

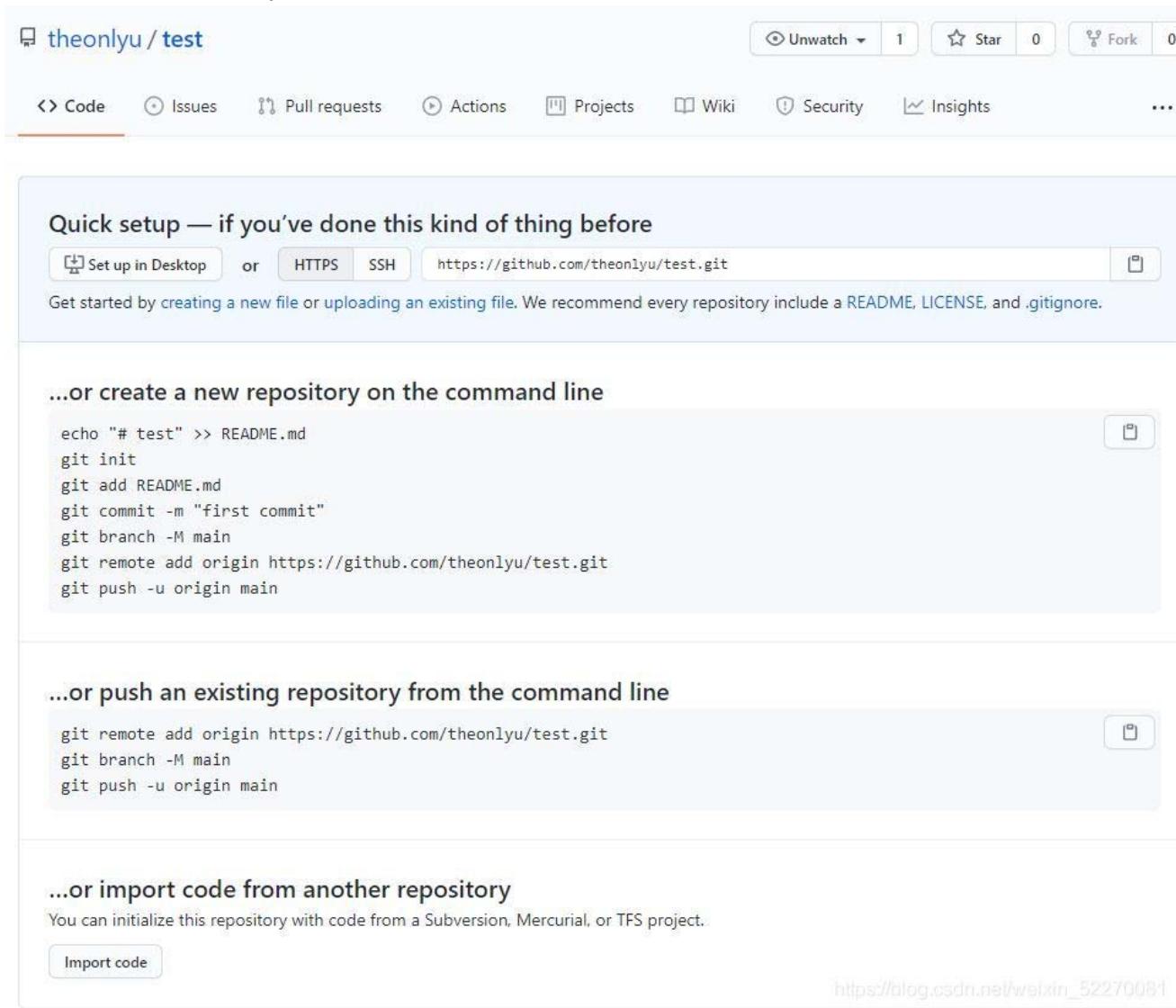
github 与git 使用方法

这里介绍windows下的git和GitHub使用。

linux下git和github搭建使用教程参考：
https://blog.csdn.net/weixin_52270081/article/details/119140724

1、注册gihub账号

github官网：<https://github.com/>自行创建即可。
登录，create repository新建仓库一个测试库readme，创建完成。




2、git的安装安装

git官方网站：<https://git-scm.com/>
选择Windows版本下载安装即可。

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.


Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.






About

The advantages of Git compared to other source control systems.




Documentation

Command reference pages, Pro Git book content, videos and other material.




Downloads

GUI clients and binary releases for all major platforms.




Community

Get involved! Bug reporting, mailing list, chat, development and more.







Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).



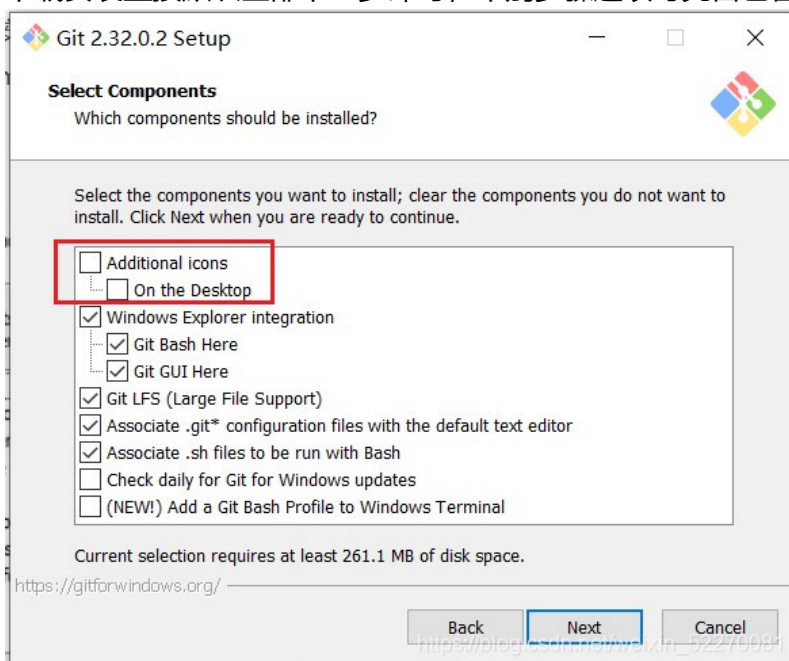
Latest source Release
2.32.0
Release Notes (2021-06-06)

Download for Windows

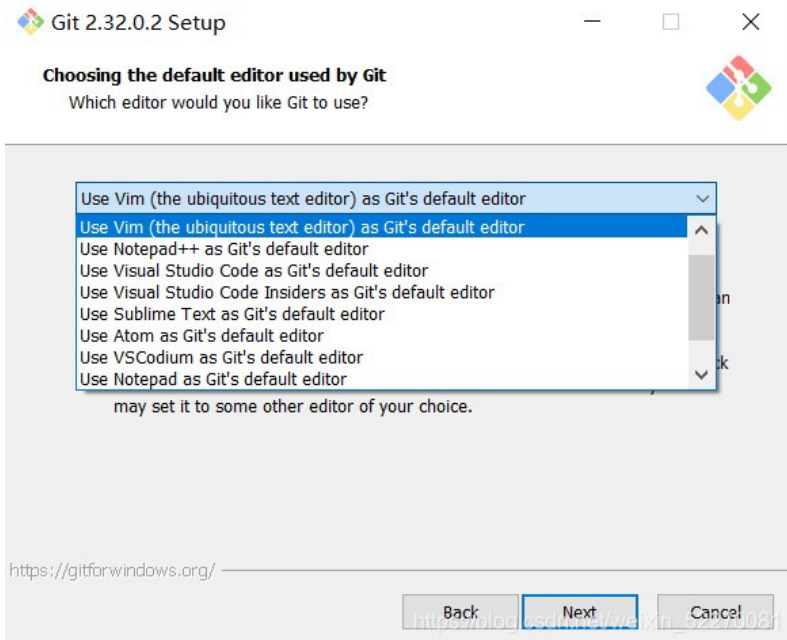
 [Windows GUIs](#)
 [Tarballs](#)

 [Mac Build](#)
 [Source Code](#)

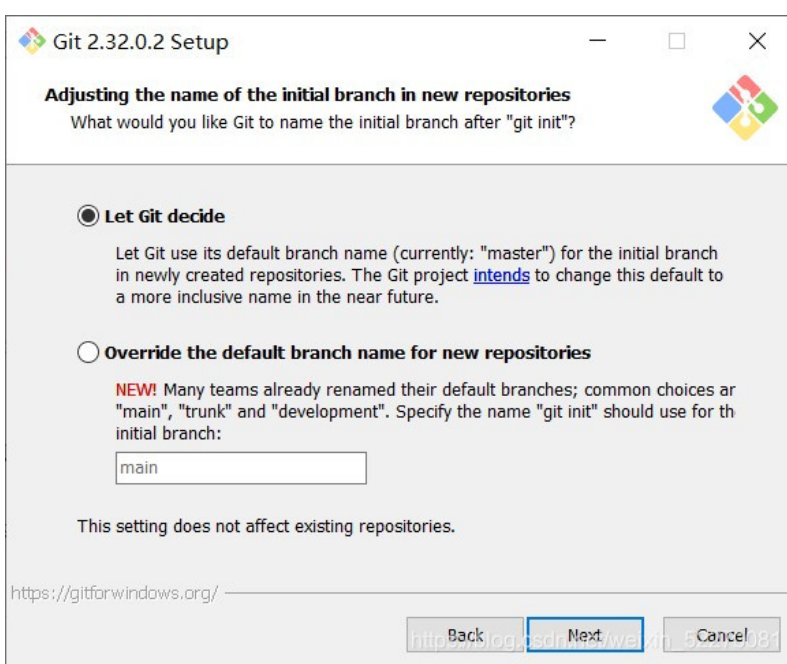
下载安装直接默认全部下一步即可，个别步骤选项可凭自己喜好选择，如：



这里选择是否创建桌面快捷方式。

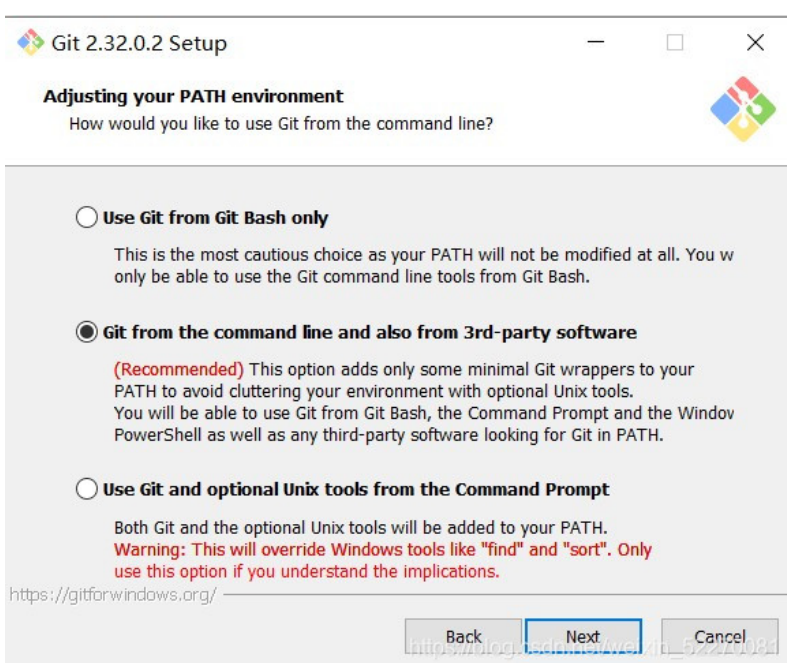


这里选择默认编辑器，默认是vim，用习惯Linux的都默认vim即可。还有notepad的编辑器。



选择第一个，由git决定分支的初始名字。

第二个是自己命名分支名字。



调整PATH环境的设置

标注 1：仅使用 Git Bash 进行操作；

标注 2：在选择使用 Git Bash 进行操作的同时，也可以使用 Windows 命令行操作，建议选择此项；

标注 3：在选择使用 Git 的同时，也把 Unix 工具加入到了我们的配置之中，而且此操作会覆盖 Windows 的一些工具，强烈不建议选择此项。

这里我们默认第二个环境即可。

其他步骤一样，全部默认即可，都是默认windows最佳选择。

安装完成后，在任务栏可以看到已经存在这几个软件

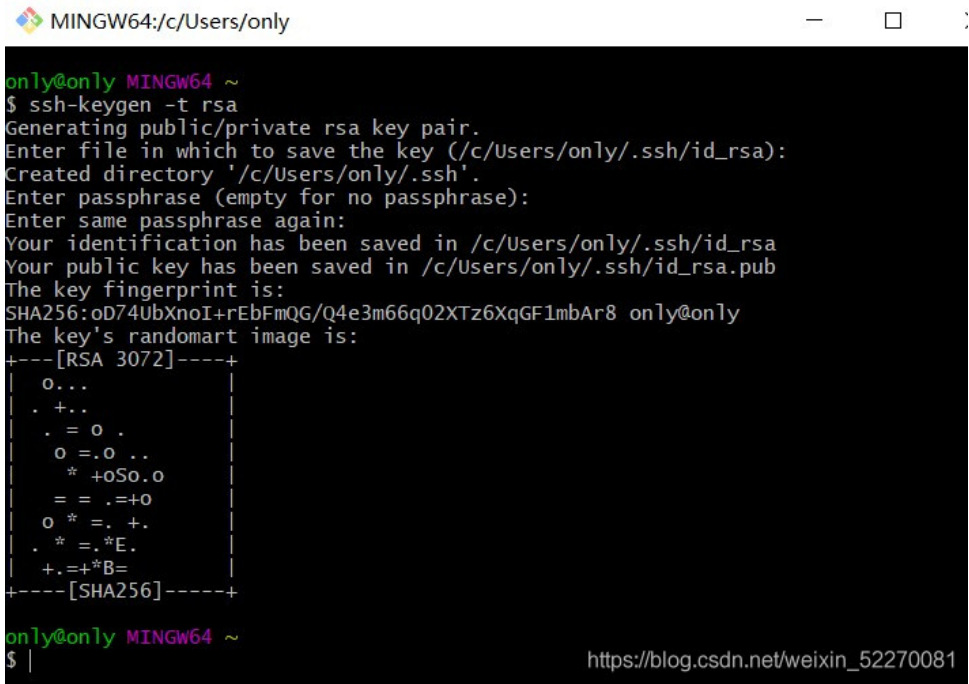


3、生成 ssh key连接GitHub

3.1 生成密钥

打开git bash生成ssh key 密钥，加密方式为rsa。

```
ssh-keygen -t rsa
```



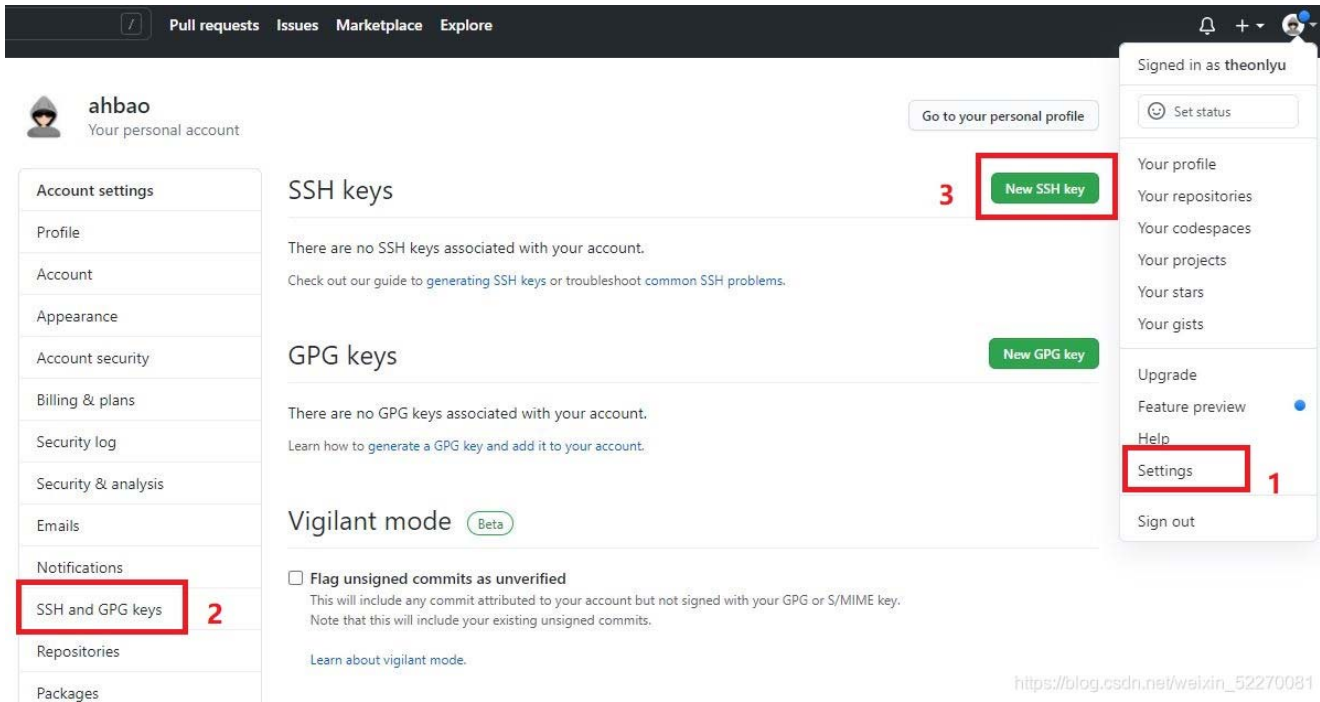
可以看到生成的密钥在C盘的用户，用户名下的.ssh名录。



3.2 复制公共密钥到git hub

公钥为id_rsa.pub的内容，以txt文本模式打开复制里面的内容。

登录github，在选项setting >> SSH and GPG key >> add new ssh添加刚才的公钥地址即可。title随便填写即可。



3.3 验证是否连接成功

输入以下命令，第一次需要确认输入yes即可。

```
ssh -T git@github.com
```



如图所示，出现 You've successfully authenticated, but GitHub does not provide shell access.则成功连接。

4、git命令上传文件至GitHub仓库

4.0 配置git参数的username,email

这是因为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。

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

自己设置自己的用户名和邮箱地址。不设置这一步，后面commit会报错如下：

```
Author identity unknown

*** Please tell me who you are.
```

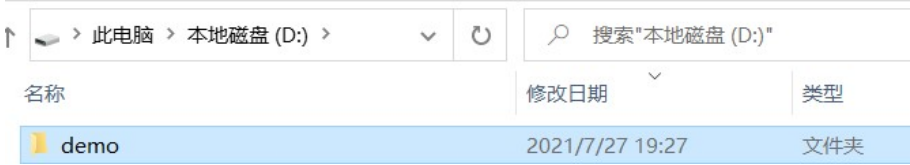
```
Run

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

4.1 选择一个盘，创建自己的仓库目录

这里我以D盘为例，创建一个demo文件夹作为项目目录。



注：在我们进行任何的git操作之前，我们都得先切换到 Git 的仓库目录。

4.2 打开Git Bash，命令进入仓库目录

前面提到需要切换进入仓库目录
命令 cd 进入仓库目录，如进入刚才创建的D盘的demo文件夹。

```
cd D:\demo
```

这时我们的工作目录已经切换至仓库目录



4.3 git init初始化仓库目录

进入任何一个新的仓库目录，第一步就是要初始化这个仓库目录。

```
git init
```

初始化后，默认进入仓库主分支，即master。

命令 `git status` 可查看仓库状态，初始化后在查看这时已经有了一个空仓库。



```
MINGW64:/d/demo
only@only MINGW64 ~
$ cd D:\demo
only@only MINGW64 /d/demo
$ git status
fatal: not a git repository (or any of the parent directories): .git
only@only MINGW64 /d/demo
$ git init
Initialized empty Git repository in D:/demo/.git/
only@only MINGW64 /d/demo (master)
$ git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)
only@only MINGW64 /d/demo (master)
$ |
```

未初始化，没有仓库可见

初始化仓库目录

初始化后，已经创建了一个空仓库。

https://blog.csdn.net/weixin_52270081

在文件夹中，这时也出现一个隐藏的文件夹.git，这个是一个仓库。



4.4 git add添加文件到本地仓库

在demo文件夹创建一个文件readme.txt，作为测试文件。

然后在Git Bash 添加文件到本地仓库。

```
git add readme.txt
```

4.5 git commit提交文件到本地仓库

提交到本地库并备注为readme commit，此时变更仍在本地。

```
git commit -m "readme commit"
```

4.6 git remote add增加一个远程服务器的别名

为了后续的方便管理，可设置一个别名。

格式为 `git remote add 别名 git@github.com:GitHub用户名/GitHub仓库名.git`

```
git remote add readme_test git@github.com:theonlyu/readme.git
```

4.7 git push推送到github仓库

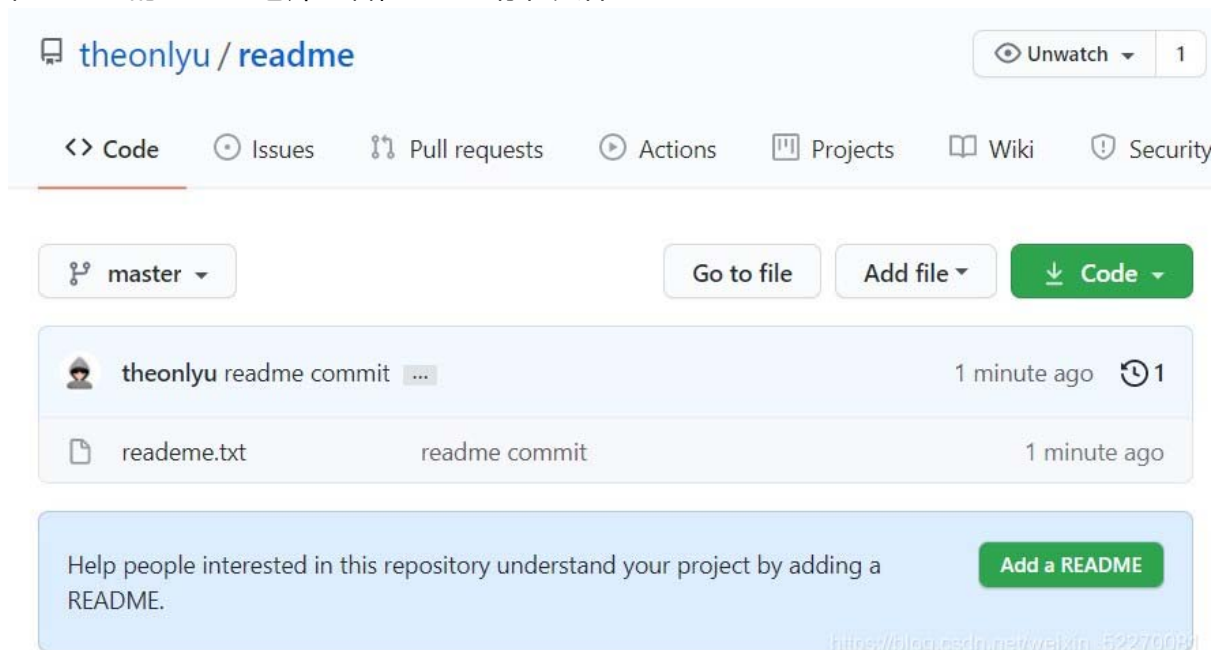
后面接刚才设置的别名就可以了。

```
git push readme_test master
```

```
only@only MINGW64 /d/demo (master)
$ git push readme_test master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 214 bytes | 71.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:theonlyu/readme.git
* [new branch]      master -> master

only@only MINGW64 /d/demo (master)
```

在GitHub的readme仓库查看，也已经存在文件了。



推送完成，命令参考：

```
git add README.md #添加文件到本地仓库
git rm README.md #本地删除库内文件
git commit -m "first commit" #提交到本地库并备注，此时变更仍在本地。
git commit -a ##自动更新变化的文件，a可以理解为auto
git remote add xxx git@github.com:xxx/xxx.git #增加一个远程服务器的别名。
git remote rm xxx ##删除远程版本库的别名
git push -u remotename master #将本地文件提交到Github的remotename版本库中。此时才更新了本地变更到github服务上
```

5、从GitHub仓库下载至本地

下载比较简单，在GitHub右边的code下可看到好几种下载方式，可以直接下载zip压缩文件，也可命令下载，一般选择ssh方式，可读写。

5.1 同样你需要进入一个目录

如直接下载到D盘的test文件夹，则先进入D盘test文件夹

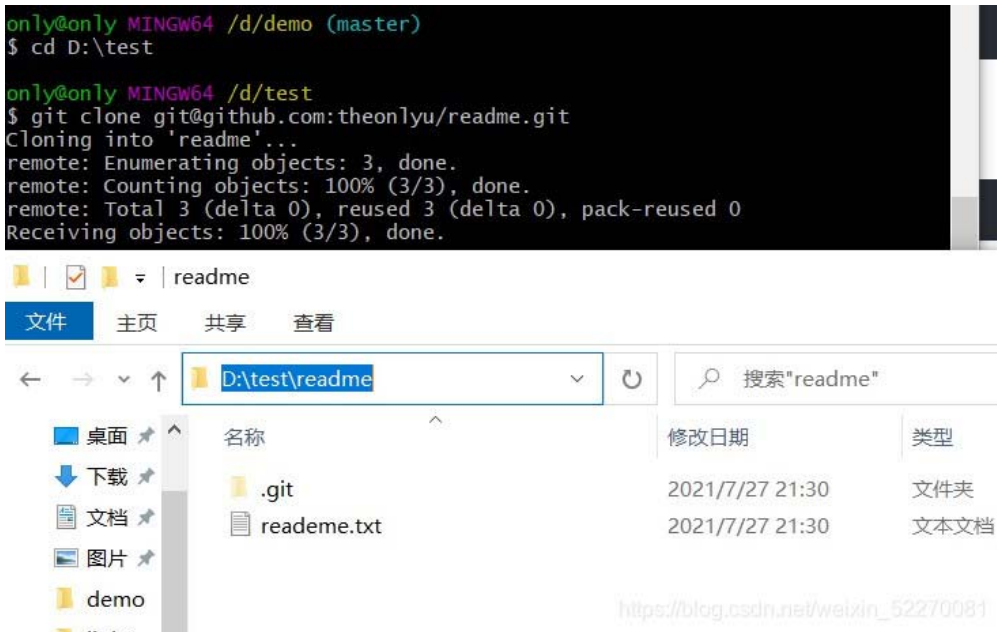
```
cd D:\test
```


5.2 git clone 下载至本地

```
git clone git@github.com:theonlyu/readme.git
```

- 1

下载后，以仓库名为文件夹的方式存在当前目录。



下载方式差别参考：

```
git clone git://github.com:xxxx/test.git ##以gitreadonly方式克隆到本地，只可以读
git clone git@github.com:xxx/test.git  ##以SSH方式克隆到本地，可以读写
git clone https://github.com/xxx/test.git ##以https方式克隆到本地，可以读写
git fetch git@github.com:xxx/xxx.git  ##获取到本地但不合并
git pull git@github.com:xxx/xxx.git  ##获取并合并内容到本地
```

6、Git的分支管理

6.1 创建分支

例子：在本地demo目录创建一个branch.txt以分支提交。

```
git branch #显示当前分支是master
git branch new_bra  #创建分支命名为new_bra
git checkout new_bra  #切换到新分支
git add branch.txt
git commit -m "added branch.txt"
git push readme_test new_bra  ##把分支提交到远程服务器，只是把分支结构和内容提交到远程，并没有发生和主干的合并行为。
```

```

only@only MINGW64 /d/demo (master)
$ git branch
* master

only@only MINGW64 /d/demo (master)
$ git branch new_bra

only@only MINGW64 /d/demo (master)
$ git checkout new_bra
Switched to branch 'new_bra'

only@only MINGW64 /d/demo (new_bra)
$ git add branch.txt

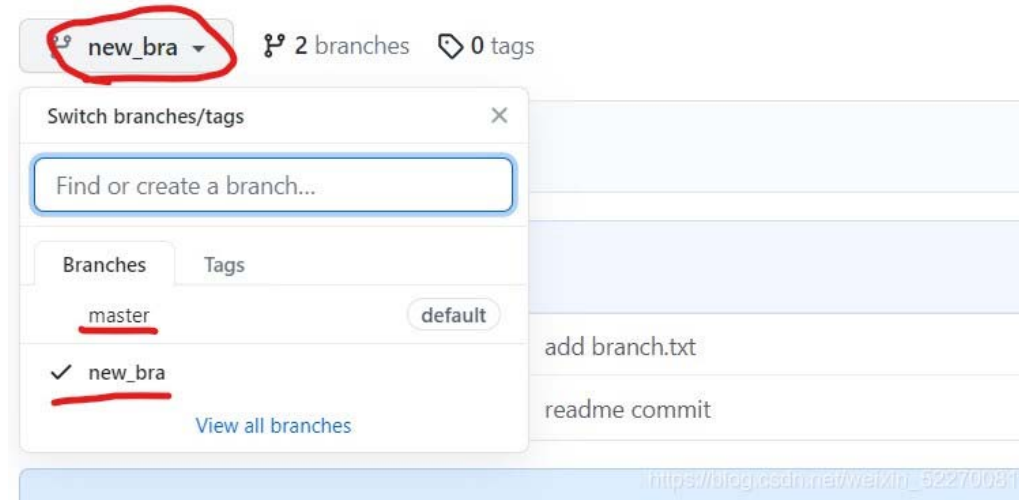
only@only MINGW64 /d/demo (new_bra)
$ git commit -m "add branch.txt"
[new_bra 57ec9bb] add branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 branch.txt

only@only MINGW64 /d/demo (new_bra)
$ git push readme_test new_bra
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 271 bytes | 90.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'new_bra' on GitHub by visiting:
remote:   https://github.com/theonlyu/readme/pull/new/new_bra
remote:
To github.com:theonlyu/readme.git
 * [new branch]      new_bra -> new_bra

only@only MINGW64 /d/demo (new_bra) https://blog.csdn.net/weixin_52270081
$

```

在github左上角可看到，new_bra分支已经上传成功，文件也存在了。主master却还没有新文件，因为没合并。



6.2 合并分支

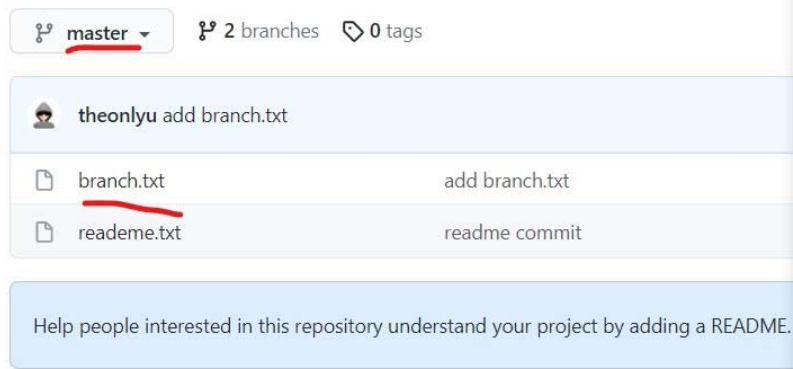
如果new_bra分支成熟了，就是代码确认下来了，觉得有必要合并进master

```

git checkout master #切换到新主干
git merge new_bra   ##把分支合并到主干
git branch          #显示当前分支是master
git push readme_test master #此时主干中也合并了new_bra的代码，readme_test为别名

```

在github中成功合并，切换至master，新文件 branch.txt 也存在了。



```
only@only MINGW64 /d/demo (new_bra)
$ git checkout master
Switched to branch 'master'

only@only MINGW64 /d/demo (master)
$ git merge new_bra
Updating fc9c14e..57ec9bb
Fast-forward
 branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 branch.txt

only@only MINGW64 /d/demo (master)
$ git branch
* master
  new_bra

only@only MINGW64 /d/demo (master)
$ git push readme_test master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:theonlyu/readme.git
 fc9c14e..57ec9bb master -> master

only@only MINGW64 /d/demo (master)
$
```

6.3 其他命令：

- #更新远程分支列表
git remote update 别名 --prune
- #查看所有分支
git branch -a
- #删除远程分支
git push 别名 --delete 分支名
- #删除本地分支
git branch -d 分支名