

GitHub 操作流程

--- Introduction on how to use Github

Peng Wang

Notes before you start to read:

- Months ago I found blogs or something in Chinese introducing Github to less experienced people like me, and basically, I just grab things here and there and put them together to complete “Section 1: Introducing Github to Dummies”. I would thank all these people who shared their experience to us, which indeed helped with my research. It also inspired me to share stuff with people, and that’s why this document is here today.
- “Section 2: Experience I have learned” are something I have learned while use Github, mainly for users of “Githun bash” (sorry I have always thought command lines are so gentle and these dialog based techniques just make us, especially researchers stupid).
- The first section will be in Chinese, the second section will be in English.
- Please contact me via grapesonwang@gmail.com for further discussion.

Section 1: Introducing Github to Dummies

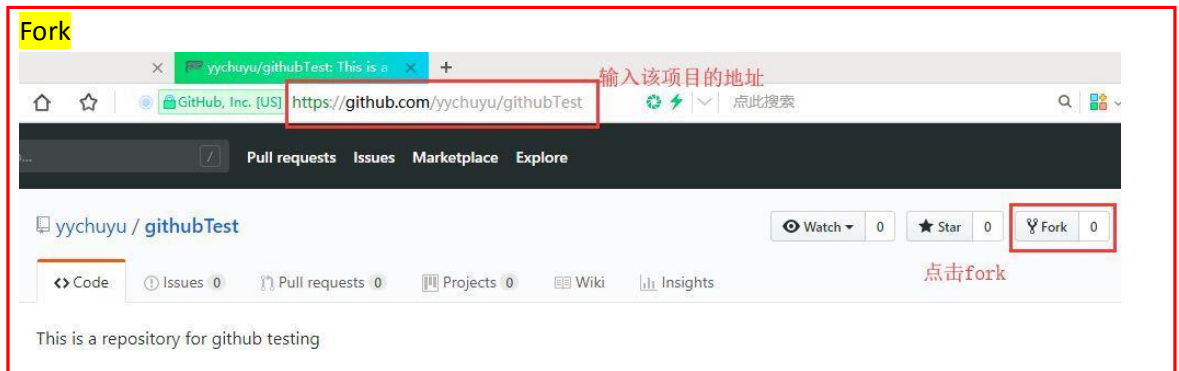
1. 存在本地版本和远程版本，为便于说明，这里假设远程版本为来自 Alvin 托管在 github 上的项目 githubTest，项目的地址为 <https://github.com/yychuyu/githubTest>。



2. 现在呢，Harry 觉得这个项目蛮有意思的，于是要么想基于该项目进行开发，要么纯粹出于兴趣，想要给项目做一点贡献。怎么做呢？

3. 前面已经说了，分为本地和远程。大概思路是：为便于多人同时工作，一般要求每个人工作在不同的分支上，然后将自己的改进 commit 到主分支上。为做到这一点，一般的步骤如下：

3.1 Harry 从 Alvin 的 repo 中 fork 一份到自己的 repo 中，这个时候，Harry 的 repo 中存在了一份 githubTest。此时的操作是在 remote 即远程进行的



3.2 此处有一个安装 git 的过程，[git download](#)

3.3 安装完了后，打开 git bash prompt，分别进行如下配置：

- git config --global user.name “Your GitHub Name”
- git config --global use.email “Your Email Address”

3.4 为了便于 Harry 进行贡献，他需要将自己远程 repo 中的代码 clone 到本地，指令一般是：git clone URL。但是这一步需要分解如下：

clone

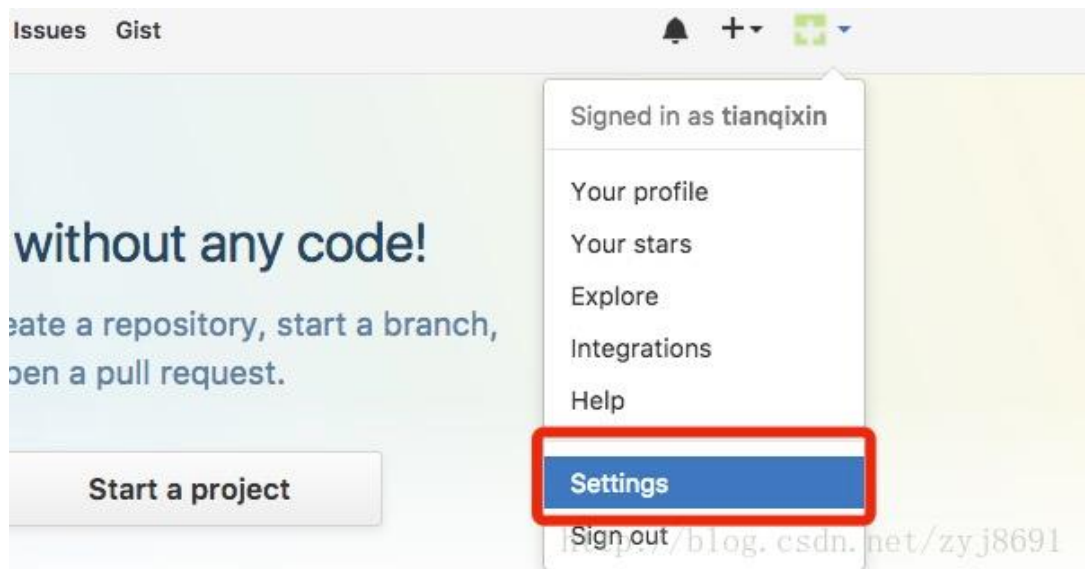
通过 fork 之后，Harry 的账号下也有了 githubTest 这个项目，但还不能对它进行编译、修改（其实是可以修改，但是不建议）

```
[harry@VM_0_16_centos ~]$ git clone git@github.com:automationroad/githubTest.git
Cloning into 'githubTest'...
Warning: Permanently added the RSA host key for IP address '192.30.253.112' to the list of known hosts.
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
```

- 创建本地仓库 H_githubTest（本质上是一个文件系统）；
- cd H_githubTest，然后 git init（此时根目录下有一个隐藏的.git 的文件夹，本地仓库初始化成功），此时指令后面会出现（master）
- 下一步需要将本地的仓库 H_githubTest 和 Harry 远程 repo 中的仓库进行关联。指令是：git remote add origin [git@github.com:Harry's GitHubName/githubTest.git](#)（也即目标 repo）
- 为了完成关联，非常直观的，本地和远程仓库之间需要有一个“握手协议”，这个一般是通过密钥来完成的。创建密钥的指令如下：（个人感觉出于安全考虑，密钥会被独立的放在 ~/.ssh 文件夹中）
 - ✓ ssh-keygen -t rsa -C “Your Email Address”
 - ✓ 一路回车，默认在 id_rsa 文件里面生成 ssh key

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/lilu/.ssh/id_rsa.
Your public key has been saved in /c/Users/lilu/.ssh/id_rsa.pub.
The key fingerprint is:
ed:49:a5:6e:49:cf:e1:72:6f:fa:4a:60:81:b8:9a:3e lilu@1ke.co
The key's randomart image is:
+--[ RSA 2048 ]-----+
|
|      . .
|     . . .
|    . . +
|   . S B .
|  o  * B .
| o   B *
| .E   . + .
|  . .   o=o
+-----+
http://blog.csdn.net/sihai12345
```

- ✓
- ✓ 可以 `cd ~/.ssh` 查看所生成的文件。
- ✓ 打开 `id_rsa.pub`，复制里面的密钥。两种方式进行打开：1. 用文本阅读器打开；2. 指令：`clip < ~/.ssh/id_rsa.pub`
- ✓ 将密钥添加到远程仓库中



- ✓
- ✓ 然后

SSH keys New SSH key

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

ⓘ Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).

- ✓
- ✓ Title 可以取得比较随意，将复制的密钥粘贴到 Key 中，然后“Add SSH Key”。搞定

3.5 接下来便是本地的操作了。包括对文件进行改进（删改），或者新建文件等，完成后需要将‘改变’commit 到远程去。该过程一般可以分解为以下一些内容：

update a file & 4. commit

接下来，Harry 就可以大显身手了，可以自由对这个项目进行修改。

但是，不建议在 master 分支直接修改，建议在 master 分支基础上切出一个 dev 分支，然后在 dev 分支上自由发挥。修改完之后，再将 dev 分支 merge 到 master 分支。

```

harry@alvin-pc:~/githubTests$ git checkout -b dev
Switched to a new branch 'dev'
harry@alvin-pc:~/githubTests$ echo This file is added by Harry >> c.c
harry@alvin-pc:~/githubTests$ git add c.c
harry@alvin-pc:~/githubTests$ git commit -m "Harry add c.c"
[dev bd352a5] Harry add c.c
 1 file changed, 1 insertion(+)
 create mode 100644 c.c
harry@alvin-pc:~/githubTests$ git status
On branch dev
nothing to commit, working tree clean
harry@alvin-pc:~/githubTests$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
harry@alvin-pc:~/githubTests$ git merge dev
Updating 8f1af3b..bd352a5
Fast-forward
 c.c | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 c.c

```

- 向仓库添加文件
 - ✓ 创建文件 touch 1.cpp
 - git status//查看状态

- ✓ 添加到暂存区
`git add 1.cpp`
 - ✓ 将文件从暂存区提交到仓库
`git commit -m 'add 1.cpp'`
 - 修改仓库文件
 - ✓ 修改 1.cpp 内容
`cat 1.cpp` //可以查看内容
 - ✓ 添加到暂存区
`git add 1.cpp`
 - ✓ 将文件从暂存区提交到仓库
`git commit -m 'change 1.cpp'`
 - 删除仓库文件
 - ✓ 删除文件
`rm -rf 1.cpp`
 - ✓ 从 Git 中删除文件
`git rm 1.cpp`
 - ✓ 提交操作
`git commit -m '删除 1.cpp'`
4. 将本地仓库的更新添加到远程仓库。指令一般是：`git push origin`（远程） `master`（本地）

push

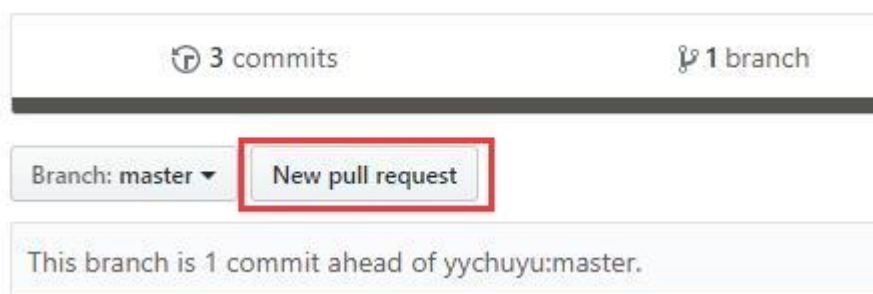
在自己的电脑上修改好代码之后，再使用 `git push` 命令将改动同步到自己的 GitHub 项目仓库里。

```
harry@alvin-pc:~/githubTests$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:automationroad/githubTest.git
 8flaf3b..bd352a5 master -> master
```

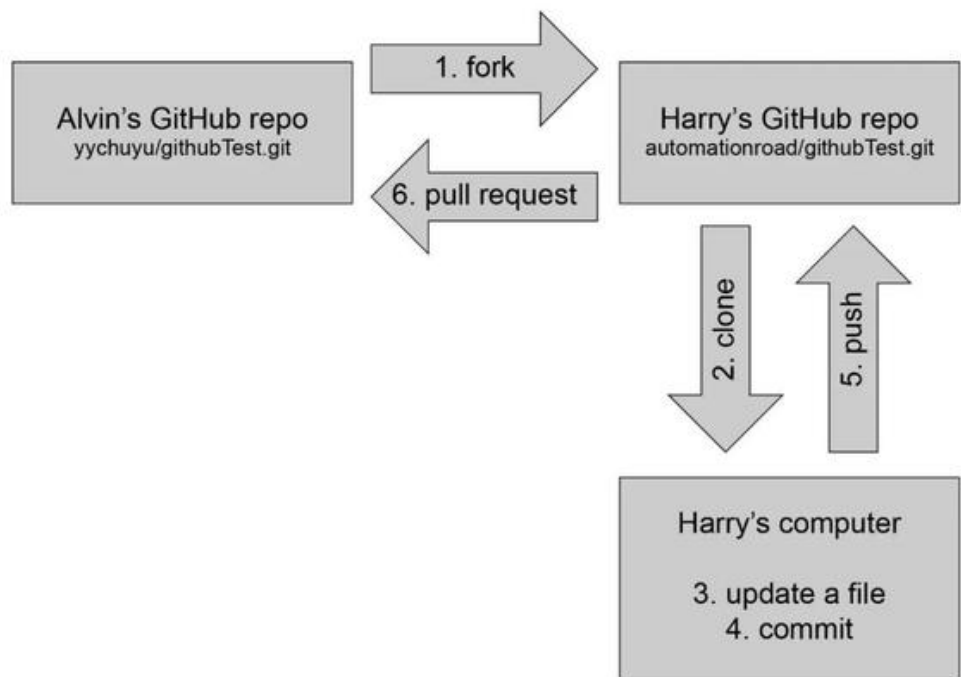
5. 然后 Harry 将自己的更新 merge 到原作者 Alvin 那边，通过 pull request。

pull request

接下来，就是向原作者 Alvin 提交你的代码了。首先点击文件列表上的「New pull request」。



整个流程大概如下表所示：



后续的一些说明：

之后，GitHub 会自动对源仓库分支（Alvin 的仓库，或者源作者的仓库）及自己仓库（Harry 的仓库，或者说别人 fork 出来的仓库）分支代码进行比对，看看是否有冲突。如果它显示「Able to merge」的话，Harry 就可以点击下面的「Create pull request」绿色按钮，进行代码提交。

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 fork: yychuyu/githubTest

base: master

head fork: automationroad/githubTest

compare: master

源仓库及分支 自己仓库及分支

✓ Able to merge. These branches can be automatically merged.

Create pull request

Discuss and review the changes in this comparison with others.

创建一个提交申请 (pull request)

1 commit

1 file changed

0 commit comments

1 contributor

Commits on Oct 02, 2018

Harry Harry: add c.c bd352a5

Showing 1 changed file with 1 addition and 0 deletions.

Unified Split

View

1 c.c

@@ -0,0 +1 @@

1 + This file is added by Harry

提交的文件内容

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 fork: yychuyu/githubTest

base: master

head fork: automationroad/githubTest

compare: master

✓ Able to merge. These branches can be automatically merged.



Harry add c.c

Write

Preview

AA B i “ <> ↻ ☰ ☷ ☶ @ 📎 ↶

Dear Alvin,

I made a niubility change, please merge it, thx!

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

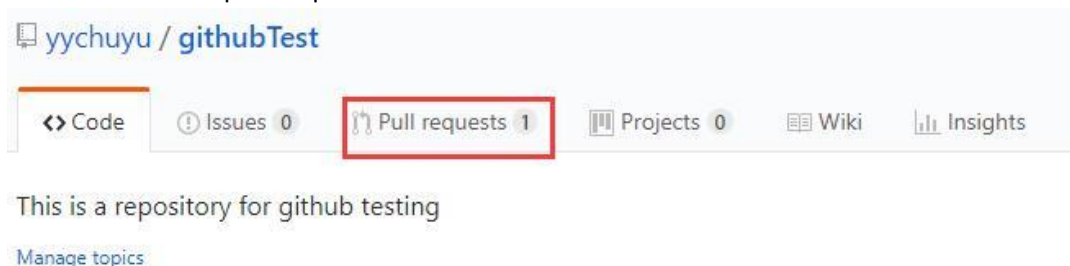
☒ Allow edits from maintainers. [Learn more](#)

Create pull request

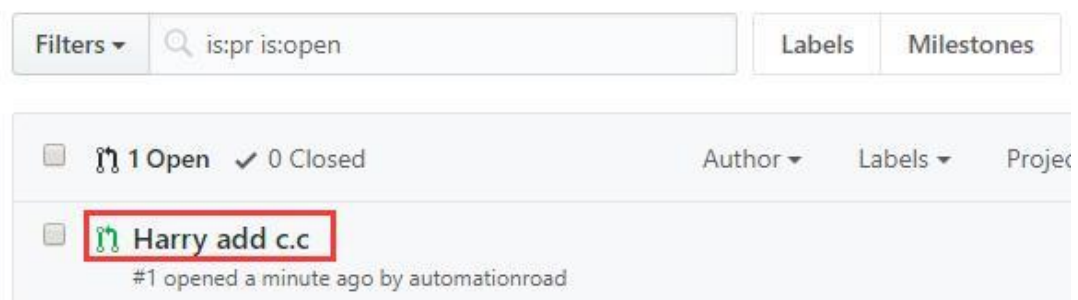
再之后，系统会要求你写一段注释，其实也可不写。但最好写一下，跟作者说明一下你改动了什么，为什么这么改。

现在就到了源作者 Alvin 这里了，前面都是 Harry 进行的操作：

1. Alvin 会收到一个 pull request，如下图




2. 然后，Alvin 可以点进去，看看 Harry 具体提交了一些什么修改。如果他觉得这个修改确实够 niubility 的话，它就可以点击「merge pull request」，将 Harry 的提交集成到自己的项目里。



Conversation 0Commits 1Checks 0Files changed 1

提交信息

文件改动



automationroad commented 4 minutes ago


First-time contributor + 😊 ...

Dear Alvin,
I made a niubility change, please merge it, thx!

Harry add c.c

bd352a5

Add more commits by pushing to the master branch on automationroad/githubTest.



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

合并Harry的提交到当前分支

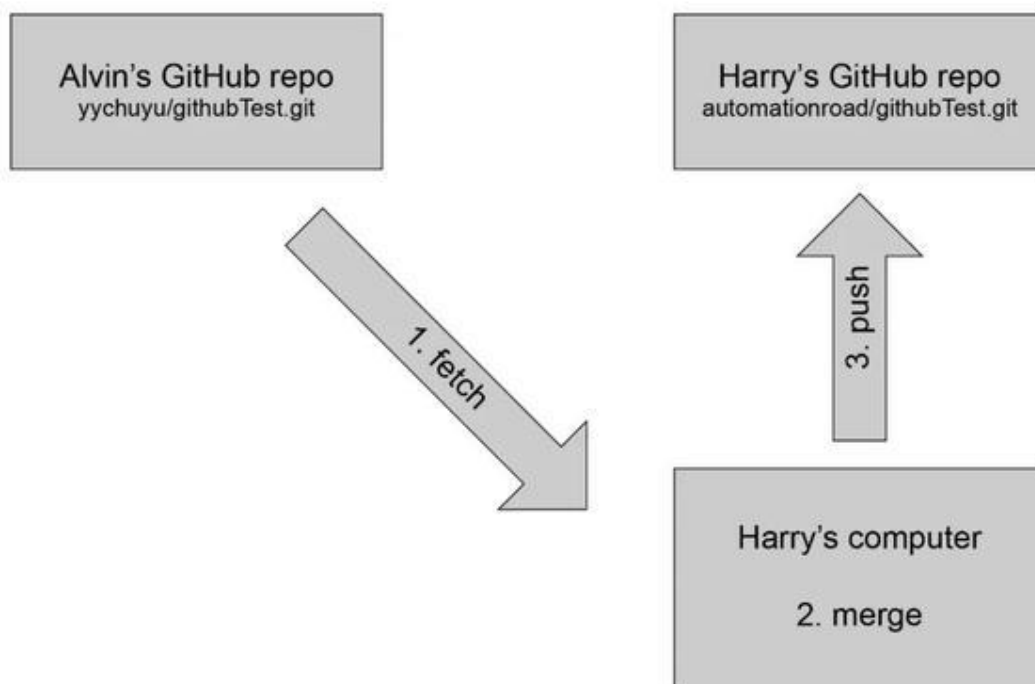
You can also open this in GitHub Desktop or view command line instructions.

至此，功德圆满，Harry 顺利完成一次对项目 githubTest 的代码贡献。

但是，这个项目实在太出众了，很多高手看到了并贡献了众多代码。比如现在 Alvin 自己提交了一个文件：

yychuyu Alvin add d.c	Latest commit bb412ad 35 seconds ago	
README.md	Initial commit	2 hours ago
a.c	init commit	2 hours ago
b.c	init commit	2 hours ago
c.c	Harry add c.c	24 minutes ago
d.c	Alvin add d.c	34 seconds ago

现在原作者项目已经发生了改变，那 Harry 账号下的 githubTest 如何与原作者 Alvin 的项目保持同步呢？Harry 需要做以下三步操作：



fetch

现在代码不同步了，我们要先把 Alvin 仓库的代码 **fetch** 到自己电脑的仓库下。注意，这是在自己电脑上操作，不是在 GitHub 上操作。指令如下：

```
git fetch git@github.com:yychuyu/githubTest.git master:latest
```

上面这条命令，**git fetch** 之后的那部分，是原作者 Alvin 项目 **git** 地址，通过点击原项目「clone or download」按钮可以看到。再之后 **master:latest** 这部分，**master** 是原项目分支，**latest** 是自己项目分支。

如果 **latest** 分支不存在的话，将自动创建。其实也可以将代码 **fetch** 到自己的 **master** 分支，但也不建议这么做。

merge

代码 **fetch** 到 **latest** 分支之后，再切到 **master** 分支，再使用 **git merge** 命令将最新代码合并到 **master** 分支。

push

现在，Harry 电脑上的代码与原项目代码保持同步了。我们再使用 **git push** 命令，就可以将最新代码推到 Harry 账号下 **githubTest** 项目里。

以上的三个步骤具体操作过程如下图示：

```

harry@alvin-pc:~/githubTests$ git fetch git@github.com:yychuyu/githubTest.git master:latest
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), done.
From github.com:yychuyu/githubTest
* [new branch]      master -> latest
harry@alvin-pc:~/githubTests$ git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.
harry@alvin-pc:~/githubTests$ git merge latest
Updating bd352a5..bb412ad
Fast-forward
 d.c | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 d.c
harry@alvin-pc:~/githubTests$ git push

```

接下来，Harry 就可以在此基础上，继续贡献自己的代码了。

Section 2: Experience I have learned

1. Large file storage

If you have done exactly according to Section 1, everything would be all right when you use Git/Github. But unfortunately, I have to upload big files (over 100 MB) to Github, which causes problems because the default maximum file size for Github is 100MB. Following is the problem when I tried to push files bigger than 100MB (203MB actually) to the remote repository.

```

$ git push -u origin master
Enumerating objects: 98, done.
Counting objects: 100% (98/98), done.
Delta compression using up to 8 threads
Compressing objects: 100% (93/93), done.
Writing objects: 100% (98/98), 449.00 MiB | 34.91 MiB/s, done.
Total 98 (delta 11), reused 0 (delta 0)
remote: Resolving deltas: 100% (11/11), done.
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
remote: error: Trace: 9bb8211e509e43ca0006b587590e828e
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File DataFromMatlab/asy_mat.xlsx is 203.62 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: File DataFromMatlab/text.xlsx is 204.29 MB; this exceeds GitHub's file size limit of 100.00 MB
To https://github.com/grapesonwang/Traffic_Gaussian_Process.git
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'https://github.com/grapesonwang/Traffic_Gaussian_Process.git'

```

There are two ways to solve the problem:

1. First way to solve the problem

✓ `rm -rf .git`

```

$ rm -rf .git
uos@acse9815 MINGW64 ~/Desktop/Google Drive PW/Mendeleev/GitHub/Traffic
$ git status
fatal: not a git repository (or any of the parent directories): .git

```

✓

✓ After this, everything in '.git' would be deleted. Because the repository initialization configuration is stored in '.git' folder, when execute the command 'rm -rf .git', the repository would be deleted (not necessarily the folder, but the configuration of the repository). The folder will no longer be a repository, you have to do 'git init' to make it a repository again.

✓ `git init`

- ✓ Then you have a brand new repository, you have to do the whole process that you deal with a new repository, which are
- ✓ `git remote add origin`
git@github.com:grapesonwang/Traffic_Gaussian_Process.git (or
https://github.com/grapesonwang/Traffic_Gaussian_Process.git)
- ✓ `git add .`
- ✓ `git commit -a -m "comments for the commision"`
- ✓ `git pull origin master`
- ✓ `git push -u origin master`
- ✓

2. *Second way to solve the problem (This relates with the large file storage (lfs))*

- ✓ You have to install the large file storage tool first, please refer to
["https://git-lfs.github.com/"](https://git-lfs.github.com/)
- ✓ `git lfs install` (you need to run this in each of your repository directory, once per repository)
- ✓ Configure large file extensions :
- ✓ `git lfs track "*.mat"`
- ✓ `git lfs track "*.xlsx"` etc., you can configure additional file extensions anytime
- ✓ `git add .gitattributes`
- ✓ Then things will be more or less the same. Just bear in mind that when you push big files, it might take a while.

3.

2.