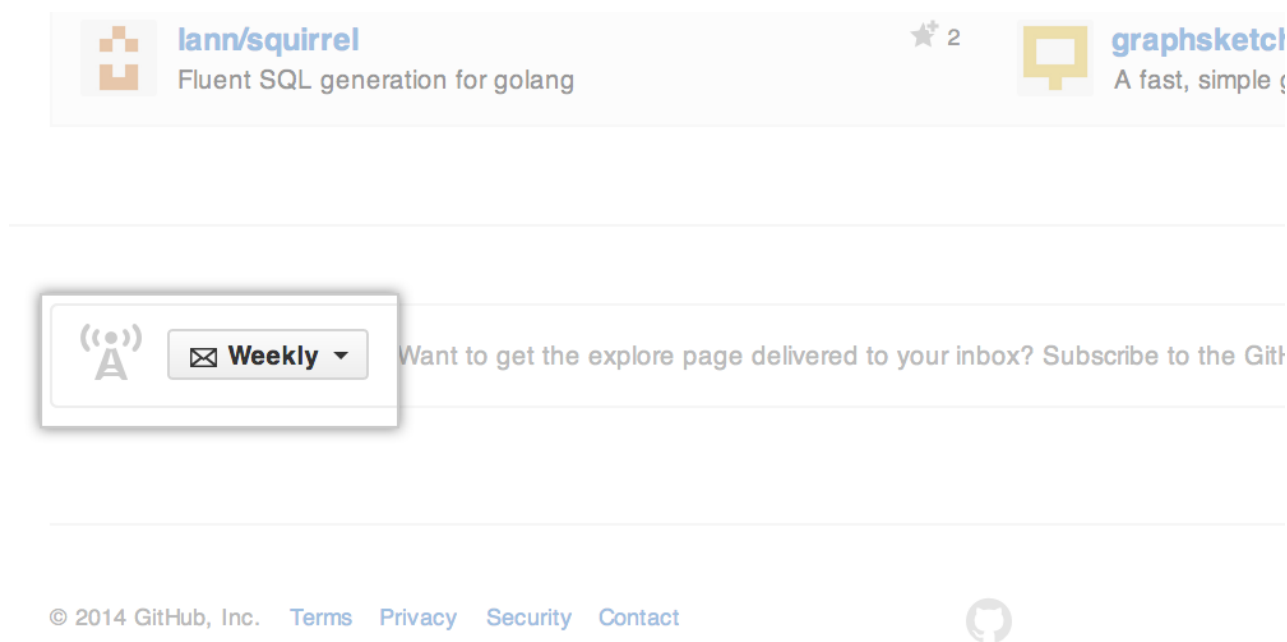


图片 5.3 jump to another user

发现 / 交流

发现页面会呈现你关注的好友关注的一些项目以及 GitHub 官方和当前比较流行的一些项目。

如果你想每天、每周、或者每月收到这些订阅，查看发现页面底部的新闻公告。



图片 5.4 订阅

你也可以稍后在[订阅页面 \(https://github.com/explore/subscribe\)](https://github.com/explore/subscribe) 配置你的订阅设置。

如果你想查看当前比较火热的项目和用户，[鼠标指向 Trending 页面 \(https://github.com/trending\)](https://github.com/trending) 查看。


Trending repositories on GitHub today

Repositories


Developers

Trending: today

1


 **HubSpot/tether** JavaScript

A positioning engine to make overlays, tooltips and dropdowns faster #hubspot-open-source


built by 

★ Star

2


 **sahat/hackathon-starter** CSS

A boilerplate for Node.js web applications

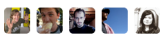
built by 

★ Star

3


 **MobileChromeApps/mobile-chrome-apps** JavaScript

Chrome apps on Android and iOS


built by 

★ Star

4

 **afaqurk/linux-dash** CSS

A drop-in, low-overhead monitoring web dashboard for a linux machine.

built by 

★ Star

All

Trending

Star

All languages

Unknown languages

Other: Languages

ProTip™ Looking for most forked repositories? [Try this search](#)

图片 5.5 鼠标指向 Trending 页面



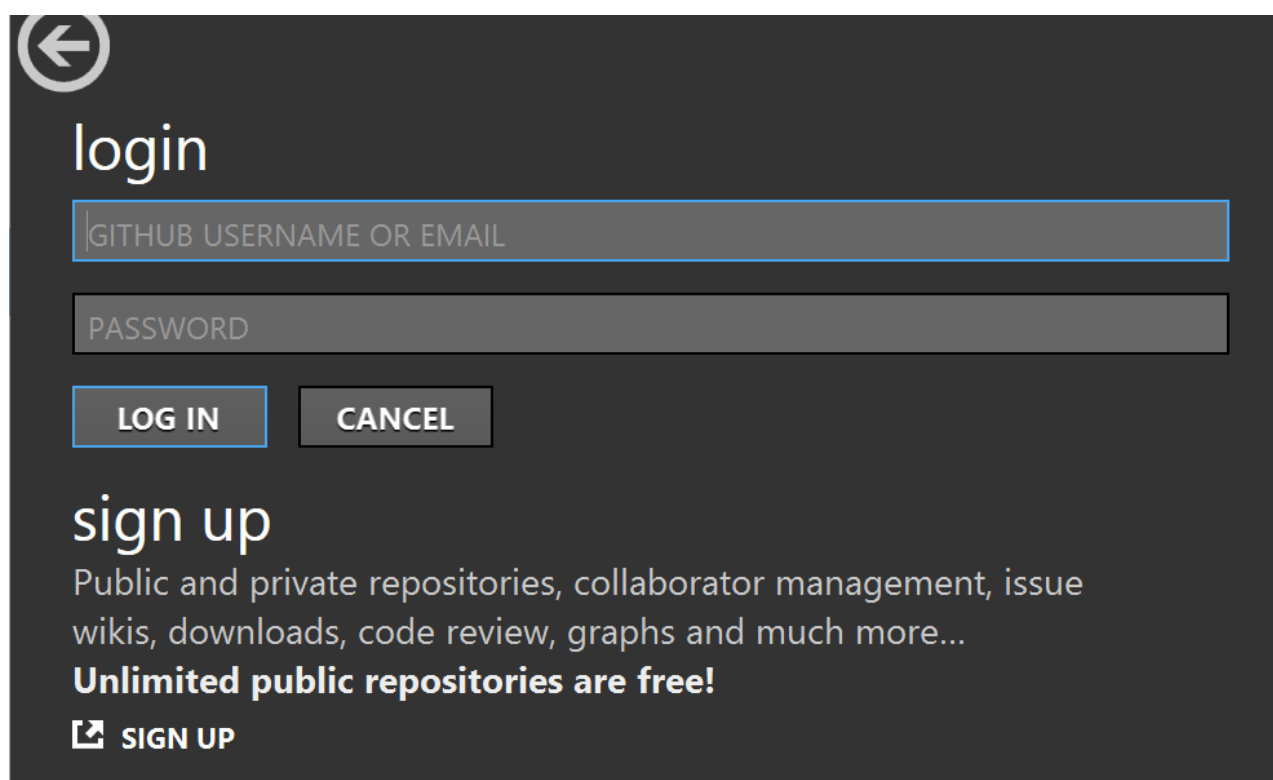
图形化工具



GitHub for Windows

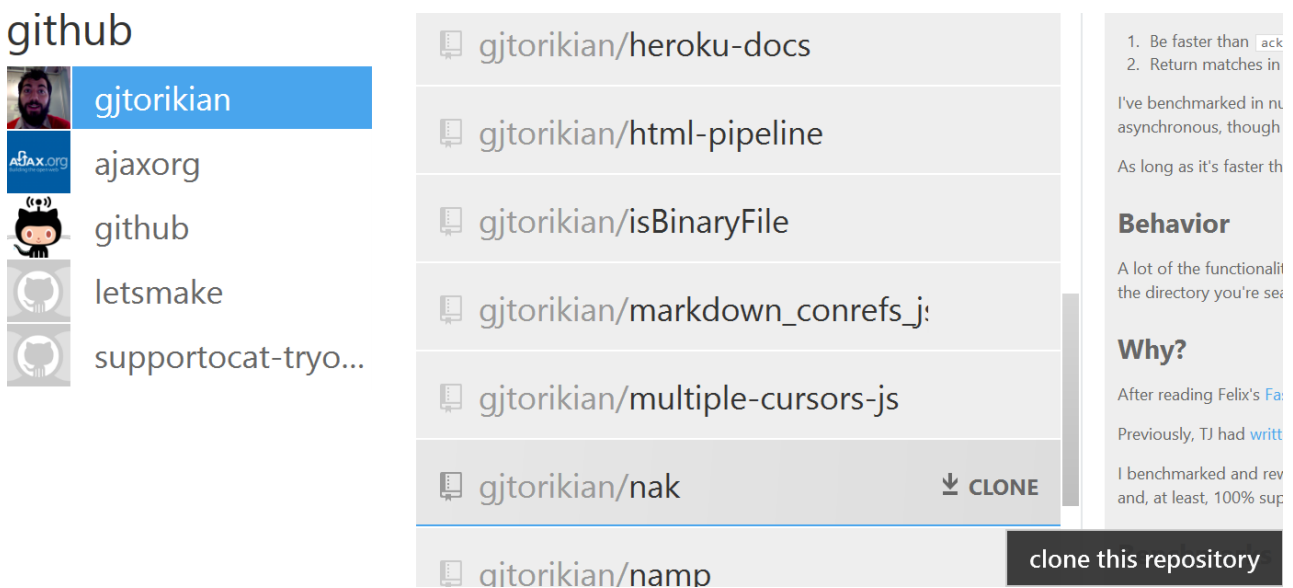
安装

- 1、从 windows.github.com (<http://windows.github.com/>) 下载最新版本的 GitHub。
- 2、当你开启软件时，你可以选择用你的 GitHub 账户登录，或者新建一个账户。



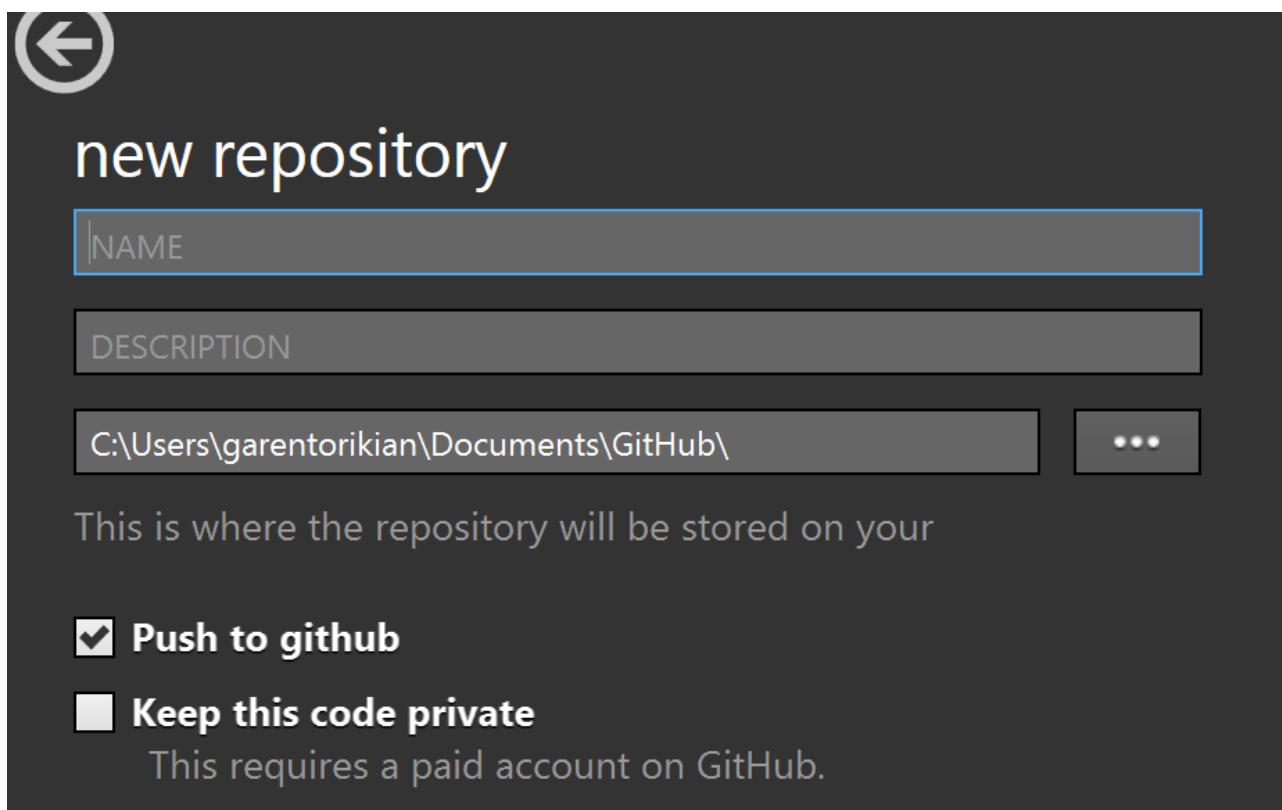
图片 6.1 选择用你的 GitHub 账户登录，或者新建一个账户

- 3、在左侧，你可以看到你的 GitHub 账户，同时你也能看到你所在组的其他（用户）。点击一个用户名，你将看到哪些仓库是可用的，点击 clone 将把对应仓库克隆到你的电脑。



图片 6.2 点击 clone 将把对应仓库克隆到你的电脑

4、另外，你可以点击顶部的 + add 来添加一个新的本地仓库。



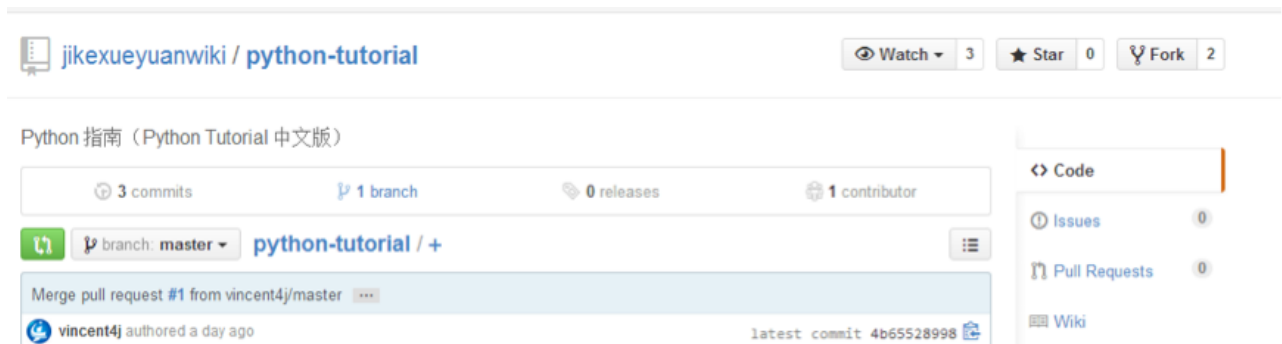
图片 6.3 添加一个新的本地仓库

你准备好通过 GitHub for Windows 开始编程和同步修改。

Fork

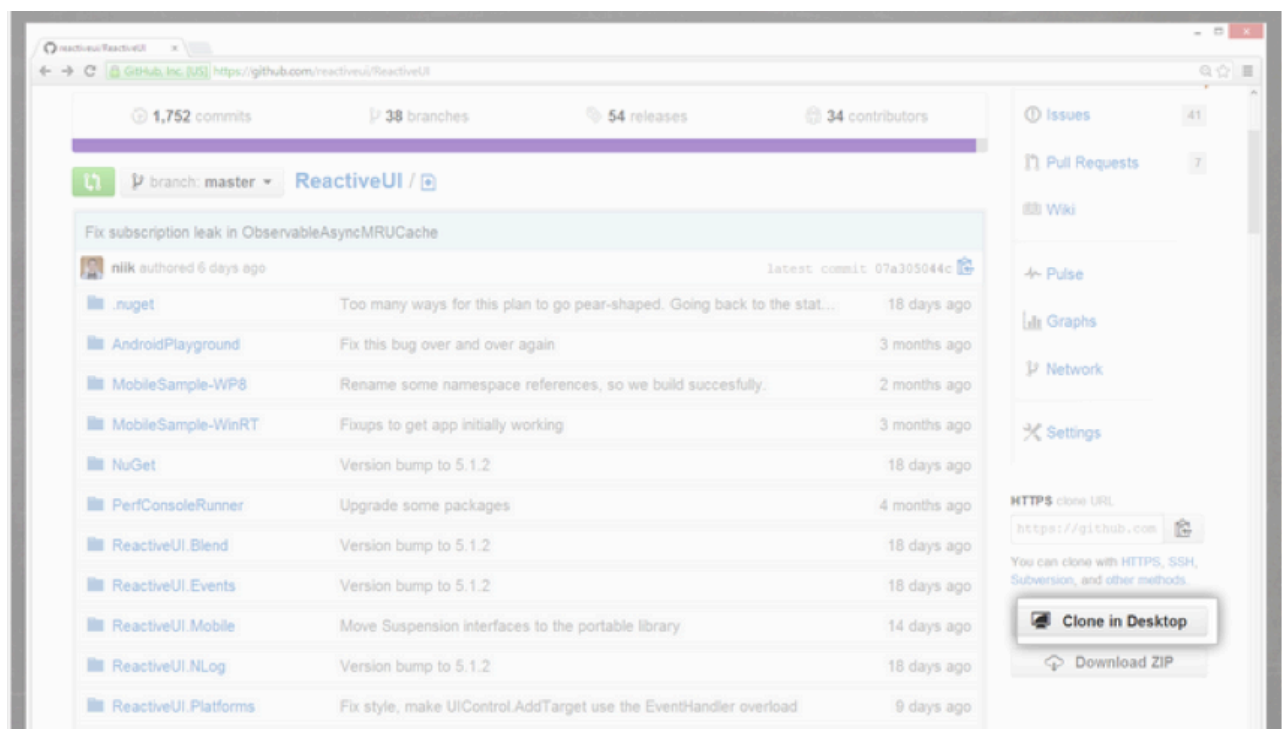
直接从 GitHub 上克隆来添加库，也是一种不错的选择。

1、你可以直接用你的个人账户或者你所属的组织，通过 GitHub 来浏览相关项目。



图片 6.4 fork 仓库

2、你也可以直接在 GitHub 上通过 Clone in Desktop 按钮来进行一键克隆。

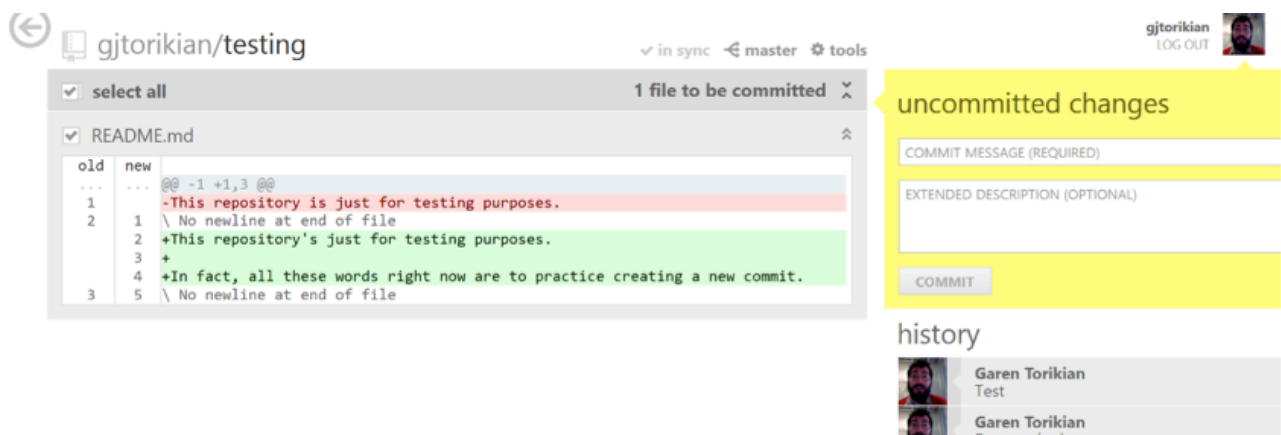


图片 6.5 Clone in Desktop

提交到本地

你在本地更新了数据，需要先提交到本地仓库：

- 1、点击你需要同步的库的名称。
- 2、你将看到一个表单，列举了你最新的变动。增添一个提交日志（另外可以选择增加一个描述），然后提交。



图片 6.6 windows-commit-local

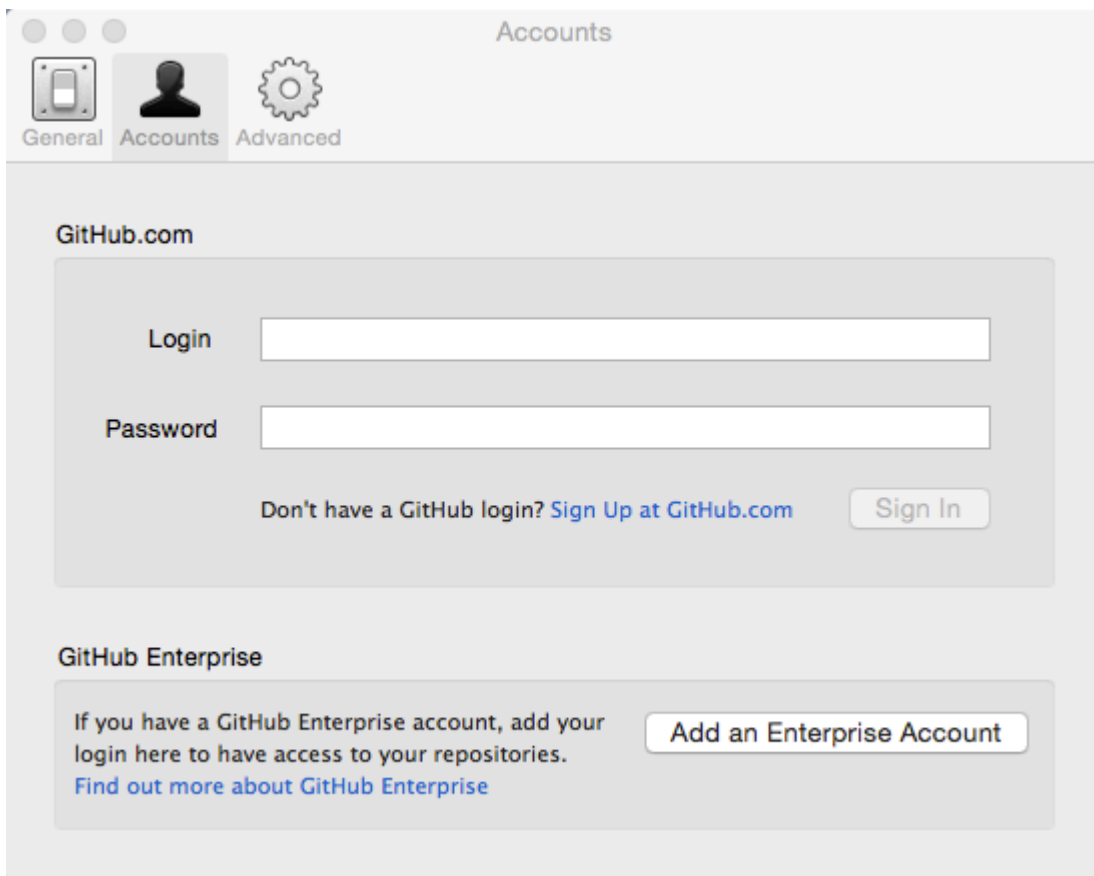
同步远程仓库

当有新的本地提交记录时，上图中的 in sync 按钮会被点亮，并且文字变成 sync，只需要点击 sync 按钮，就同步到了 GitHub 上的远程仓库。

GitHub for Mac

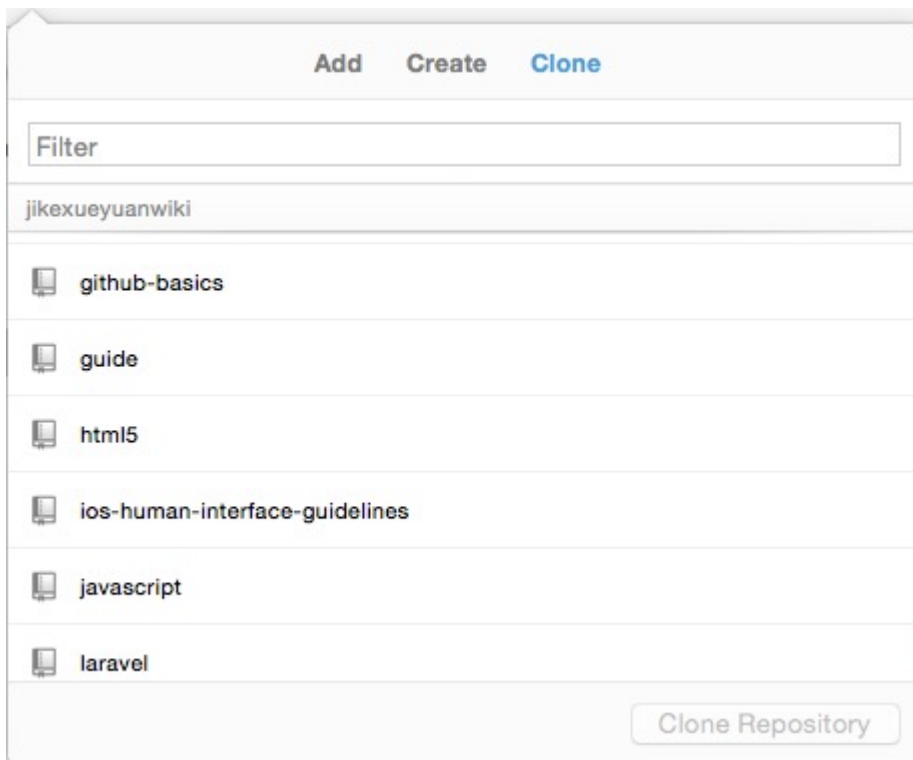
安装

- 1、从 mac.github.com (<http://mac.github.com/>) 下载最新版本的 GitHub。
- 2、当你开启软件时，你可以选择用你的 GitHub 账户登录，或者新建一个账户。



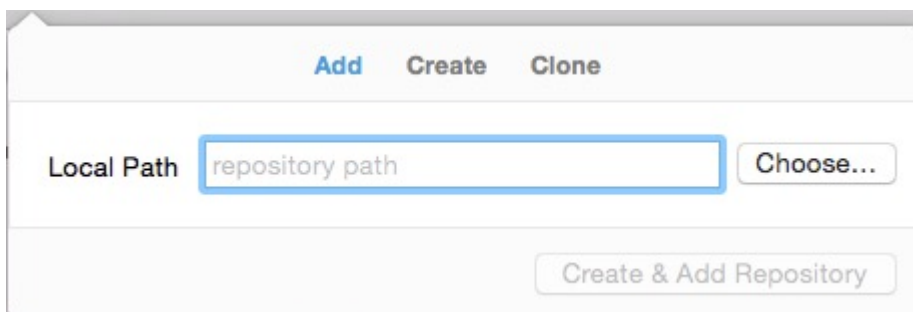
图片 6.7 选择用你的 GitHub 账户登录，或者新建一个账户

- 3、在左侧，你可以看到你的 GitHub 账户，同时你也能看到你所在组的其他（用户）。点击一个用户名，你将看到哪些仓库是可用的，点击 clone 将把对应仓库克隆到你的电脑。



图片 6.8 点击 clone 将把对应仓库克隆到你的电脑

4、另外，你可以点击顶部的 + add 来添加一个新的本地仓库。



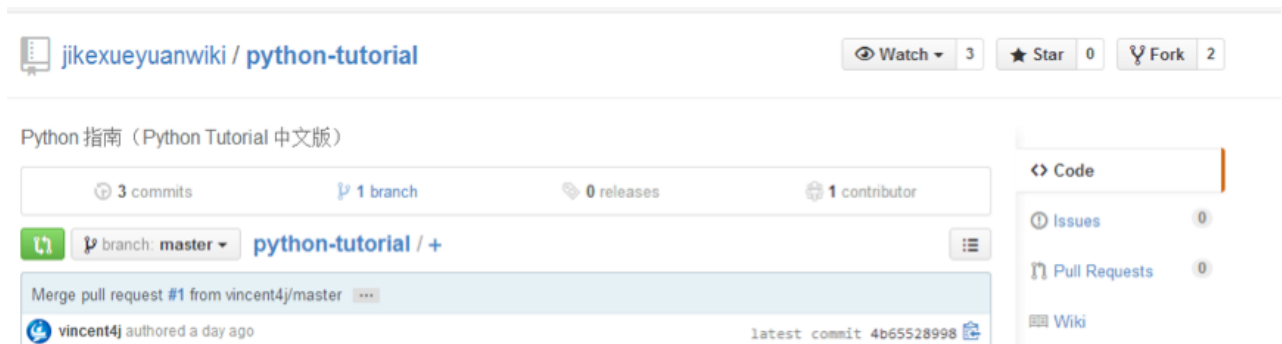
图片 6.9 添加一个新的本地仓库

你准备好通过 GitHub for Mac 开始编程和同步修改。

Fork

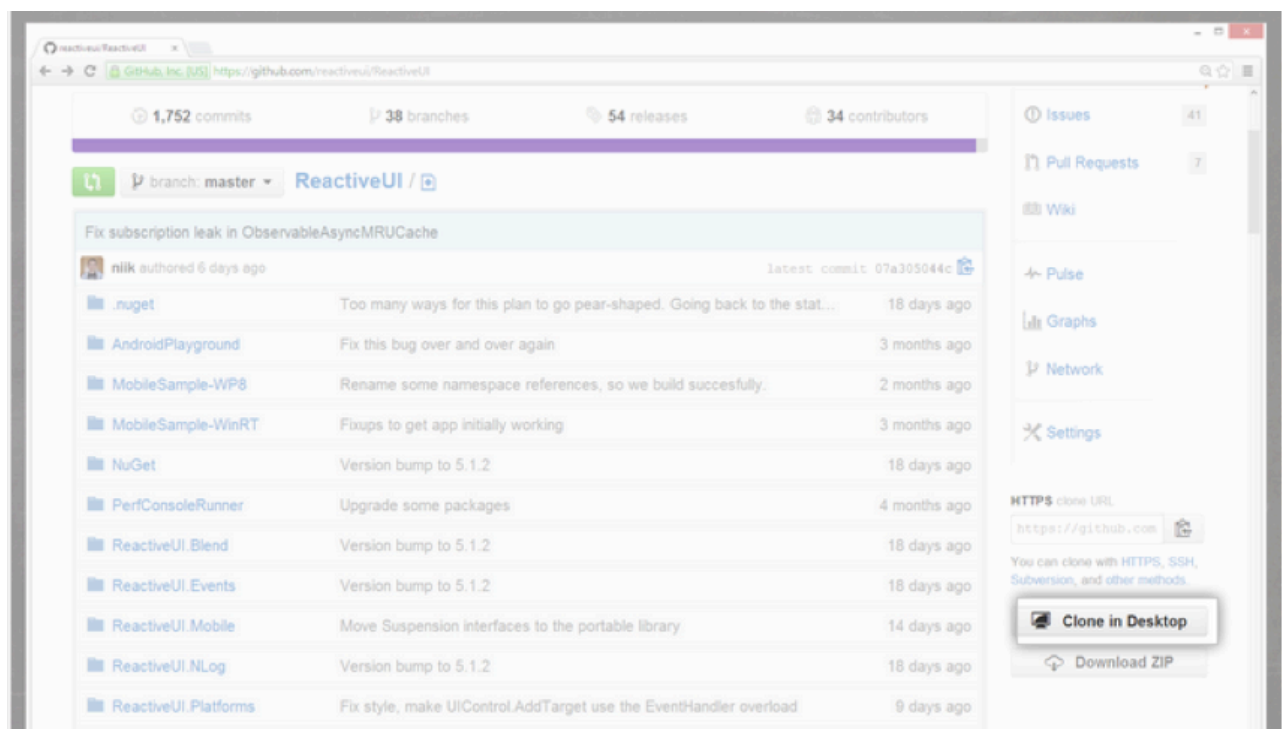
直接从 GitHub 上克隆来添加库，也是一种不错的选择。

1、你可以直接用你的个人账户或者你所属的组织，通过 GitHub 来浏览相关项目。



图片 6.10 fork 仓库

2、你也可以直接在 GitHub 上通过 Clone in Desktop 按钮来进行一键克隆。

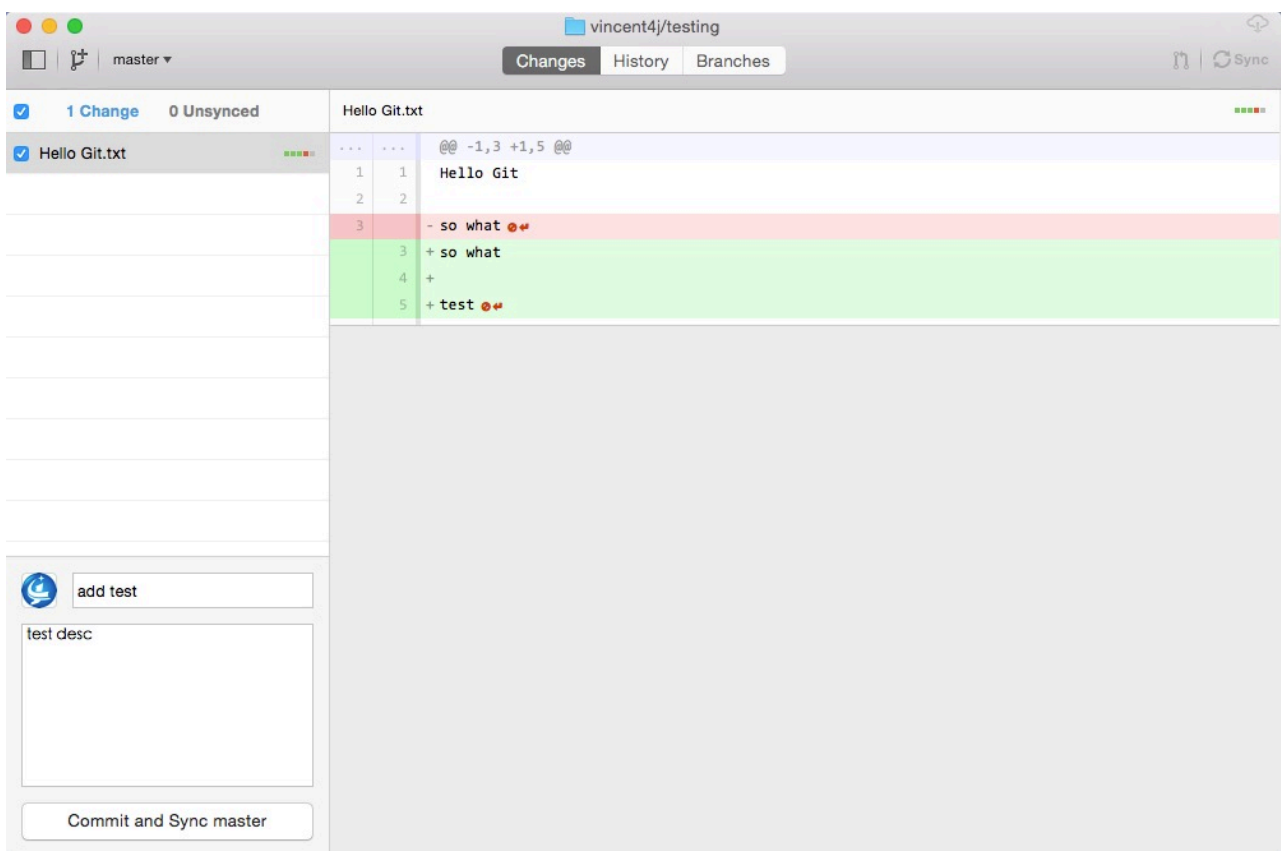


图片 6.11 Clone in Desktop

提交到本地

你在本地更新了数据，需要先提交到本地仓库：

- 1、点击你需要同步的库的名称。
- 2、你将看到一个表单，列举了你最新的变动。增添一个提交日志（另外可以选择增加一个描述），然后点击 Commit and Sync master，提交修改到本地仓库并同时同步到远程仓库。



图片 6.12 mac-commit-local

同步远程仓库

当有新的本地提交记录时，右上角的 **Sync** 按钮会点亮，点击它，就同步到了 GitHub 上的远程仓库。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台



更多信息请访问 

<http://wiki.jikexueyuan.com/project/github-basics/>