

# 基于 git 和 github 的团队写作开发

雷博文

## 1. 建一个 github 账号、下载 git

①Github 网站: <https://github.com/>

git 下载地址 (下最新版就可): <https://git-scm.com/downloads>

下载完 git 之后安装的选项都默认就好

放一张 git 常用命令图:

### Git 常用命令速查表

master :默认开发分支	Head :默认开发分支
origin :默认远程版本库	Head^ :Head 的父提交

#### 创建版本库

```
$ git clone <url> #克隆远程版本库
$ git init #初始化本地版本库
```

#### 修改和提交

```
$ git status #查看状态
$ git diff #查看变更内容
$ git add . #跟踪所有改动过的文件
$ git add <file> #跟踪指定的文件
$ git mv <old> <new> #文件改名
$ git rm <file> #删除文件
$ git rm --cached <file> #停止跟踪文件但不删除
$ git commit -m "commit message" #提交所有更新过的文件
$ git commit --amend #修改最后一次提交
```

#### 查看提交历史

```
$ git log #查看提交历史
$ git log -p <file> #查看指定文件的提交历史
$ git blame <file> #以列表方式查看指定文件的提交历史
```

#### 撤销

```
$ git reset --hard HEAD #撤销工作目录中所有未提交文件的修改内容
$ git checkout HEAD <file> #撤销指定的未提交文件的修改内容
$ git revert <commit> #撤销指定的提交
```

#### 分支与标签

```
$ git branch #显示所有本地分支
$ git checkout <branch/tag> #切换到指定分支或标签
$ git branch <new-branch> #创建新分支
$ git branch -d <branch> #删除本地分支
$ git tag #列出所有本地标签
$ git tag <tagname> #基于最新提交创建标签
$ git tag -d <tagname> #删除标签
```

#### 合并与衍合

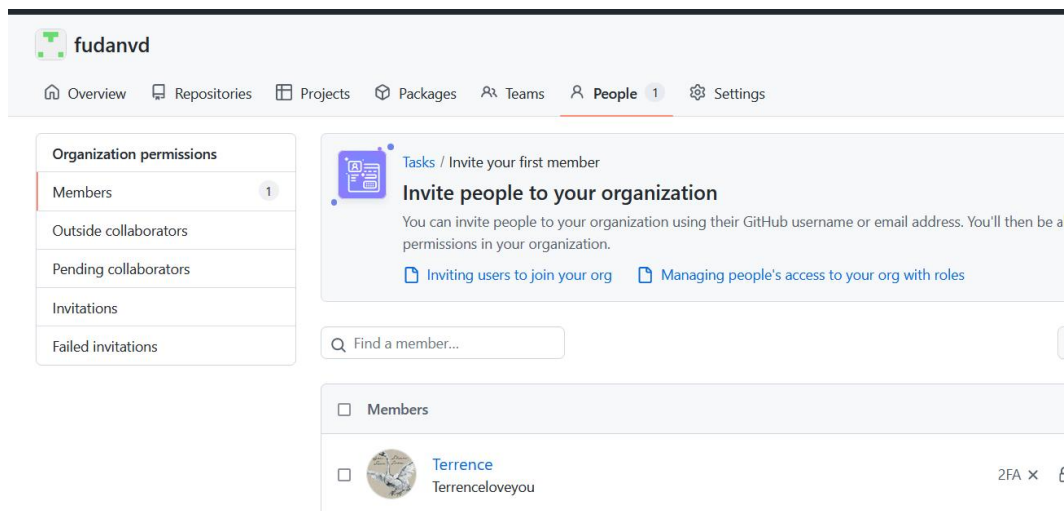
```
$ git merge <branch> #合并指定分支到当前分支
$ git rebase <branch> #衍合指定分支到当前分支
```

#### 远程操作

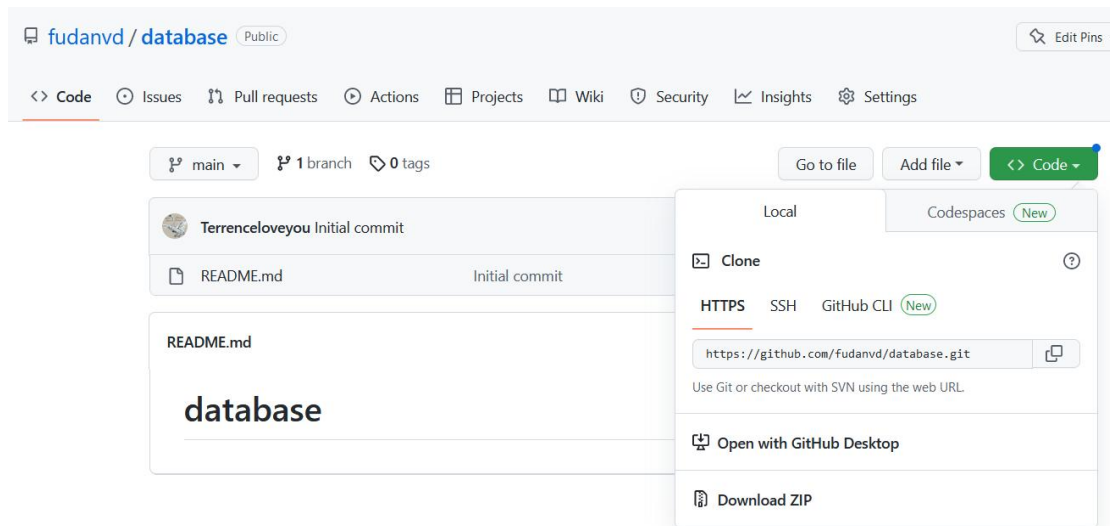
```
$ git remote -v #查看远程版本库信息
$ git remote show <remote> #查看指定远程版本库信息
$ git remote add <remote> <url> #添加远程版本库
$ git fetch <remote> #从远程库获取代码
$ git pull <remote> <branch> #下载代码及快速合并
$ git push <remote> <branch> #上传代码及快速合并
$ git push <remote> :<branch/tag-name> #删除远程分支或标签
$ git push --tags #上传所有标签
```

# Git Cheat Sheet <CN> (Version 0.1) # 2012/10/26 -- by @riku < riku@gitcafe.com / http://riku.wowubuntu.com >

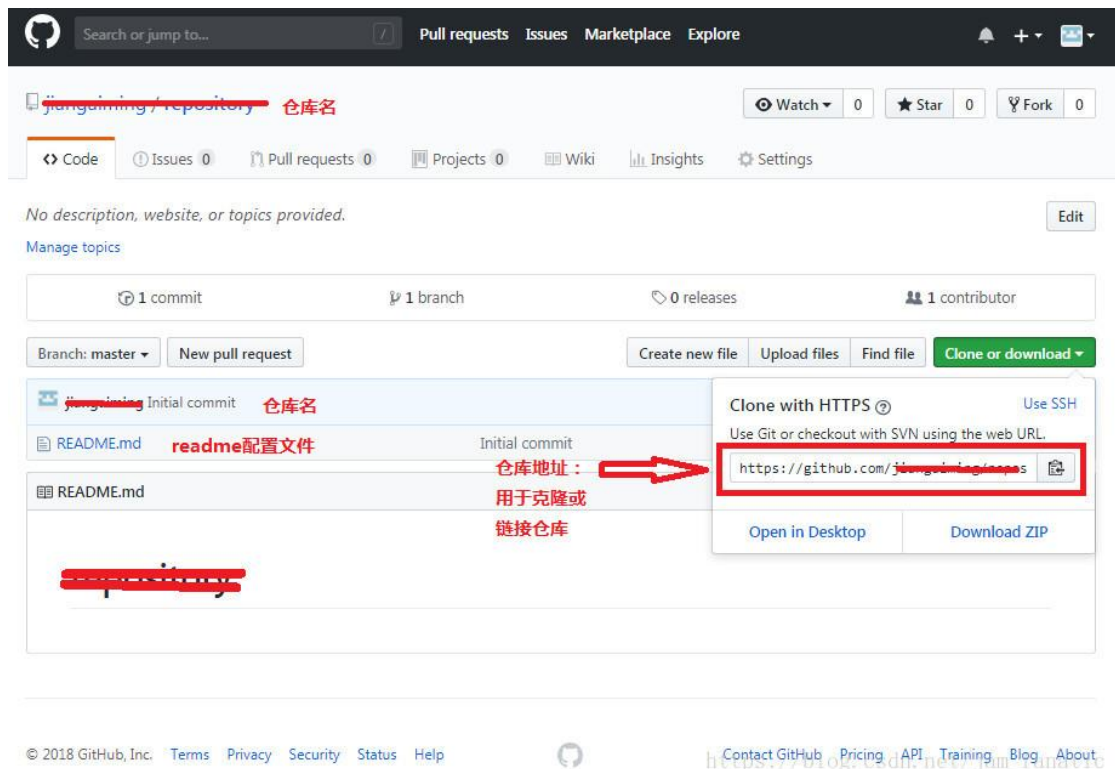
## ②github 设置



俺在 github 上面创建了一个 organization，大家可以把自己创建好的 github 账号/邮箱发给我，然后就可以邀请进来这个 organization（大家会收到一个确认的邮件），这样我们都可以编辑代码，就不需要很繁琐的 fork 之类的操作了。



然后我们在这里面创建了一个叫做 database 的仓库，我们所写的代码会在这里面存放，图中的仓库各个展示的信息可以参照下图中的注释来看。

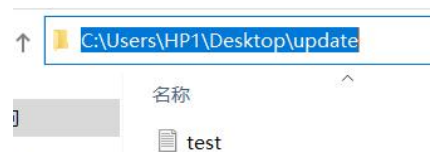


## 2. Organization 创建者上传自己的代码到 github

参考: <https://blog.csdn.net/gpwner/article/details/52829187>

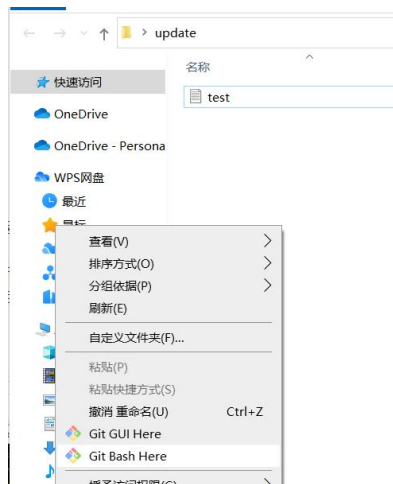
比如说我们想将一个 project 上传到 github 我们建好的仓库中, 那么需要用 git 的 push 操作:

假设我们要将这个文件传上去:



### ①初始化

在当前 project 所在的文件夹中右键点击 git bash here, 输入 \$ git init



这样在当前路径下初始化了一个 Git 仓库，在当前路径下还会多一个.git 目录（默认这个文件夹是隐藏的，参考

[https://blog.csdn.net/xiaomin\\_er/article/details/105591105](https://blog.csdn.net/xiaomin_er/article/details/105591105)）

update

名称	修改日期	类型	大小
.git	2023/1/14 14:40	文件夹	
test	2023/1/14 14:35	文本文档	1 KB

```

MINGW64:/c/Users/HP1/Desktop/update
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git init
Reinitialized existing Git repository in C:/Users/HP1/Desktop/update/.git/
  
```

（上图中我是输入了两遍 git init 所以重新初始化了）

## ②创建并配置 SSH

在刚刚的 gitbash 中继续输入：

```
$ ssh-keygen -t rsa -C "youremail@example.com"
```

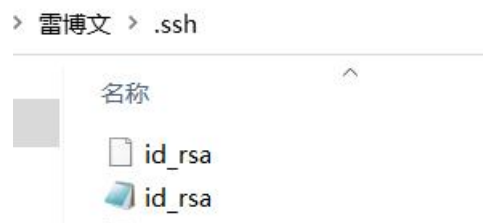
（youremail@example.com 是在 github 注册时候用的邮箱）

补全他给的路径，一直回车直到有这玩意出现：

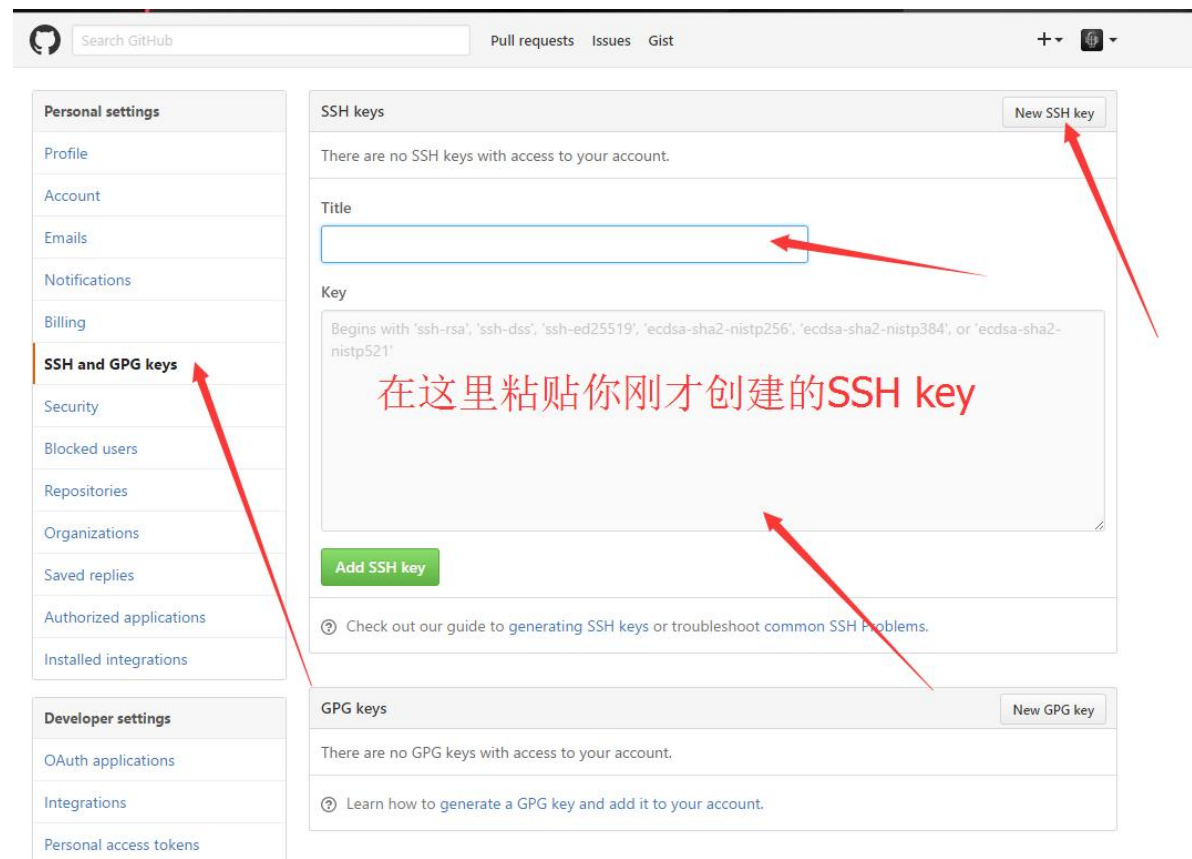
```

Enter file in which to save the key (/c/Users/HP1/.ssh/id_rsa): /c/Users/HP1/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/HP1/.ssh/id_rsa
Your public key has been saved in /c/Users/HP1/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zE3Q4v6IXcN+kWgi76or7p2ymS3afYPbR6tQxuxGaBA 1252275143@qq.com
The key's randomart image is:
+---[RSA 3072]---+
| E      ..      |
| .      ...     |
| .      ...     |
| . + o.o       |
| o *.S...      |
| . =. + * o     |
| ..O* O .      |
| .+*. =+ * o .  |
| .+B=O=+* .     |
+---[SHA256]-----+
  
```

然后在对应的的文件夹中出现了这两个文件：

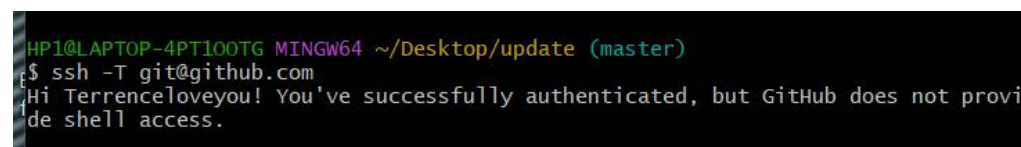


我们点开这个 pub 文件然后复制下来其中的东西，  
然后进入 github，进入 [github.com](https://github.com) -> 个人设置 -> SSH and GPG keys -> new SSH key -> add SSH，title 随便填，key 粘贴刚刚复制的 pub 文件中的内容



添加完毕后，我们验证一下是否成功：

输入：\$ ssh -T git@github.com



说明 ssh 添加成功

③连接仓库、上传文件

我们的仓库已经建立好了，然后我们设置一下 username 和 email，这样的话 github 每次就会记录上传的人

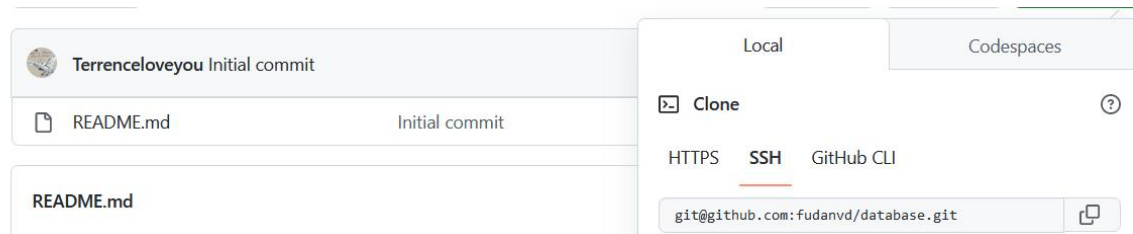


```
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git config --global user.name "Terrence"

HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git config --global user.email "1252275143@qq.com"
```

然后添加 git 的 ssh 远程地址（地址可以在 github 中建立的 database 仓库找到）：

```
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git remote add origin git@github.com:fudanvd/database.git
```



参照 push 用法，我们首先将远程仓库地址命名为 origin，这样下次就不需要很长的地址输入；然后我们使用几个操作就可以上传本地的 project：

输入命令行

```
$ git add .
```

//注意add后面是有"."的，而且和add之间有一个空格

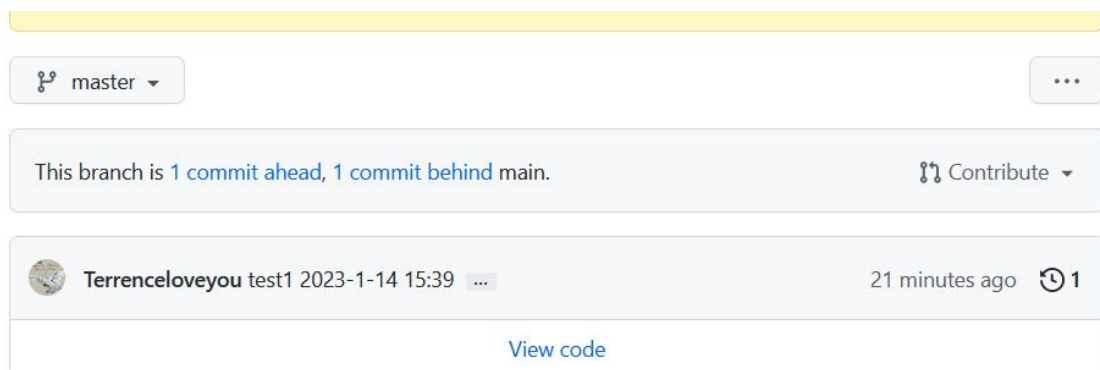
```
$ git commit -m "2016-10-16 10:49:29"
```

```
$ git push origin master
```

```
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git remote add origin git@github.com:fudanvd/database.git

HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/update (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 233 bytes | 233.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/fudanvd/database/pull/new/master
remote:
To github.com:fudanvd/database.git
* [new branch]   master -> master
```

然后我们上 github 看一看：



发现上传成功。如果你想单独 Push 一个文件，那么可以 `$ git add 文件名+文件后缀`。

需要注意的是，打开 github 之后会发现有两个 branch，一个是 github（在 2020 年改名）的默认分支 main，另外一个 git 默认的分支 master。而如果在 git 中 push 到 main 分支就会报错，相关信息和解决方法：

<https://blog.csdn.net/gongdamrgao/article/details/115032436>

（本来觉得两个 branch 无所谓，但是到后面越来越麻烦，于是手动把 github 上面的 master 删掉然后 main 改个名字，然后这个 git 就崩了，我也崩了...

花了很久很久找解决方案，最后结果就是 master 分支变成了主分支，然后其他的都没有变，然后我把 main 给删掉了，这样子就很顺畅了）

#### ④更新本地仓库

`git pull` 远程仓库地址 分支名称

参考 3 中写代码、上传代码的顺序，建议 organization 的创建者在修改代码前也先 pull 然后再修改上传。

### 3. 其他同学下载代码、上传代码

参考 <https://blog.csdn.net/Gpwner/article/details/53140016>

[https://blog.csdn.net/weixin\\_47505105/article/details/122893403](https://blog.csdn.net/weixin_47505105/article/details/122893403)

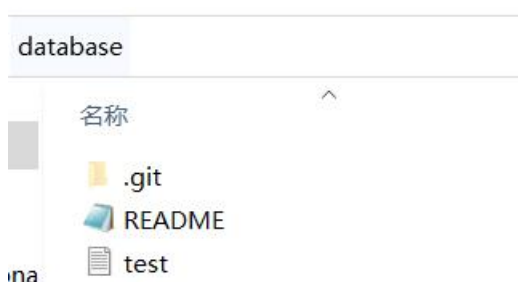
在项目发起者已经在 github 上面传完代码、各个队员已经在 organization 中协作人员里面、配置好了 ssh key，配置远程地址，设置用户名和邮箱（参照上面 2 中的配置方法）之后：

#### ①clone 操作

在 pc 随便位置打开 `git bash here`，然后输入：`git clone git@github.com:fudanvd/database.git` (git 的 ssh 远程地址，见上面的 2.③)，等待在当前目录下 clone 下来仓库中所有的东西，然后会发现出现了个 database 的文件夹

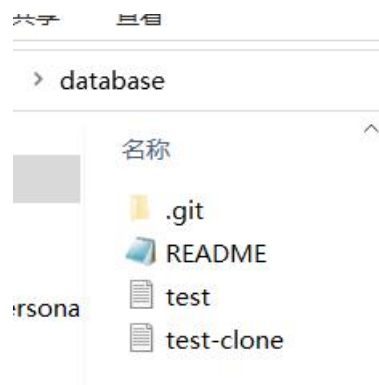
```
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop
$ git clone git@github.com:fudanvd/database.git
Cloning into 'database'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (7/7), done.
Receiving objects: 100% (11/11), done.
remote: Total 11 (delta 0), reused 8 (delta 0), pack-reused 0
```

在 database 文件夹中包含的内容和 github 中 master 的分支中内容是一样的（.git 文件默认不显示，见上面 2.①）



## ②更新代码

在 clone 下来的文件夹（含有.git 文件夹）中右键打开 git bash here  
在本地更新完代码后（我在本地加入了个 test-clone.txt 文件）：



一定要每次提交代码前都要先同步一下仓库（git pull），看看有没有其他人更新，否则无法上传代码。

下面的就是提交更新过的代码的操作：

git pull （更新仓库）

git add .

git commit -m "bbb" （""中间的内容是提交的 comment，写提交的时间即可）

git push

如果 git push、git pull 操作不成功，有可能是没有简化推送本地仓库到远程仓库的命令，参考：[https://blog.csdn.net/weixin\\_47505105/article/details/122893403](https://blog.csdn.net/weixin_47505105/article/details/122893403) 的 3 部分

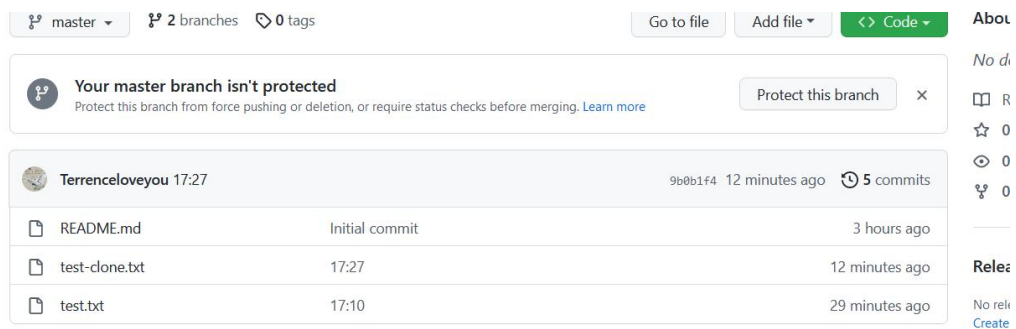
提交完成后，发现在 github 上面已经有了修改后的文件：

```
HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/database (master)
$ git add .

HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/database (master)
$ git commit -m "17:27"
[master 9b0b1f4] 17:27
1 file changed, 1 insertion(+)
create mode 100644 test-clone.txt

HP1@LAPTOP-4PT100TG MINGW64 ~/Desktop/database (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes | 320.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:fudanvd/database.git
ff5fdcc..9b0b1f4 master -> master
```





这样就算是提交完成了！

## 4. 冲突的情况

### ①冲突的发生

在多人协作的情况下其实冲突的发生概率还是蛮大的，冲突是怎么产生的：

“Git 其实是有自动合并功能的，比如 A 修改了 a 文件，B 修改了 b 文件，这两个修改互不相干，是可以自动合并的。但是 A 和 B 同时修改 a 文件，两人修改的地方不相同的话有些时候也是能自动合并，如果连修改的地方都相同，那程序就无能为力了，只能由人来确定到底应该保留谁的修改，这就是冲突。”

为了解决这种情况的发生，首先就是我们一定在修改代码之前记得先 pull 下来最新的版本；然后尽量不同的同学不要在同一段时间内修改同一个代码再提交，这样很容易冲突。

下图就是当我在本地修改了某个文件之后，另外有同学也修改了同一个文件并且上传了，而且我没有把他上传的文件 pull 下来，这时候我再去 pull 就会发生冲突：

```
HP1@LAPTOP-4PTI00TG MINGW64 ~/Desktop/update (master)
$ git pull origin master
From github.com:fudanvd/database
* branch          master      -> FETCH_HEAD
error: Your local changes to the following files would be overwritten by merge:
    test.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 9b0b1f4..37112bd
```

而且听闻还有可能出现如果本地和终端的文件内容不一样，pull 下来之后覆盖掉了本地的内容的情况发生，一定要注意这种情况！！！如果发现 push 不上去，需要重新 pull 的话，一定要借助下面给出的解决方法！！！！

### ②解决方法

1.听说借助 vs code 来打开仓库的话对应冲突的文件会有相应的解决方法：“保留当前更改”、“保留传入更改”……这样比较容易一些（我的 vs code 环境属实有点问题）

2.不借助其他 ide 的情况下，

参考：<https://blog.csdn.net/zhanglixin999/article/details/124356347>

输入：

git stash

git pull

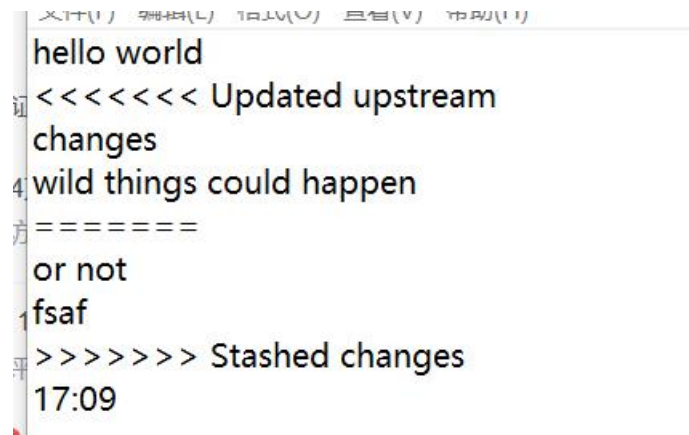
git stash pop

解析: git stash: 将改动藏起来

git pull: 用新代码覆盖本地代码

git stash pop: 将刚藏起来的改动恢复

这样操作的效果是在最新的仓库代码的基础仍保留本地的改动。



```
hello world
<<<<<< Updated upstream
changes
4 wild things could happen
=====
or not
1 fsaf
>>>>>> Stashed changes
17:09
```

上图是操作完之后 pull 下来的版本，其中对应：

从<<<<<<到=====中间的内容是 pull 下来的版本

=====到>>>>>>中间的内容是存储的自己本地修改的版本，这样可以对应进行手动的处理这些冲突。

解决完冲突之后再 push 上传到 github 中即可。

如果不保留本地的修改，直接覆盖的话输入：

git reset --hard

git pull