



农业农村部食物与营养发展研究所
Institute of Food and Nutrition Development , Ministry of Agriculture and Rural Affairs

Github 上传项目教程

(课题组内部资料)

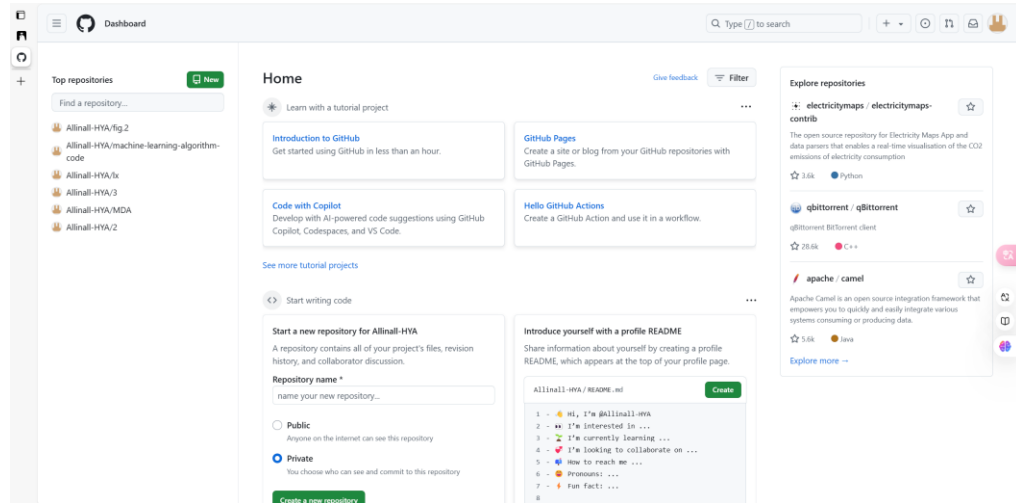
作 者：郝赢奥

时 间：2024.11.27

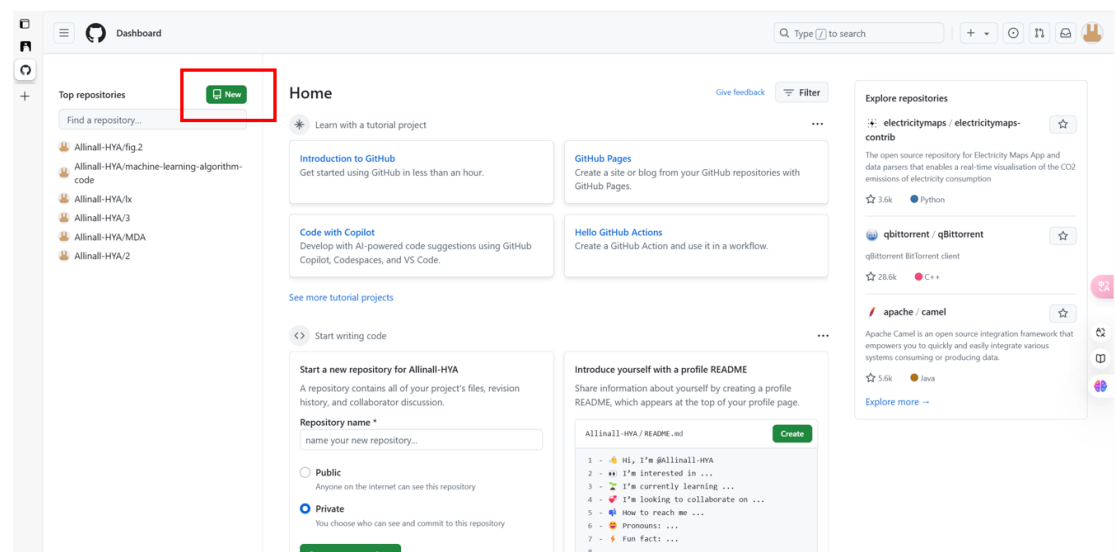
任广旭课题组制

一、注册 Github

可以登录 Github 官网 <https://github.com/>，使用邮箱申请一个 Github 账号，记住账号和密码，然后登陆，进入这个页面



二、创建 Github 项目



点击 New 跳转到创建仓库页面，填写一下相关的信息，同时关注一下自己的账号信息，包括邮箱和用户名

Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name * 填写新建仓库的名字

Great repository names are short and memorable. Need inspiration? How about [symmetrical-octo-winner](#)?

Description (optional) 这个描述可以描述一下上传文件都是什么，也可以不填

☒ Public Anyone on the Internet can see this repository. You choose who can commit.
☐ Private Only those who can see and commit to this repository.

Initialize this repository with ☒ Add a README file 这个README可以勾选一下，描述文件，让查阅的人知道如何使用你的文件
This is where you can write a README file.

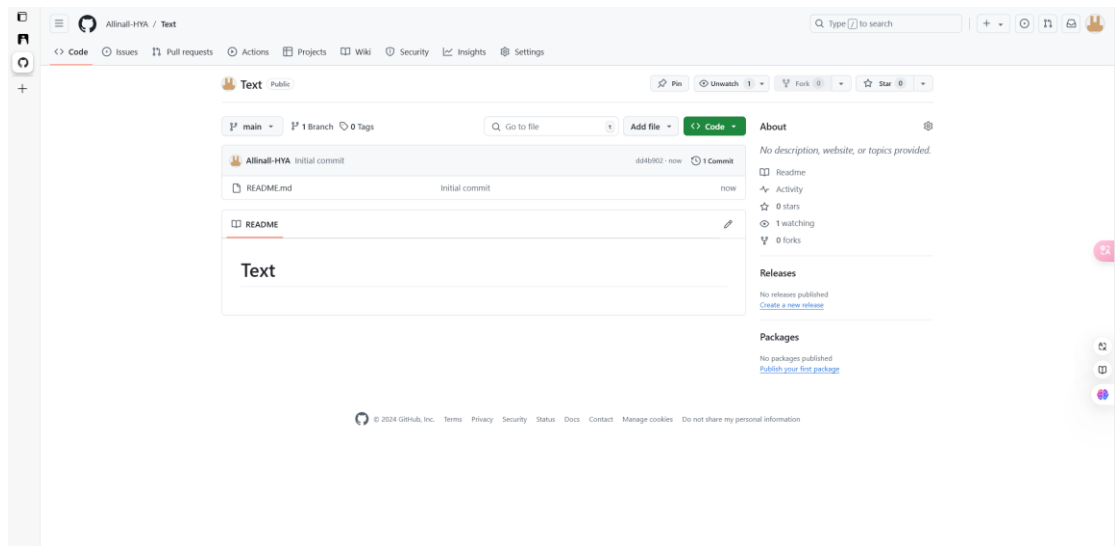
Add .gitignore
gitignore template: [None](#)
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: [None](#)
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)
This will set `main` as the default branch. Change the default name in your [settings](#).

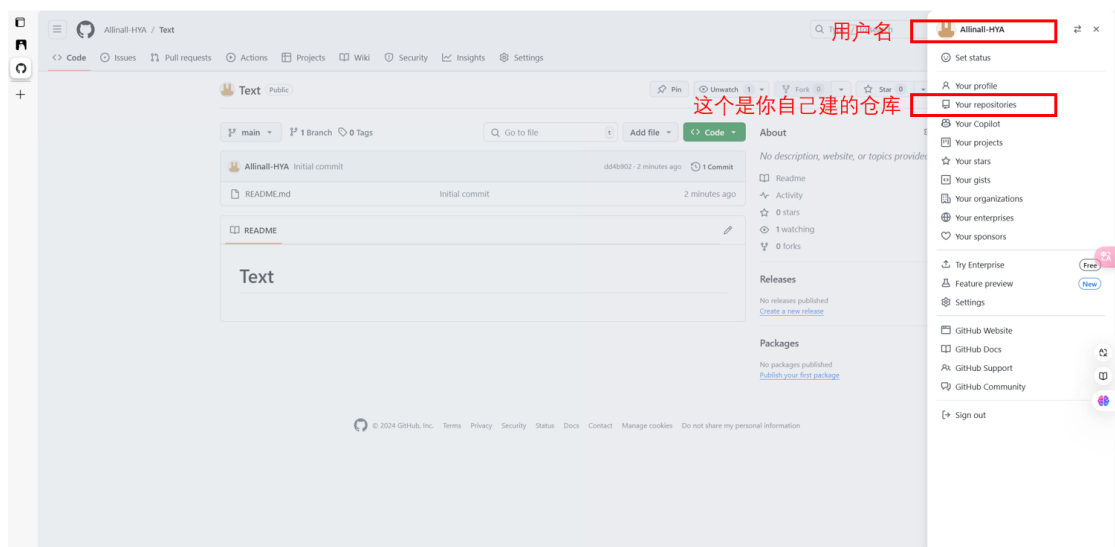
☐ You are creating a public repository in your personal account.

点击就可以创建好了

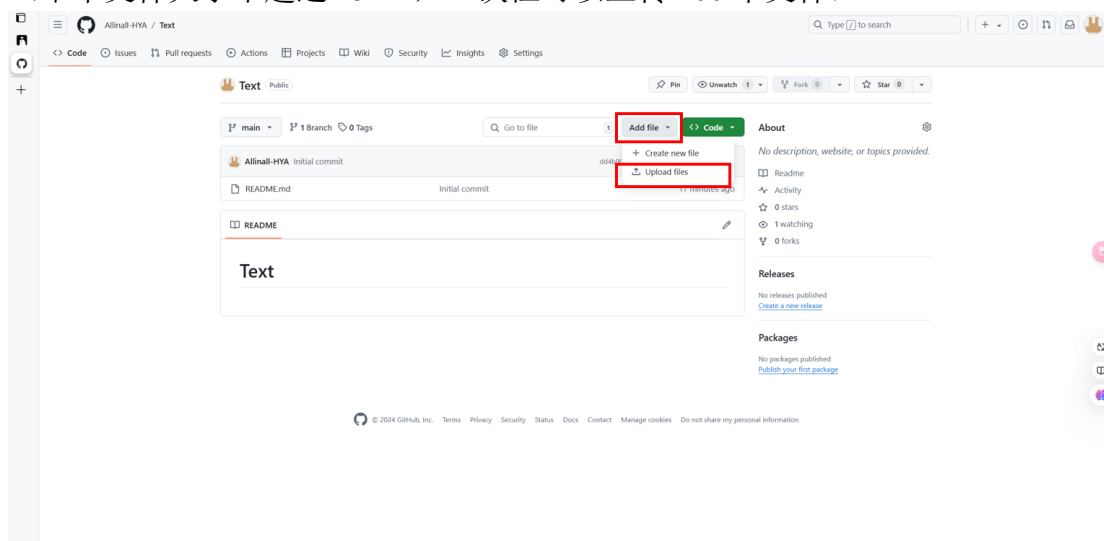
创建之后长这个样子



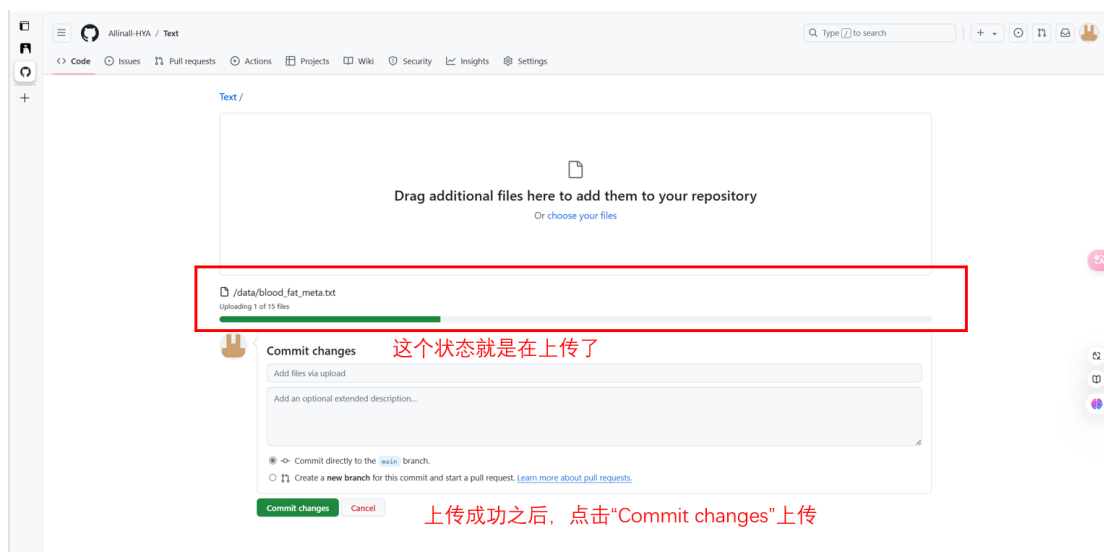
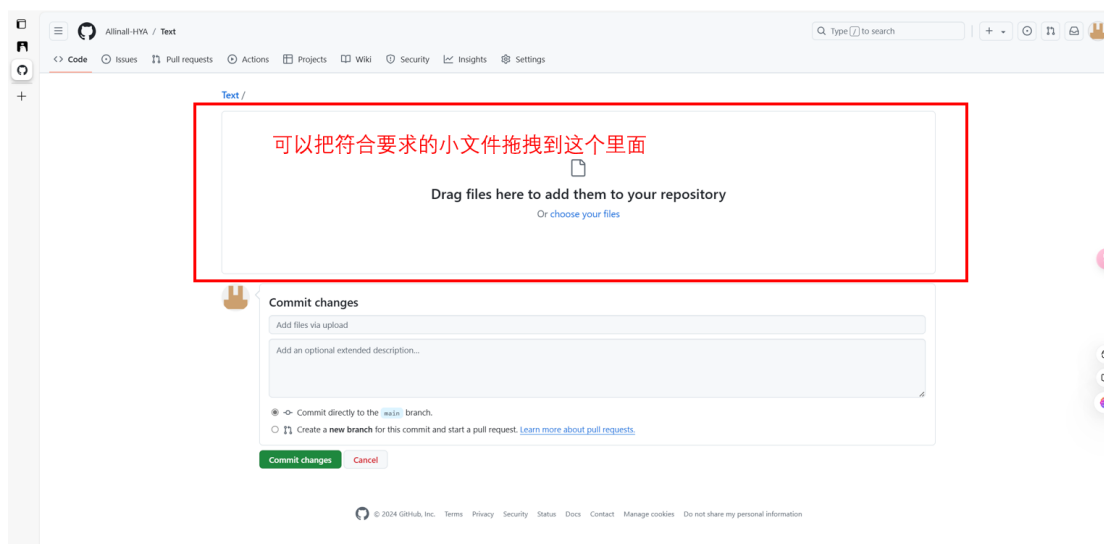
这是一个空的仓库，点击右上角的头像，能显示出你的用户名和一些设置选项，这里只列举了常用的，其他的大家自己探索吧



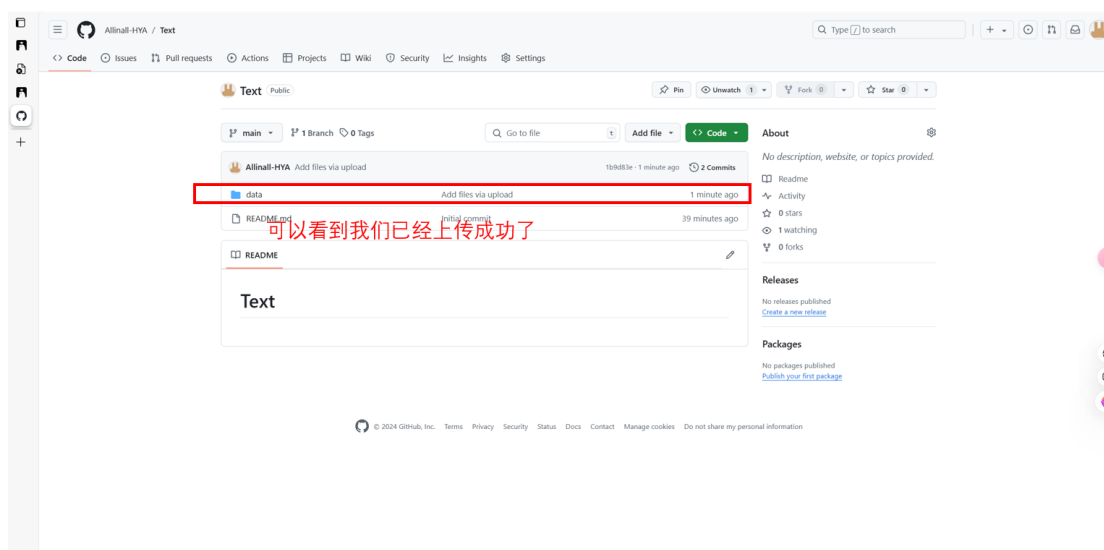
点击“Add file”可以出现两个选项，点击“Upload files”可以上传一些小的文件（单个文件大小不超过 25Mb，一次性可以上传 100 个文件）



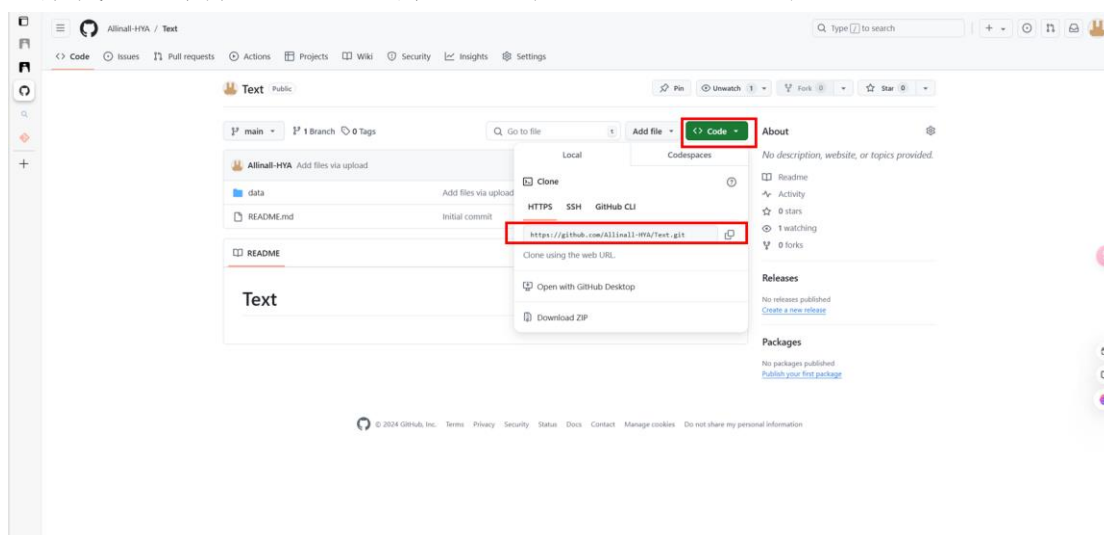
点击后跳转到下一个上传页面



上传成功之后会自动跳转到仓库，可以看到我们上传的文件



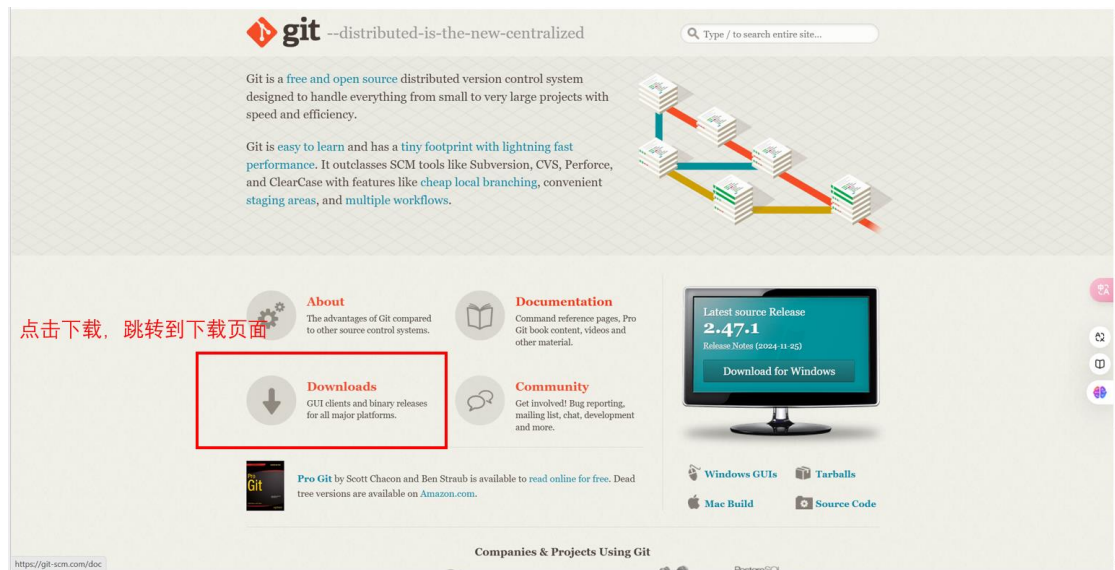
当然这种方法只适用于小文件，如果文件大小超过了 25Mb 怎么办，我们可以通过敲代码命令符的形式去上传，接下来我给大家进行一个演示



点击“Code”会弹出一个框，我们需要关注一下“HTTPS”选项卡下面的网址

二、构建 Git 环境

首先，第一次选择上传文件到 Github，我们需要在电脑上构建一个 Git 环境，可以浏览器访问 Git 官网 <https://git-scm.com/>，点击“Downloads”跳转到下载页面，根据自己使用的电脑选择下载的版本（以 win 为例），下载“64-bit Git for Windows Setup.”版本，下载后安装，安装就一直无脑点下一步就可以了，这里就不展示了，下载之后如果找不到可以在电脑里检索“git”，并把它固定在开始任务栏里，这样就可以找到它了

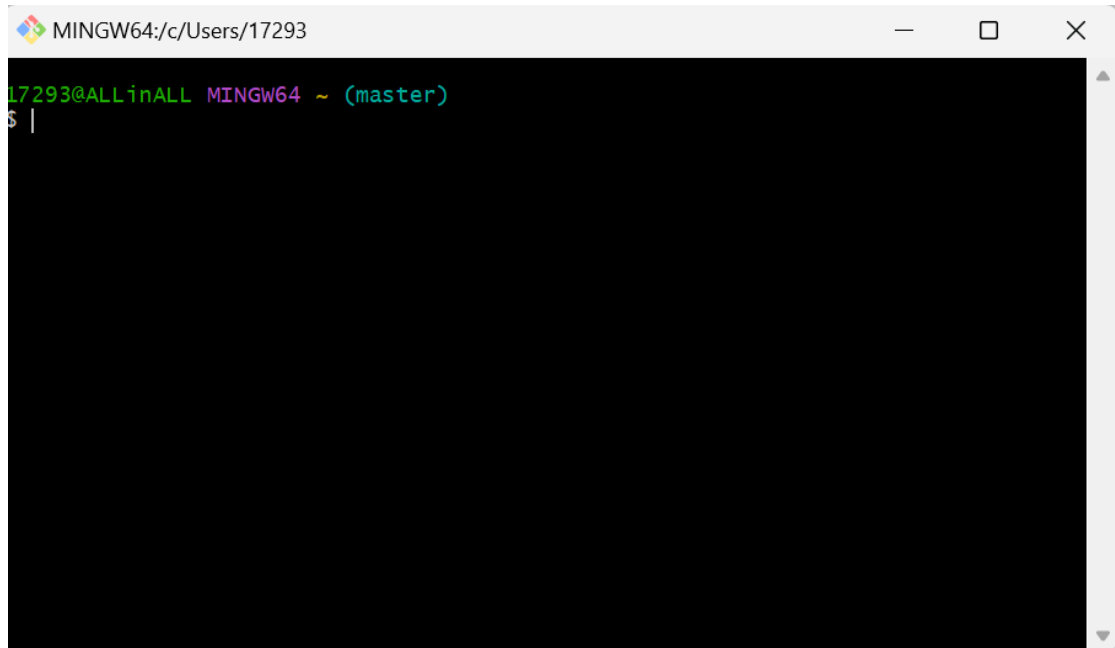


三、本地环境配置

在这个部分，我们就需要用代码命令符来配置本地环境了，这个过程其实比

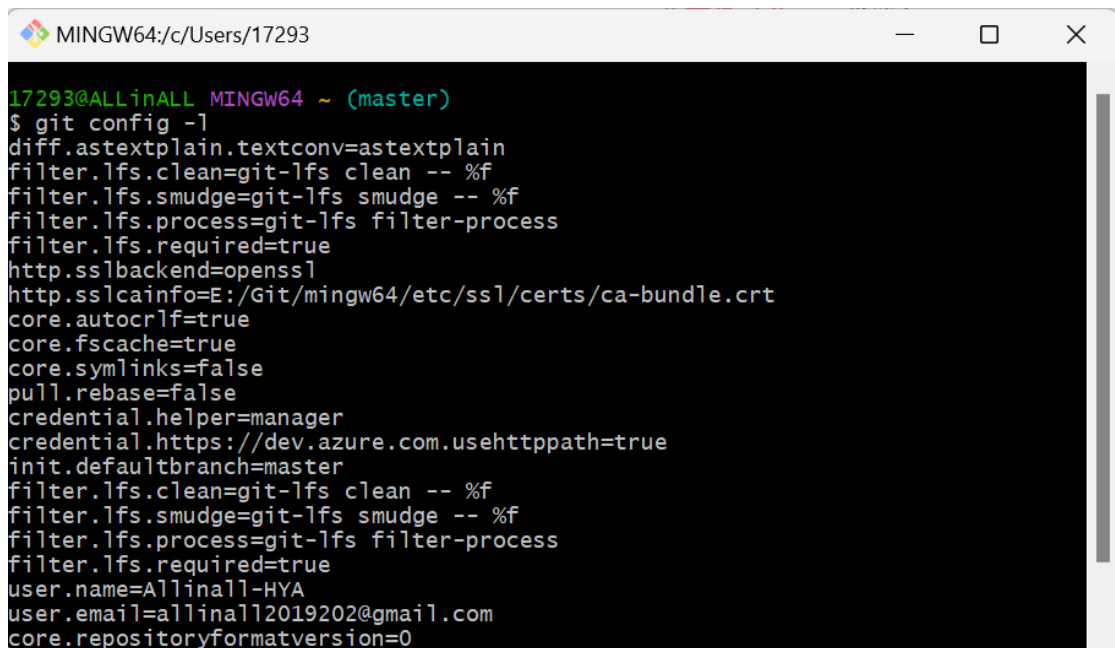
较简单,但是特别容易出现报错,那么我们就根据他的提示去修改我们的指令,接下来我就展示一下具体的代码操作

点开“Git Bash”



可以先看一下它的配置情况,输入以下代码:

git config -l



这些是系统自动弹出的配置信息,上传文件的时候需要与你的 Github 进行连接,需要输入你的用户名和邮箱,输入以下命令可以实现:

git config --global user.name “Allinall-HYA”

git config --global user.email “allinall2019202@gmail.com”

```
MINGW64:/c/Users/17293

17293@ALLinALL MINGW64 ~ (master)
$ git config -global user.name "Allinall-HYA"
git config -global user.email "allinall2019202@gmail.com"
error: key does not contain a section: -global
error: key does not contain a section: -global

17293@ALLinALL MINGW64 ~ (master)
$ git config --global user.name "Allinall-HYA"
error: key does not contain a section: --global

17293@ALLinALL MINGW64 ~ (master)
$ git config --global user.name "Allinall-HYA"

17293@ALLinALL MINGW64 ~ (master)
$ git config --global user.email "allinall2019202@gmail.com"

17293@ALLinALL MINGW64 ~ (master)
$
```

配置完用户名跟邮箱后，再次输入

git config -l

检查配置

```
MINGW64:/c/Users/17293

diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=E:/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name="Allinall-HYA"
user.email="allinall2019202@gmail.com"
core.repositoryformatversion=0
core.filemode=false
core.bare=false
:
```

可以看到我们的信息已经搞好了（我这个跟之前一样是因为我弄过了），这样就配置好了，下一步我们需要配置一下公钥和私钥，这是与 GitHub 链接所需要的，我们要输入：

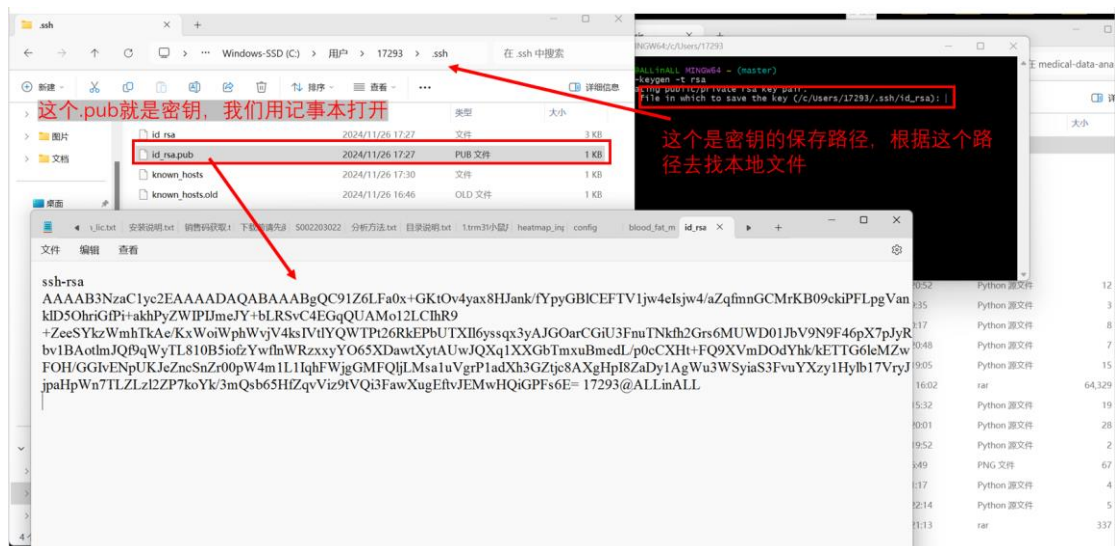
ssh-keygen -t rsa

输入之后直接摁回车就可以，出现以下界面就说明密钥创建好了（ps.这张图片是引用 UP 主的，我的已经创建好了，在后面我会放上 UP 主的教学视频）

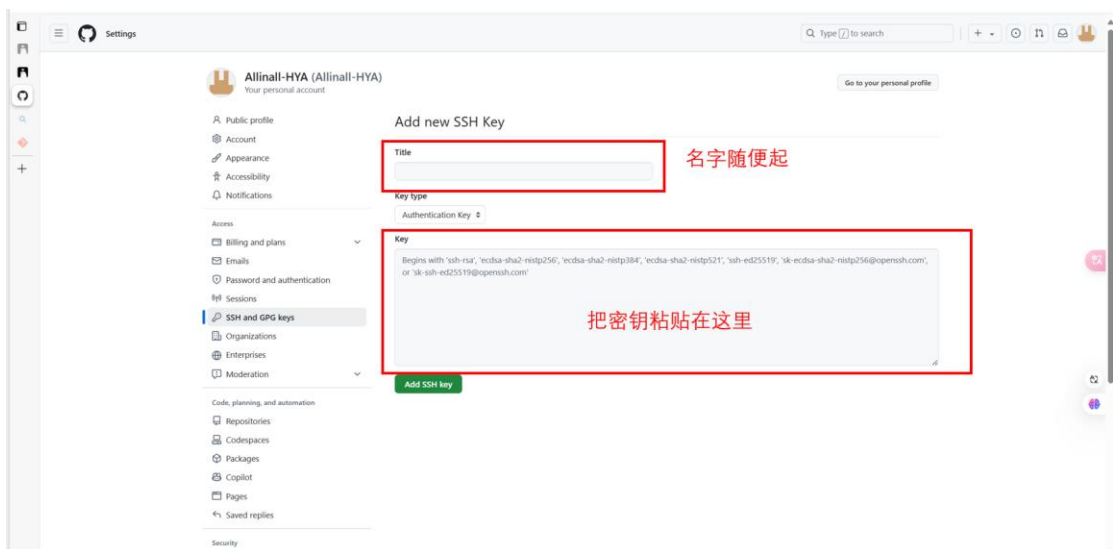
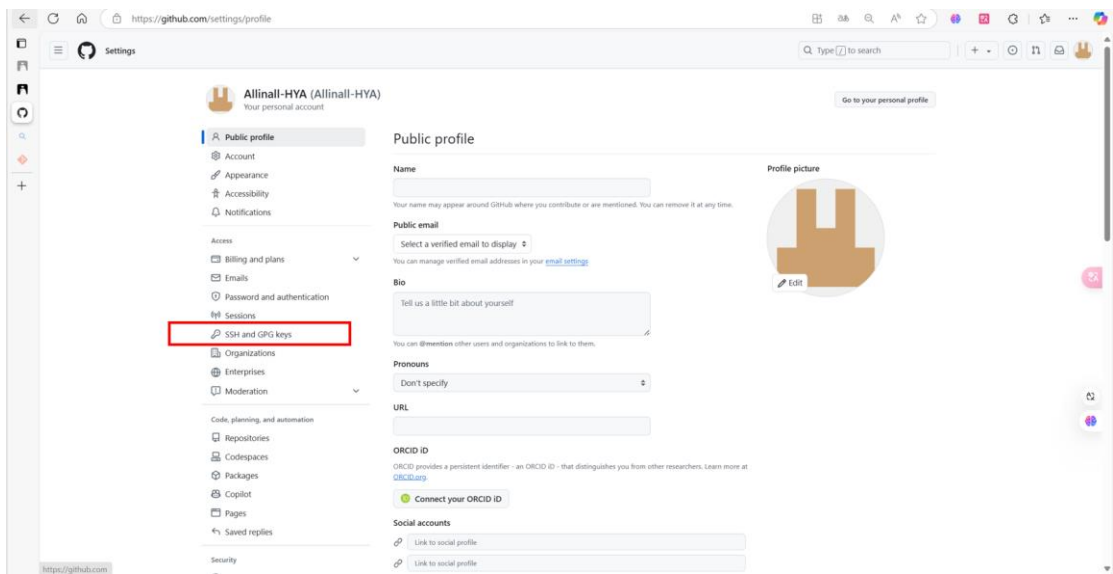
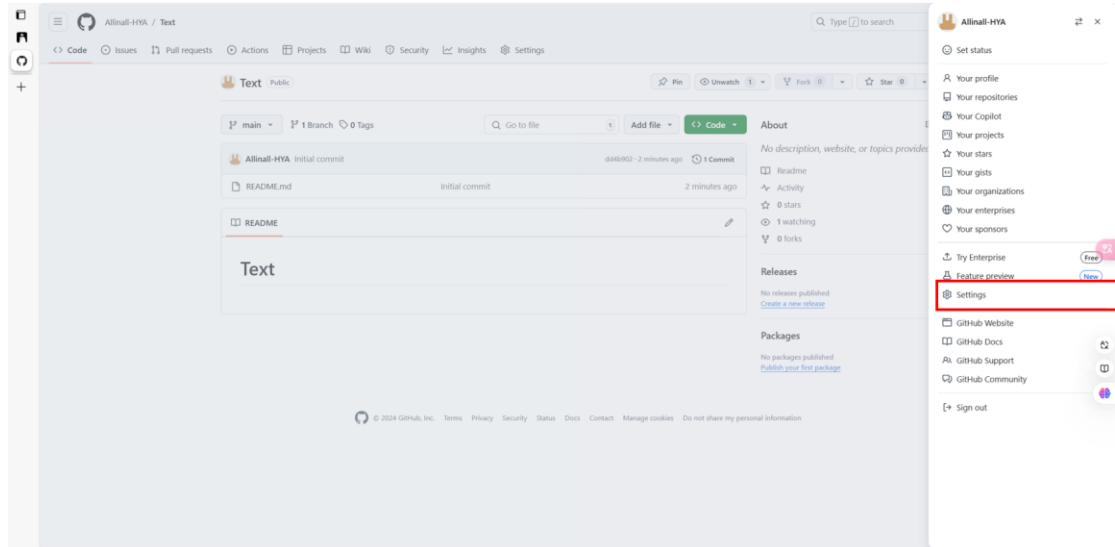

```
MINGW64:/c/Users/zjy
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/zjy/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/zjy/.ssh/id_rsa
Your public key has been saved in /c/Users/zjy/.ssh/id_rsa.pub
The key's fingerprint is:
SHA256:AF9Wjw0qfAo8gga9qgG8oJi+hxOP4iXAGfsAt5bHXAI zjy@LAPTOP-A5EU7BL8
The key's randomart image is:
+---[RSA 3072]-----+
|..E . . o.o|
|...o + o . =|
|+oo.= * o . o|
|*o=* = =|
|*B= + . S|
|B+o.|
|oo=o|
|o=oo|
|oo+|
+-----[SHA256]-----+

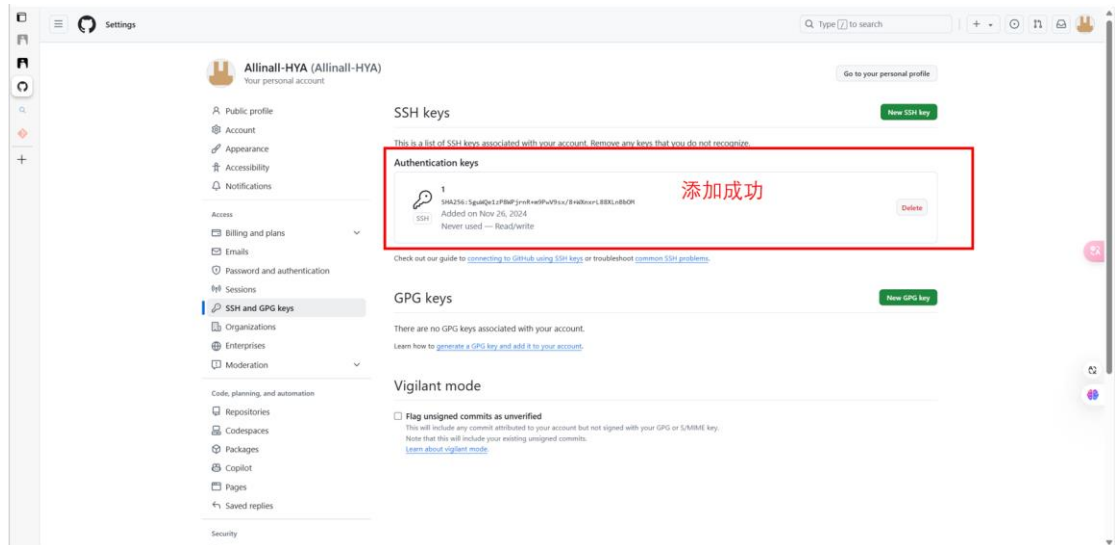
zjy@LAPTOP-A5EU7BL8 MINGW64 ~
$
```

根据提示的路径去寻找密钥并复制



去到 GitHub 上点击头像，选择“Settings”，点击后跳转页面，点击“SSH and GPG keys”，填写好名字和密钥后，点击“Add SSH key”就可以添加密钥了





添加成功之后，需要在本地 Git 上验证是否与 Github 连接成功，输入：
ssh -T [git@github.com](https://github.com)

```
MINGW64:/c/Users/17293

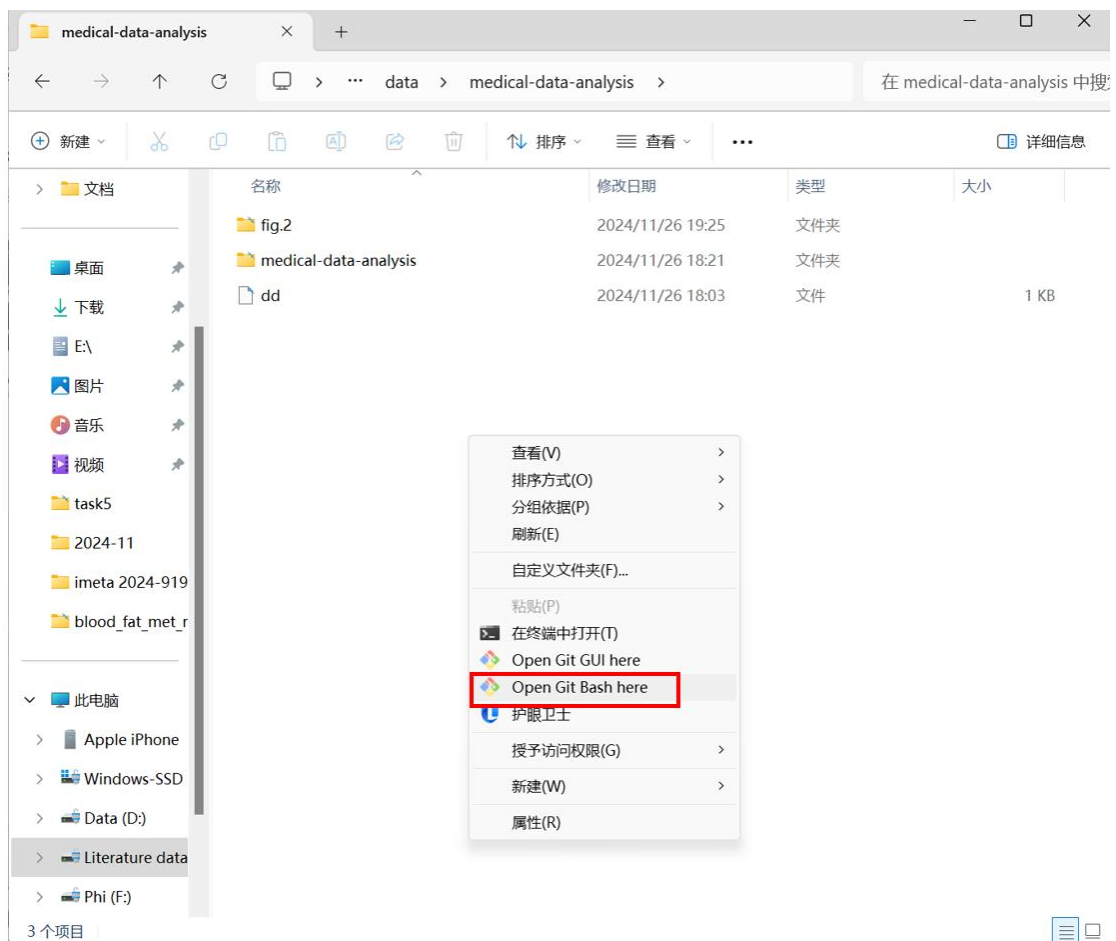
17293@ALLinALL MINGW64 ~ (master)
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/17293/.ssh/id_rsa):
/c/Users/17293/.ssh/id_rsa already exists.
Overwrite (y/n)?

17293@ALLinALL MINGW64 ~ (master)
$

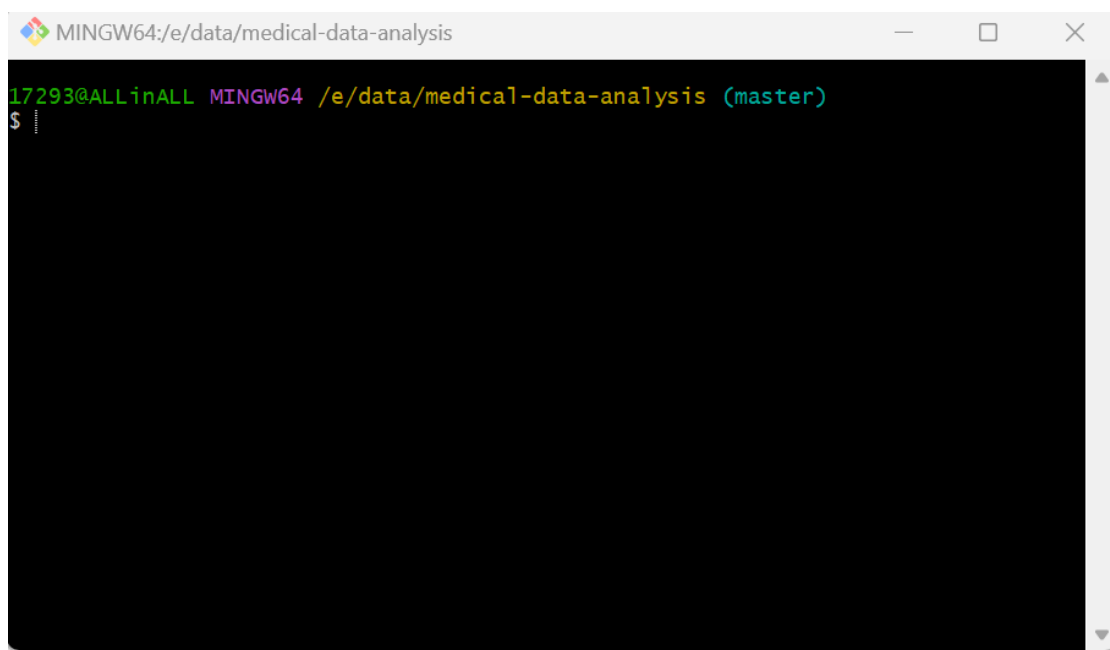
17293@ALLinALL MINGW64 ~ (master)
$ ssh -T git@github.com
Hi Allinall-HYA! You've successfully authenticated, but GitHub does not provide
shell access.

17293@ALLinALL MINGW64 ~ (master)
$
```

出现这两行的时候就说明 Git 与 Github 是可以进行连接的，连接成功之后，把它关掉，进入目标文件夹，点击鼠标右键，选择“Open Git Bash here”

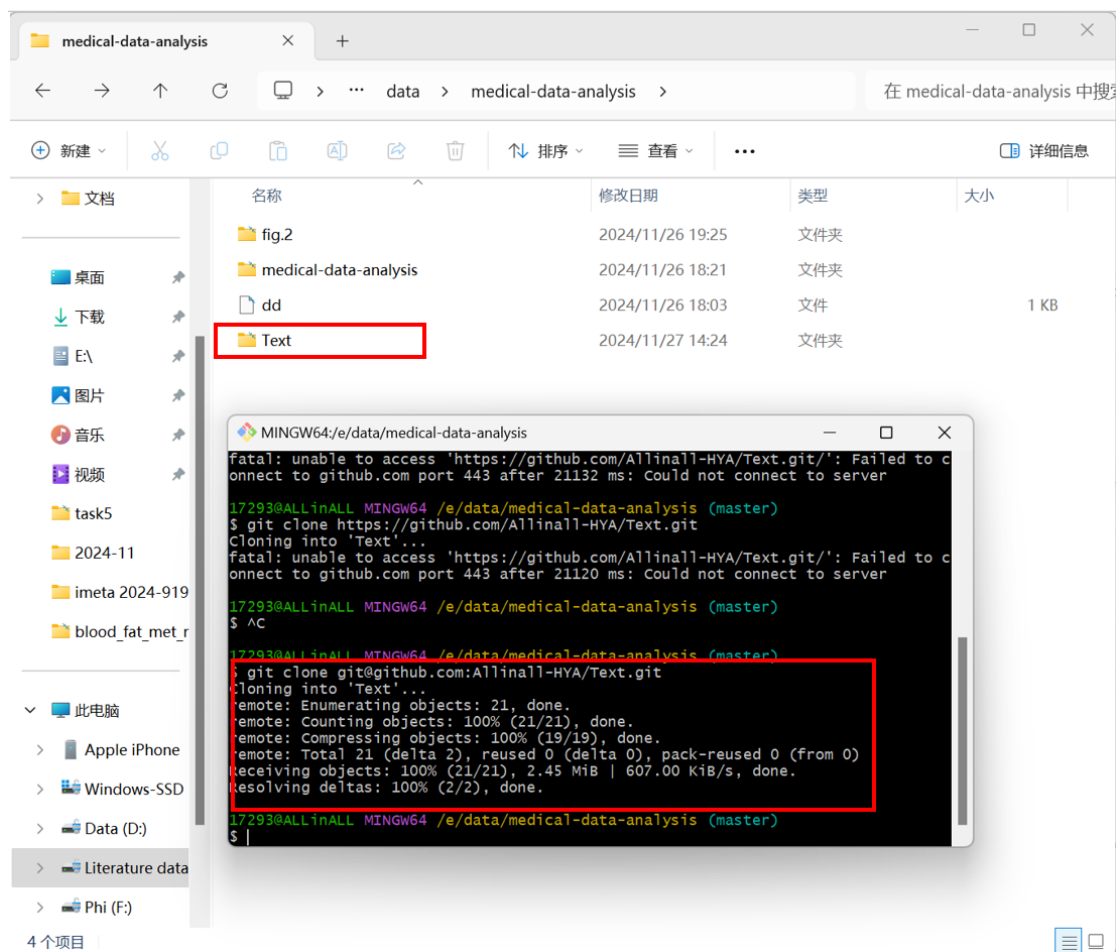


可以看到，这个路径变成了当前文件夹

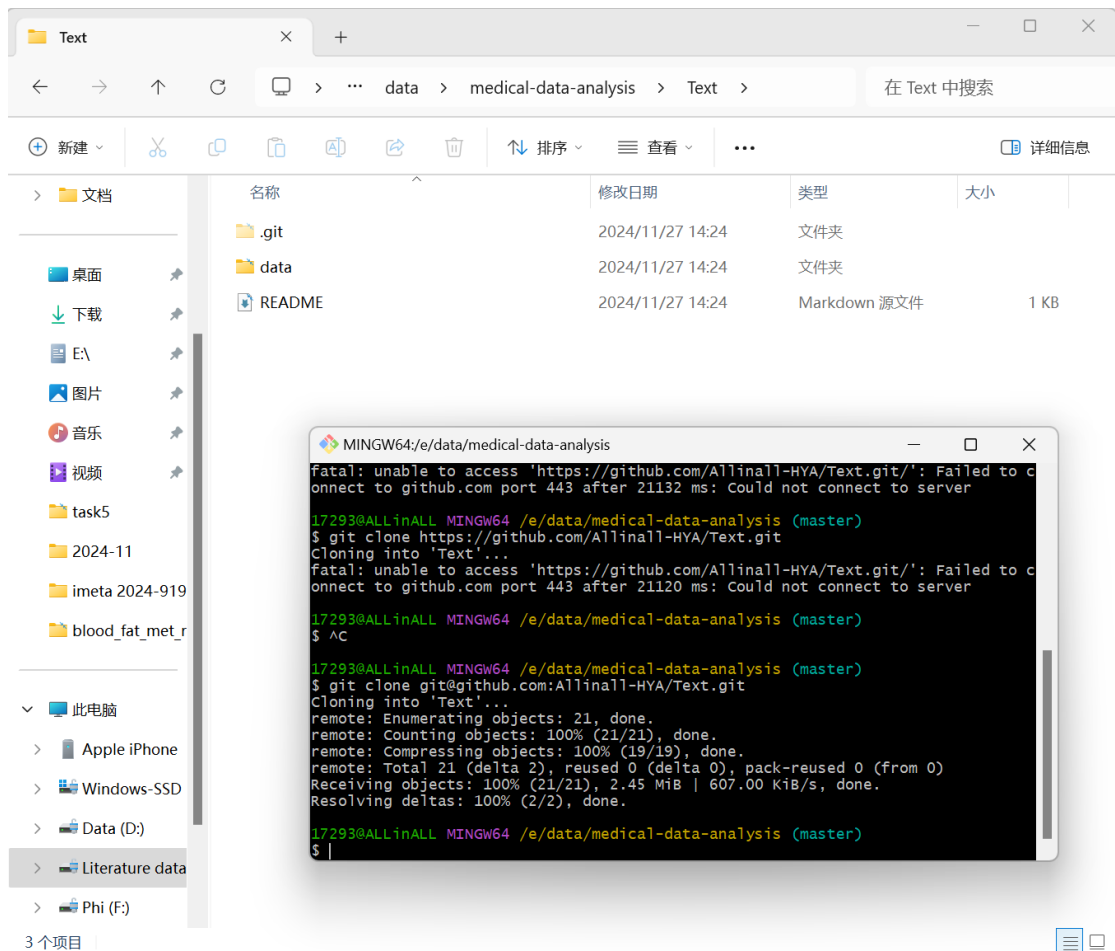


复制一下 Github 上的仓库，输入

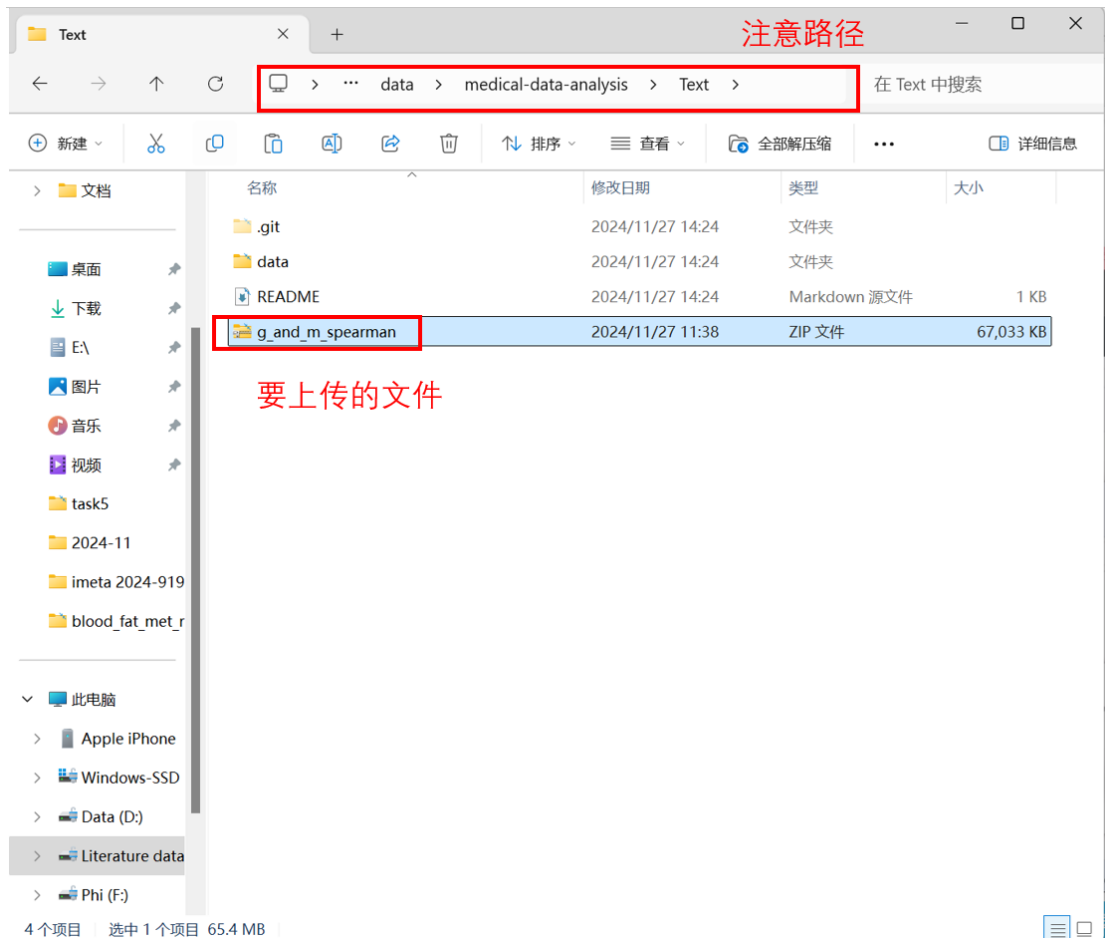
git clone <https://github.com/Allinall-HYA/Text.git> (这个链接是之前需要关注的那一个，可以向上翻找一下)，这里粘贴需要手动右键粘贴，不要使用快捷键，不然会报错



在这里我使用了 SSH 密钥来克隆仓库，因为我使用 HTTPS 显示连接不上，这两种方式都可以，如果出现同样问题的朋友们可以借鉴一下，出现以上代码说明已经克隆好了，在文件夹中出现了仓库“Text”



在“Text”文件夹中会出现一个“.git”的隐藏文件夹和我们仓库中已经上传的文件，接下来要把文件进行上传，只需要把上传的文件复制到“Text”文件中



复制进去之后，在“Text”文件夹中重启一下“Git Bash”，注意是在“Text”文件夹中重启!!!!!!! 打开之后输入以下代码查看未追踪的文件：

git status

```
MINGW64:/e/data/medical-data-analysis/Text

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

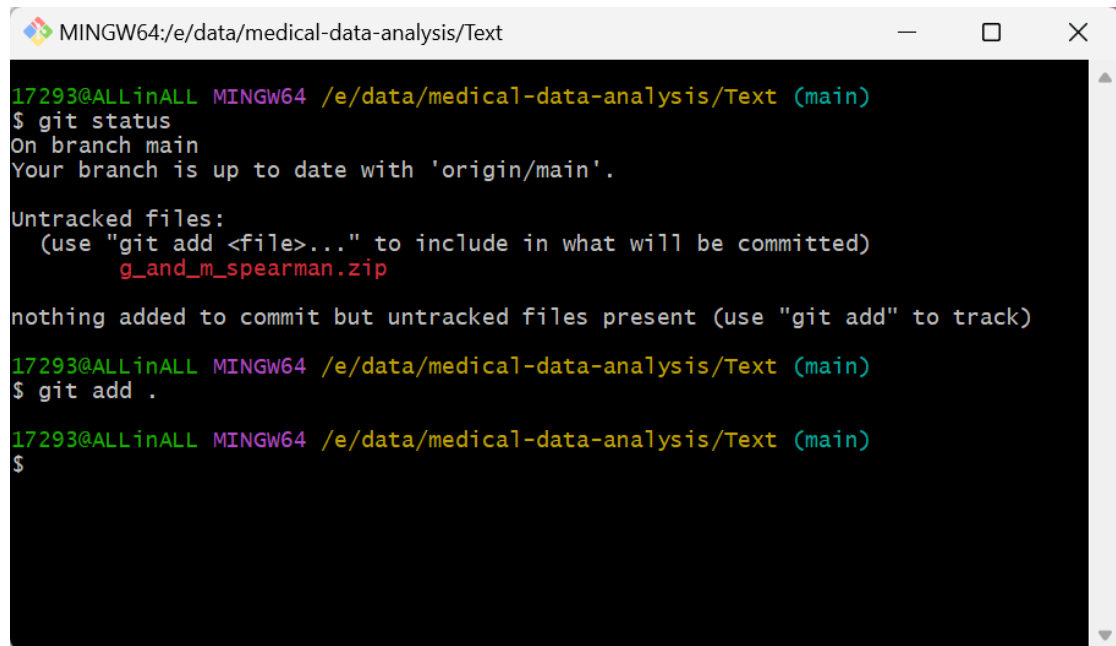
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      g_and_m_spearman.zip

nothing added to commit but untracked files present (use "git add" to track)
17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ |
```

已经检测到了我们新加入的文件，是红色的，接下来输入以下代码选择所有

文件:

git add .



```
MINGW64:/e/data/medical-data-analysis/Text

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

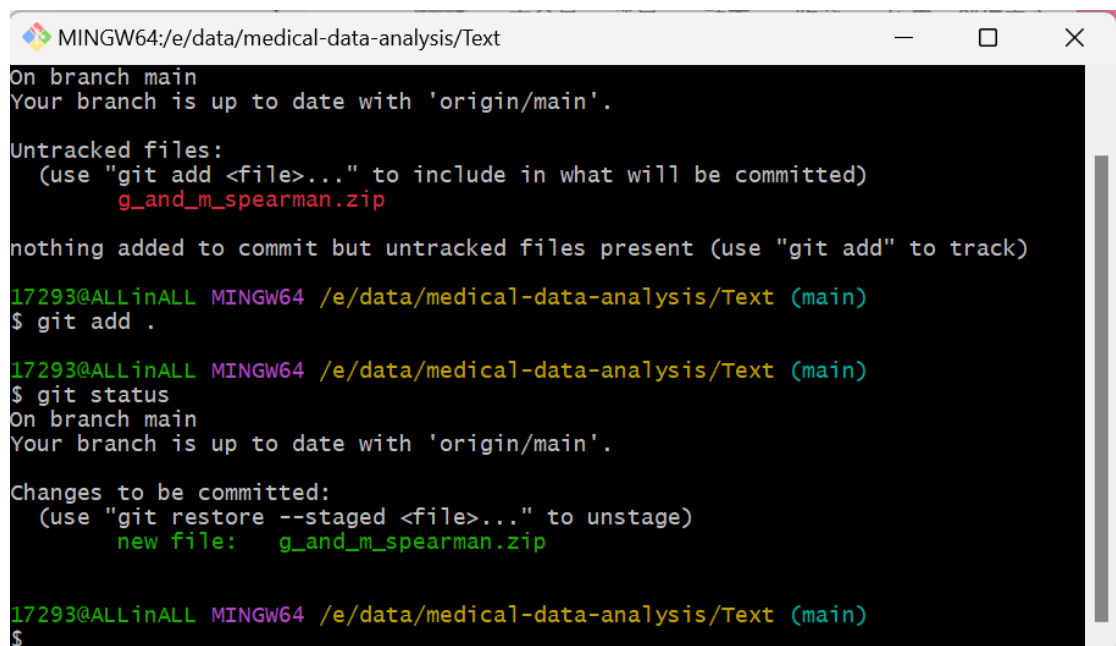
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        g_and_m_spearman.zip

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

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git add .

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$
```

再一次进行验证，看一下目标文件是否被追踪到



```
MINGW64:/e/data/medical-data-analysis/Text

On branch main
Your branch is up to date with 'origin/main'.

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

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

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git add .

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   g_and_m_spearman.zip

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$
```

可以看到文件的名字变成了绿色，表明追踪到了，接下来输入以下代码把文件放到本地仓库，并描述一下，引号里的东西写什么都可以

git commit -m "Text"


```
MINGW64:/e/data/medical-data-analysis/Text

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

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git add .

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

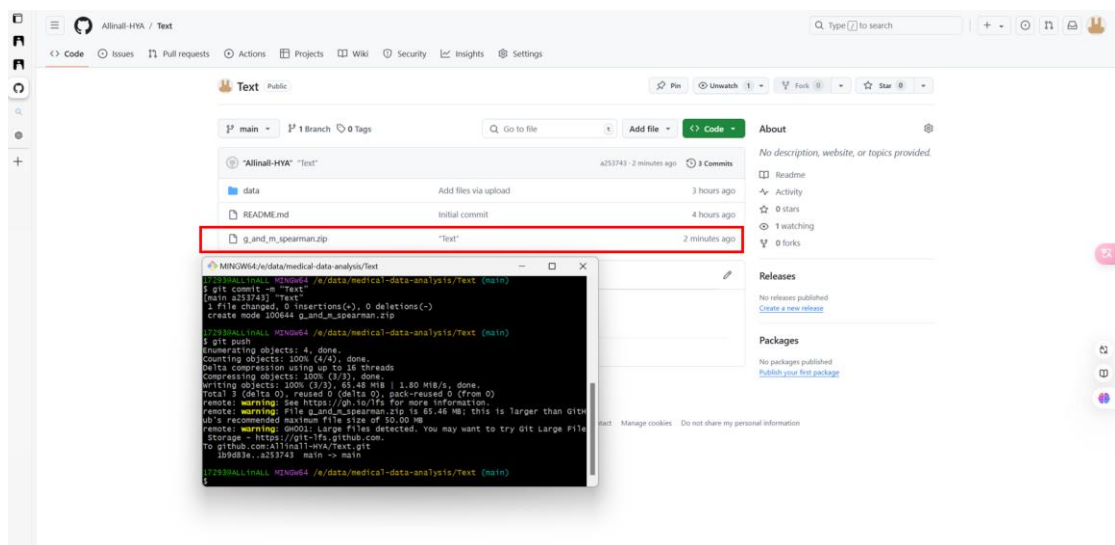
changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   g_and_m_spearman.zip

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ git commit -m "Text"
[main a253743] "Text"
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 g_and_m_spearman.zip

17293@ALLinALL MINGW64 /e/data/medical-data-analysis/Text (main)
$ |
```

这个时候文件就被放到了本地仓库中，接下来请我们使用以下代码把文件放到远程仓库中：

git push



此时文件已经上传到了 Github 上，Github 需要刷新一下才可以看到

到此为止，我们的文件就已经完成上传了

总结一下，这里详细的描述了如何上传大文件和小文件，个人建议没有代码基础的小伙伴如果文件较小，可以选择直接手动上传，如果文件较大在使用代码命令符号来操作，当然，两种方法结合也是可以的。

最后附上 B 站 UP 主北京低才生的教学视频链接，个人认为这个 UP 主讲得非常明白清晰，有需要的话大家可以跟着一起试着操作一下。

https://www.bilibili.com/video/BV1J14y1a7ZG/?spm_id_from=333.337.search-card.all.click&vd_source=72f2bbe12740acb3916f64dcaa13018f