

## 目录

1. Git & Github
  - Git
  - Github
  - 两者关系
2. Repository
  - Create a repo
  - Create a new file in github
  - Modify a file in github
  - Upload a new file in github
3. Branch
  - 创建分支
  - 合并分支
4. Fork
5. Pull Request
6. Issue
7. Git与使用命令
  - Git概述
  - 基本Linux命令
  - 初次使Git
8. Clone
9. Status & Add & Commit
  - 一些概念
  - 本地修改
  - 进入缓存区
  - 提交修改
  - 查看修改日志
10. Push/Pull
  - Push
  - Pull
11. Init

## 1. Git & Github

Git与Github是两个不同的概念。

### 1) Git

Git is one of the most popular **version control system(VCS)**. It is a mature, actively maintained, open source project compatible with many operating systems and IDEs.

Git 是一款免费的、开源的分布式**版本控制系统**。

#### 什么是版本控制?

简单来说, 版本控制是指, 将每次对项目的修改, 包括修改内容、修改者、修改时间等内容储存起来, 以便track或roll back。

## 2) Github

Github是基于Git版本控制的代码托管平台。即，Github上上传的所有项目代码均是基于Git进行版本控制的。但Github的功能除了Git为它提供的版本控制外，还有其他许多强大的功能，比如项目协作等。

## 3) 两者关系

Github使用Git进行版本控制，同时提供其他强大功能。

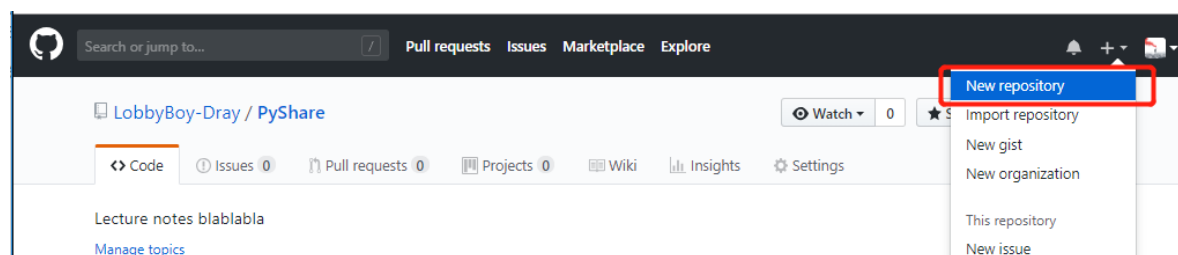
“大概就是「魔兽争霸」与「对战平台」的关系吧”——某知乎网友。

# 2. Repository

**Repository**, or "**Repo**": 可以简单理解为**Project**，存储着一个项目，可以包含很多文件，即Repository of files；如果想在Github上拥有一个项目，就必须创建一个Repository。

## 1) Create a repo

在加号的下拉菜单中点击New repository:



进入创建repo的界面:

### Create a new repository

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

Owner: LobbyBoy-Dray / Repository name: Lyrics repo名称

Description (optional): This is a repo for my lyrics. repo简介

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

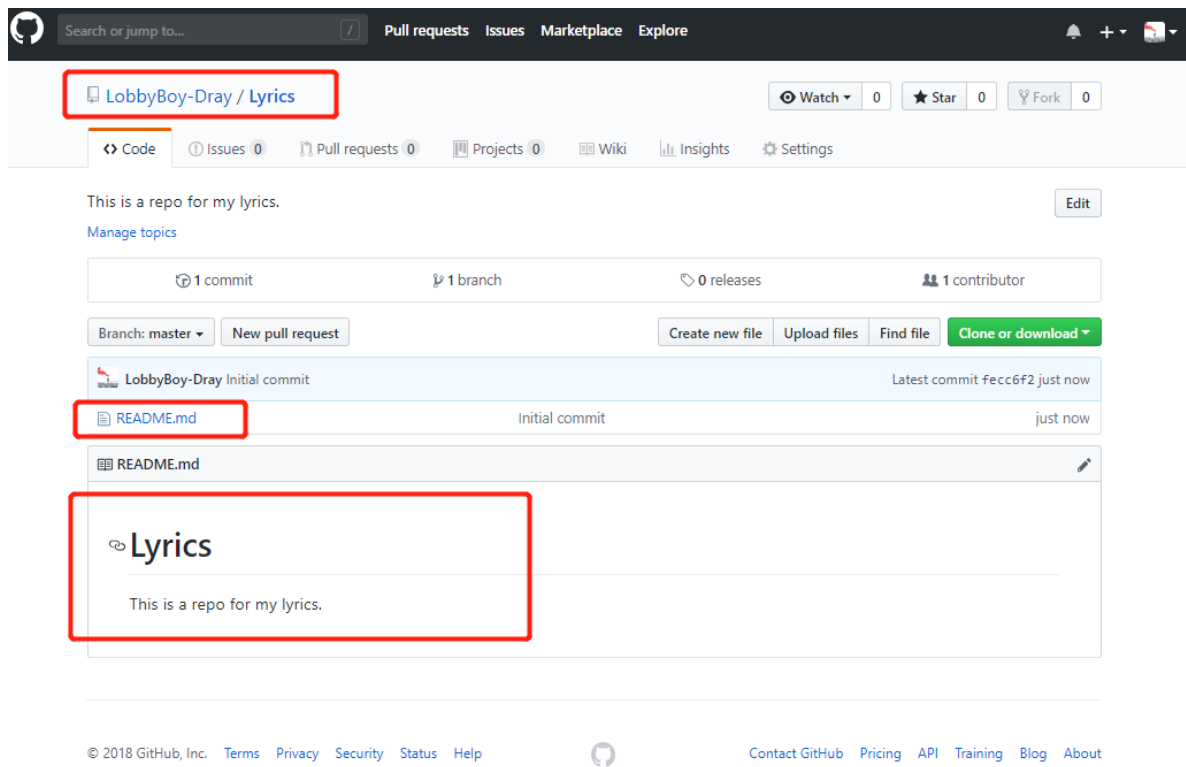
☐ Private: You choose who can see and commit to this repository.

☒ Initialize this repository with a README 是否创建readme文件，最好选中

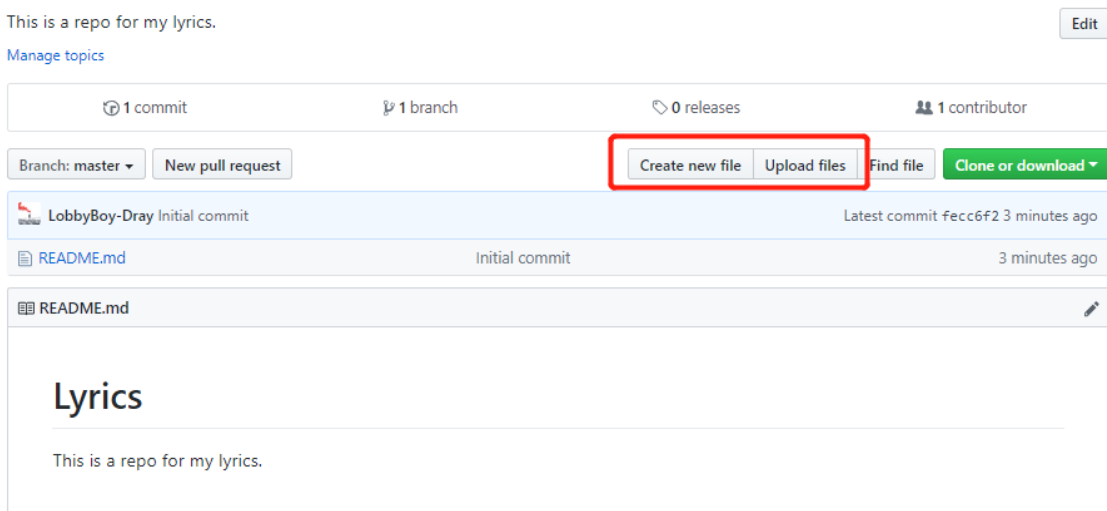
Add .gitignore: None | Add a license: None

Create repository 创建repo

创建成功后自动跳转到该repo内部:



可以在github中创建文件或上传文件：



## 2) Create a new file in github

点击Create new file，进入文本编辑界面，输入内容即可（注意文件名要加后缀）。完成后，点击commit new file

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Lyrics That girl.md [Cancel](#) 文件名

[Edit new file](#) [Preview](#) 预览模式 Spaces 2 No wrap

```
1 > This is the first song for that girl.
2
3 There's a girl but I let her get away
4 It's all my fault cause pride got in the way
5
6
7
8
```

内容

Commit new file

This is the first version, not too bad

Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

[Commit new file](#) [Cancel](#) 提交(执行)该创建

对此次创建(修改)的描述(备注)

创建完毕:

LobbyBoy-Dray / Lyrics Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

This is a repo for my lyrics. [Edit](#)

[Manage topics](#)

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file [Clone or download](#)

LobbyBoy-Dray This is the first version, not too bad Latest commit 1b774a7 just now

README.md	Initial commit	4 minutes ago
That girl.md	This is the first version, not too bad	just now

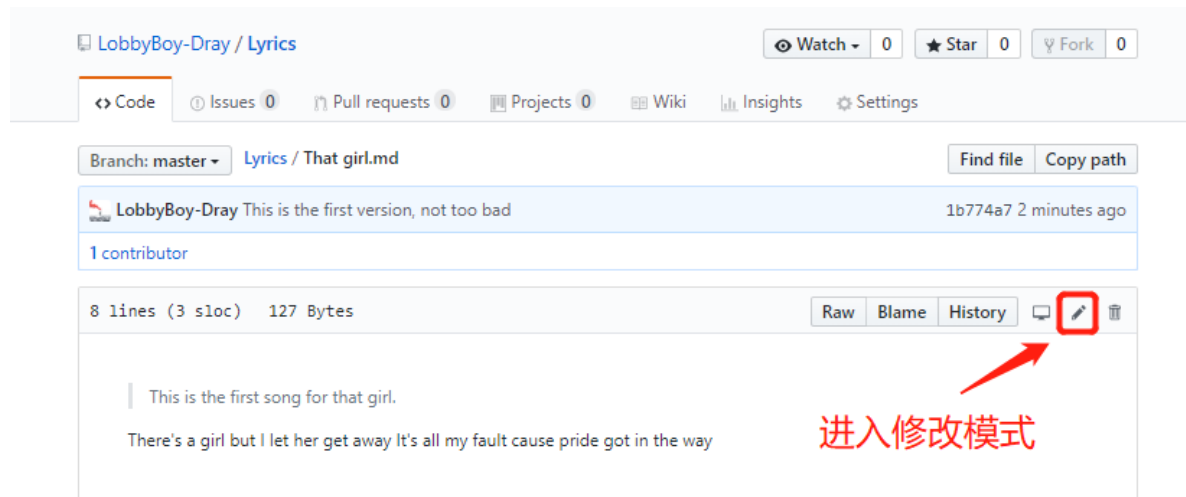
README.md

## Lyrics

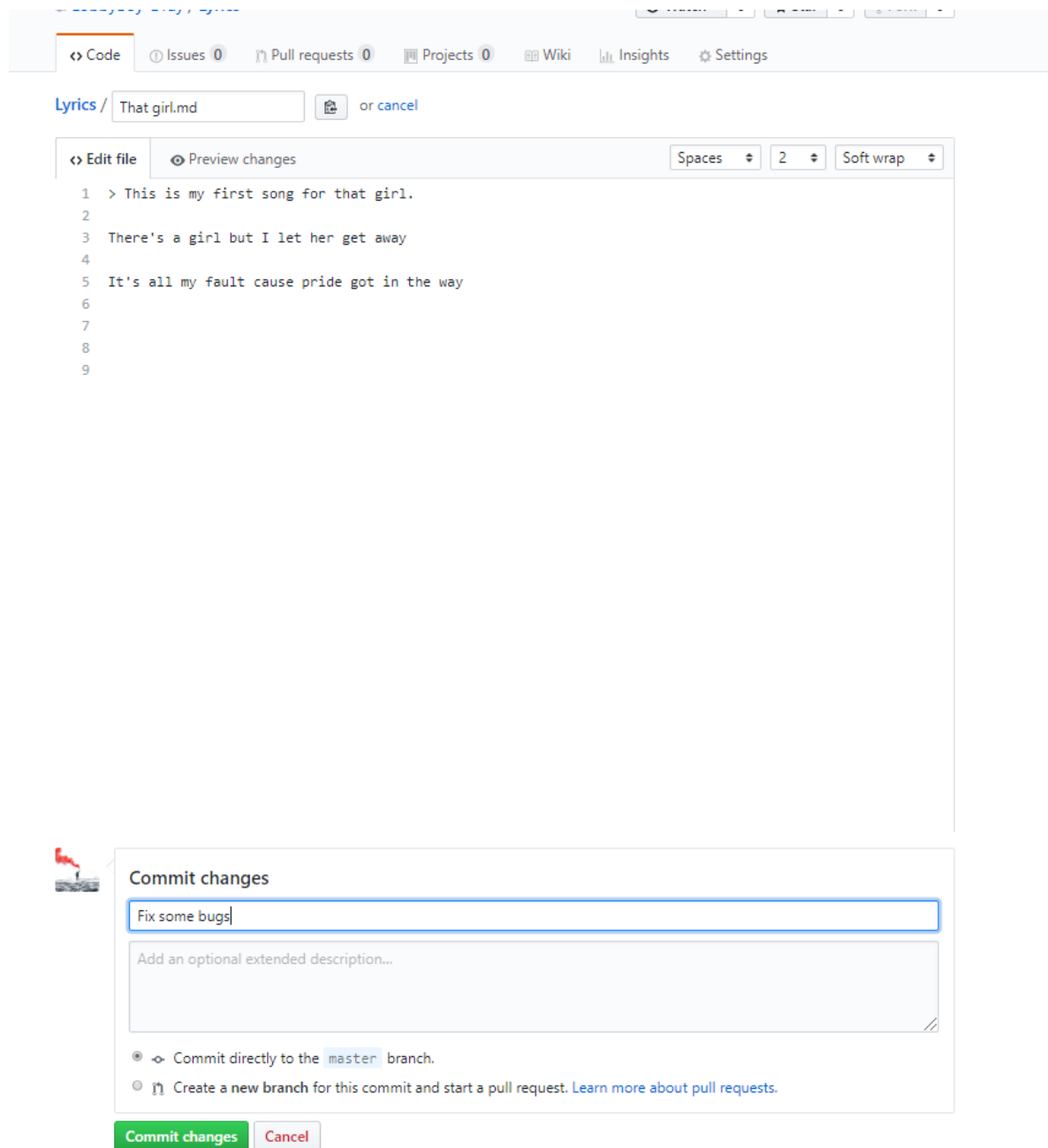
This is a repo for my lyrics.

### 3) Modify a file in github

点击文件，再点击“笔”，进入修改模式：



修改完成，点击commit changes：



修改历史（历史版本）可以在History中查看：

LobbyBoy-Dray / Lyrics

Watch 0Star 0Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

Branch: master Lyrics / That girl.mdFind fileCopy path

LobbyBoy-Dray Fix some bugs

a9641ca 9 hours ago

1 contributor

9 lines (3 sloc) 127 BytesRawBlameHistory

This is my first song for that girl.  
There's a girl but I let her get away  
It's all my fault cause pride got in the way

每次修改记录均可见：

LobbyBoy-Dray / Lyrics

Watch 0Star 0Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

History for Lyrics / That girl.md

Commits on Oct 15, 2018

Fix some bugs

Verified

a9641ca

<>

LobbyBoy-Dray committed 9 hours ago

This is the first version, not too bad

Verified

1b774a7

<>

LobbyBoy-Dray committed 9 hours ago

NewerOlder

点击进入可以查看更加详细的修改信息：

LobbyBoy-Dray / Lyrics

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Fix some bugs

master

LobbyBoy-Dray committed 9 hours ago Verified

1 parent 1b774a7 commit a9641caa54e9af7b40fcdc19833cea139b09fe70

Showing 1 changed file with 2 additions and 1 deletion. Unified Split

3 That girl.md

```
... @@ -1,6 +1,7 @@
1 - > This is the first song for that girl.
1 + > This is my first song for that girl.
2
3   There's a girl but I let her get away
4 +
4   It's all my fault cause pride got in the way
5
6
```

0 comments on commit a9641ca Lock conversation

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported Comment on this commit

## 4) Upload a new file in github

从本地上传文件至github:

LobbyBoy-Dray / Lyrics

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

This is a repo for my lyrics. Edit

Manage topics

3 commits 1 branch 0 releases 1 contributor

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

LobbyBoy-Dray Fix some bugs Latest commit a9641ca 9 hours ago

README.md	Initial commit	9 hours ago
That girl.md	Fix some bugs	9 hours ago

README.md

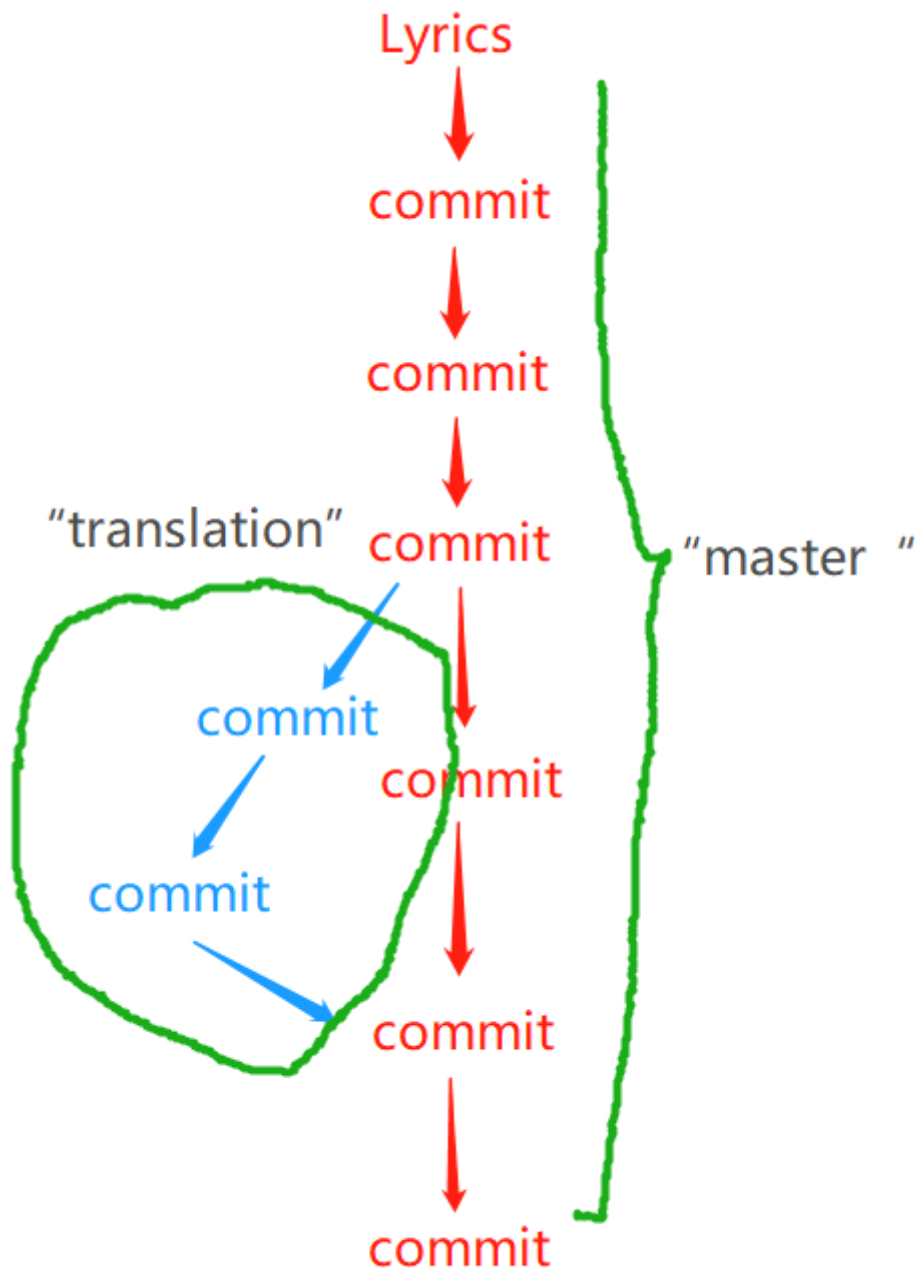
### Lyrics

This is a repo for my lyrics.

## 3. Branch

分支。主要用于项目的团队开发，也可以用于个人work on different aspects of the project。

任何一个Repo都有一个主分支，名为“master”。You can think the history of a project as a linear list of commit，而这个linear list就是master，如下图所示：



即分支是一个流程，是由无数次的修改提交（commit）构成一个时间轴。

而当你想在Repo中为你的项目加入一些新特性，而又isn't sure whether you should keep it的时候，你需要一块专门的空间来做experimental try，在这个空间里，你做的任何改动不会影响“大局”，即“master”，你可以放心对文件进行修改，直到修改到满意为止。这个空间就是你新建的分支（branch），可以最终通过merge，将该分支合并到master上，相当于把原来在“试验田”中的新特性正式加到你的项目中。

对于单人开发，可能只需要master和develop两个分支，平时的开发在develop分支进行，开发测试完成后，在发布前将develop合并到master分支即可。

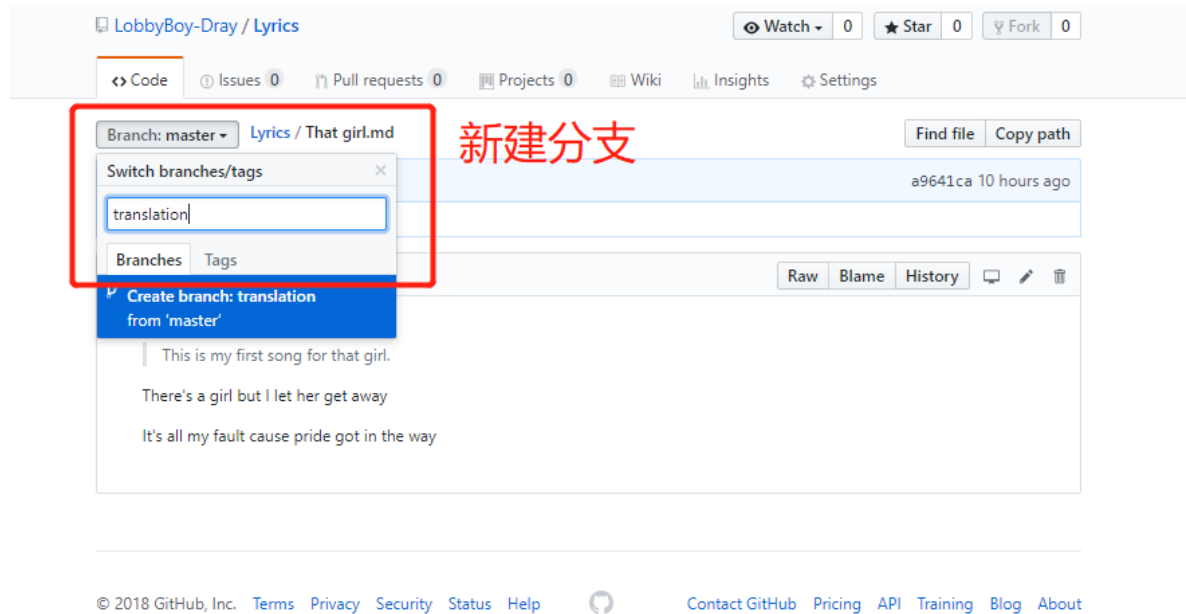
对于团队开发，可以为每个开发者创建一个分支，每个人在自己的分支上开发自己的部分，然后逐步merge到master。



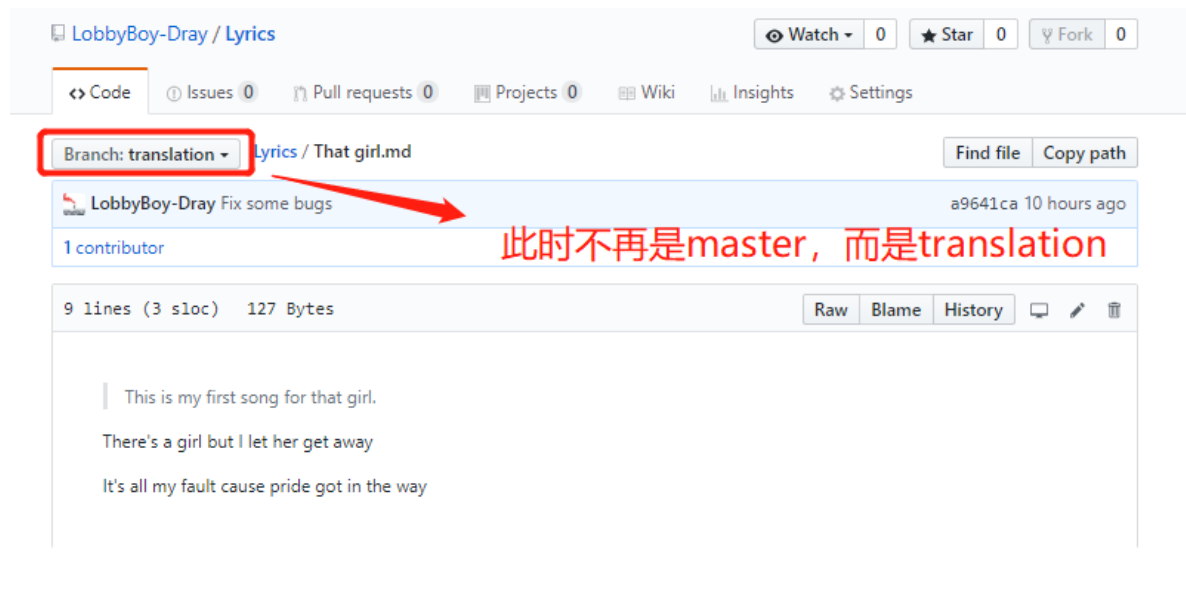
因此分支相当于生产零部件，master相当于主车间里的主体，最终零部件都是要进入主车间进行装配的。

## 1) 创建分支

举例：现在我想为我的歌词加入中文翻译，因此我可以创建一个分支，名为translation：



创建完毕后：



已经进入translation分支，下面在该分支里对文件进行修改并提交：

Lyrics / That girl.md or cancel

1 > This is my first song for that girl.  
2  
3 There's a girl but I let her get away  
4  
5 曾经心爱的女孩 我却让她擦肩而过  
6  
7 It's all my fault cause pride got in the way  
8  
9 自尊心作祟 一切都是我的错  
10  
11  
12

Commit changes

Add translation

Add an optional extended description...

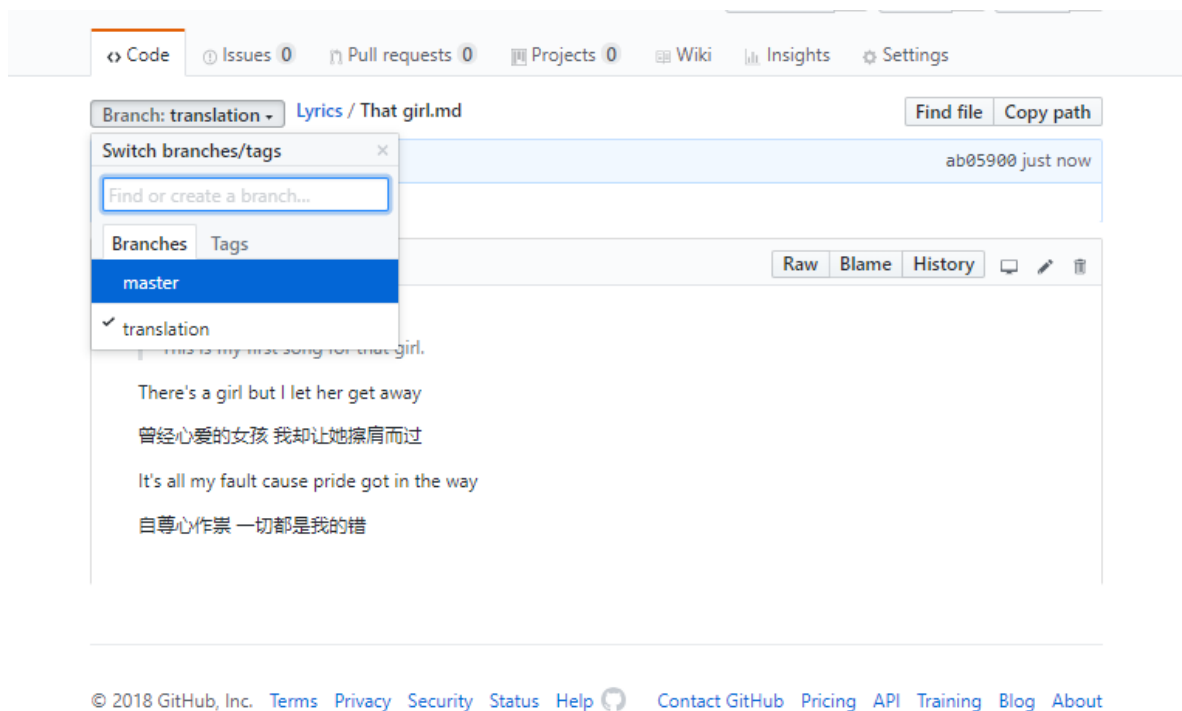
☒ Commit directly to the translation branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

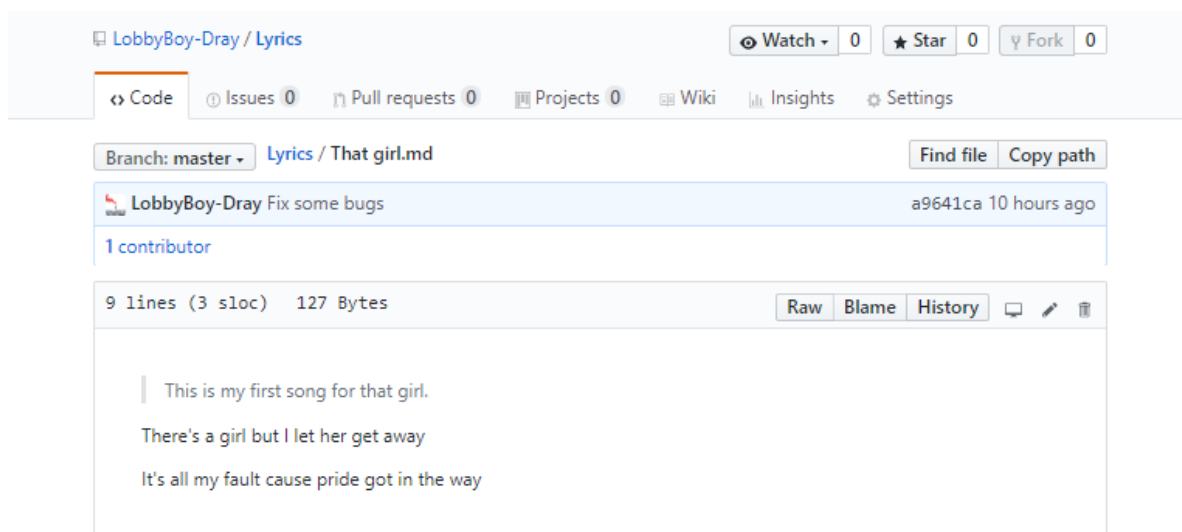
Commit changes Cancel

此时已经是commit到 translation分支，而不是master 分支，故不影响master

点击Branch可以切换回master分支：

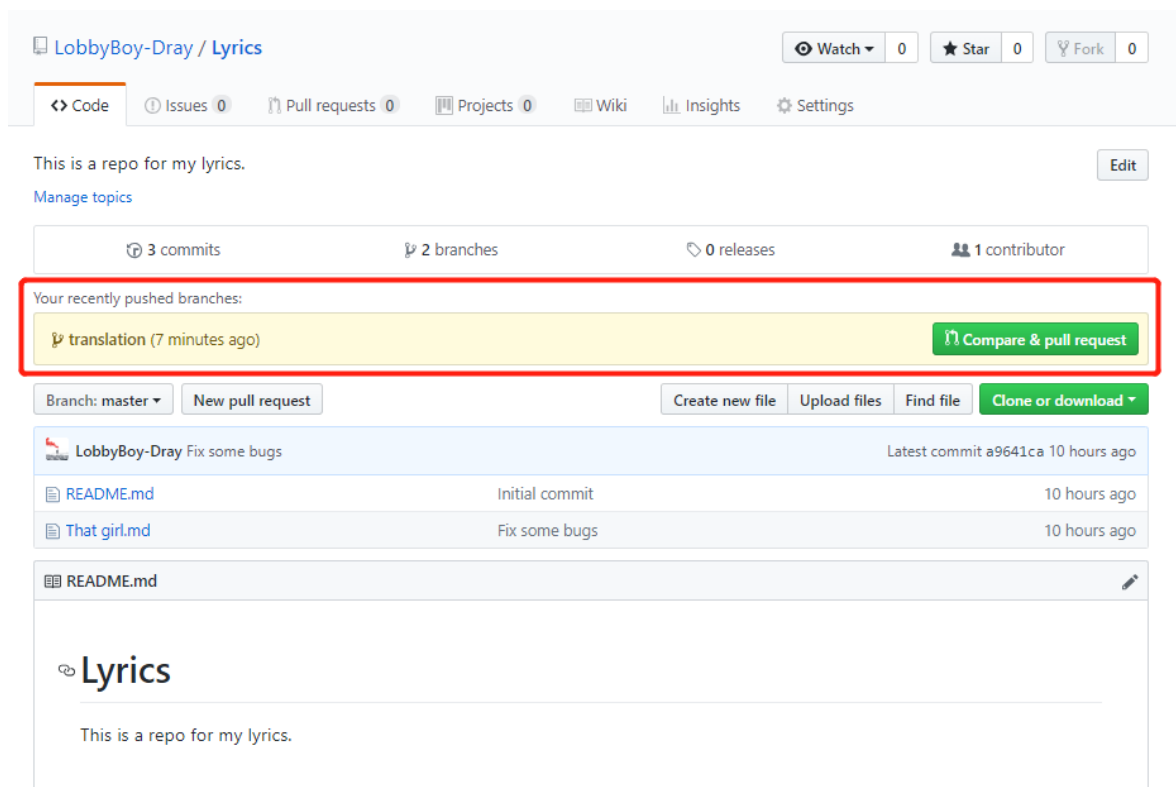


可以看到在master分支中，中文翻译并没有被加入：



## 2) 合并分支

新建的分支可以在Repo主页面看到：



上图中的绿框Compare & pull request就好像这个分支的创建者（这里是我自己）对我说：would you please take my changes? would you please pull my request and merge into the master branch?

所以pull request的含义是：take some changes from the particular branch and bring them into another branch（usually master branch）。

如果想要将该分支合并，点击该绿框，进入pull request页面：

LobbyBoy-Dray / Lyrics

Watch 0

Star 0

Fork 0

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ← compare: translation ✓ Able to merge. These branches can be automatically merged.

Add translation

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

Create pull request

1 commit

1 file changed

0 commit comments

1 contributor

Commits on Oct 15, 2018

LobbyBoy-Dray

Add translation

Verified

ab05900

Showing 1 changed file with 3 additions and 0 deletions.

Unified Split

3 That girl1.md

< >

📄

View

🗨

⌵

@@ -2,7 +2,10 @@

2 2

点击create pull request。但此时还没有完成merge，点击后github会检测两个合并的分支是否存在冲突，若不存在冲突，继续点击merge pull request、confirm merge：

LobbyBoy-Dray / Lyrics

Watch 0Star 0Fork 0

CodeIssues 0Pull requests 1Projects 0WikiInsightsSettings

## Add translation #1

Open

LobbyBoy-Dray wants to merge 1 commit into master from translation

Conversation 0Commits 1Checks 0Files changed 1+3 -0

LobbyBoy-Dray commented a minute ago

No description provided.

Add translation

Verifiedab05900

Add more commits by pushing to the translation branch on LobbyBoy-Dray/Lyrics.

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 requestComment

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

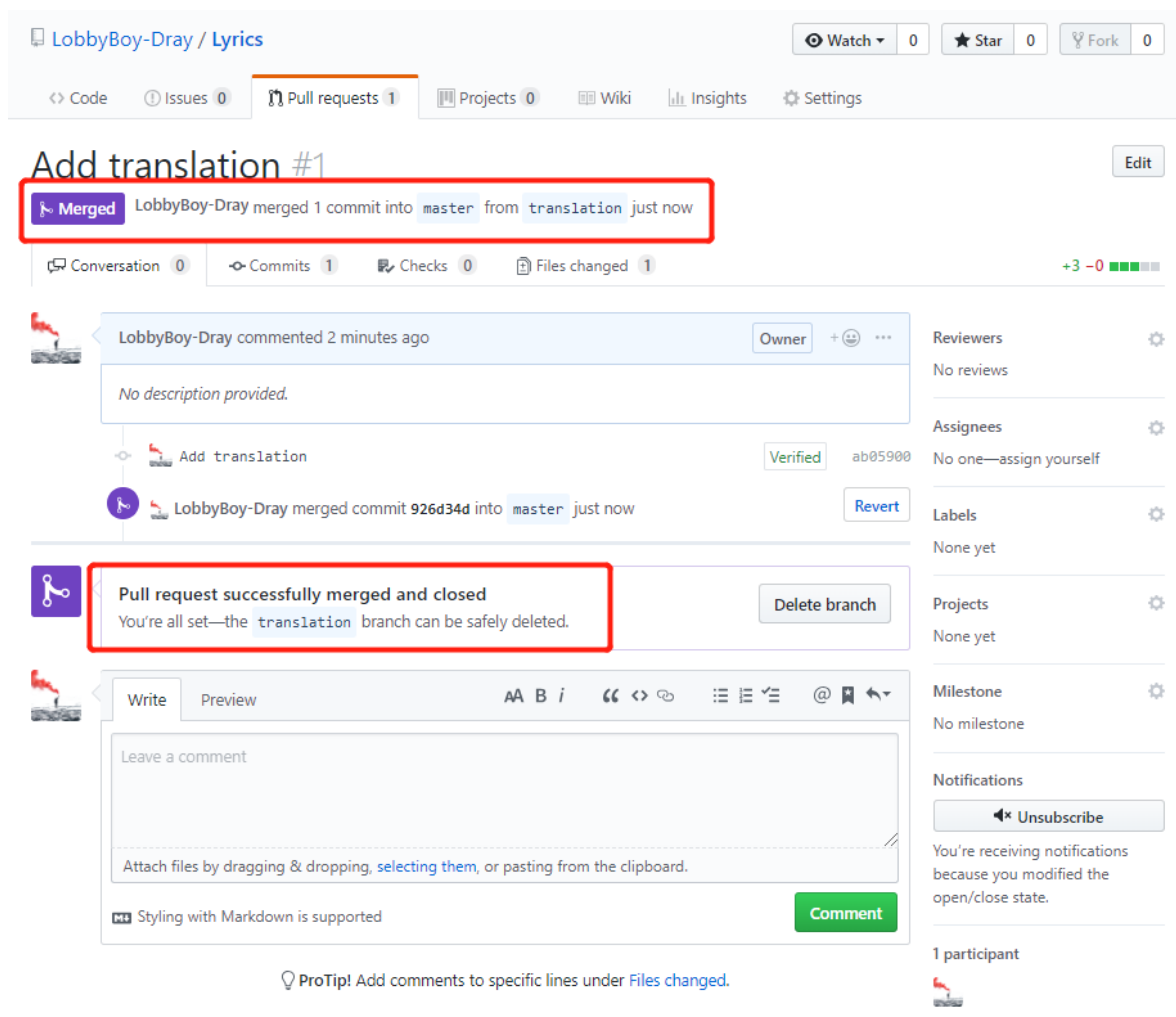
Unsubscribe

You're receiving notifications because you authored the thread.

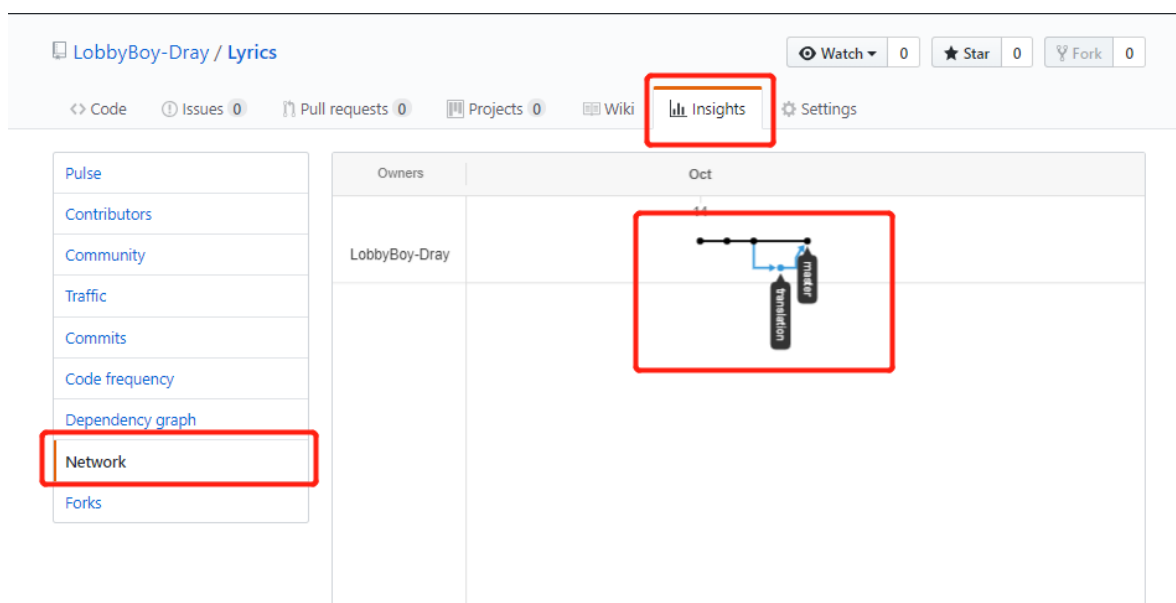
1 participant

Lock conversation

合并完成，translation分支可以被删除了：



在Insights-network里可以看到这一过程的图示：

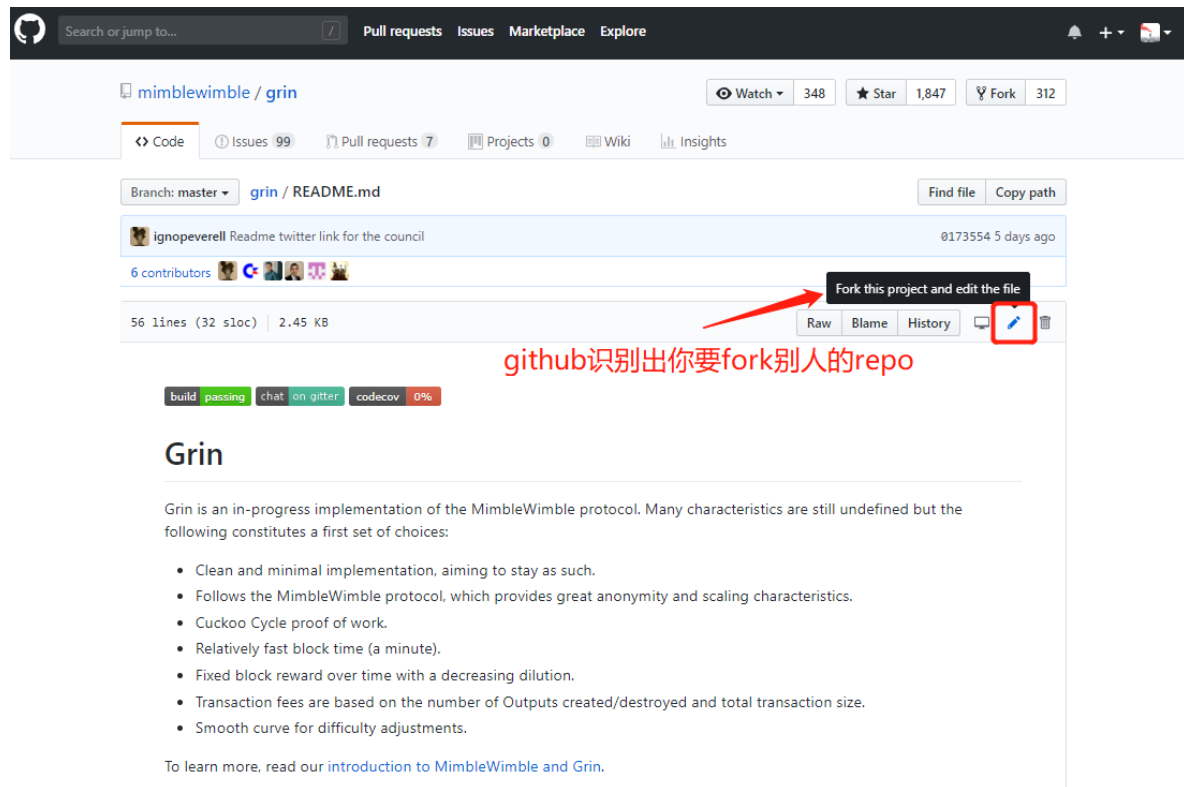


## 4. Fork

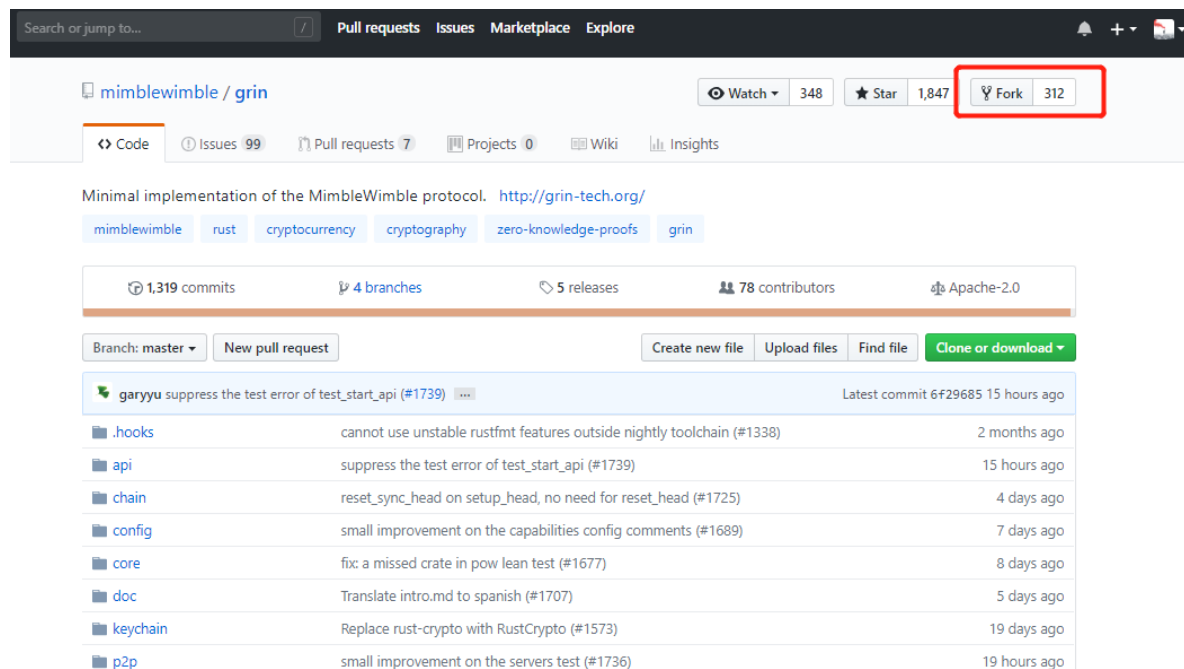
相当于复刻，将别人的Repo完完整整地复刻到自己的Github中，供自己参考，或者在别人的项目基础上进行改进。此时任你如何修改这个fork过来的repo，丝毫不会影响原作者那边的Repo，因为这个Repo相当于副本，也可以理解为使一个branch，在branch上改，当然不影响其他的branch。

怎样fork别人的Repo？

进入该Repo，随便点击一个文件，点击“笔”：



或者点击右上角的fork：

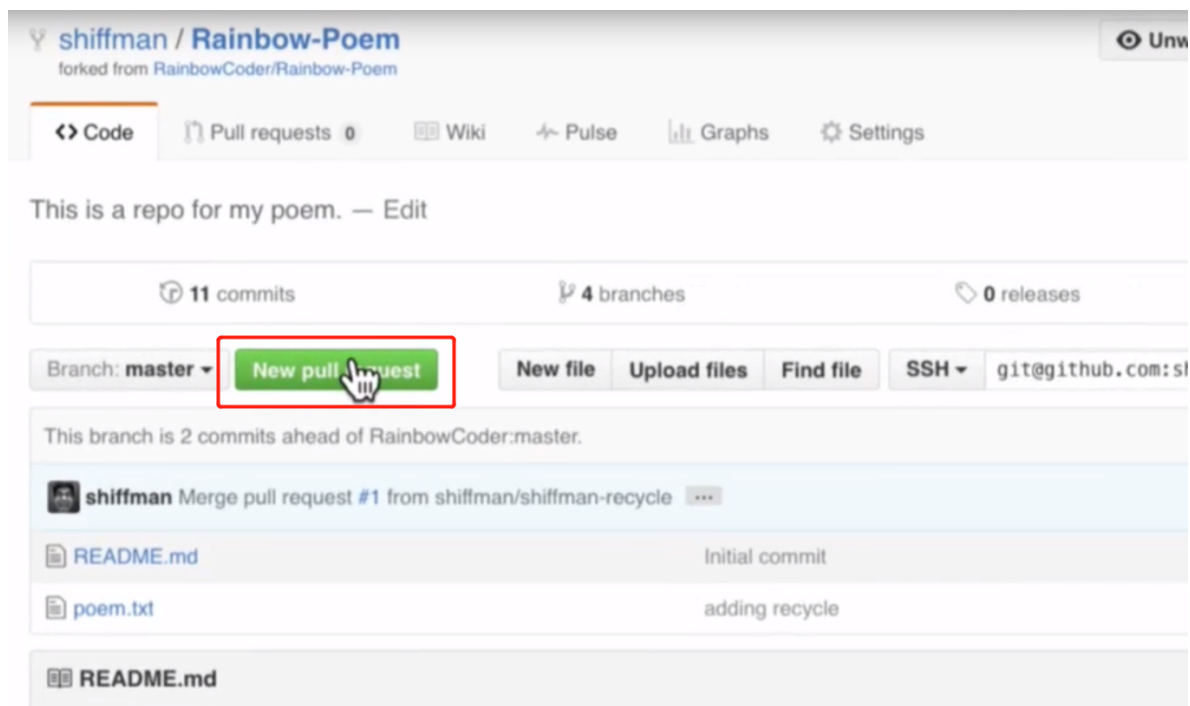


## 5. Pull Request

在Branches部分已经简单提及Pull Request的概念，可以理解为“获取请求”。假设别人fork了你的Repo，然后在这个fork上面做了一些改进，他觉得改进很棒，也有利于你的代码，因此他想把他在你的fork上的修改提供给你——contribute it back to the original repo。此时他需要push request，即“推出请求”，即向你发出请求，请求你允许他将他的那部分代码合并到你的原Repo中；而你如果同意的话，就需要pull request，即“获取请求”。

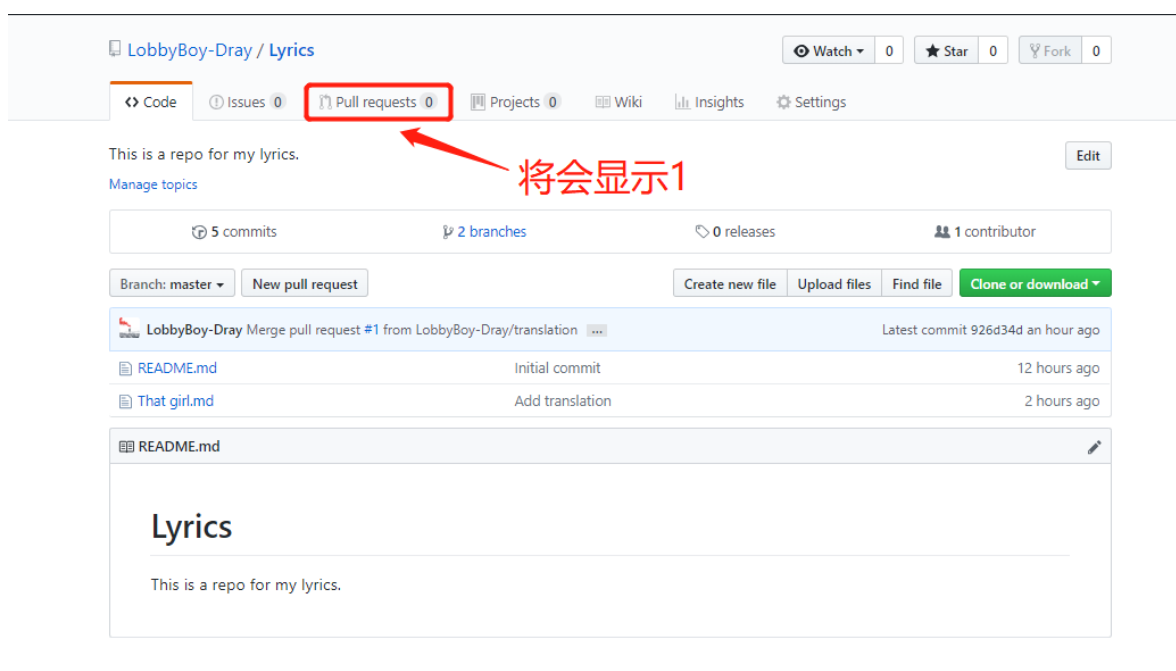
当修改完毕fork的Repo后，





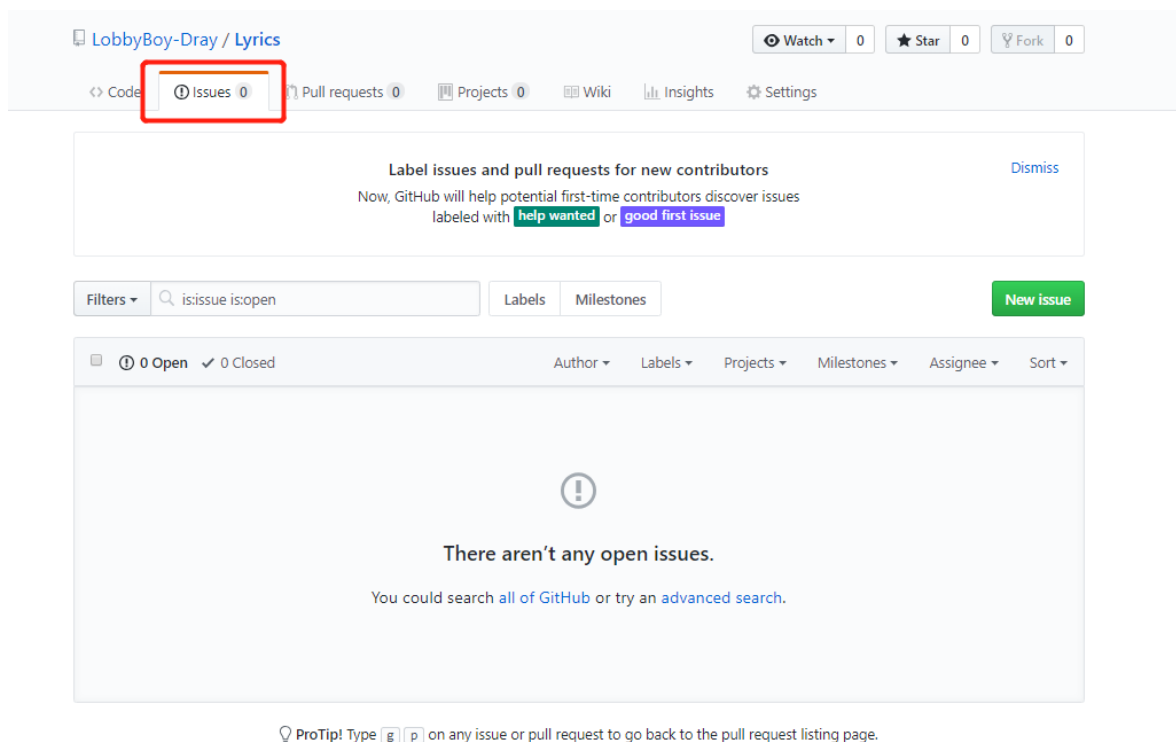
点击后并按操作继续，可以向原Repo的主人push request，接着你就只需默默祈祷他能够pull your request。

在原主人那里，他会接收到你push的request，在下图的Pull requests里可以看到，他会根据他的需求选择是否接受你的修改。



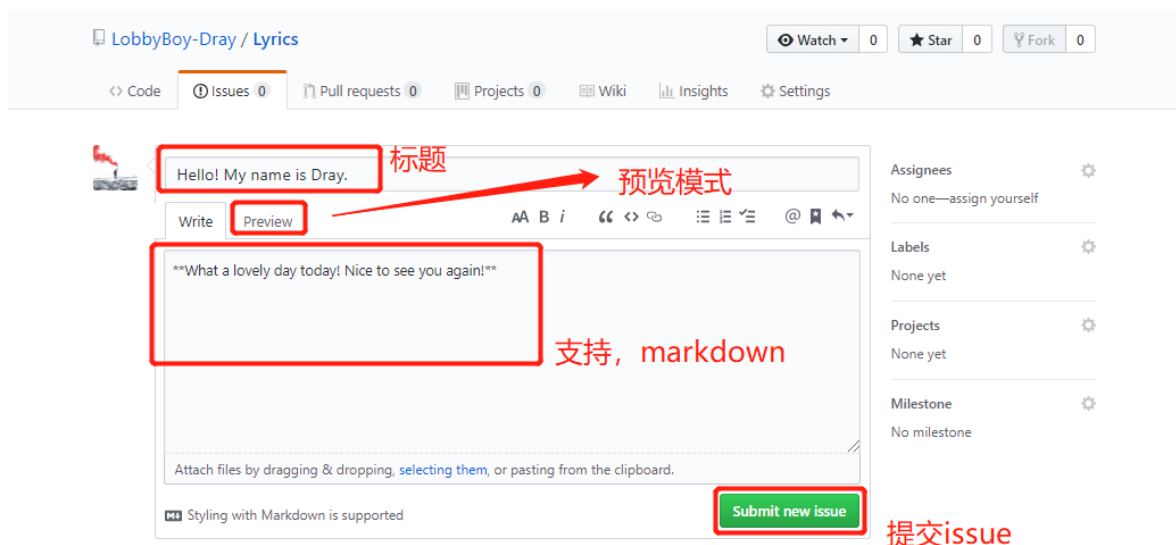
## 6. Issue

问题，事件。

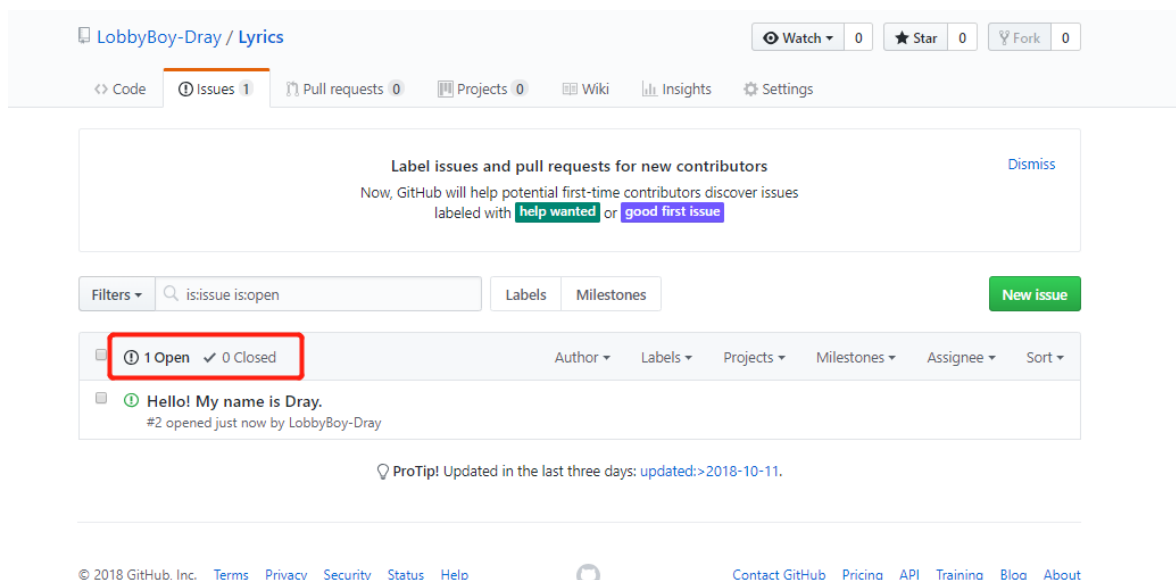


Issues is a place to leave a comment about the project. It can be some one who finds a bug, or you would like to start a vote.

点击New issue创建一个issue:



创建完毕后，在issues界面可以看到有一个issue是open的:



点进去后可以关闭这个issue。对于别人发起的issue，你看到了这些问题可以去逐个修复，修复完成后就可以一个个的close掉。

## 7. Git与使用命令行

### 1) Git概述

你可以在Github上使用Git进行版本管理，也可以在自己的电脑上（本地）使用Git进行版本管理，并与Github进行交互，但前提是你的电脑上已经安装了Git。

Mac系统自带Git，但Win不一定。在命令行中键入git并回车，如果没有报错（如下图），则你的电脑已经安装Git；如果报错，则需要在Git官网根据自己的电脑类型下载相应安装包：

```
1. bash
Last login: Mon Oct 15 14:40:04 on ttys000
GaodeMacBook-Air:~ draymondgao$ 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
```

[Git for Win](#)

强烈推荐使用命令行（Command Line）操作Git而非GUI界面，因为使用命令行操作可以加深对于Git的理解。

Git的基本操作命令如下（不用全部记住，浏览一下就好）：

```
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
```

可以发现，Git命令均以git开头。

## 2) 基本Linux命令

Win系统下，ls命令无法使用，等价命令为dir。

推荐一款Win的终端替代软件——cmdr，免费，颜值高，功能强大，解决ls在Win下无法使用的问题等等。下载地址为：[Download](#)

必须知道的四个命令：

- cd：切换当前工作目录至xxx，cd ..表示返回上一层目录；
- pwd：显示当前工作目录；
- ls：显示当前工作目录中所有文件的名称，ls -a显示包括隐藏文件的名称；
- clear：清除屏幕（历史记录）。

```
1. bash
GaodeMacBook-Air:~ draymondgao$ cd Desktop
GaodeMacBook-Air:Desktop draymondgao$ ls
DataScience      Kaggle-Titanic  L4.txt          PyShare          all              第一期
GaodeMacBook-Air:Desktop draymondgao$ pwd
/Users/draymondgao/Desktop
GaodeMacBook-Air:Desktop draymondgao$ cd PyShare
GaodeMacBook-Air:PyShare draymondgao$ ls
Git & github      Lecture1        Lecture2        Markdown         README.md
GaodeMacBook-Air:PyShare draymondgao$ ls -a
.                  .DS_Store      Git & github    Lecture2         README.md
..                 .git           Lecture1        Markdown
GaodeMacBook-Air:PyShare draymondgao$ pwd
/Users/draymondgao/Desktop/PyShare
GaodeMacBook-Air:PyShare draymondgao$
```

### 3) 初次使用Git

初次使用Git需要设置姓名和邮箱，告诉Git你是谁：

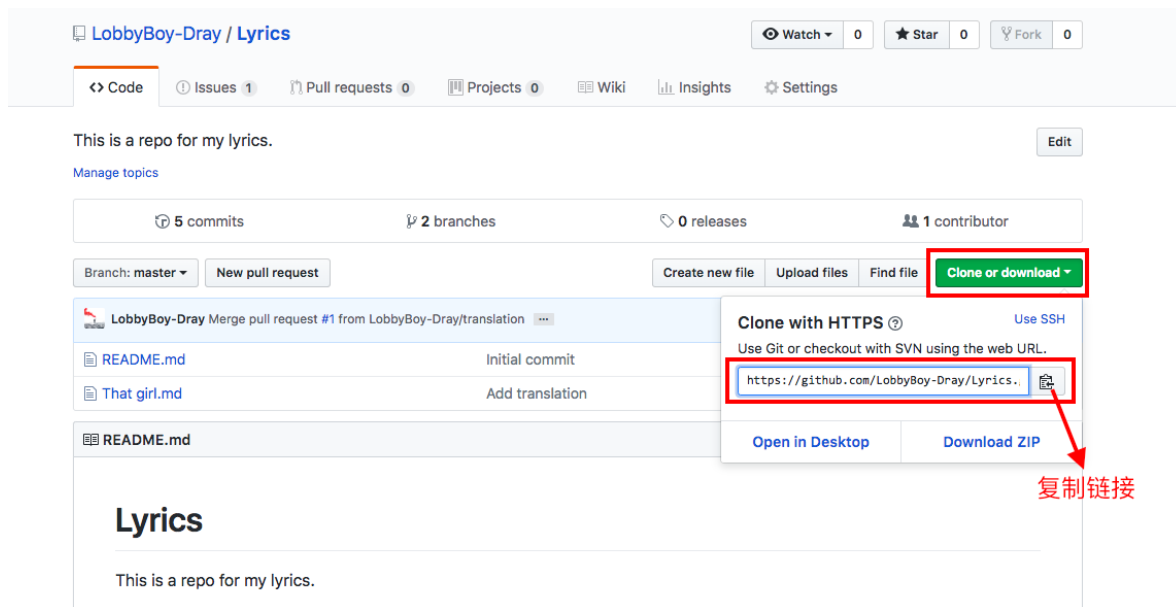
- `git config -global user.name "你的姓名"`
- `git config -global user.email "你的邮箱"`

## 8. Clone

`git clone`: Clone a repository into a new directory.

Clone命令将一个GitHub上的Repo克隆（download）到本地，变为本地仓库。

Clone一个Repo需要给到该Repo的url，查看方法如下：



因此要Clone该Repo（Lyrics）到我的桌面，步骤为：

- 打开终端；
- `cd`到桌面；
- 找到该Repo的链接：<https://github.com/LobbyBoy-Dray/Lyrics.git>;
- `git clone https://github.com/LobbyBoy-Dray/Lyrics.git`;

```
1. bash
Last login: Mon Oct 15 15:33:01 on ttys000
GaodeMacBook-Air:~ draymondgao$ cd Desktop/
GaodeMacBook-Air:Desktop draymondgao$ git clone https://github.com/LobbyBoy-Dray/Lyrics.git
Cloning into 'Lyrics'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 13 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (13/13), done.
GaodeMacBook-Air:Desktop draymondgao$
```

## 9. Status & Add & Commit

`git status`: Show the working tree status.

### 1) 一些概念



根据上图理解以下概念:

- **working tree:** 工作区, 即Repo文件夹。在上例中, working tree就是Lyrics这个文件夹, 包括里面的所有文件夹和文件。每当你修改了其中的某个文件, working tree的状态就会改变;
- **stage:** 缓存区, 一个中转站;
- **local repository:** 本地仓库, 也叫版本库, 存储着各种不同的版本, 即History。

回到上面的例子, 对于刚刚clone下来的Lyrics仓库, cd进去后, 键入`git status`:

```
GaodeMacBook-Air:Desktop draymondgao$ cd Lyrics/
GaodeMacBook-Air:Lyrics draymondgao$ ls
README.md      That girl.md
GaodeMacBook-Air:Lyrics draymondgao$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
GaodeMacBook-Air:Lyrics draymondgao$
```

因为对刚刚clone下来的Lyrics仓库，我还没有做任何改动，所以会显示nothing to commit, working tree clean。

PS: 建议大家没事就输一下这个命令来查看你当前工作区的状态。

## 2) 本地修改

两处变化：

- 修改Lyrics中的That girl.md文件（加了两句英文歌词）；
- 增加了一个文件，名为That girl\_2.md。

此时，再次键入git status：

```
1. bash
GaodeMacBook-Air:Lyrics draymondgao$ git status
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:   That girl.md
Untracked files:
  (use "git add <file>..." to include in what will be committed)
       That girl_2.md
no changes added to commit (use "git add" and/or "git commit -a")
GaodeMacBook-Air:Lyrics draymondgao$
```

上图说明了两点，也是Git帮你发现的两个改变：

- 工作区中的That girl.md文件被修改，该文件的状态变为**modified**，且该修改还**not staged**；
- 工作区中的That girl\_2.md是个新文件，刚才还不在工作区里，所以被Git标记为**untracked**，说明该文件虽然在工作区里，但是没有加入到git库中，不参与版本控制。

可见，工作区中的文件有三种类型：

- **untracked**：未跟踪状态，一般是新建的文件或文件夹。虽然该文件在工作区中，但并没有加入到Git库中，不参与版本控制，需要进行后续一些操作将其入库；
- **unmodified**：未修改状态，比如Lyrics中的readme文件，我没有修改它，所以它是未修改状态。这样的文件已经入库，参与版本控制，但没有被修改，git status命令也不会显示这些文件；
- **modified**：修改状态，说明该入库的文件已经被修改。



### 3) 进入缓存区

工作区发生变化，即出现untracked或modified文件时，Git会提示你将他们提交到**缓存区**，即**stage**。

使用`git add .`将所有改动，包括提交到缓存区；使用`git add 文件名`将特定的改动文件提交到缓存区。这里我使用前一种命令，将一个untracked文件和一个modified文件提交到stage中，并使用`git status`查看状态：

```
GaodeMacBook-Air:Lyrics draymondgao$ git add .
GaodeMacBook-Air:Lyrics draymondgao$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   That girl.md
    new file:   That girl_2.md
```

可见，字体颜色由红变绿，说明改动被成功提交至缓存区。此时上述两个文件来到了一个新的状态——**staged**，暂存状态。

处于该状态下的修改，仅仅为“暂存”，并没有最终生成新的版本。还需要再进行一步`git commit`，将修改最终同步到库中。

因此，可以将缓存区理解为临时保存你的改动的区域，这样做的优点是防止失误提交。

如果“后悔了”，可以使用`git reset HEAD <file>...` to unstage将修改移出缓存区，回到第一步中的工作区成为untracked状态或modified状态。

### 4) 提交修改

使用`git commit -a -m "这里写一些注释"`来完成最终提交。

```
GaodeMacBook-Air:Lyrics draymondgao$ git commit -a -m "Make some changes."
[master 744f5e4] Make some changes.
 2 files changed, 11 insertions(+)
 create mode 100644 That girl_2.md
GaodeMacBook-Air:Lyrics draymondgao$ git status
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 tree clean
GaodeMacBook-Air:Lyrics draymondgao$
```

在最终提交完毕后，那些staged状态的文件，最终会转化为unmodified状态，回到工作区，其历史版本信息则被保存在local repository中。

在commit后，使用`git status`再次查看状态，可以发现，`nothing to commit, working tree clean`，一切重归于好。

### 5) 查看修改日志

使用`git log`：



```
GaodeMacBook-Air:Lyrics draymondgao$ git log
commit 744f5e4b6de5b51fb8f40c8196d2a534fc0adb50 (HEAD -> master)
Author: LobbyBoy-Dray <gjw2014sis@163.com>
Date: Mon Oct 15 16:57:45 2018 +0800

    Make some changes.

commit 926d34dbaf31819663f883f5afb8d324742699a2 (origin/master, origin/HEAD)
Merge: a9641ca ab05900
Author: Draymond Gao <42985789+LobbyBoy-Dray@users.noreply.github.com>
Date: Mon Oct 15 11:00:48 2018 +0800

    Merge pull request #1 from LobbyBoy-Dray/translation

    Add translation

commit ab0590044eb31ecfb957a0992e472d60dd9da253 (origin/translation)
Author: Draymond Gao <42985789+LobbyBoy-Dray@users.noreply.github.com>
Date: Mon Oct 15 10:35:16 2018 +0800

    Add translation

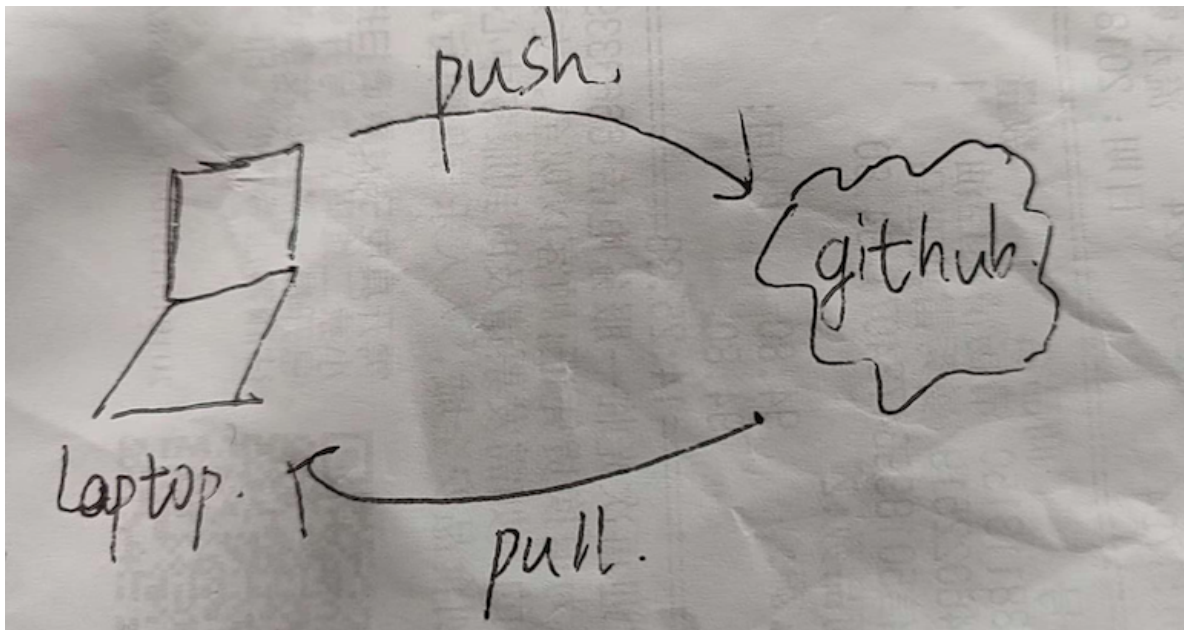
commit a9641caa54e9af7b40fcdc19833cea139b09fe70
Author: Draymond Gao <42985789+LobbyBoy-Dray@users.noreply.github.com>
Date: Mon Oct 15 00:24:33 2018 +0800
```

可以看到，除了刚刚的修改被记录了下来，早上在Github上操作的修改也被记录了下来，这是因为Clone的效果，Clone把一切信息都复制了下来。

## 10. Push/Pull

### 1) Push

此时，你想把刚刚在本地对Repo的修改同步到Github上，需要使用git push命令。



push: 推。如果你本地代码有更新，那么就需要把本地代码推到远程仓库（remote repository），这样本地仓库（local repository）跟远程仓库就可以保持同步了。

一般直接使用git push origin master，意思是将本地代码“推”到远程仓库的master分支上。

```
GaodeMacBook-Air:Lyrics draymondgao$ git push origin master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 437 bytes | 437.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/LobbyBoy-Dray/Lyrics.git
    926d34d..744f5e4  master -> master
GaodeMacBook-Air:Lyrics draymondgao$
```

## 2) Pull

pull: 拉。如果别人提交代码到远程仓库，这个时候你需要把远程仓库的最新代码同步到自己本地电脑上的Repo中。

常用命令: `git pull origin master`

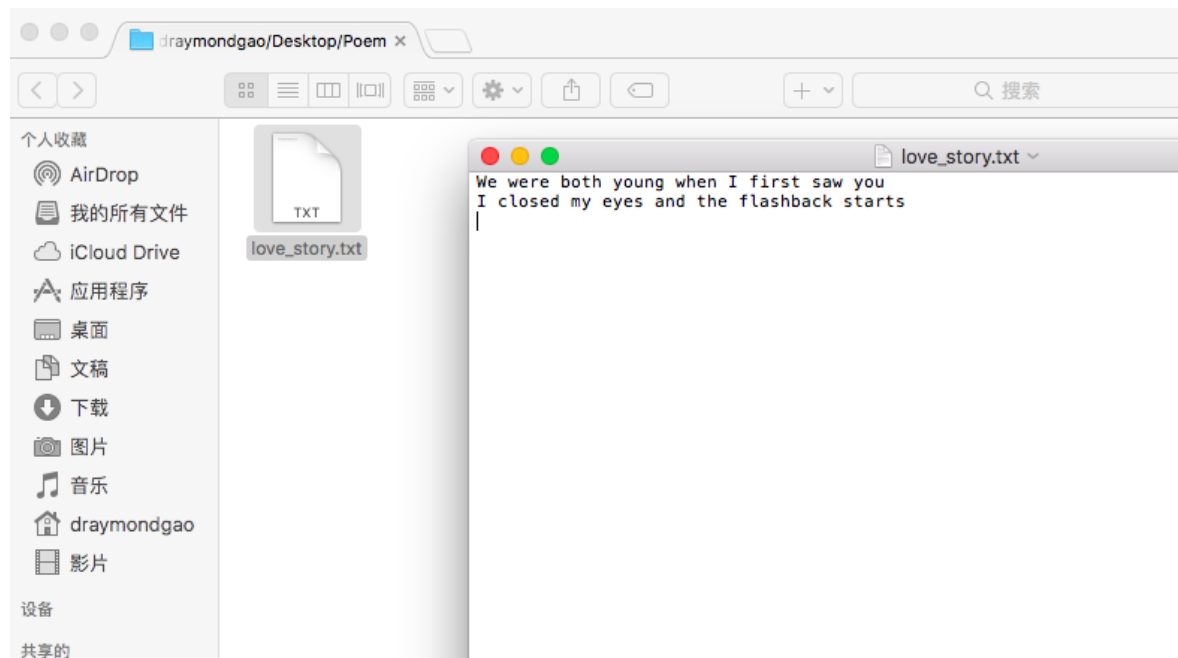
一般在push前都会先pull一下，这样不容易产生冲突。

## 11. Init

如果想要在本地电脑从零开始创建一个Repo，再将其上传至github，就要使用`git init`命令。

**举例：在本地新建名为Poem的Repo**

首先，在桌面新建名为Poem的文件夹，并在其中新建名为`love_story.txt`的文件，输入一些内容并保存。



打开终端，cd到Poem文件夹（在进行任何git操作前，都要先切换到仓库目录，也就是要cd到项目文件夹的目录下），键入`git status`查看Repo的状态：

```
1. bash
Last login: Mon Oct 15 15:47:50 on ttys001
GaodeMacBook-Air:~ draymondgao$ cd Desktop/
GaodeMacBook-Air:Desktop draymondgao$ cd Poem/
GaodeMacBook-Air:Poem draymondgao$ git status
fatal: Not a git repository (or any of the parent directories): .git
GaodeMacBook-Air:Poem draymondgao$
```

系统会提示Not a git repository，意思就是当前目录还不是一个Repo——肯定的啊，不是随便在自己电脑上建一个文件夹它就是Git仓库的，需要手动设置，标定它成为Repo。

在根目录下使用git init：

```
GaodeMacBook-Air:Poem draymondgao$ git init
Initialized empty Git repository in /Users/draymondgao/Desktop/Poem/.git/
GaodeMacBook-Air:Poem draymondgao$
```

可以看到，Poem文件夹已经被初始化一个Git仓库了。

此时git status：

```
1. bash
GaodeMacBook-Air:Poem draymondgao$ git status
On branch master

No commits yet

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

    love_story.txt

nothing added to commit but untracked files present (use "git add" to track)
GaodeMacBook-Air:Poem draymondgao$
```

上图说明，该Repo中的所有文件都没有被tracked，处于untracked状态，因此需要add并commit：

git add .: 将所有文件放入缓存区；

git commit -a -m "xxxxxxx": 提交修改；

接下来使用git push origin master命令上传到Github，发现出现错误：

```
GaodeMacBook-Air:Poem draymondgao$ git push origin master
error: src refspec master does not match any.
error: failed to push some refs to 'origin'
GaodeMacBook-Air:Poem draymondgao$
```

这是因为，这个刚刚创建的本地仓库，并没有和远程仓库——即github上的Repo建立关联（not associated）。使用`git remote`或`git remote -v`查看与本地仓库关联的远程仓库：

```
GaodeMacBook-Air:Poem draymondgao$ git remote
GaodeMacBook-Air:Poem draymondgao$ git remote -v
GaodeMacBook-Air:Poem draymondgao$
```

发现的确该本地Repo还没有关联任何远程Repo。


此时需要打开Github，新建一个同名的Repo，注意此时不用选中Initialize this repository with a readme。

## Create a new repository

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

Owner

Repository name


 LobbyBoy-Dray / Poem ✓

Great repository names are short and memorable. Need inspiration? How about **turbo-doodle**.

Description (optional)

I love poem!

☒  **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

新建后转到如下页面：

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


 Set up in Desktop or **HTTPS** **SSH**   远程仓库链接

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

---

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

```
echo "# Poem" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/LobbyBoy-Dray/Poem.git
git push -u origin master
```




本地创建Repo并同步至github的步骤

---

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

```
git remote add origin https://github.com/LobbyBoy-Dray/Poem.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.

拿到远程仓库链接后，使用`git remote add origin 链接命令`，将本地仓库与远程仓库连接起来：

```
GaodeMacBook-Air:Poem draymondgao$ git remote add origin https://github.com/LobbyBoy-Dray/Poem.git
GaodeMacBook-Air:Poem draymondgao$ git remote
origin
GaodeMacBook-Air:Poem draymondgao$ git remote -v
origin https://github.com/LobbyBoy-Dray/Poem.git (fetch)
origin https://github.com/LobbyBoy-Dray/Poem.git (push)
```

此时再使用`git remote`和`git remote -v`查看，发现已经关联完毕。

最后使用`git push origin master`命令将该Repo上传到Github，成功：

```
GaodeMacBook-Air:Poem draymondgao$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/LobbyBoy-Dray/Poem/pull/new/master
remote:
To https://github.com/LobbyBoy-Dray/Poem.git
 * [new branch]      master -> master
GaodeMacBook-Air:Poem draymondgao$
```

总结一下步骤（以Poem为例）：

1. cd到Poem下；
2. `git init`；
3. 在github上创建Repo，拿到远程仓库链接；
4. `git remote add origin 链接`；
5. `git add .`；

```
6.git commit -a -m "注释";  
7.git push origin master;
```

### **为什么clone下来的文件夹不需要init?**

因为clone操作可以看做**高级复制**，clone下来的项目本身已经被git处理好了，已经是一个git仓库了，所以不需执行git init命令。