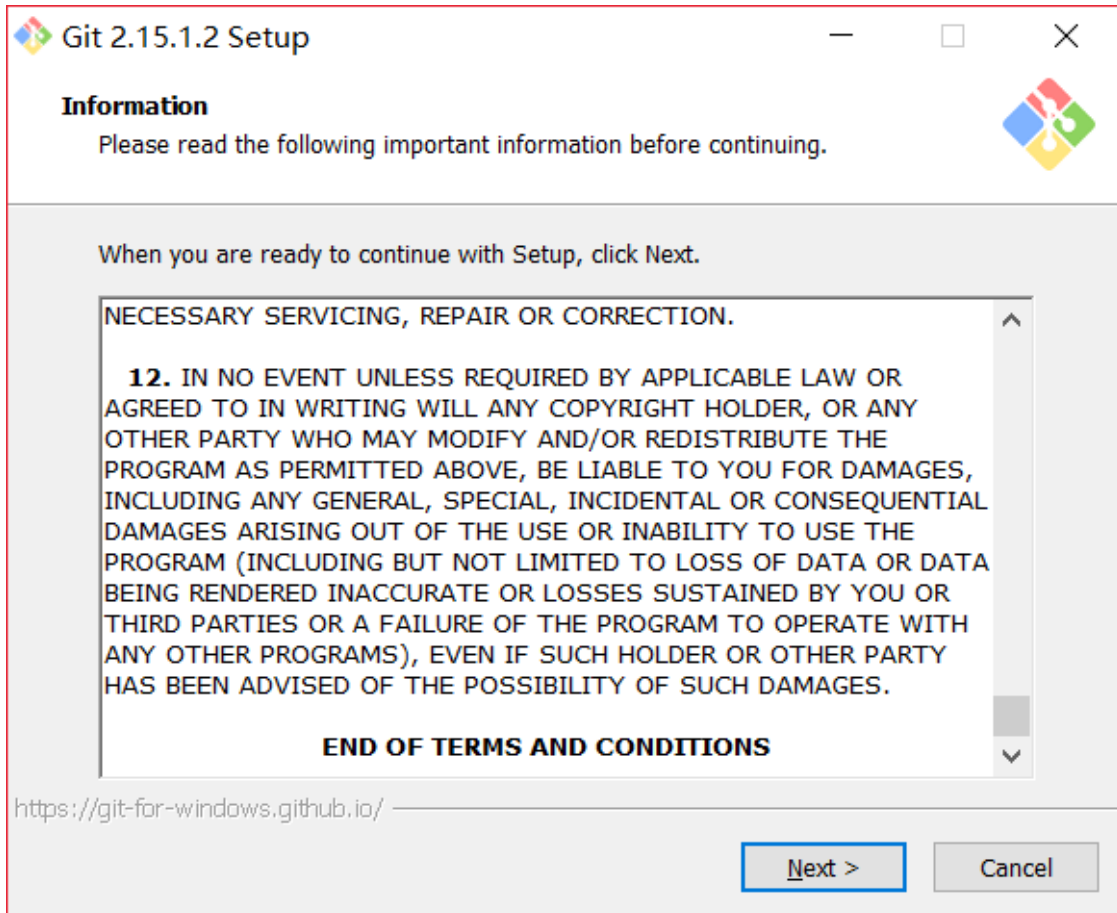


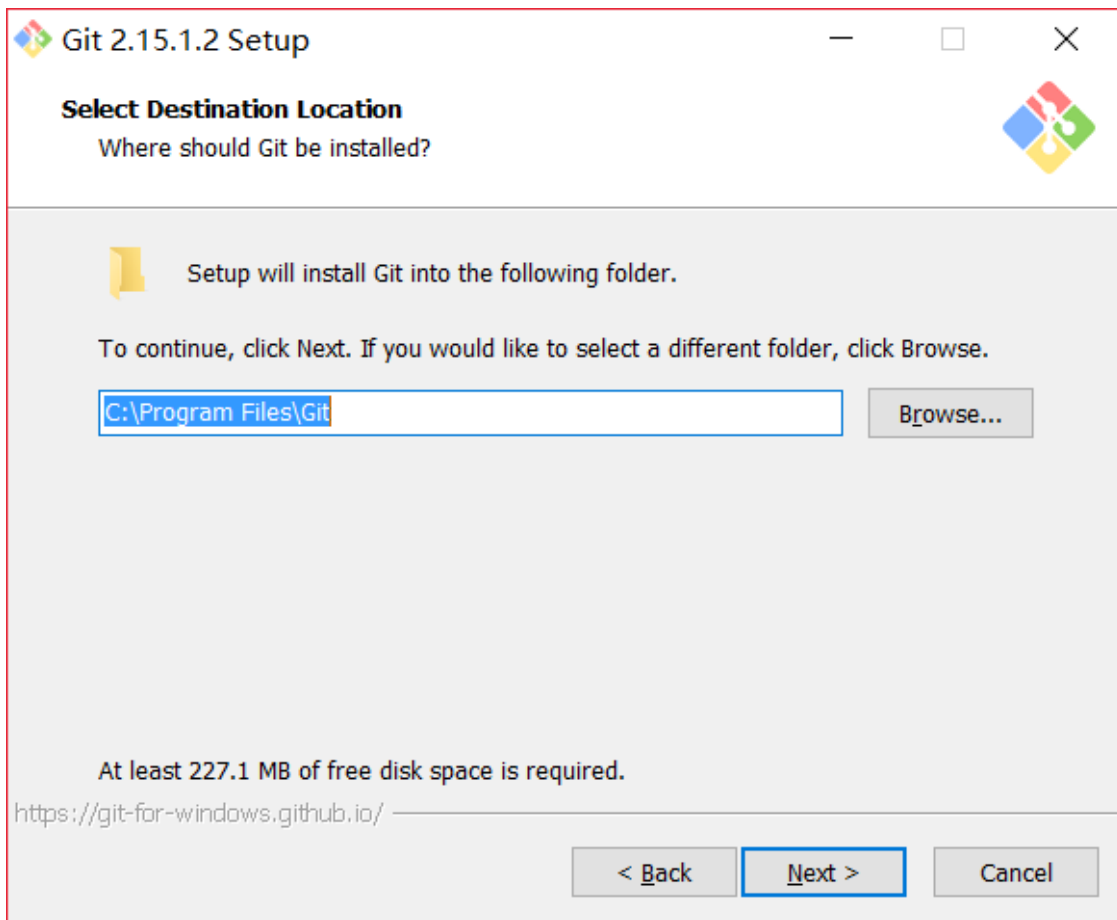
Git 是当今最流行的版本控制软件，它包含了许多高级工具，这里小编就讲一下 Git 的安装。

下载地址：<https://git-scm.com/downloads>

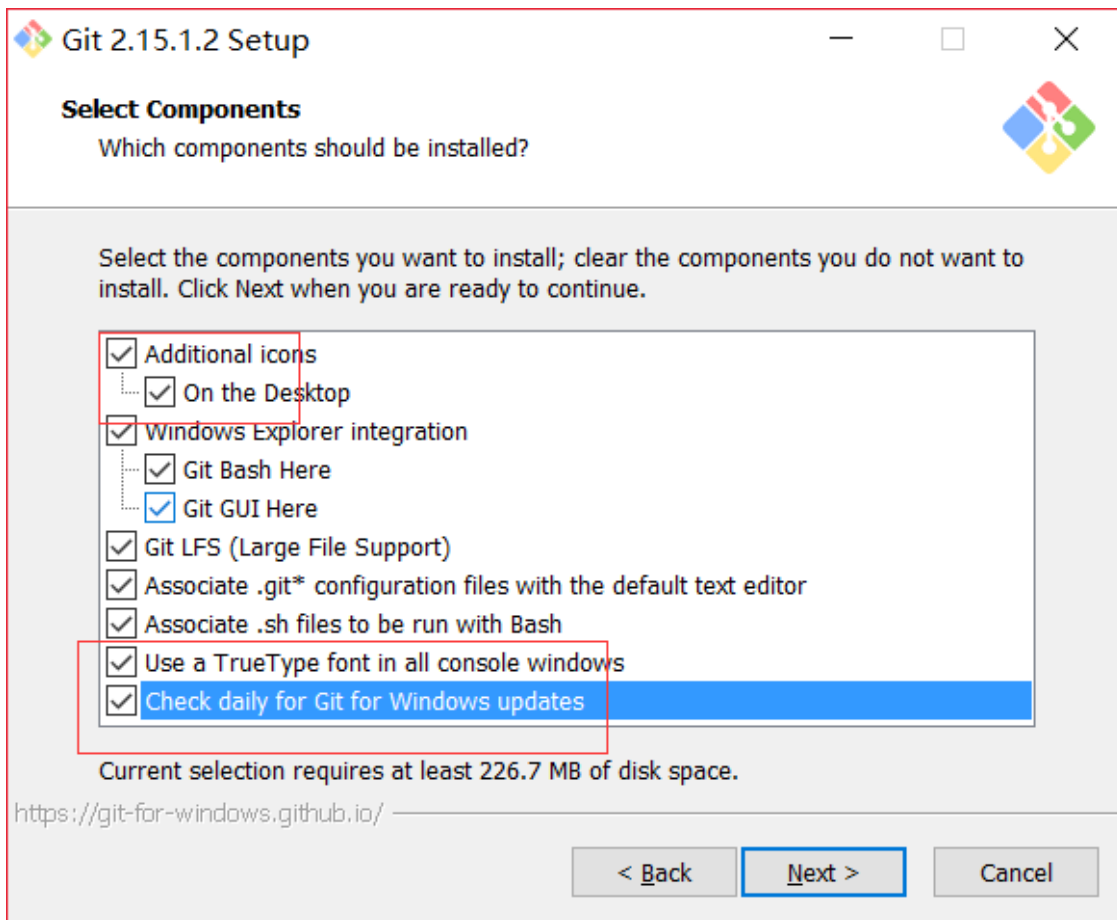
首先如下图：（点击 next）



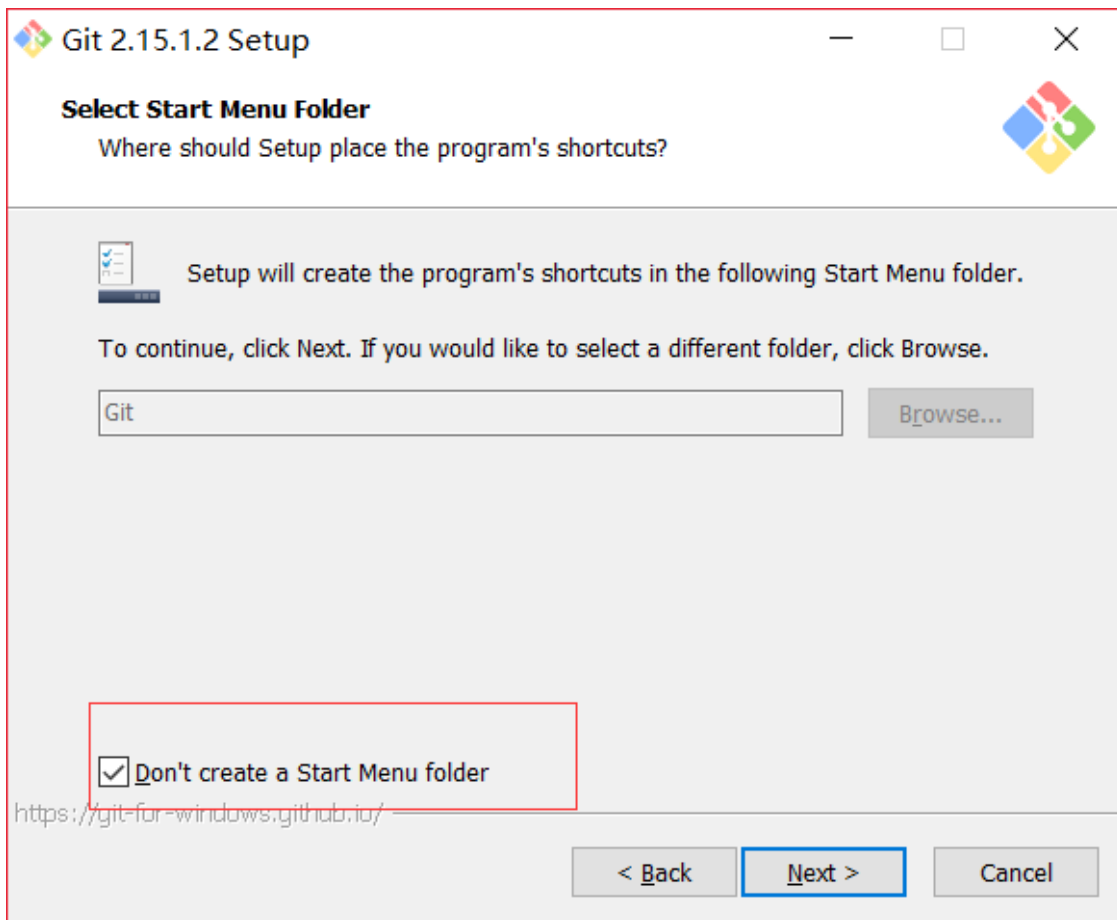
第二步：文件位置存储，可根据自己盘的情况安装



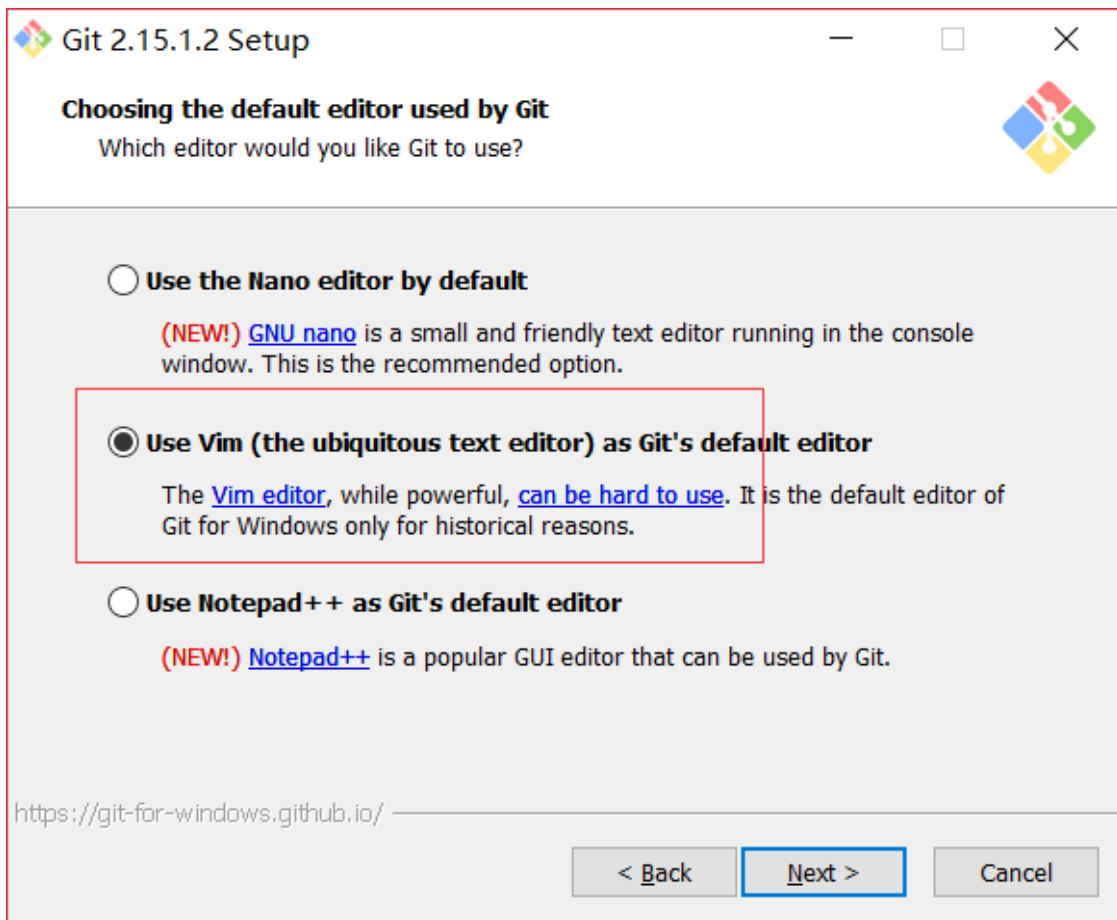
第三步：安装配置文件，自己需要的都选上，下一步



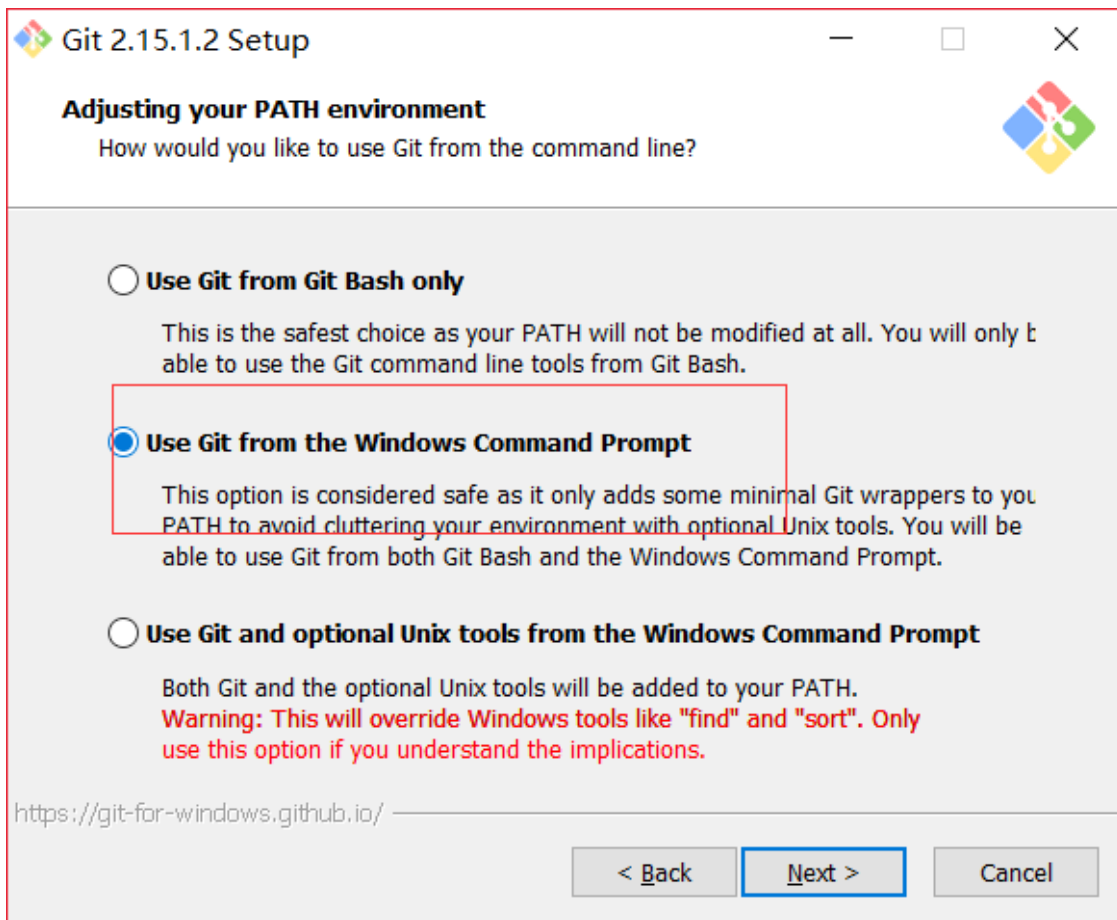
第四步：不创建启动文件夹，下一步：



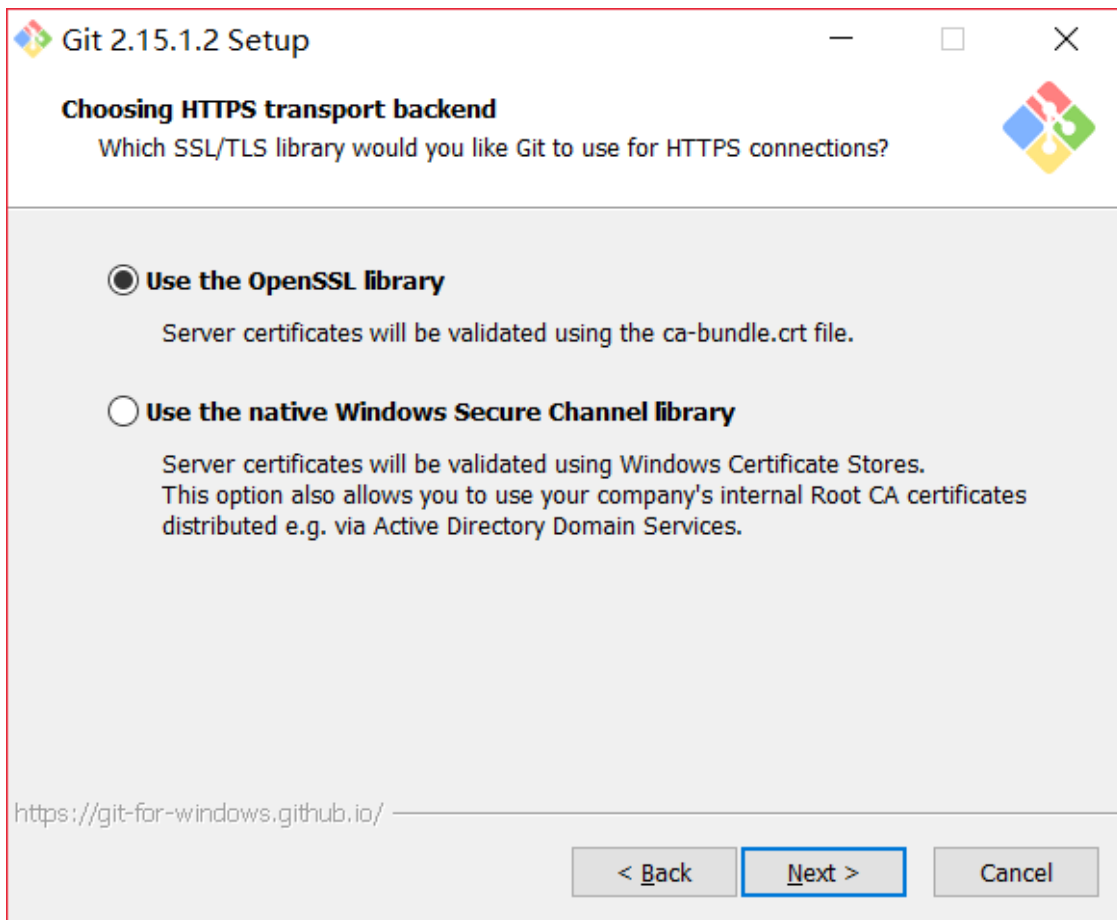
第五步：选择默认的编辑器，我们直接用推荐的就行，下一步



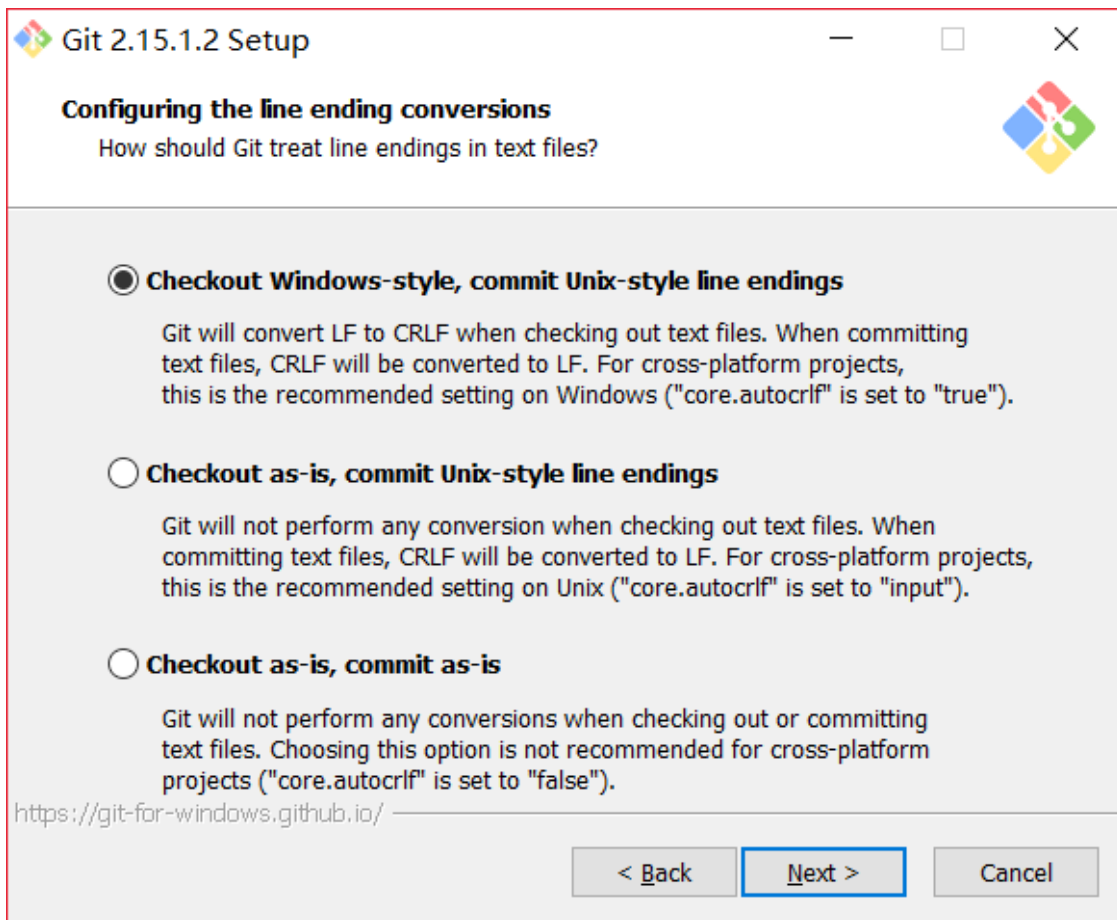
第六步：勾上第二项，这样就可以在 cmd 中操作，下一步



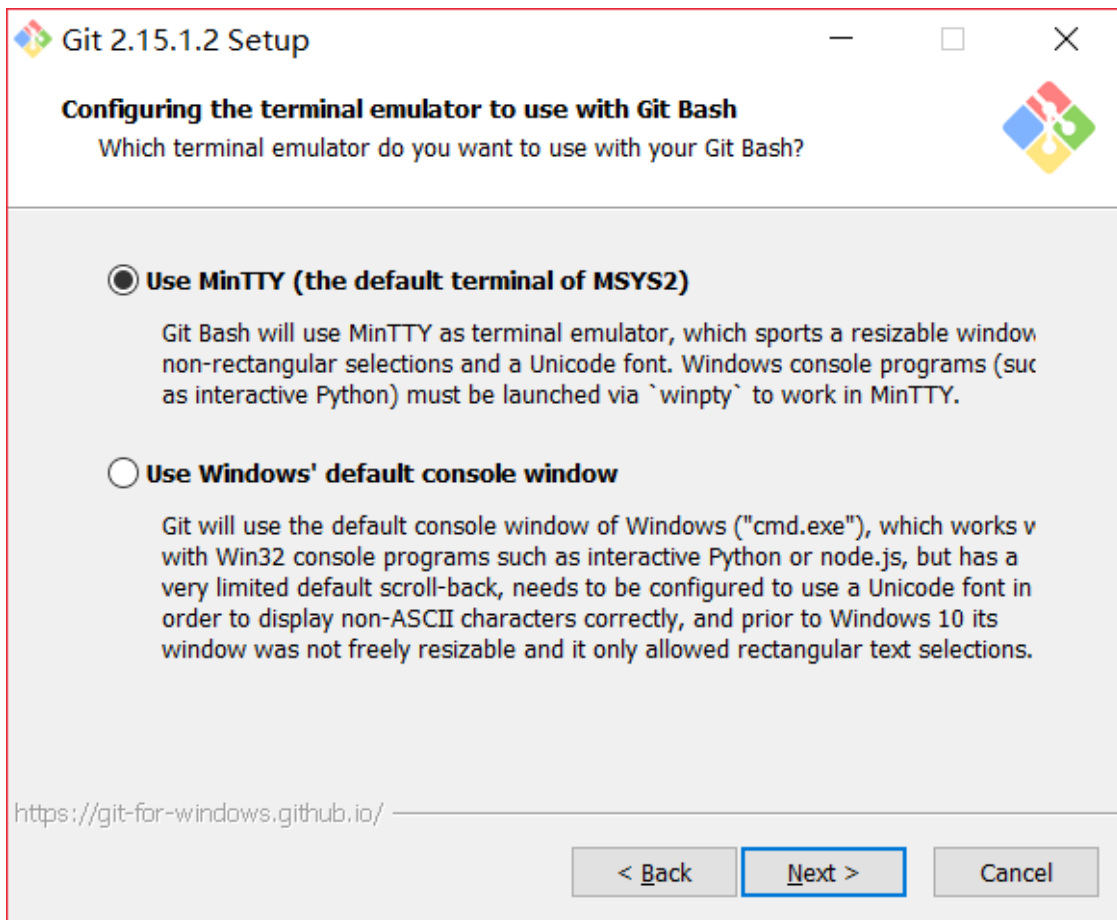
第七步：使用默认设置就行，下一步：



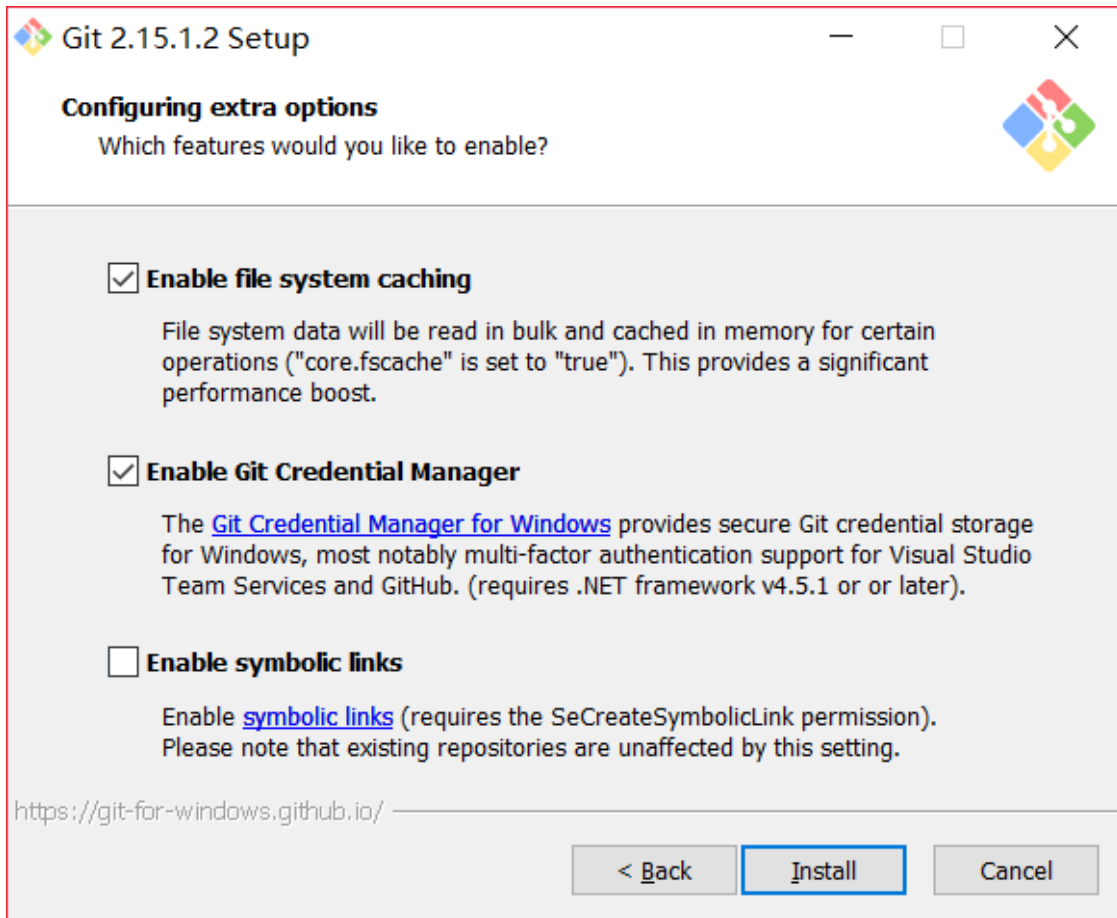
第八步：配置行结束标记，保持默认“Checkout”



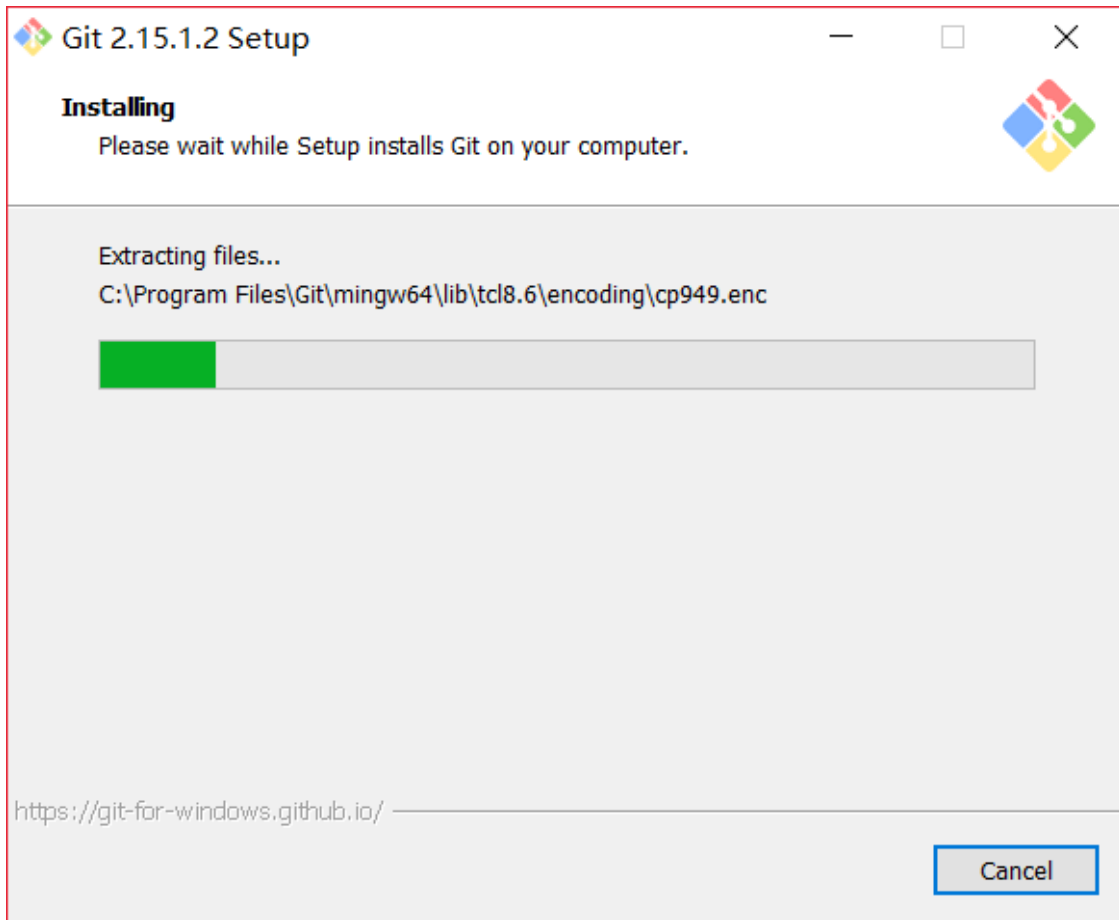
第九步：在终端模拟器选择页面，默认即可，配置后 Git



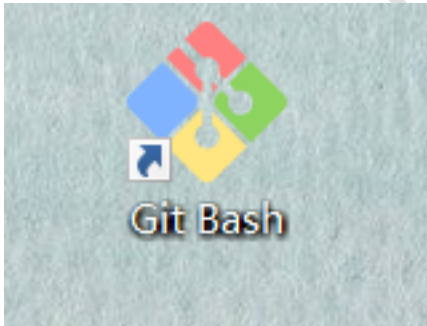
第十步：最后配置 Git 额外选择默认即可，然后安装。

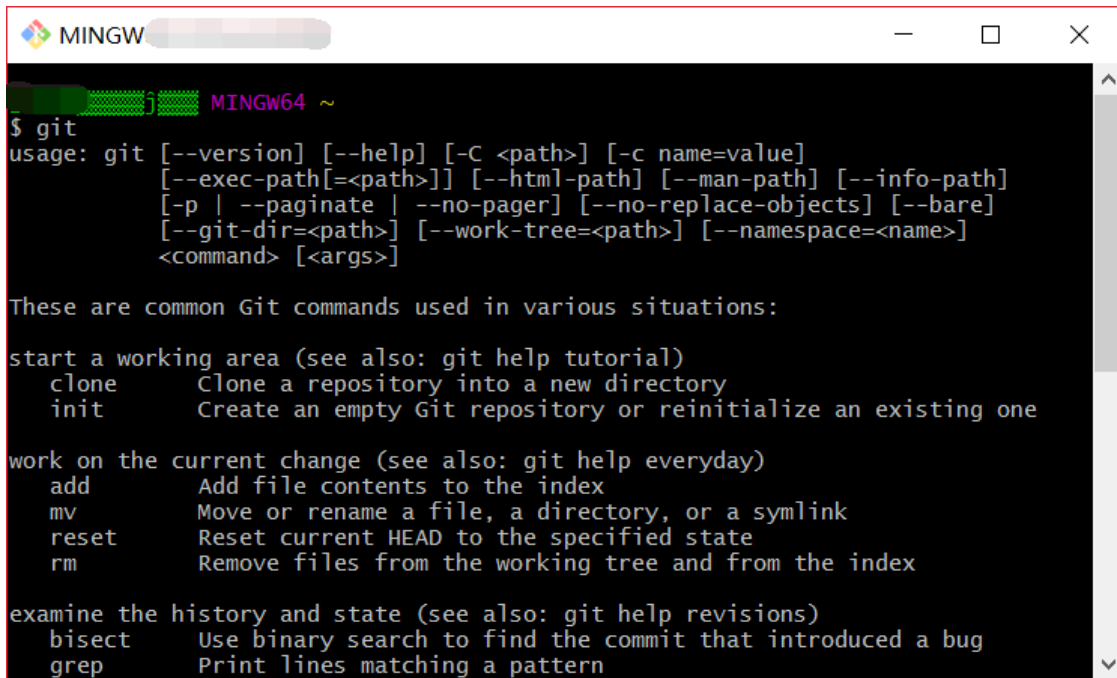


第十一步：安装过程：



第十二步：使用 Git,桌面快捷方式界面如下，打开就可以使用





```
$ 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

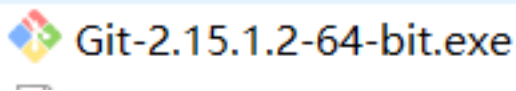

examine the history and state (see also: git help revisions)
    bisect     Use binary search to find the commit that introduced a bug
    grep       Print lines matching a pattern
```

上述过程便是在 windows 环境下安装 git 的大致过程，对于不同的 git 版本安装过程可能会略有差异，但整体步骤均相同。

GitHub 的名字源于 Git，Git 是一个分布式版本控制系统，让程序员团队能够协作开发项目，Git 帮助大家管理为项目所做的工作，避免一个人所做的修改影响其他人所做的修改。你在项目中实现一个新功能的时候，Git 将跟踪你对每个文件所做的修改。确定代码可行后，你将提交所做的修改，而 Git 将记录项目最新的状态，如果你犯了错，想撤销所做的修改，可轻松的返回以前的任何可行状态。GitHub 上的项目都存储在仓库中，后者包含与项目相关联的一切：代码，项目参与者的信息，问题和 bug 报告等

下面讲一下安装 Git

第一步：下载



第二步：安装过程，此处省略

第三步：配置

Git 跟踪谁修改了项目，哪怕参与项目开发的只有一个人。为此，Git 需要知道你的用户名和电子邮件。你必须提供用户名，但可以使用虚构的电子邮件地址：

```
git config --global user.name "Username"
git config --global user.email "Username@example.com"
```

因为 Git 是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和 Email 地址。你也许会担心，如果有人故意冒充别人怎么办？这个不必担心，首先我们相信大家都是善良无知的群众，其次，真的有冒充的也是有办法可查的。

注意 `git config` 命令的 `--global` 参数，用了这个参数，表示你这台机器上所有的 Git 仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和 Email 地址。

第四步：创建项目

我们来创建一个要进行版本控制的项目（又称版本库）。在你的系统创建一个文件夹，并将其命名为 `learngit`。

我创建 `learngit` 的程序如下：

```
mkdir learngit    #创建文件 learngit
cd learngit       #进入 learngit 文件里面
pwd               #显示 learngit 的存在目录
```

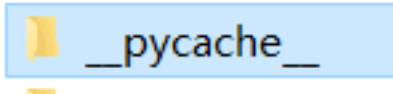
我创建了一个 `hello.world.py` 程序，如下：

```
print("hello world")
```

第五步：忽略文件

忽略扩展名为.pyc 的文件，它是根据.py 文件自动生成啊，我们无需让 Git 去跟踪。这些文件存储在目录 **__pycache__** 中，未来让 git 忽略它，我们创建一个名为.gitignore 的特殊文件，（这个文件是以局点打头，没有扩展名，并且让在其中添加下面一行内容）

```
__pycache__/
```



第六步：初始化仓库

我们创建了一个文件，并通过 `git init` 命令把这个目录变成 Git 可以管理的仓库：

```
$ git init
Initialized empty Git repository in C:/Users/learngit/.git/
```

瞬间 Git 就把仓库建好了，而且告诉你是一个空的仓库（empty Git repository），细心的读者可以发现当前目录下多了一个.git 的目录，这个目录是 Git 来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把 Git 仓库给破坏了。要是删除这个东西，则丢弃项目的所有记录。

如果你没有看到.git 目录，那是因为这个目录默认是隐藏的，用 `ls -ah` 命令就可以看见。

也不一定必须在空目录下创建 Git 仓库，选择一个已经有东西的目录也是可以的。不过，不建议你使用自己正在开发的公司项目来学习 Git，否则造成的一切后果概不负责。

第七步：检查状态

在执行其他操作之前，先来看一下状态：

```
$ git status
On branch master

No commits yet

Untracked files:
```

(use "git add <file>..." to include in what will be committed)

```
hello_world.py
```

nothing added to commit but untracked files present (use "git add" to track)

在 Git 中，分支是项目的一个版本，从这里的输出我们可以知道，我们位于分支的，master

我们每次查看项目的状态时候，输出的都是我们位于分支 master 上，接下来的输出表明，我们将进行初始项目的日叫，**提交**是项目在特定时间的快照。

Git 指出了项目中未被跟踪的文件，因为我们还没有告诉他要跟踪那些文件，接下来我们被告知没有任何东西添加到当前提交里面，但我们可能需要将为跟踪的文件加入仓库

第八步：将文件加入到仓库

```
$ git add .
```

```
...@j MINGW64 ~/learngit (master)
```

```
$ git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   hello_world.py
```

命令 git add. 将项目中未被跟踪的文件都加入到仓库中，它不提交这些文件，而只是让 git 开始关注他们。现在我们检查这个项目的状态，发现 Git 找到了需要提交的文件的一些修改，标签 new file 表示这些文件是新加入的。

第九步：执行提交

```
$ git commit -m "Started project"
```

```
[master (root-commit) 5d6ceca] Started project
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 hello_world.py
```

我们在执行 `git commit -m "Started project"` 的时候以拍摄项目的快照。标志 `-m` 让 Git 接下里的消息（“Started project”）记录到项目中的历史记录中，输出表明我们在分支 `master` 上，而且有一个文件被修改了

简单解释一下 `git commit` 命令，`-m` 后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

第十步：查看提交历史

```
$ git log
commit 5d6cecad80427924b94b14c6fd2bb82a4fa86840 (HEAD -> master)
Author: username <xxxxxxxxxx.example.com>
Date:   Sat Dec 9 20:16:17 2017 +0800
```

Started project

我们每次提交的时候，Git 都会生成一个包含 40 字符的独一无二的引用 ID，它记录提交是谁执行的，提交的时间以及提交的指定消息，并非在任何情况下你都需要所有的这些信息，因此 Git 提供一个选项，让我们能够打印提交历史条目的更简单的版本。

```
$ git log --pretty=oneline
5d6cecad80427924b94b14c6fd2bb82a4fa86840 (HEAD -> master) Started proje
ct
```

标志 `--pretty=oneline` 指定显示一项最重要的信息，提交的引用 ID 以及为提交记录的消息。

第十一步：第二次提交

为了显示版本控制的强大，我们需要对项目进行修改，并提交所做的修改。为此，我们在文件 `hello.world.py` 中再添加一行代码。

```
print("hello world")
print("hello git")
```

如果我们查看项目的状态，将发现 Git 注意到这个文件的变化

```
$ git status
On branch 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 direct
ory)
```

```
modified:  hello_world.py
```

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

这个之处了我们当前所在的分支为 `master`,其中做出修改的文件是 `hello_world.py`,而且指出所做的修改还没有提交。

接下来我们提交所做操作,并在查看操作。

这一步,我们执行了提交,并且在执行命令 `git commit` 的时候指定了标志`-am`.标志`-a` 让 `Git` 将仓库中所有修改了的文件都加入当前提交中,(如果我们两次提交之间加入了新文件,我们执行 `get add`.操作,将新文件加入到仓库中)标志`-m` 让 `Git` 咱提交历史中记录一条消息。

```
$ git commit -am "Extrended greeting."
[master b4ee15d] Extrended greeting.
1 file changed, 2 insertions(+), 1 deletion(-)
$ git status
On branch master
nothing to commit, working tree clean
$ git log --pretty=oneline
b4ee15ddf5274f488db8c74c327065c6f331ec5e (HEAD -> master) Extrended gre
eting.
5d6cecad80427924b94b14c6fd2bb82a4fa86840 Started project
```

我们在查看项目的状态的时候,发现工作目录也是干净的,最后我们发现提交历史中包含两个提交。

第十二步: 撤销修改

下面来看看如何放弃所做的修改,恢复到一个可行状态,为此,我们首先在 `hello_world.py` 中添加一行代码

```
hello_world.py

print("hello world")
print("hello git")

print("the world is bad")
```

保存并运行这个文件

我们查看状态,发现 `Git` 注意到所做的修改

```
$ git status
On branch 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:   hello_world.py
```

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

Git 注意到我们修改了 `hello_world.py`，我可以提交所做的修改，但是我们不提交所做的修改，而要恢复到最后一个提交，为此我们不对 `hello_world.py` 执行任何操作——不删除刚添加的代码行，也不使用文本编辑器的撤销功能，而是在终端会话中执行如下命令：

```
$ git checkout .
```

命令 `git checkout` 能够让我们恢复到以前的任何提交。命令 `git checkout` 放弃最后一次提交所做的所有操作，将项目恢复到最后一次提交的状态。

```
$ git status
On branch master
nothing to commit, working tree clean
```

就这个项目而言，我们恢复到前一个状态微不足道，但是如果我们开发的是大型项目，其中数十个文件都被修改了，那么恢复到前一个状态，将撤销来自最后一次提交的对这个文件所做的所有修改，这个功能很有用，比如：实现新功能，我们可以根据需要做任意数量的修改，如果这些修改都不行，可以撤销他们，而不会对项目有任何伤害，你无需记住做了那些修改，因而不必手工撤销所做的修改，Git 会替我们完成所有的工作。

第十三步：检查以前的提交

我们可以检查提交历史中的任何一次提交，而不仅仅是最后一次，为此我们可以在命令 `git check` 末尾指定该提交的引用 ID 的前 6 个字符（而不是局点）。通过检查出以前的提交，我们可以对其进行审核么然后返回到最后一次提交，或者放弃最近所做的工作，并选择以前的提交。

```
$ git log --pretty=oneline
b4ee15ddf5274f488db8c74c327065c6f331ec5e (HEAD -> master) Extended gre
eting.
5d6cecad80427924b94b14c6fd2bb82a4fa86840 Started project
```

```
$ git checkout 5d6cec
Note: checking out '5d6cec'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-b` with the checkout command again. Example:

```
git checkout -b <new-branch-name>
```

```
HEAD is now at 5d6ceca... Started project
```

检查出以前的提交，我们将离开分支 `master`，并进入 Git 所说的分离头指针（`detached HEAD`）状态，`HEAD` 表示项目的当前状态，之所以说我们处于分离状态是因为我们离开了一个命名分支（这里是 `master`）

要回到分支 `master`，可以检查出它：

```
`$ git checkout master` Previous HEAD position was `5d6ceca`... Started project` Switched to branch `master`
```

这就会让你回到分支 `master`。除非使用 Git 的高级功能，否则在提交以前的项目后，最好不要对项目做任何修改，然而，如果参与项目开发的人只有我们自己，而我自己又想放弃所有提交，并恢复到以前的状态，也可以将项目重置到以前的状态，为此，可在处于分支 `master` 上的任何情况下，执行如下命令。

```
$ git status
On branch master
nothing to commit, working tree clean
$ git log --pretty=oneline
b4ee15ddf5274f488db8c74c327065c6f331ec5e (HEAD -> master) Extended greeting.
5d6cecad80427924b94b14c6fd2bb82a4fa86840 Started project
$ git reset --hard 5d6cec
HEAD is now at 5d6ceca Started project
$ git status
On branch master
nothing to commit, working tree clean
$ git log --pretty=oneline
5d6cecad80427924b94b14c6fd2bb82a4fa86840 (HEAD -> master) Started project
```

首先，我们查看了状态，确认我们在分支 `master` 上，查看历史提交时，我们看见了两个提交。

然后，我们执行了命令 `git reset --hard`，并在其指定了要永久性的恢复到的提交的引用 ID 的前 6 个字符。

接下来，我们在次查看状态，大仙，我们在分支 `master` 上，并且没有需要任何修改，

最后，我们再次查看提交的历史状态时候，我们发现我们处于重新开始的提交中。

第十四步：删除仓库

有时候，仓库的历史纪录被我们搞乱了，而我们又不知道如何恢复，这时候我们首先应该考虑百度一下，看看自己的问题出在那里，如果无法恢复，而且参与项目的人只有自己，可以继续使用这些文件，但需要将这些项目的历史纪录删除——删除 `.git` 这不会影响任何文件的当前状态，而只会删除文件的所有提交，因此我们将无法检查出项目的其他任何状态。

为此，可以打开一个文件浏览器，并将目录 `.git` 删除，也可以通过命令完成这个任务。但是这样做过我们需要创建一个新的仓库，以重新对这些修改进行跟踪。

下面演示了如何在终端会话中完成这个过程。

```
$ git status
On branch master
nothing to commit, working tree clean
$ rm -rf .git
rm -rf .git 表示删除目录.git,删除后我们将继续查看状态。
$ git status
fatal: Not a git repository (or any of the parent directories): .git
```

这个意思是被告知我们，这不是一个仓库，（`git` 用来跟踪仓库的信息都存储爱文件夹 `.git` 中，因此删除该文件夹也将会删除整个仓库）

接下来，我们用命令 `git init` 来创建一个新的仓库，并查看状态

```
$ git init
Initialized empty Git repository in C:/Users//learngit/.git/
```

```
$ git status
On branch master
```

```
No commits yet
```

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
hello_world.py
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

从状态中我们发现，又回到了初始状态，等待第一次提交，我们下面将所有的文件都添加到仓库中，并执行第一次提交，最后检查状态，如下：

```
$ git add .
$ git commit -m "Starting over"
[master (root-commit) 81350ab] Starting over
 1 file changed, 1 insertion(+)
 create mode 100644 hello_world.py
$ git status
On branch master
nothing to commit, working tree clean
```

从检查状态我们发现，我们在分支 `master` 上，并且没有任何未提交的修改。

这就是版本控制的基本操作，希望能多多练习，这样我们才能学会版本控制。

- **推荐一个版本控制工具**
- **CODING：**研发管理系统：<https://coding.net/>，以 Git 代码托管与发布为核心，涵盖代码提交、代码审查、Bug 追踪等开发场景，通过任务、文件、Wiki 等工具，获取需求管理、任务追踪、知识库管理等一系列写作功能。基于 Git 的版本控制保障了公司代码资产安全，实现自动构建，减少人工干预，提升业务系统质量。

GitHub 简介

GitHub 是用于版本控制和协作的代码托管平台，它可以让您和其他人在任何地方协同工作。GitHub 可以托管各种 Git 版本库，并提供一个 web 界面，但与其它像 SourceForge 或 Google Code 这样的服务不同，GitHub 的独特卖点在于从另外一个项目进行分支的简易性。

为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将 GitHub 称为代码玩家的 MySpace。

一：创建新的 Git 仓库

本文将学习使用 GitHub 基本知识，如存储库，分支，提交和 Pull 请求，我将创建自己的 hello world 存储库并学习 GitHub 的 Pull Request 工作流，这是一种创建和检查代码的流行方法。

1：创建存储库

一个库通常用于举办单个项目，存储库可以包含文件夹和文件，图像，视频，电子表格和数据集等等，你的项目需要的任何内容，我们认为包括 README 或者包含项目信息的文件。GitHub 可以在创建新存储库的同时轻松添加一个。

如下图所示：我们可以添加一个新的仓库，并且添加描述，最后单击 **Create repository**。

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

king-jian ▾

/

Repository name

hello-world ✓

Great repository names are short and memorable. Need inspiration? How about `scaling-octo-pancake`.

Description (optional)

It is just is a practice code 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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾

?

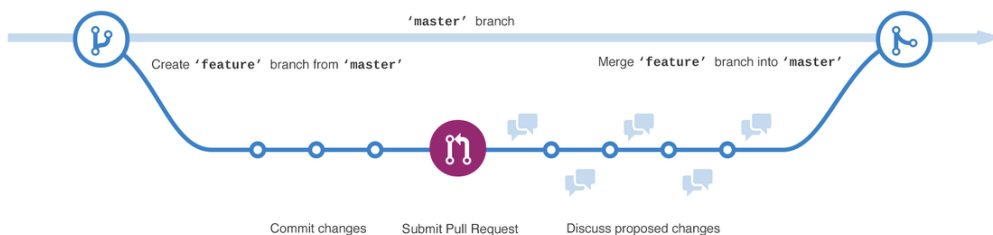
Create repository

2: 创建一个分支 (Create a Branch)

分支是一次处理不同版本的存储库的方法。

默认情况下，我们的存储库有一个名为 **master** 的分支 **branch**，该分支被认为是权威分支。我们在使用分支进行试验并在提交之前进行编辑 **master**。

当你在分支机构上创建 **master** 分支时,我们正在制作该 **master** 时间点的副本或者快照，如果其他人 **master** 在我们的分支机构上工作时对 **branch** 做了更改，则可以提取这些更新。

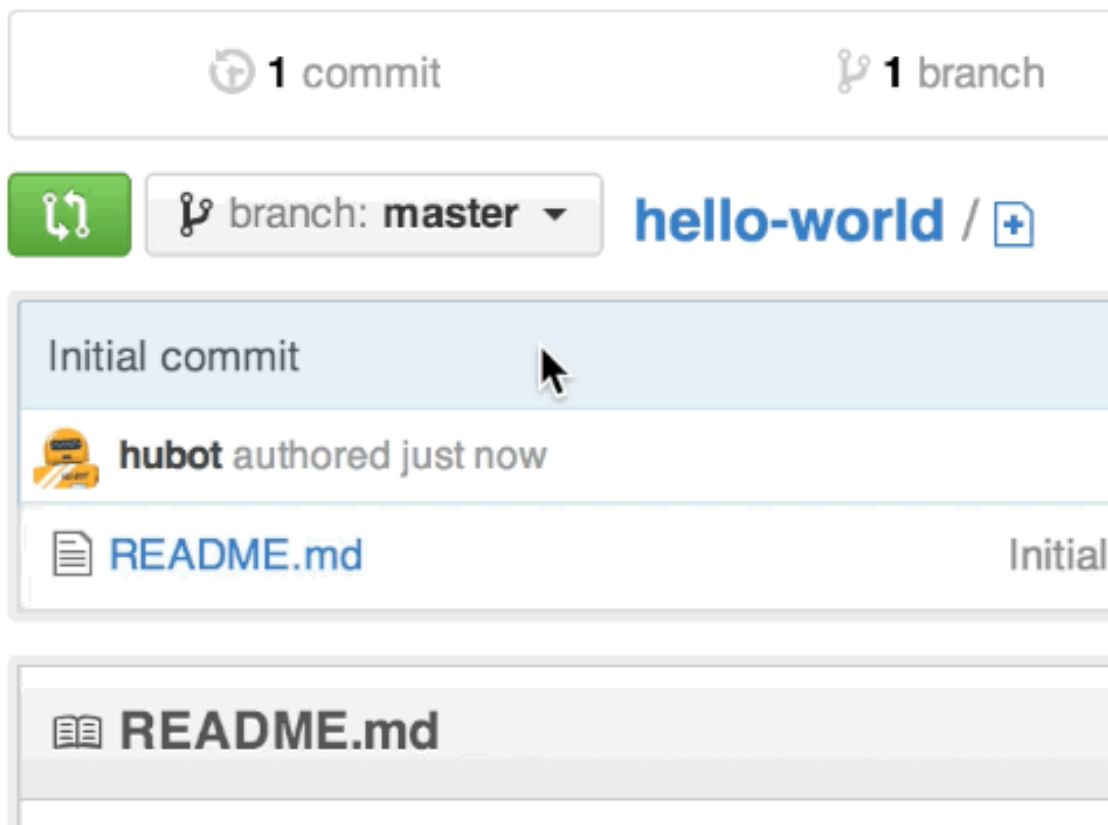


在 GitHub，我们的开发人员，编写人员和设计人员使用分支来保持错误修复和功能工作与我们的 `master` 分支分开，当更改准备就绪时候，他们讲其分支合并到 `master`。

3: 创建一个新分支

- 1，转到新的仓库，也就是我们的仓库 `hello-world`
- 2，单击文件列表顶部的下拉列表：`master`
- 3，在新分支文本框中输入分支名称 `readme-edits`
- 4，选择蓝色的创建分支框或者按键盘的 `Enter`

Just another repository — Edit



The screenshot shows the GitHub interface for a repository named 'hello-world'. At the top, it indicates '1 commit' and '1 branch'. Below this, there's a green button with a fork icon, a dropdown menu showing 'branch: master', and the repository name 'hello-world' with a plus icon. The main content area shows 'Initial commit' with a mouse cursor hovering over it. Below the commit message, it says 'hubot authored just now'. A file named 'README.md' is listed with the word 'Initial' to its right. At the bottom, there's a section for 'README.md' with a book icon.

现在我们有二个分支，`master` 和 `readme-edits`，他们看起来一样，但是时间不会很长，接下来我们增加我们的改变在新的分支上。

4: 制作并提交更改

现在我们在 `readme-edits` 分支的代码视图中，这是一个 `master` 的副本，让我们做一些修改吧。

- 1, 单击该 `README.md` 文件
- 2, 点击要编辑的文件上糊涂右上角的铅笔图标
- 3, 在编译器中，写一点东西
- 4, 编写描述更改的提交消息
- 5, 单击提交按钮

The screenshot shows the GitHub web interface for a repository named 'king-jian / hello-world'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. The 'Code' tab is selected, and the file 'README.md' is open for editing. The editor shows the following content:

```
1 # hello-world
2 It is just is a practice code repository
3
4 hello everybody!
5 today i try to use GitHub, i hope you will give me some advice!
```

Below the editor is a 'Commit changes' dialog box. It has a text input field for the commit message with the value 'Finish README'. Below this is a larger text area for an optional extended description. At the bottom of the dialog, there are two radio buttons: the first is selected and labeled 'Commit directly to the `readme-edits` branch.', and the second is labeled 'Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)'. At the very bottom of the dialog are two buttons: 'Commit changes' (green) and 'Cancel' (red).

5: 打开 Pull 请求

`Pull Requests` 是 `GitHub` 上合作的核心，当我们打开拉取请求时候，你提出了更改并请求某人审核并提取我们的贡献，并将其合并到他们的分支中，拉请求显示来自于两个分支的内容的差异。

5.1 点击 Pull Requests

orld

Unwatch 1

Star 0

Fork 0

as 0

Pull requests 0

Wiki

Pulse

Graphs

Settings

hello-world / README.md

Find file

Copy path

DME

710d852 3 minutes ago

5.2 在 Example Comparisons 中比对代码



Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.



EXAMPLE COMPARISONS



readme-edits	7 minutes ago
master@{1day}...master	24 hours ago





5.3 在比较页面查看差异，确保其是我们要提交的内容




Comparing changes


Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).





 base: master  compare: readme-edits ✓ Able to merge. These branches can be automatically merged.

 **Create pull request** Discuss and review the changes in this comparison with others. 

 1 commit  1 file changed  0 commit comments  1 contributor

 Commits on Nov 05, 2018
  king-jian Finish README Verified 89cedff

 Showing 1 changed file with 3 additions and 0 deletions. Unified Split

3  README.md   View 

... .. @@ -1,2 +1,5 @@

1 1 # hello-world

2 2 It is just is a practice code repository

3 +

4 + hello everybody!



5 + today i try to use GitHub, i hope you will give me some advice!



No commit comments for this range





5.4 如果是，点击 **Create pull request**




Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

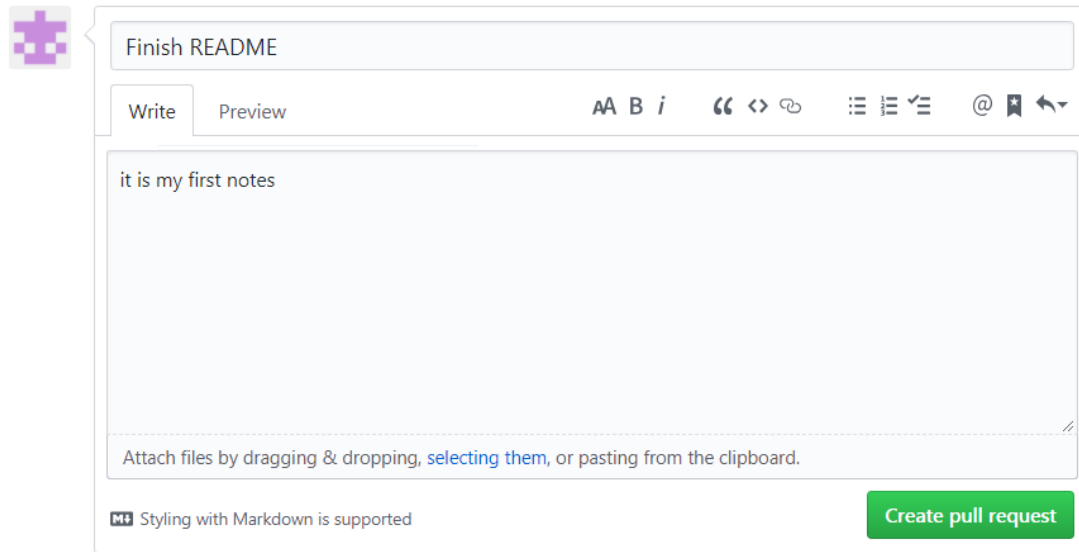
 base: master  compare: readme-edits ✓ Able to merge. These branches can be automatically merged.

 **Create pull request** Discuss and review the changes in this comparison with others. 

 1 commit  1 file changed  0 commit comments  1 contributor

 Commits on Nov 05, 2018
  king-jian Finish README Verified 89cedff

5.5 为我们的拉取请求提供标题，并写下更改的简要说明



Finish README

Write Preview

AA B i “ <> 🔗 ☰ ☷ ☹ @ 📎 ↶

it is my first notes

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

📄 Styling with Markdown is supported


Create pull request


6: 合并我们的 Pull Requests


最后将我们的更改结合在一起，将我们的 `readme-edits` 分支合并到 `master` 分支上。

6.1 单击 Merge pull request，然后单击 Confirm merge

Add more commits by pushing to the `readme-edits` branch on `king-jian/hello-world`.




**Continuous integration has not been set up**
Several [apps](#) are available to automatically catch bugs and enforce style.

**This branch has no conflicts with the base branch**
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).




WritePreview

AA B i “ < > 🔗 ⋮ ≡ ☑ @ 🚩 ↩

Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.


 Styling with Markdown is supported

Close pull request

Comment


💡 ProTip! Add `.patch` or `.diff` to the end of URLs for Git's plaintext views.

6.2 Delete branch



Pull request successfully merged and closed
You're all set—the `readme-edits` branch can be safely deleted.

Delete branch




WritePreview

AA B i “ < > 🔗 ⋮ ≡ ☑ @ 🚩 ↩

Leave a comment

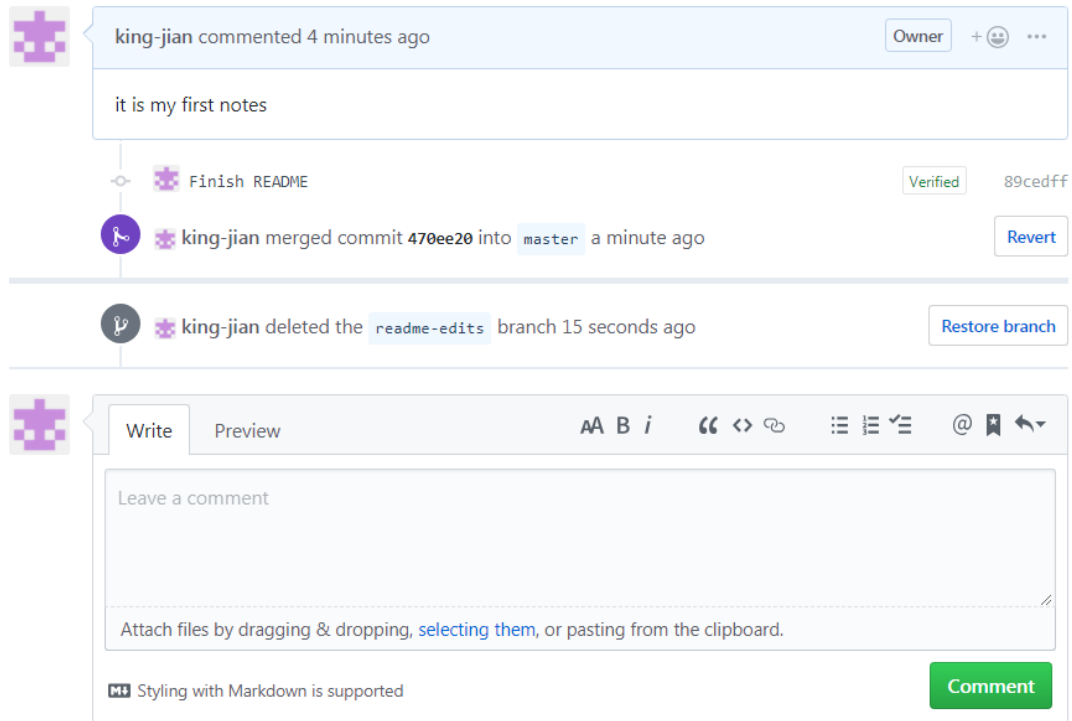
Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

 Styling with Markdown is supported

Comment

💡 ProTip! Add `.patch` or `.diff` to the end of URLs for Git's plaintext views.

6.3 恢复的话点击 **Restore branch**



The screenshot shows a GitHub interface. At the top, a commit by 'king-jian' is shown with the message 'it is my first notes'. Below this, a commit titled 'Finish README' is shown with a 'Verified' badge and hash '89cedff'. Another commit by 'king-jian' is shown with the message 'king-jian merged commit 470ee20 into master a minute ago' and a 'Revert' button. Below this, a commit is shown with the message 'king-jian deleted the readme-edits branch 15 seconds ago' and a 'Restore branch' button. At the bottom, a comment box is shown with a 'Write' tab, a 'Preview' tab, and a 'Comment' button. The comment box contains the text 'Leave a comment' and 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.'.

king-jian commented 4 minutes ago

Owner + 🧑🏻 🍌

it is my first notes

Finish README Verified 89cedff

king-jian merged commit 470ee20 into master a minute ago Revert

king-jian deleted the readme-edits branch 15 seconds ago Restore branch

Write Preview AA B i “ < > 🔗 ☰ ☷ ☹ @ 📌 ↶

Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

📄 Styling with Markdown is supported Comment

💡 ProTip! Add [.patch](#) or [.diff](#) to the end of URLs for Git's plaintext views.

二：如何一步步的在 GitHub 上传自己的项目



1 创建一个新的项目，填写项目名称，描述等

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner


Repository name


 sword-king / learn-chouti-project 

Great repository names are short and memorable. Need inspiration? How about [bookish-eureka](#).

Description (optional)

It is just to learn chouti project

☒  **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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼


Add a license: **None** ▼



Create repository

WWW.11

2 创建完成后，跳转到下面页面

 sword-king / learn-chouti-project

Watch 0Star 0Fork 0

CodeIssues 0Pull requests 0Projects 0WikiInsightsSettings

Quick setup — if you've done this kind of thing before

Set up in DesktoporHTTPSSSHhttps://github.com/sword-king/learn-chouti-project.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# learn-chouti-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/sword-king/learn-chouti-project.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/sword-king/learn-chouti-project.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

那么请记住下面的地址：

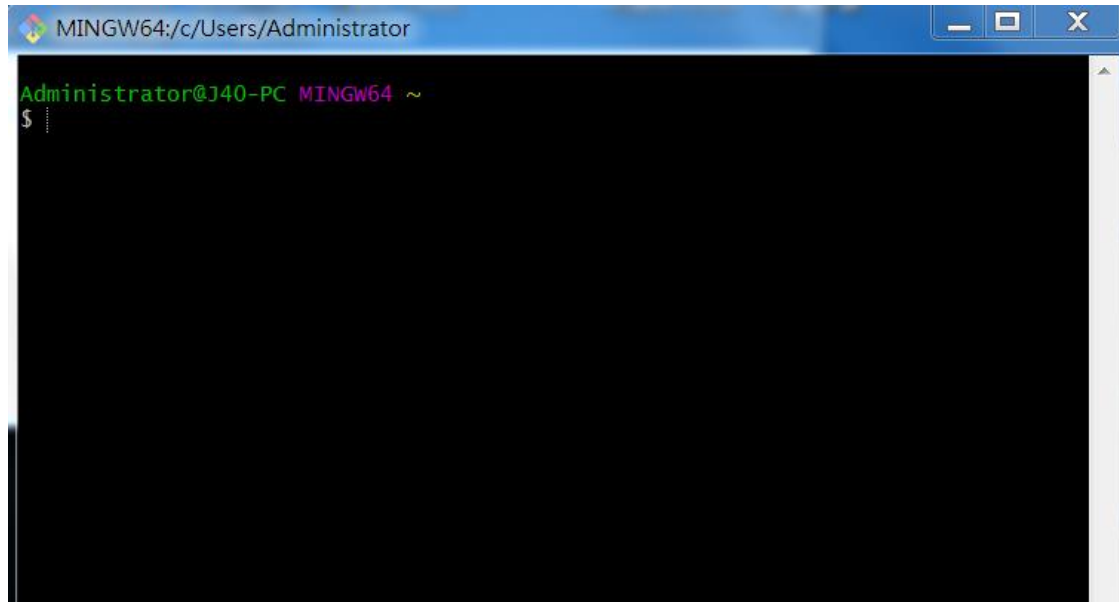
Quick setup — if you've done this kind of thing before

Set up in DesktoporHTTPSSSHhttps://github.com/sword-king/learn-chouti-project.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

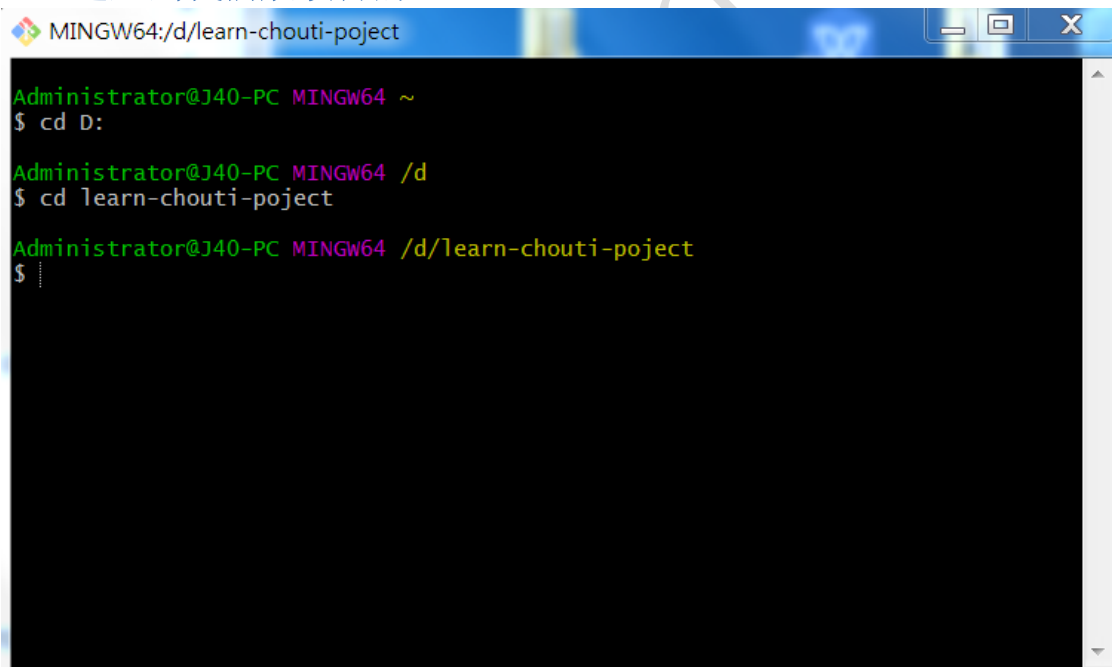
3 我们需要下载 Git，并安装。

4 进入 Git Bash，出现如下界面



```
MINGW64:/c/Users/Administrator
Administrator@J40-PC MINGW64 ~
$
```

5 cd 进入到我们放项目的地址

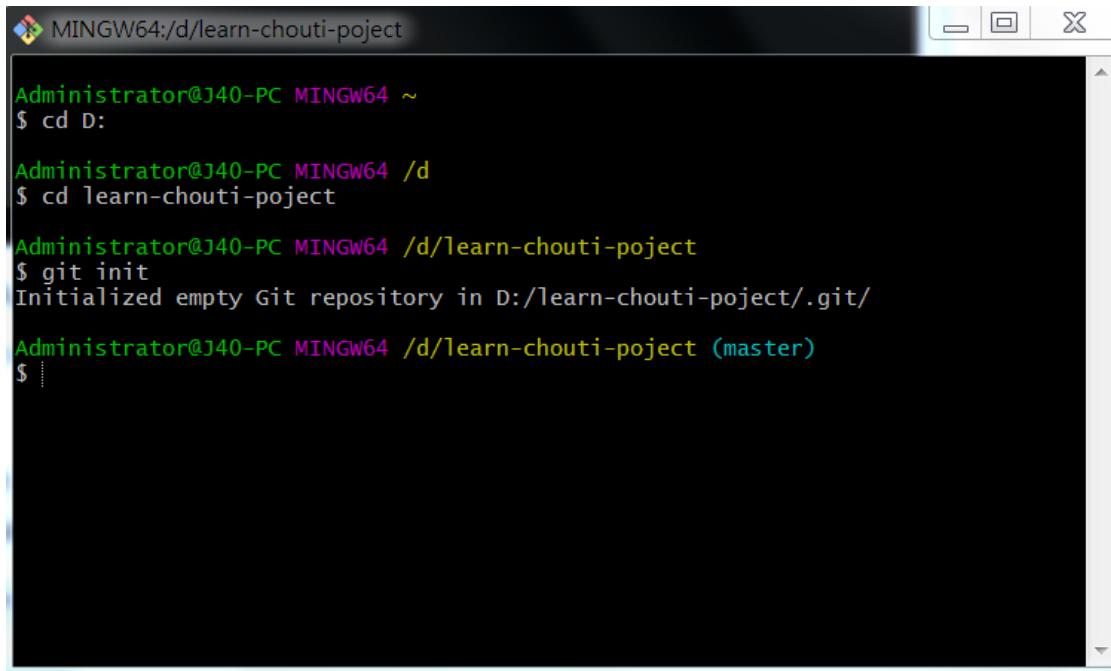


```
MINGW64:/d/learn-chouti-poject
Administrator@J40-PC MINGW64 ~
$ cd D:
Administrator@J40-PC MINGW64 /d
$ cd learn-chouti-poject
Administrator@J40-PC MINGW64 /d/learn-chouti-poject
$
```

6 输入 git init

在当前项目工程下履行这个号令，相当于把当前项目 **git** 化

在当前项目的目录中生成本地的 `git` 管理（我们会发现当前目录下多了一个 `.git` 文件夹）

A screenshot of a Windows command prompt window titled "MINGW64:/d/learn-chouti-poject". The prompt shows the user navigating to the directory and initializing a git repository. The text in the terminal is as follows:

```
Administrator@J40-PC MINGW64 ~  
$ cd D:  
  
Administrator@J40-PC MINGW64 /d  
$ cd learn-chouti-poject  
  
Administrator@J40-PC MINGW64 /d/learn-chouti-poject  
$ git init  
Initialized empty Git repository in D:/learn-chouti-poject/.git/  
  
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)  
$
```

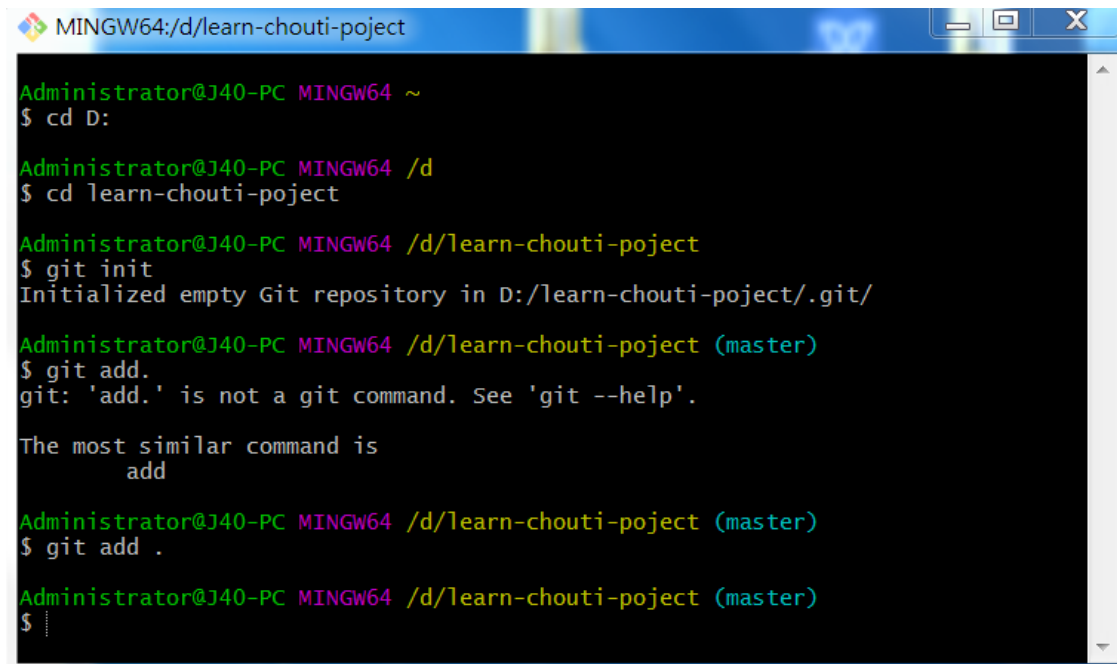


7 输入 `git add .`

把当前目录下代码参加 `git` 的跟踪中，意思就是交给 `git` 经管，提交到本地库

这个是将项目上所有的文件添加到仓库中，如果只想添加某个特定的文件，只需要将.换成特定的名称即可。

（下面会报错，我们发现 `add` 和 `.` 之间有空格）

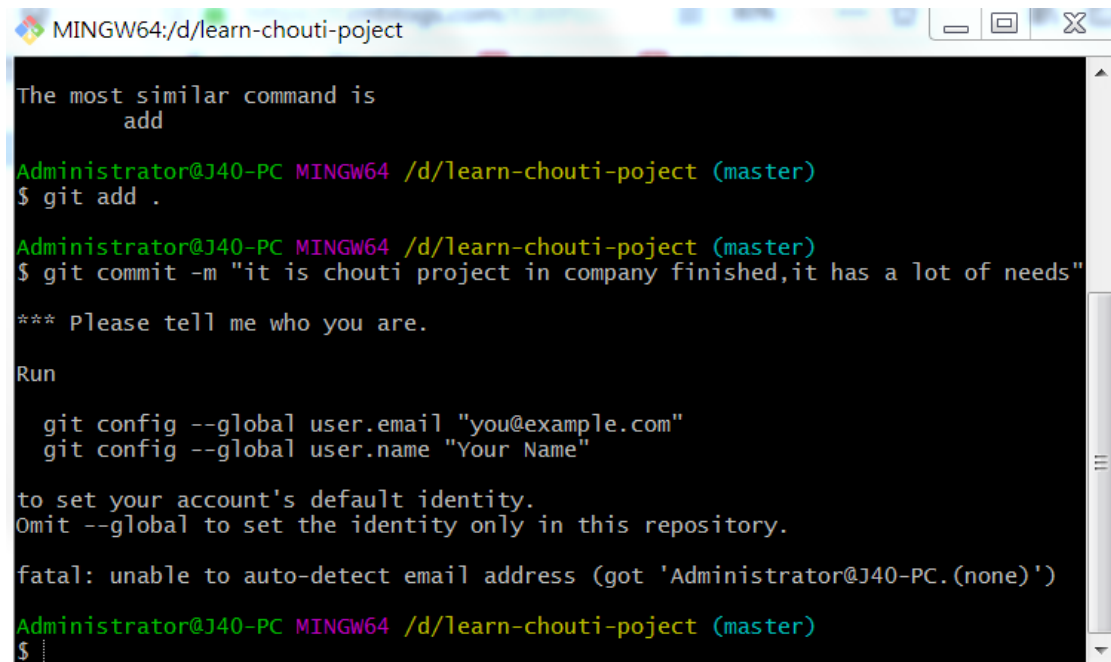


```
MINGW64:/d/learn-chouti-poject
Administrator@J40-PC MINGW64 ~
$ cd D:
Administrator@J40-PC MINGW64 /d
$ cd learn-chouti-poject
Administrator@J40-PC MINGW64 /d/learn-chouti-poject
$ git init
Initialized empty Git repository in D:/learn-chouti-poject/.git/
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git add.
git: 'add.' is not a git command. See 'git --help'.
The most similar command is
    add
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git add .
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$
```

8 输入 `git commit -m "first commit"`

相当于写点提交信息

表示我们对这次提交的注释，双引号里面的内容可以根据个人的需求改



```
MINGW64:/d/learn-chouti-poject

The most similar command is
  add

Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git add .

Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git commit -m "it is chouti project in company finished,it has a lot of needs"

*** Please tell me who you are.

Run

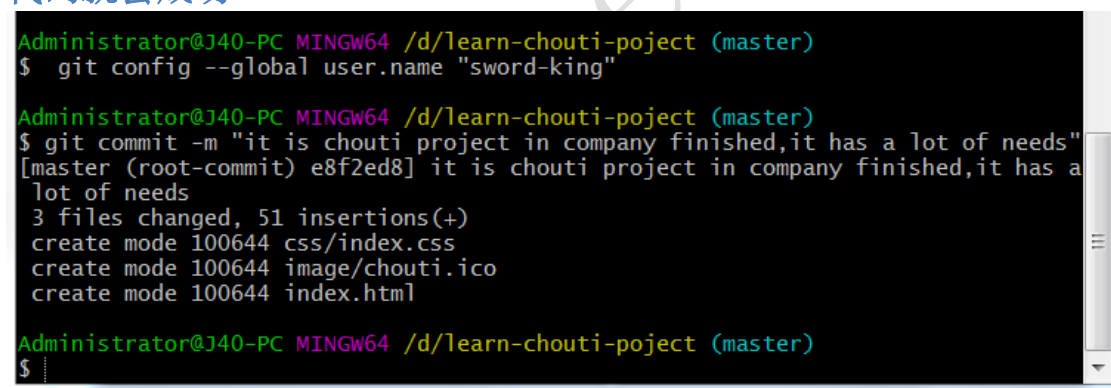
  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Administrator@J40-PC.(none)')

Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$
```

9 出现上面的内容，我们需要输出自己的账号或者名字，再执行上面的代码就会成功



```
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git config --global user.name "sword-king"

Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git commit -m "it is chouti project in company finished,it has a lot of needs"
[master (root-commit) e8f2ed8] it is chouti project in company finished,it has a lot of needs
3 files changed, 51 insertions(+)
create mode 100644 css/index.css
create mode 100644 image/chouti.ico
create mode 100644 index.html

Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$
```


10 关联自己的仓库 url 地址

这里自己找自己的 url 地址

git remote add origin https://自己的仓库 url 地址

下面展示本人的：

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** 

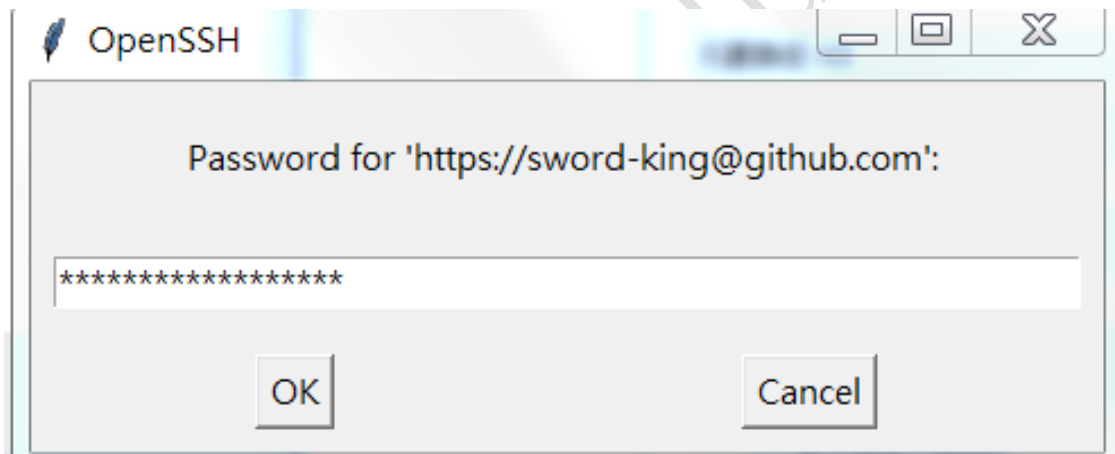
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

11 上传代码 输入 `git push -u origin master`（意思：上传到 GitHub 仓库）

将本地库提交到 `github` 上。

执行完毕后，如果没有异常，会等待几秒，然后跳出一个让我们输入 Username 和 password 的窗口，我们只需要输入个人的 `github` 登录账号和密码即可。

```
Administrator@J40-PC MINGW64 /d/learn-chouti-poject (master)
$ git push -u origin master
Username for 'https://github.com': sword-king
```



最后上传完毕

```
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 1.58 KiB | 808.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
remote: This repository moved. Please use the new location:
remote: https://github.com/sword-king/learn-chouti-projects.git
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote: https://github.com/sword-king/learn-chouti-projects/pull/new/master
remote:
To https://github.com/sword-king/learn-chouti-project.git
* [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$
```

12 上传成功，进入到 GitHub 中查看

sword-king / learn-chouti-projects

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

It is just to learn chouti project

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

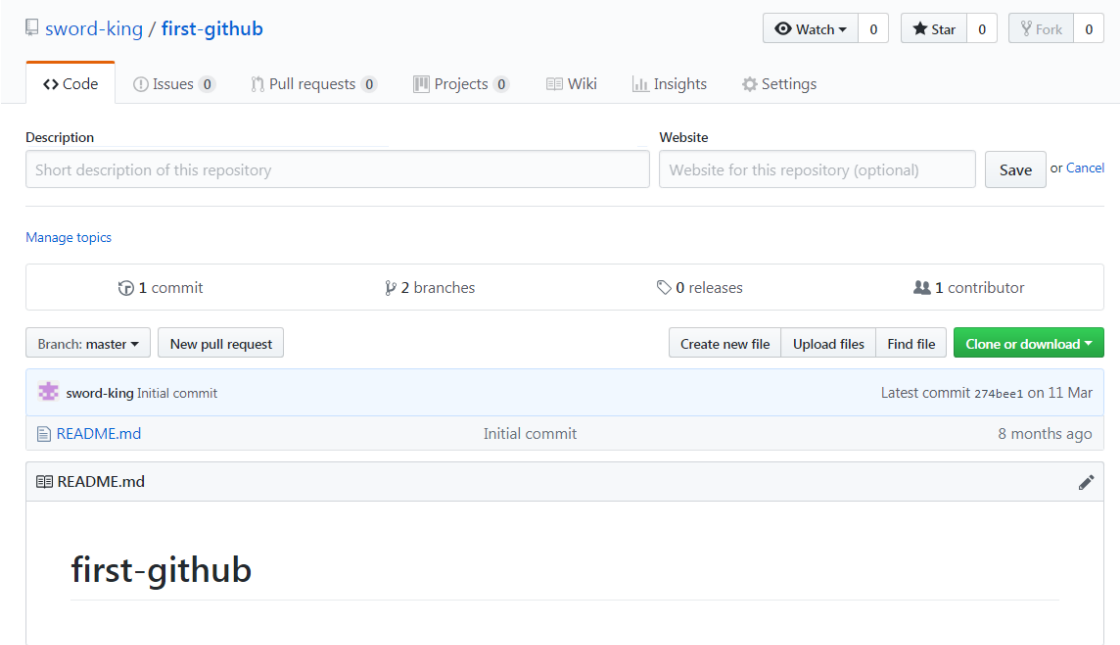
sword-king it is chouti project in company finished,it has a lot of needs Latest commit e8f2ed8 3 hours ago

css	it is chouti project in company finished,it has a lot of needs	3 hours ago
image	it is chouti project in company finished,it has a lot of needs	3 hours ago
index.html	it is chouti project in company finished,it has a lot of needs	3 hours ago

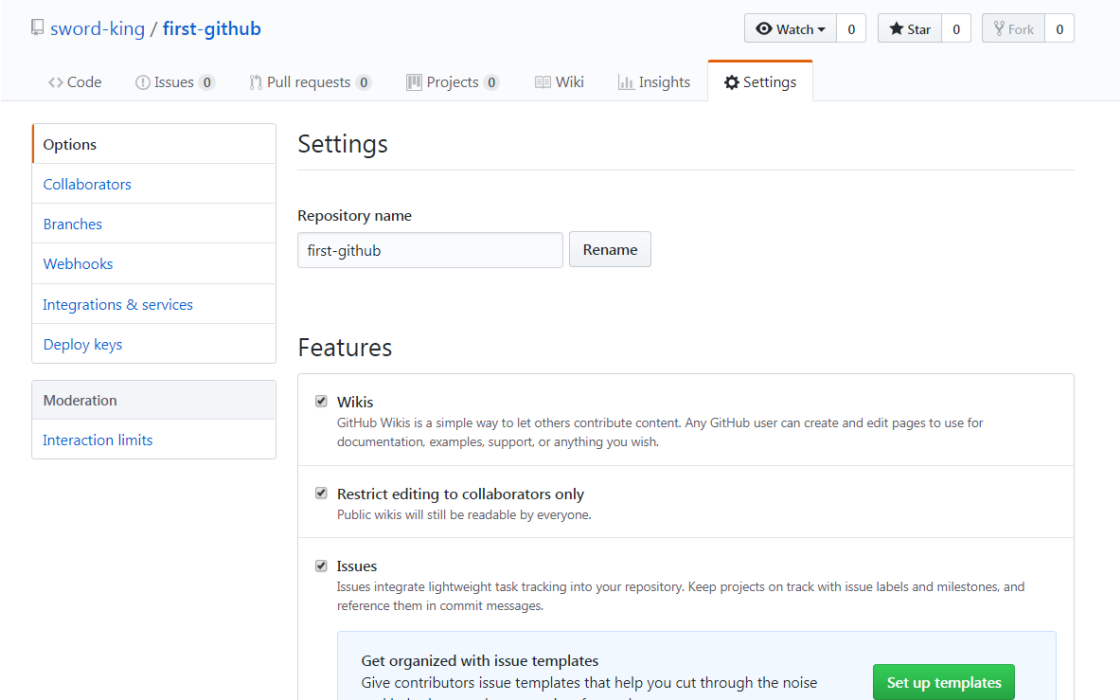
Help people interested in this repository understand your project by adding a README. Add a README

三：GitHub 如何删除项目

1，首先找到需要删除的项目，点开



2，找到 settings，点开



3, 将滚动条滑到底部, 找到 Danger Zone 下的 Delete this repository

Danger Zone

Make this repository private Please upgrade your plan to make this repository private.	
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	<button>Transfer</button>
Archive this repository Mark this repository as archived and read-only.	<button>Archive this repository</button>
Delete this repository Once you delete a repository, there is no going back. Please be certain.	<button>Delete this repository</button>

4, 点击, 会弹出一个警告框, 将项目名称输入进行确认

Are you absolutely sure?

×

Unexpected bad things will happen if you don't read this!


This action cannot be undone. This will permanently delete the `sword-king/first-github` repository, wiki, issues, and comments, and remove all collaborator associations.

Please type in the name of the repository to confirm.

first-github

I understand the consequences, delete this repository

5，这里会弹出账号重新进行确认，输入密码进行确认即可。



Confirm password to continue

Password [Forgot password?](#)

Confirm password

Tip: You are entering [sudo mode](#). We won't ask for your password again for a few hours.

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

6，删除成功后，会重新回到个人主界面提醒项目删除成功

Your repository "sword-king/first-github" was successfully deleted.

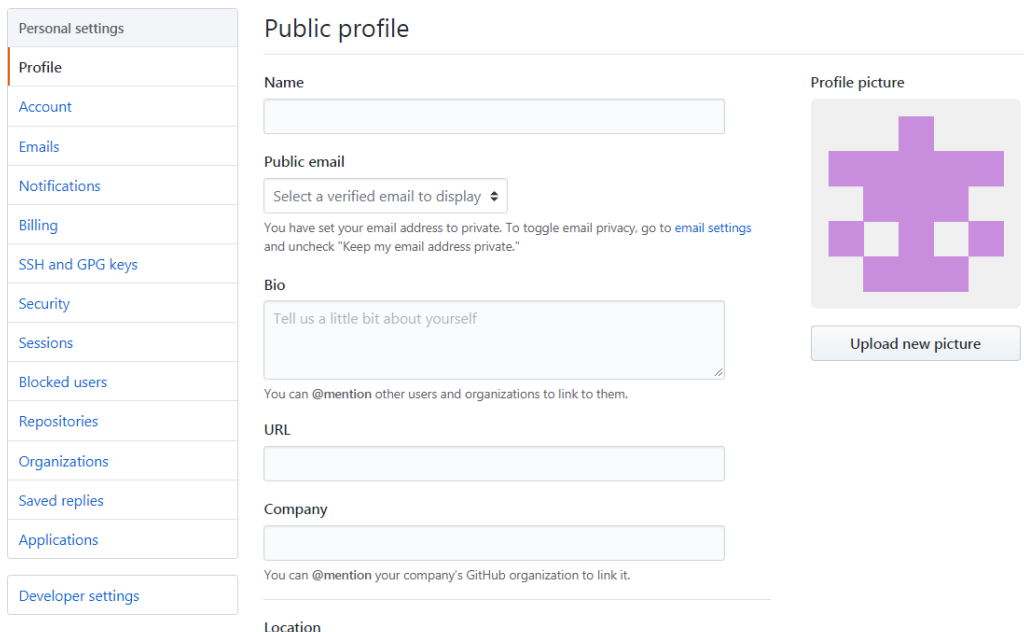
四，报错 `push declined due to email privacy restrictions` 的解决方法

当你上传代码到最后一步，发现无法 `push`，并且会出现如下错误：

```
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 1.58 KiB | 808.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
remote: This repository moved. Please use the new location:
remote:   https://github.com/sword-king/learn-chouti-projects.git
remote: error: GH007: Your push would publish a private email address.
remote: You can make your email public or disable this protection by visiting:
remote: http://github.com/settings/emails
To https://github.com/sword-king/learn-chouti-project.git
! [remote rejected] master -> master (push declined due to email privacy restrictions)
error: failed to push some refs to 'https://github.com/sword-king/learn-chouti-project.git'
```

那么如何解决呢？

1，进入 GitHub 主页，进入 `setting`



Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Sessions
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

Developer settings

Public profile

Name

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, go to [email settings](#) and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

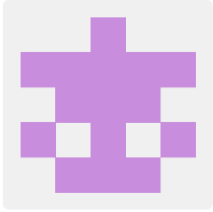
URL

Company

You can @mention your company's GitHub organization to link it.

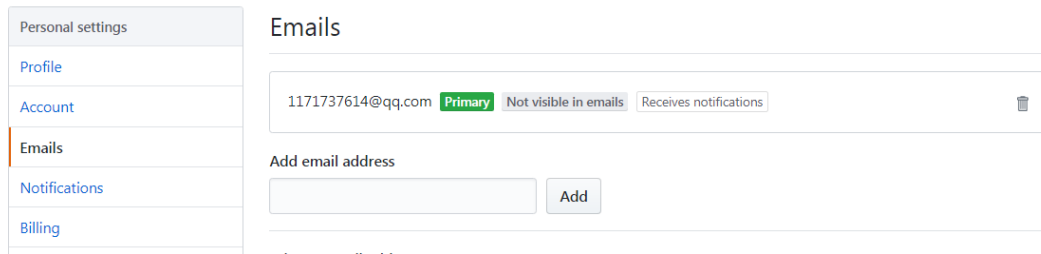
Location

Profile picture



Upload new picture

2, 点击 emails



3, 取消 Block command line pushes that expose my email 的勾即可

五, 报错 error: src refspec master does not match any.的解决方法

上传代码到最后一步, 出现此错误, 如何解决呢? (就是无法匹配 master)

```
$ git push -u origin master
error: src refspec master does not match any.
error: failed to push some refs to 'https://github.com/health-king/DL-algorithm-learn.git'
```

1: 错误产生的原因

引起该错误的原因是, 目录中没有文件, 空目录是不能提交上去的。

2: 解决方法

这个仔细检查, 本地的文件名称是否和 GitHub 上的对应。我的就是因为名字不对应导致的错误, 也就是自己粗心!!!

其二, 就是之前没有使用如下代码:

```
git add .
```

没有对代码进行跟踪, 我找了两天的, 才发现自己每次都少了这一步, 也会报同样的错误, 非常粗心!!!

六, 报错 error: failed to push some refs to 的解决方法

上传代码到最后一步, 出现此错误, 如何解决呢?

```
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$ git push -u origin master
Username for 'https://github.com': health-king
To https://github.com/health-king/chouti-bymyself.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/health-king/chouti-bymyself.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

1: 错误产生的原因

引起该错误的原因是，我们在 GitHub 中对代码进行了在线的修改；或者我们直接在 GitHub 上的某个库照片那个添加了 readme 文件或者其他文件，但是没有对本地库进行同步，所以这时候我们要想 commit 到 remote 的 GitHub 库中就会有 push 失败的问题，

2: 解决方法

这个问题就是因为远程库与本地库不一致造成的，我们只需要把远程库同步到本地库就可以了。指令如下：

```
git pull --rebase origin master
```

git pull --rebase origin master 意为先取消 commit 记录，并且把它们临时保存为补丁(patch)(这些补丁放到“.git/rebase”目录中)，之后同步远程库到本地，最后合并补丁到本地库之中。

如图所示：

```
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$ git pull --rebase origin master
From https://github.com/health-king/chouti-bymyself
 * branch          master      -> FETCH_HEAD
First, rewinding head to replay your work on top of it...
Applying: test
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
```

然后再上传，指令如下：

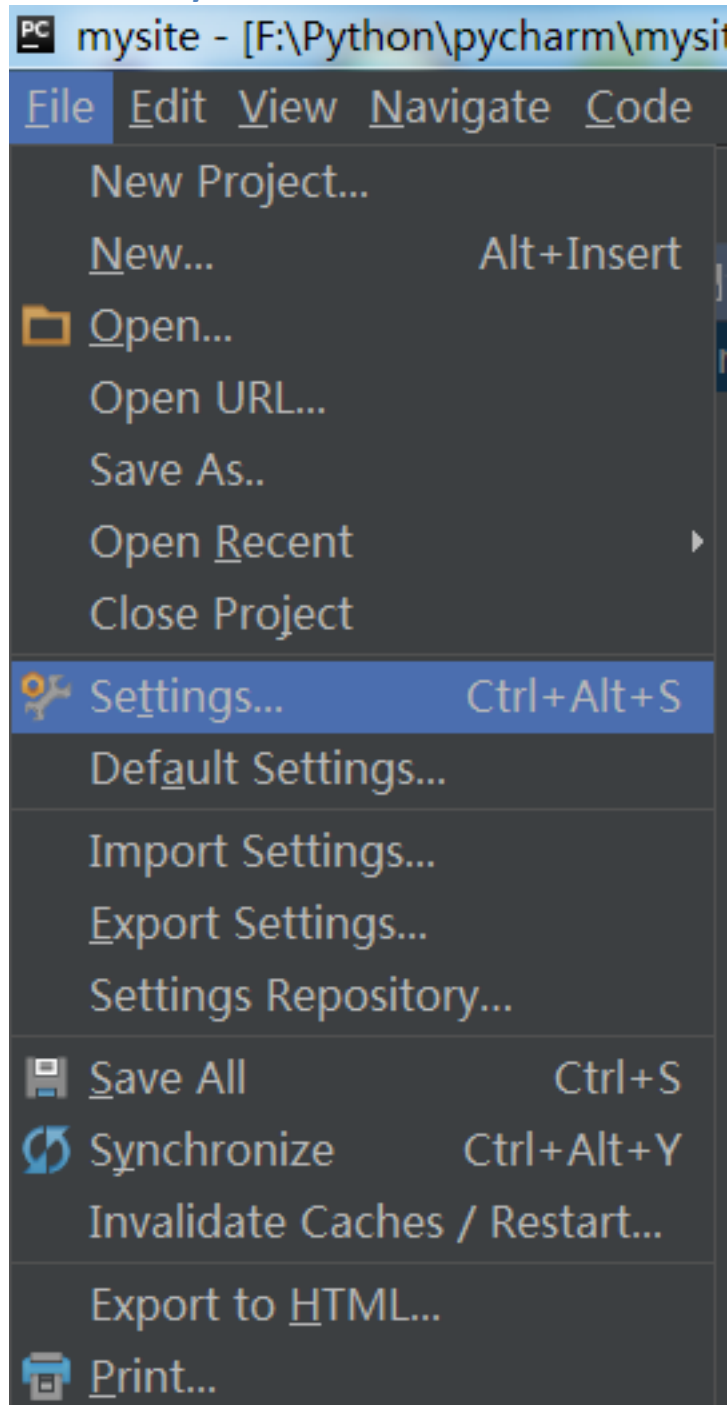
```
git push -u origin master
```

结果如图所示（此时解决问题）：

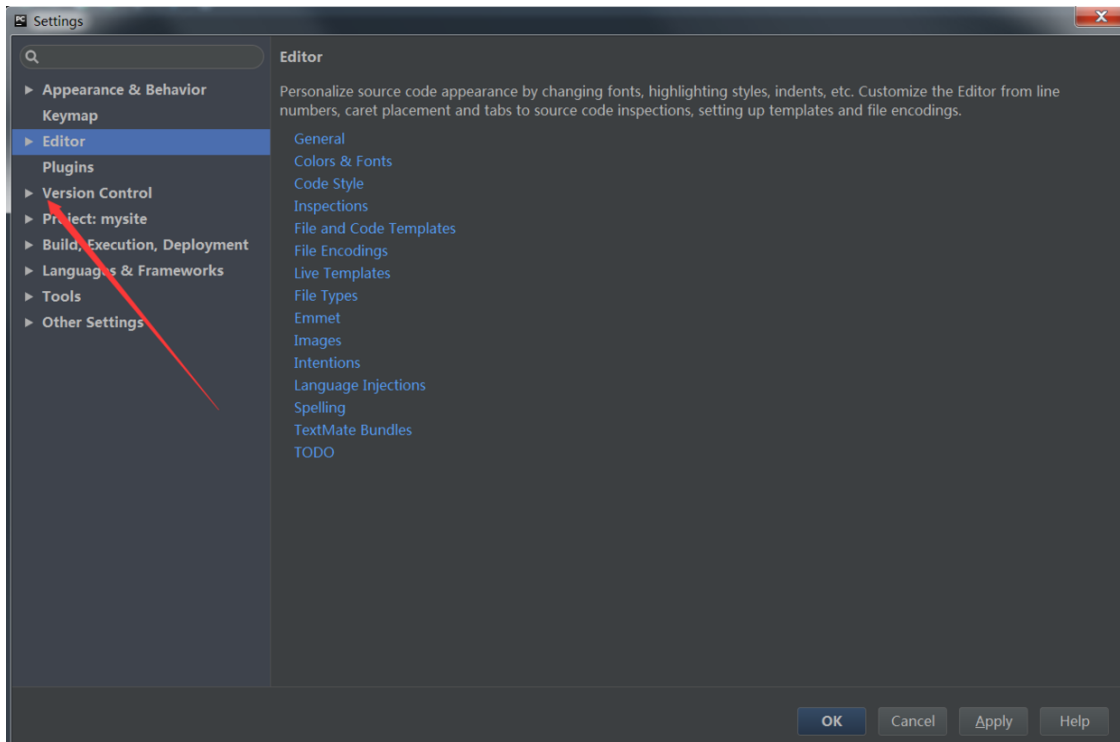
```
Administrator@J40-PC MINGW64 /d/learn-chouti-project (master)
$ git push -u origin master
Username for 'https://github.com': health-king
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 7.02 KiB | 3.51 MiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/health-king/chouti-bymyself.git
   c4b8854..19fbf7a  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

环境: pycharm 2016, git 2.8, github 账户, windows7

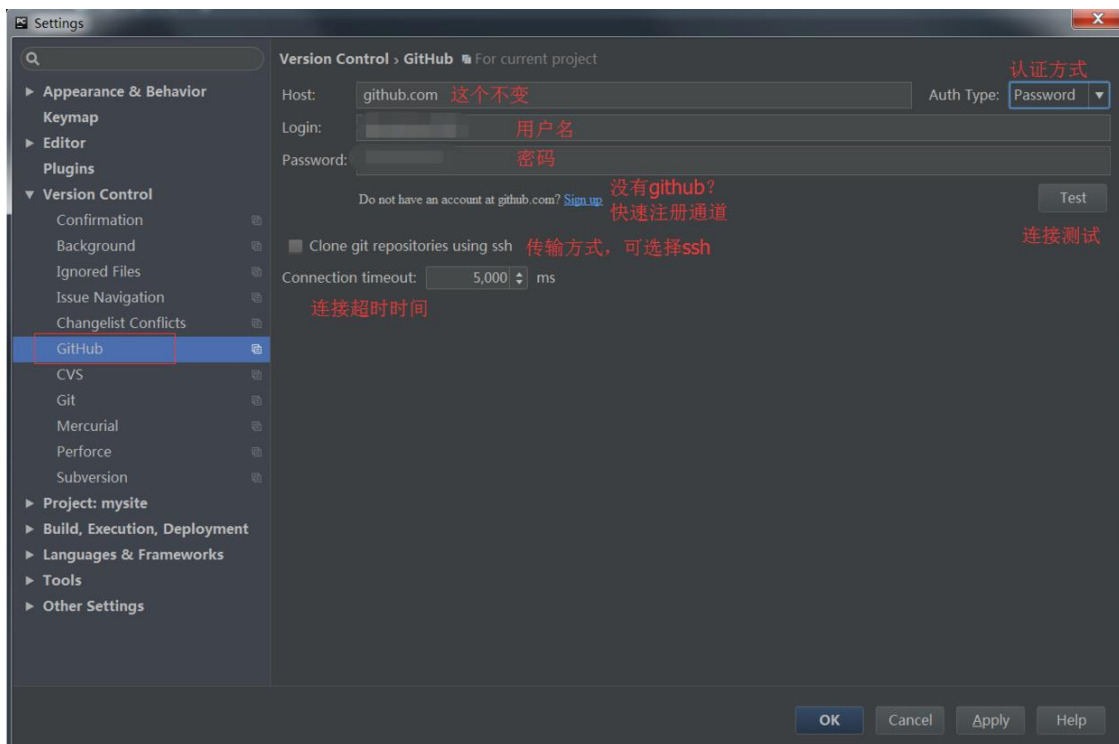
一、配置 Pycharm



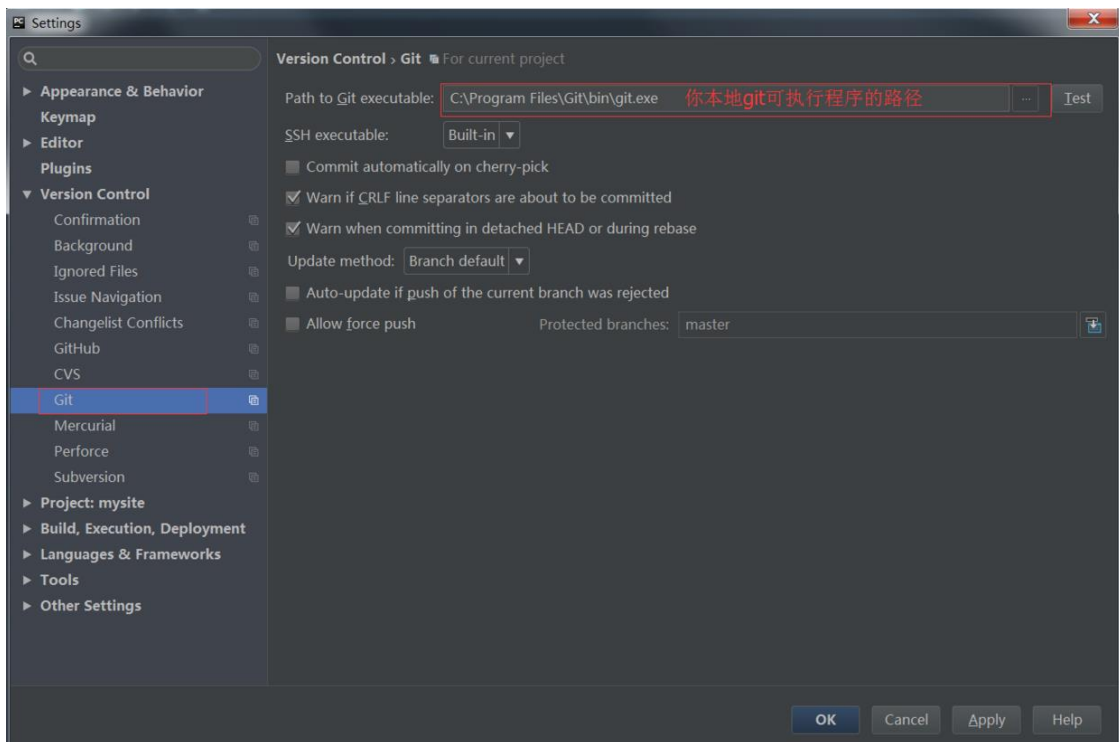
不管你用哪种方法，进入 pycharm 的配置菜单。



选择上图中的 **version control**。（这里插一句，不管有多难，在程序员的世界，请不要汉化，坚持使用英文原版）



按照图中所示，配置好 github 相关内容。没有 github 的同学，建议你自己前往官网注册，而不要通过它的快速注册通道。



这一步很关键，很多同学配置完 github 就直接开始使用了，结果却是各种错误。

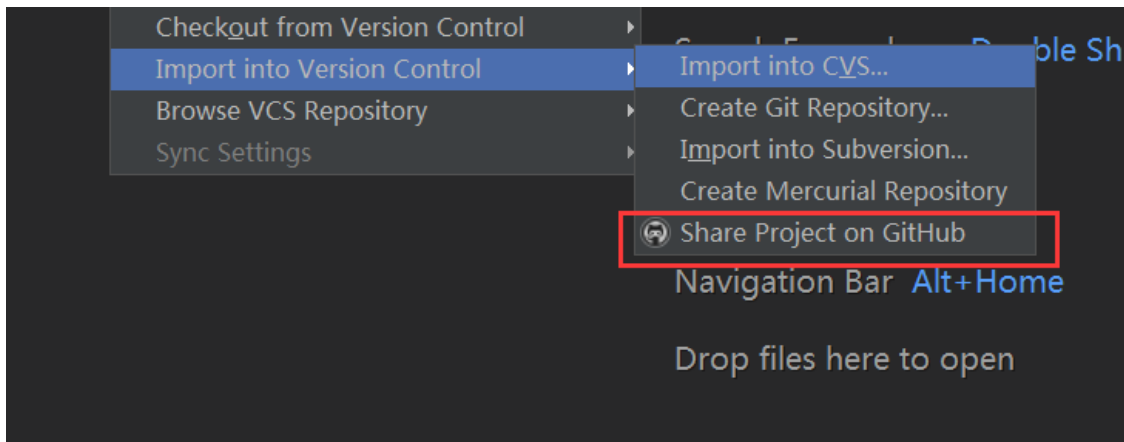
这是因为，除了配置 github 外还要配置 git。github 的配置只是告诉了 pycharm 你的账号和密码，

pycharm 还不知道如何进行版本控制操作，它底层还是需要调用 git 的功能的。

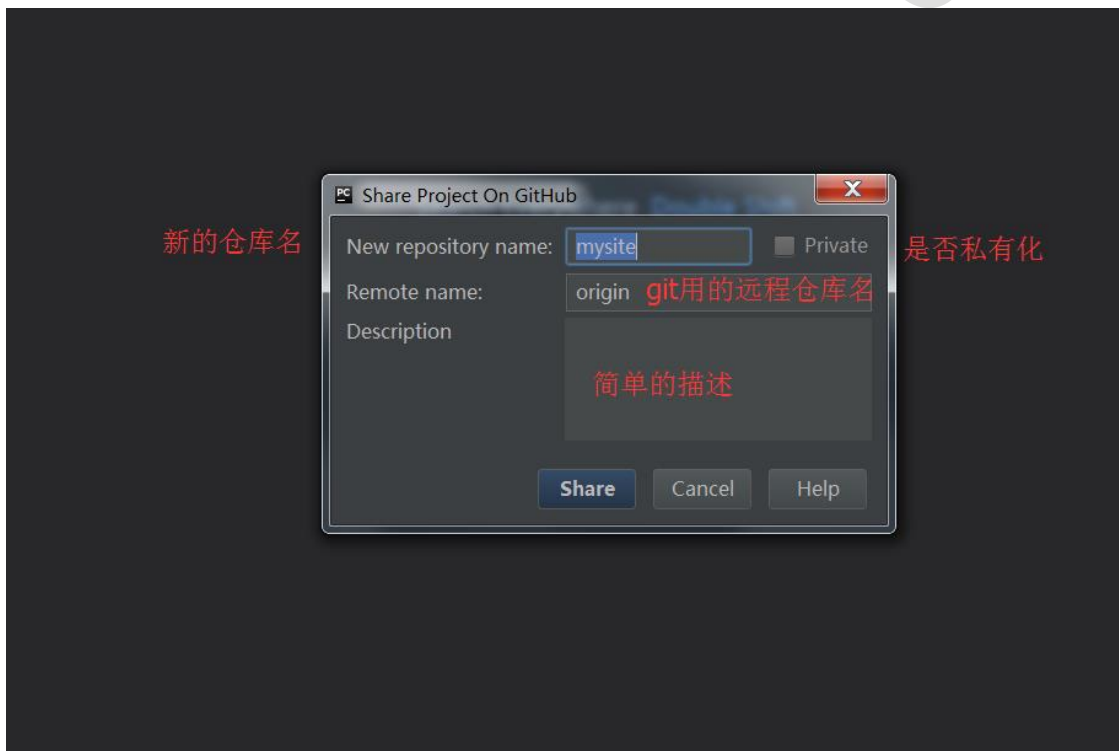
因此，请提前下载并安装 git 程序到你的本机。

二、建立远程仓库并提交代码

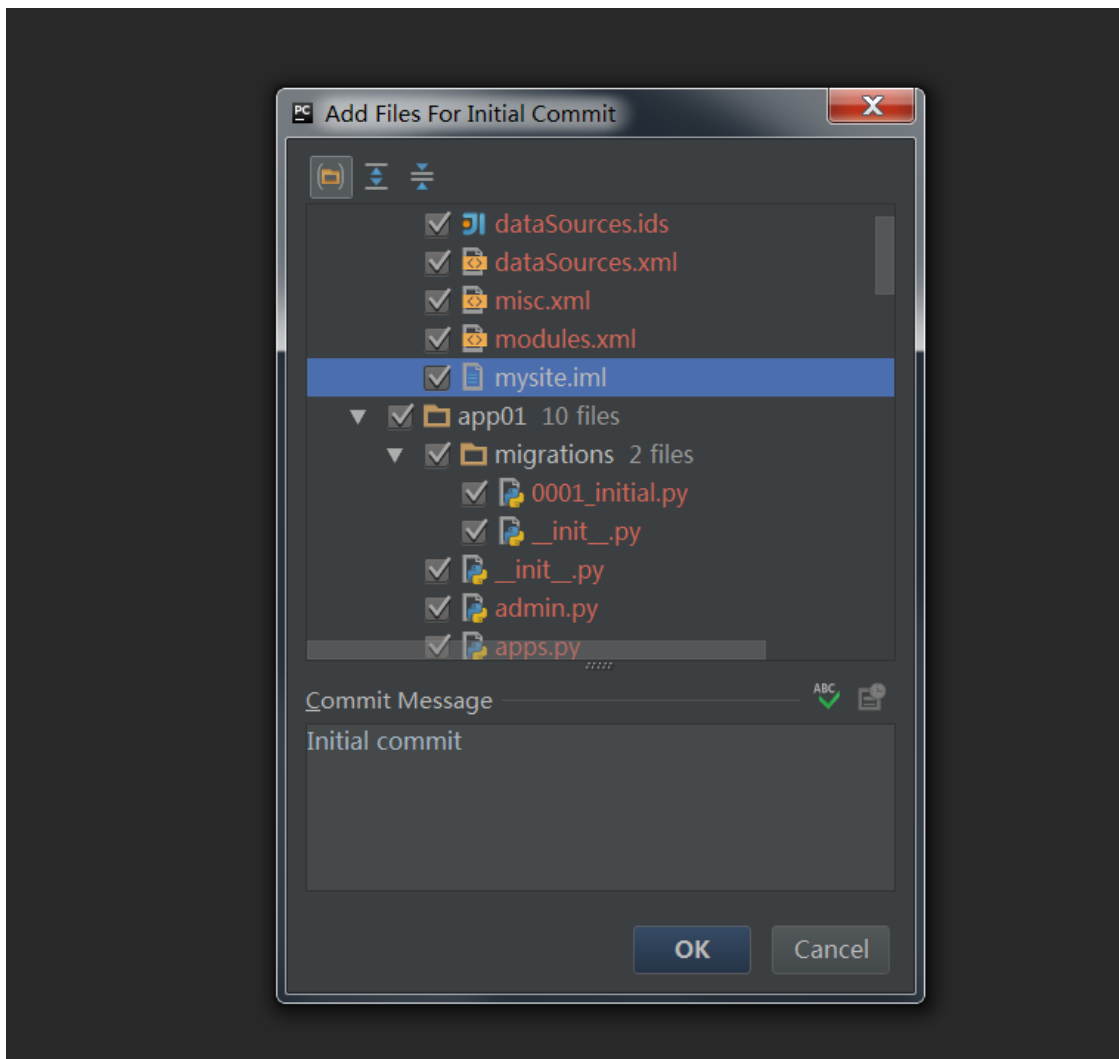
配置完了，就可以正常使用了，点击顶部菜单栏的 VCS 选项。



在 import into version control 下有一个在 github 中共享项目的栏目了，点击进去。



按照图中的注释，建立一个新的仓库，点击 share 按钮。



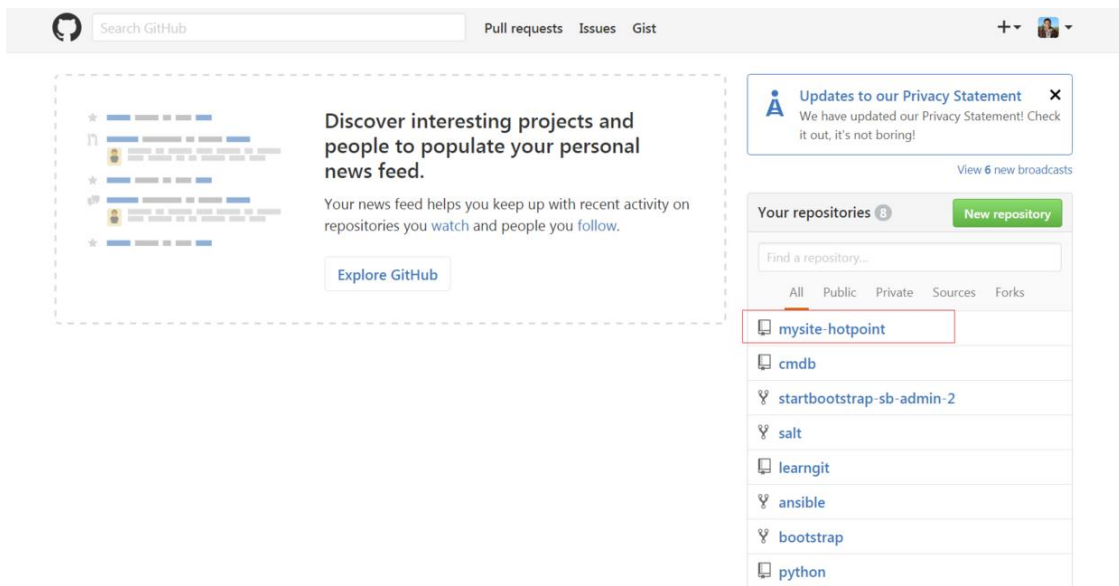
在这里，可以选择你要上传的文件，一般是直接全部上传，当然也可以取消那些不必要的文件。输入提交信息，

点击 OK 确认。等待片刻，根据你的网络情况和文件大小，pycharm 将文件传输给 github，成功后会弹出小的提示

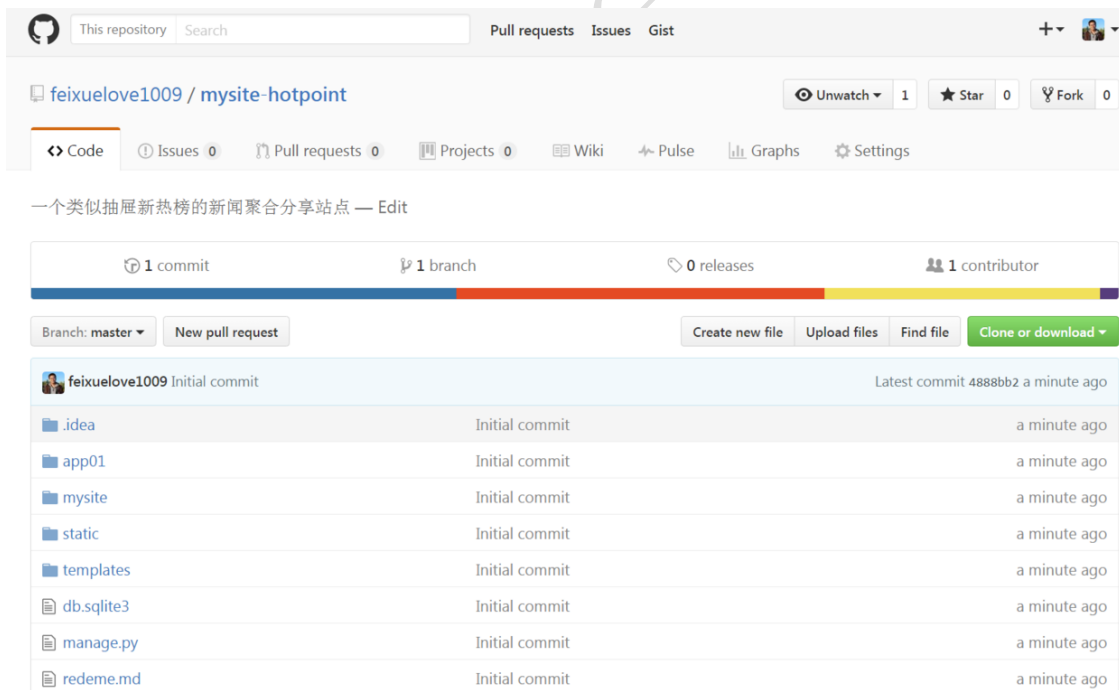
消息，这就大功告成了。很简单吧？！

三、在 github 中查看上传的新仓库

进入 github 官网，登录自己的账号：

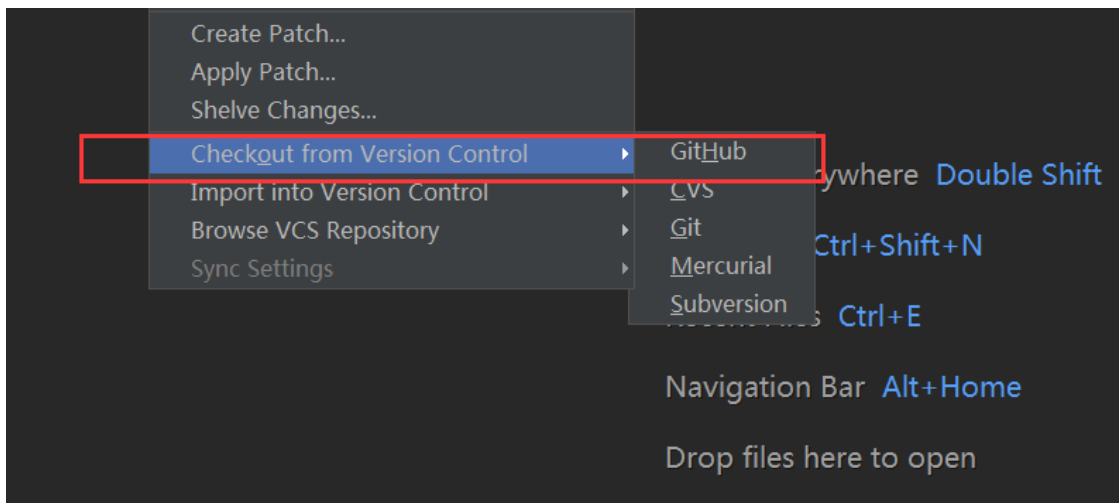


图中红框是我们刚才通过 **pycharm** 建立的新仓库，点击进去，可以看到如下图，整个 **pycharm** 项目的所有文件都在里面：

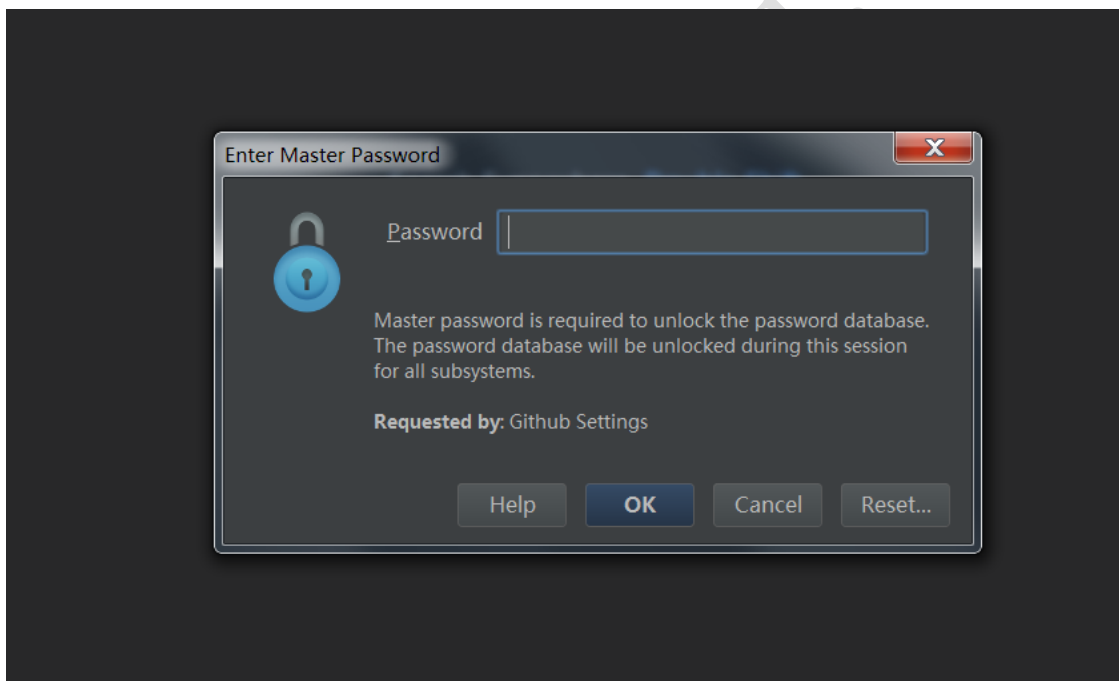


四、使用 **pycharm** 克隆 **github** 仓库

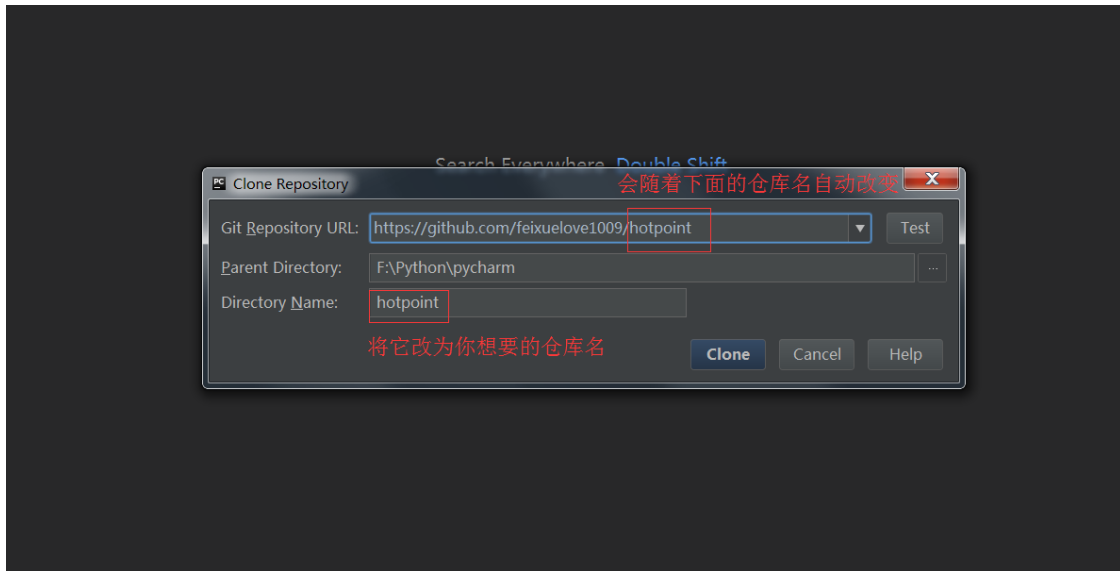
前面说的是上传，要下载呢？在 **VCS** 菜单中：



根据上图红框选择。



输入密码。



pycharm 会登录你的 github 账户，读取你的仓库信息，你可以从下拉框中选择仓库，也可以在 directory name 框内

直接输入仓库名。点击 clone，pycharm 就自动下载仓库内容了。

五、在 pycharm 内进行 git 的相关操作

以上只是 commit 和 clone 的操作，具体的 push，add，status 等 git 常用操作都在 CVS 菜单里可以找得到，相信具有

git 基础的同学一看就会，这里就不介绍了。

