FOXCONN®

Gitlab多人协同工作

Master & Developer

BEACON

FOXCONN

目录

BEACON

FOXCONN®

开发前准备

BEACON

开发前准备

Gitlab多人协同工作，所有开发者会被添加进一个共同的工作组。协同工作开始前，群组管理员及开发者都有一些准备工作要做。

群组管理员（Master）：

1 - 在Gitlab上创建群组
2 - 为群组添加成员并赋予权限
3 - 为群组新建项目
4 - 为群组项目创建分支

开发者（Developer）：

1. 去Gitlab注册个人账号并添加个人SSH Key（参照本PPT附录）
2. 请群组管理员将你的账号添加进open-source群组

FOXCONN®

开发中过程

BEACON

开发中过程

日常开发中开发者与群组管理员的分工。

开发者（Developer）：

1. 克隆远程仓库到本地（只在第一次做一次）
2. 获取远程最新版本并merge到本地
3. 切换到自己的分支
4. 在自己的分支上写代码并提交
5. 将自己分支的更新push到远程
6. 在Gitlab上发起请求，请求合并自己的分支到远程的master分支

群组管理员（Master）：

1. 管理员收到合并请求，审查代码并决定是否通过合并请求

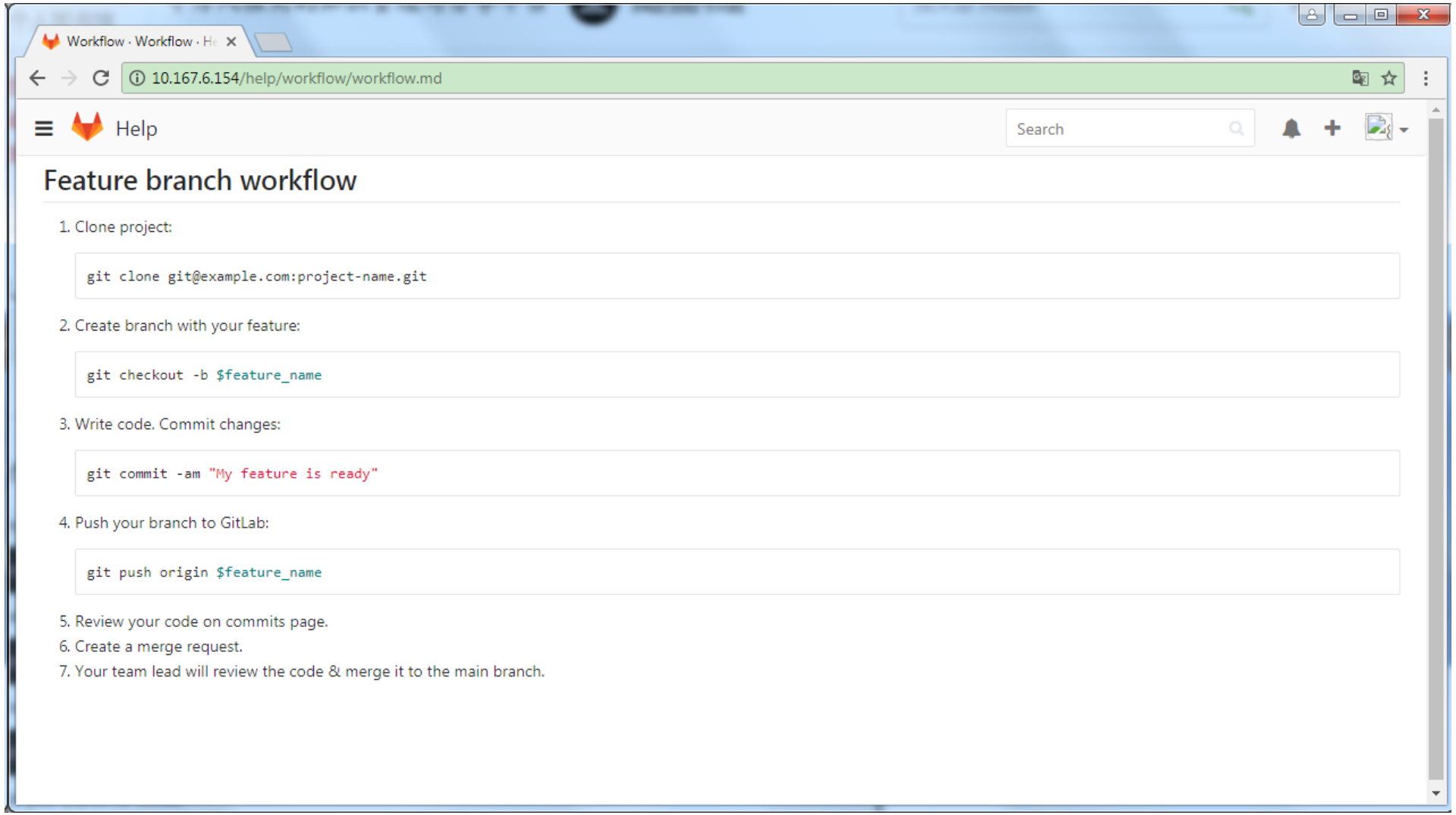
FOXCONN®

Developer

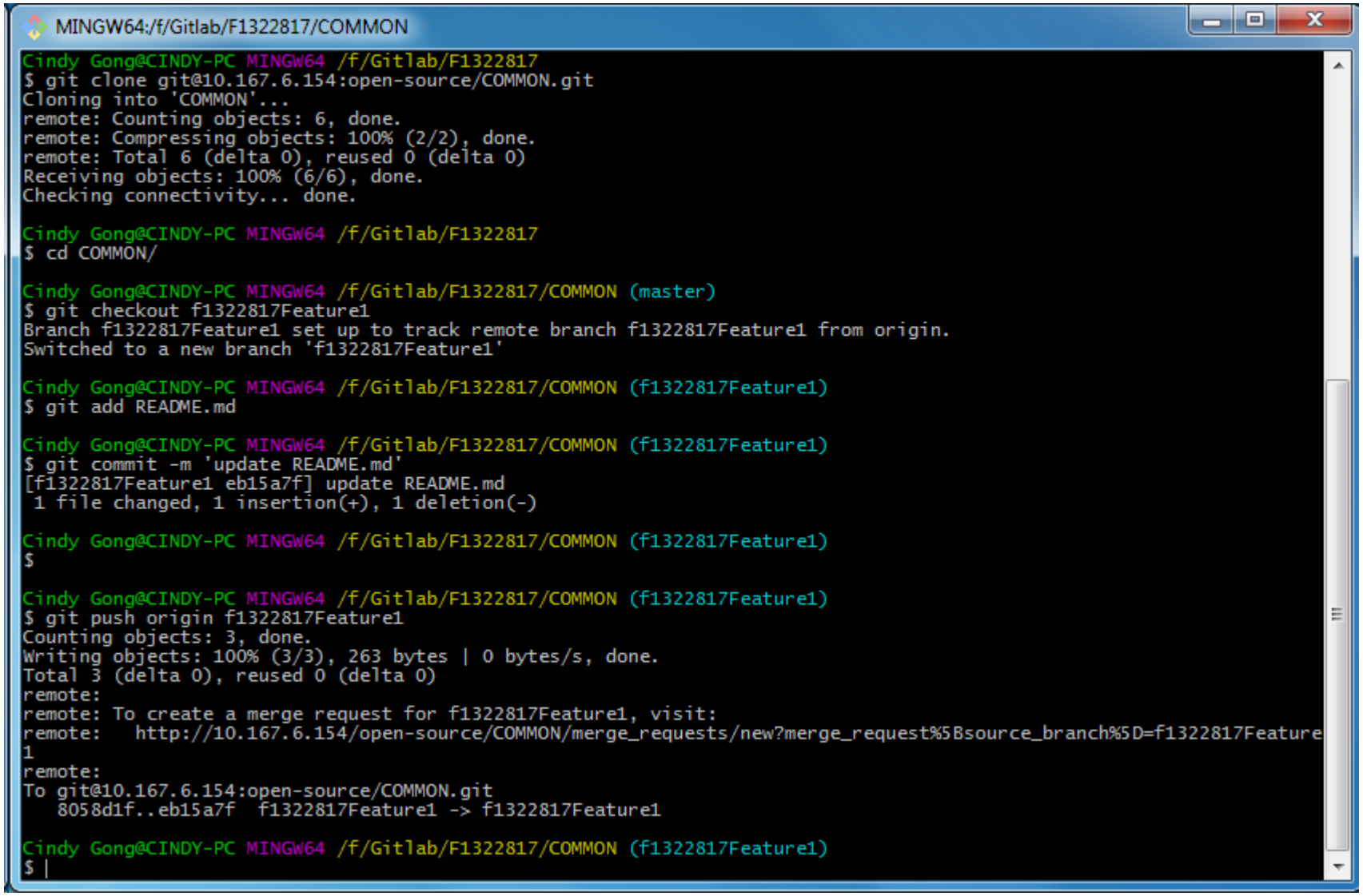
BEACON

开发者Workflow

Gitlab建议的协同工作流



开发者Workflow



① 克隆管理员创建好的项目到本地

② 切换到自己的分支

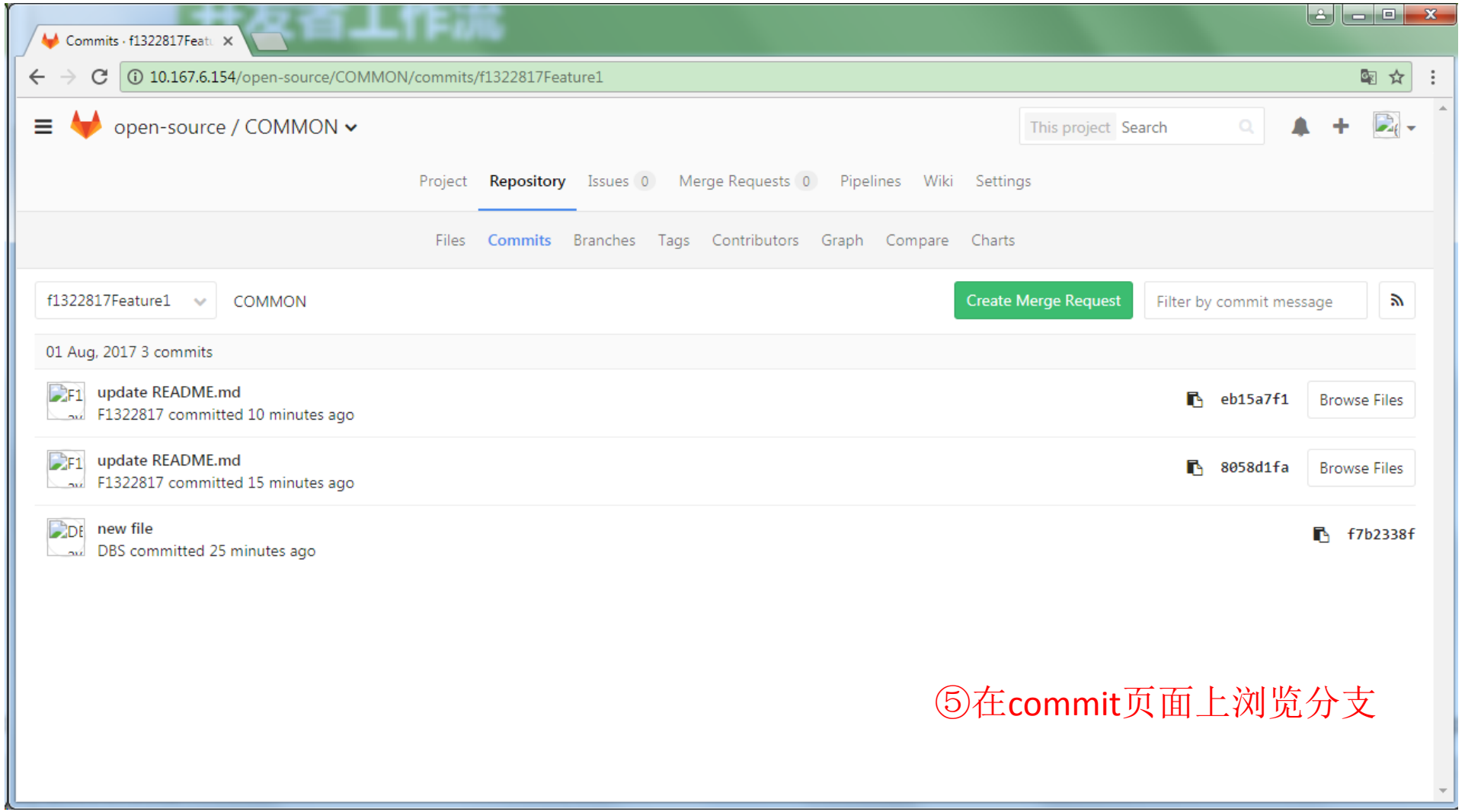
③ 在自己的分支上写代码并提交

④ push到远程服务器(分支是自己的分支)

BEACON

开发者Workflow

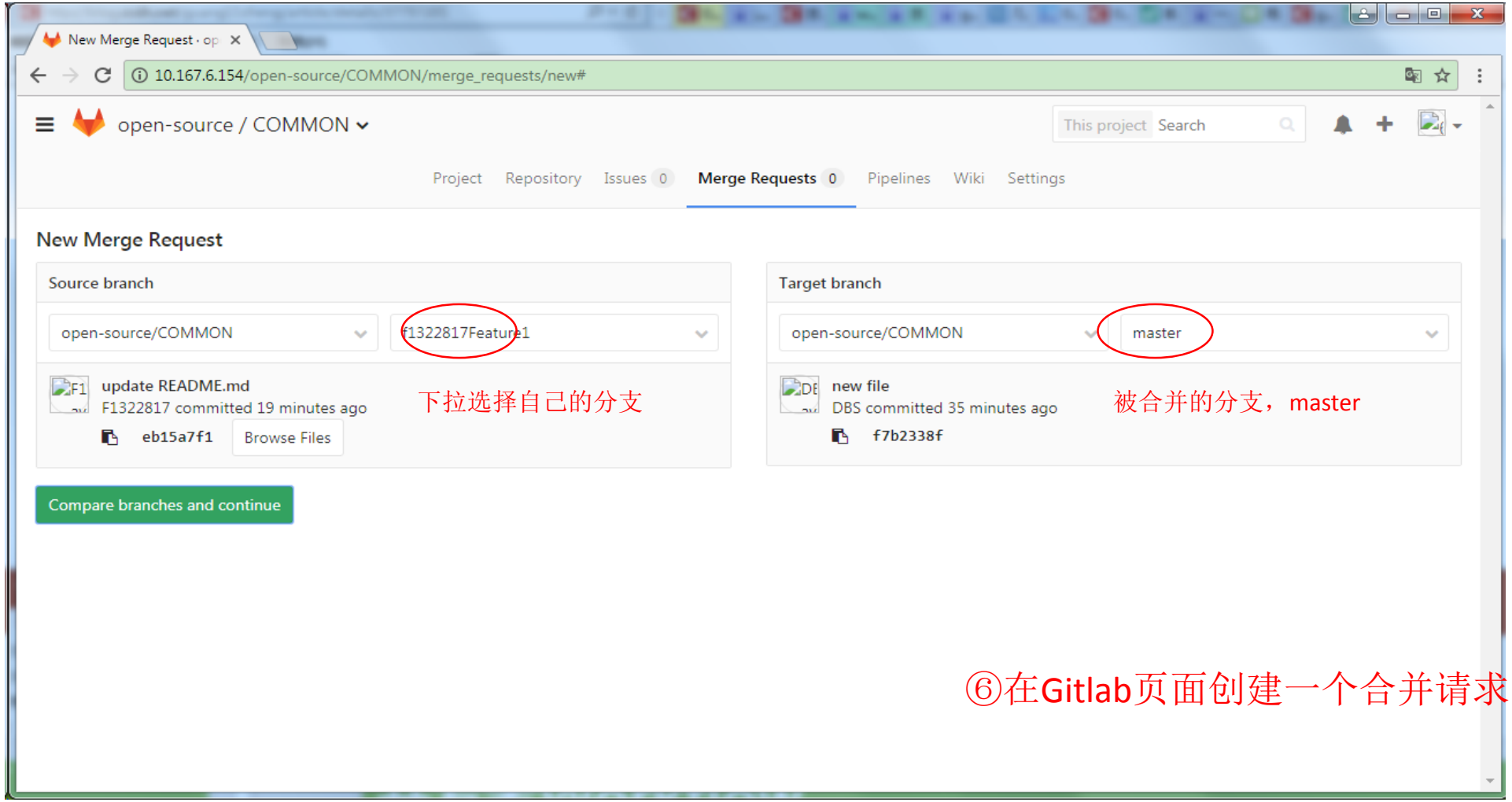
Projects -> 项目名称 -> Repository -> Commits -> 选择自己的分支，点击单个链接可以看到更新的具体内容。



BEACON

开发者Workflow

点击Merge Requests -> New Merge Request，在New Merge Request页面选择自己要合并的分支和要并入的分支（master），点击Compare branches and continue继续。



开发者Workflow

开发者操作总结：

1. 克隆管理员创建好的项目到自己本地
2. 切换到自己的分支（分支由管理员创建）
3. 在自己的分支上写代码，并提交
4. 推送到远程服务器，分支是自己的分支
5. 在commit页面上浏览分支
6. 创建一个合并请求

其中：远程仓库由管理员创建，开发者使用的分支也由管理员事先创建。开发者在Gitlab页面上创建了合并请求之后，由管理员审查代码并最终通过合并请求，完成分支到主分支的合并。

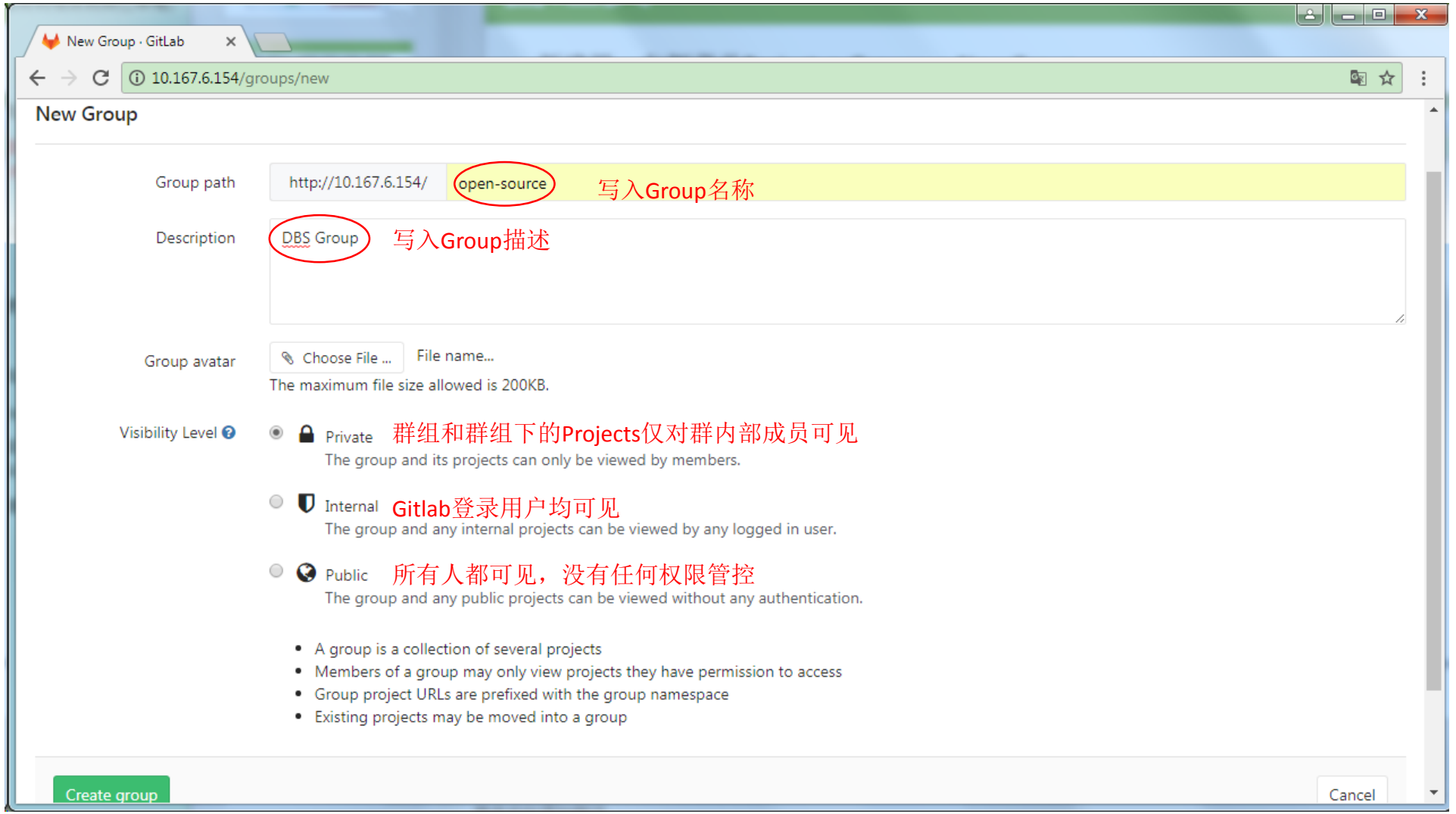
FOXCONN®

Master

BEACON

Master – 创建群组

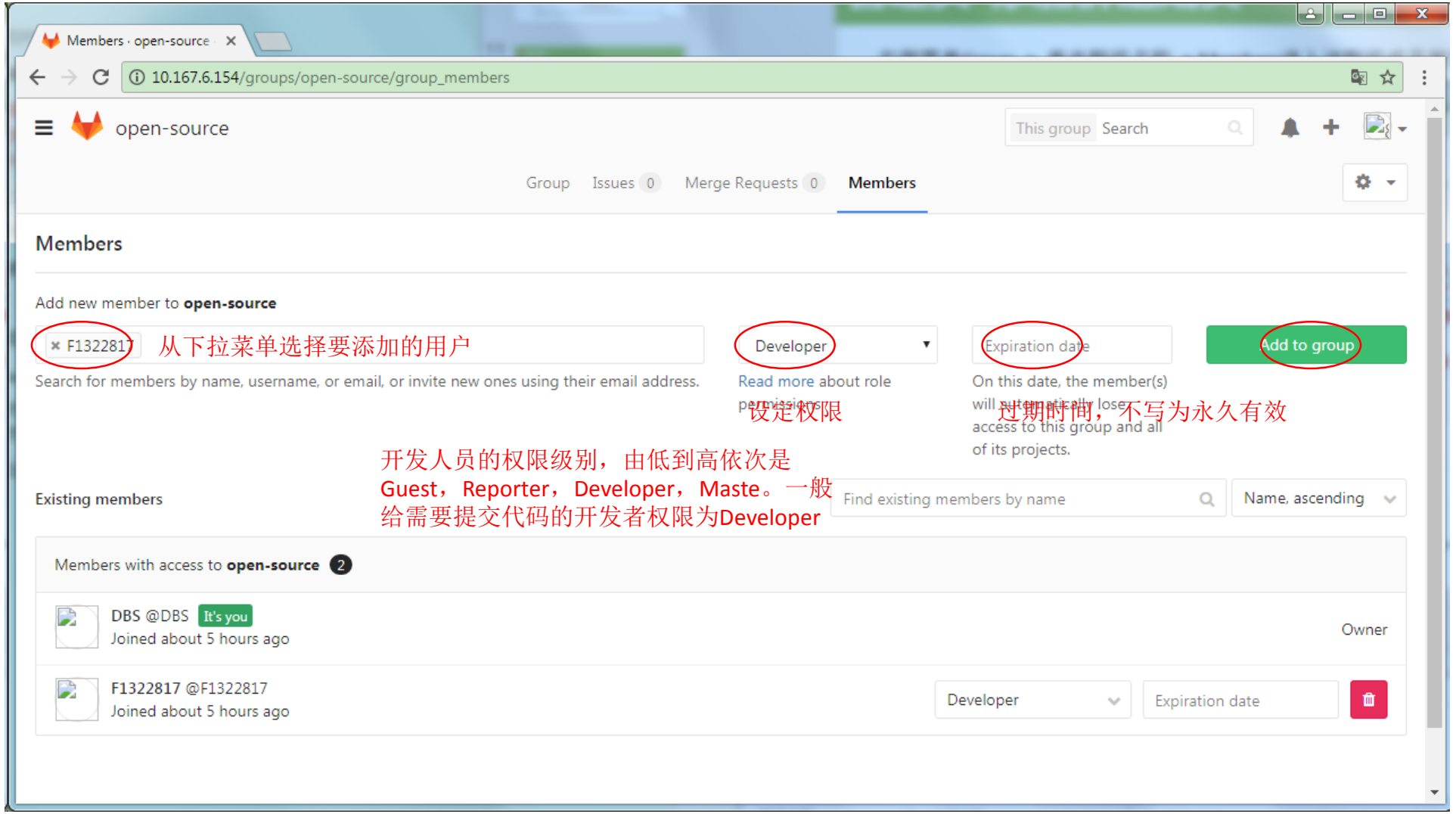
左侧菜单Project -> Groups -> New Group，填入群组名称和描述，选择权限选项点击Create group提交。



BEACON

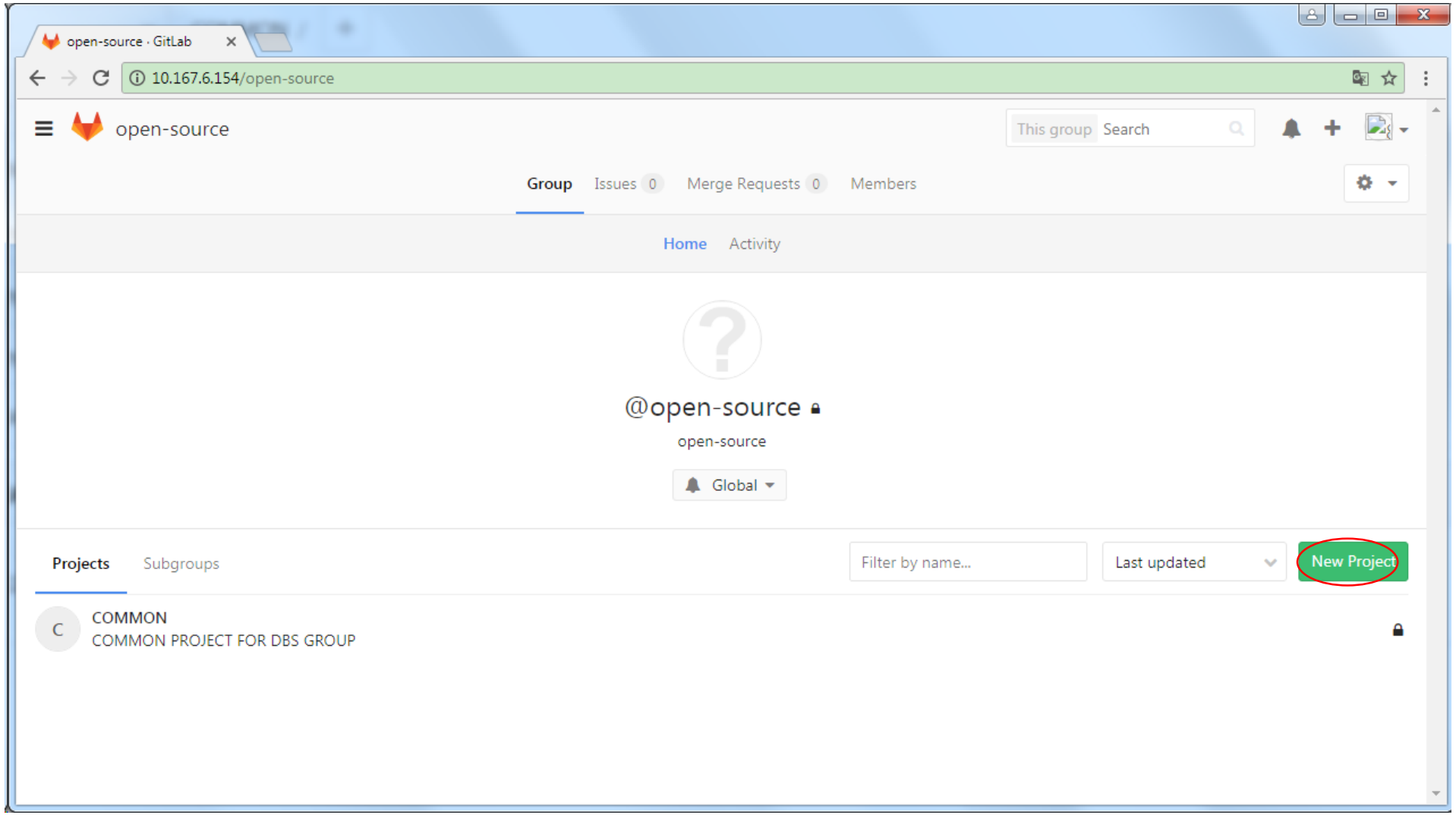
Master - 添加群组成员

左侧菜单Group -> 单击群组名称 -> Members进入该群组成员管理页面，按照指引添加成员



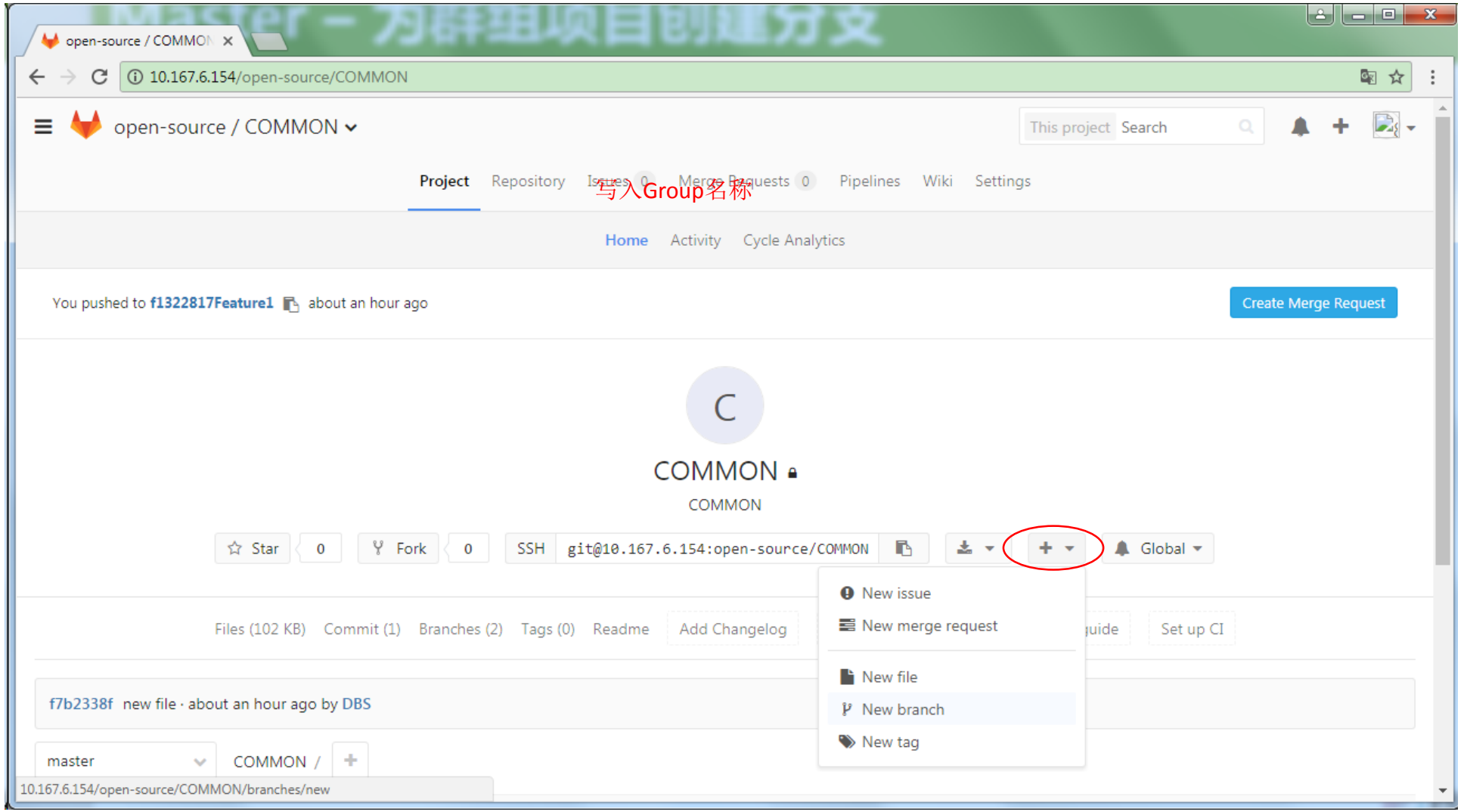
Master - 新建群组项目

单击左边菜单栏，选择Group -> 单击群组名称 -> New Project，按照指引新建项目。



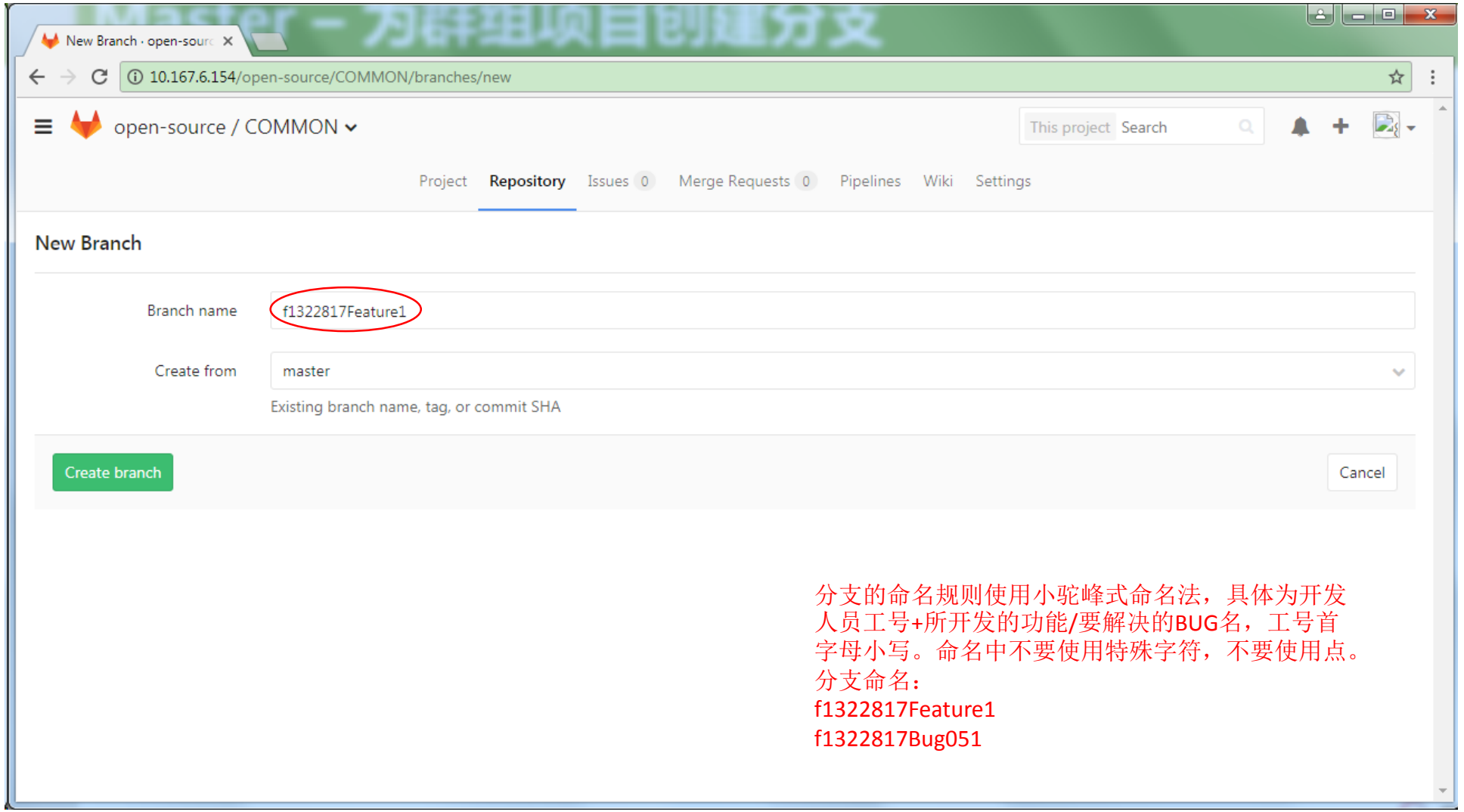
Master – 为群组项目创建分支

进入Project页面，在SSH地址栏右边点+，下拉菜单中选择New branch，创建新的分支。



Master – 为群组项目创建分支

进入Project页面，在SSH地址栏右边点+，下拉菜单中选择New branch，创建新的分支。



分支的命名规则使用小驼峰式命名法，具体为开发人员工号+所开发的功能/要解决的BUG名，工号首字母小写。命名中不要使用特殊字符，不要使用点。

分支命名：

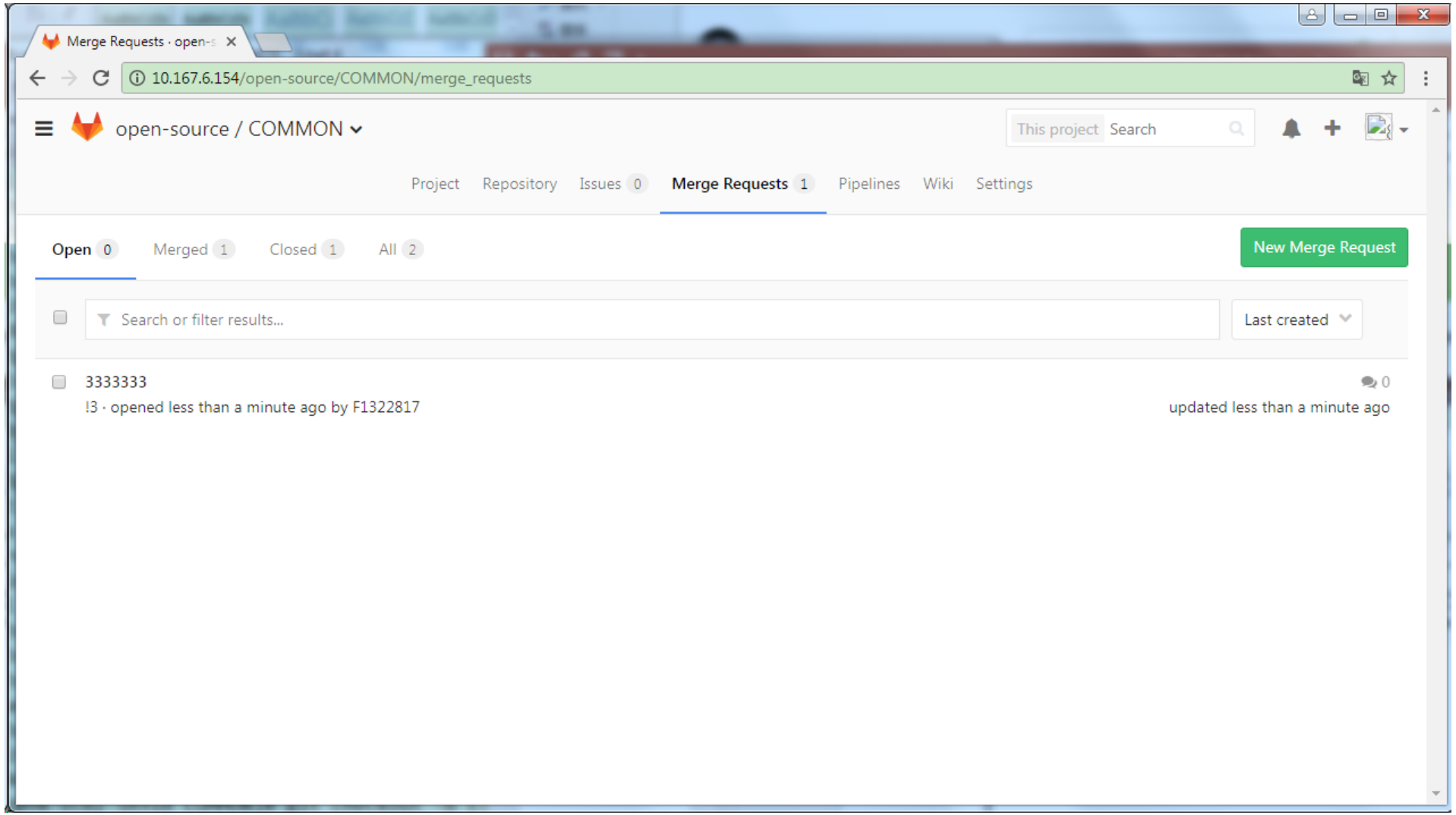
f1322817Feature1

f1322817Bug051

BEACON

Master – 合并分支

管理员收到合并请求，审查代码并决定是否通过合并请求。点击Accept Merge Request通过，代码完成合并。

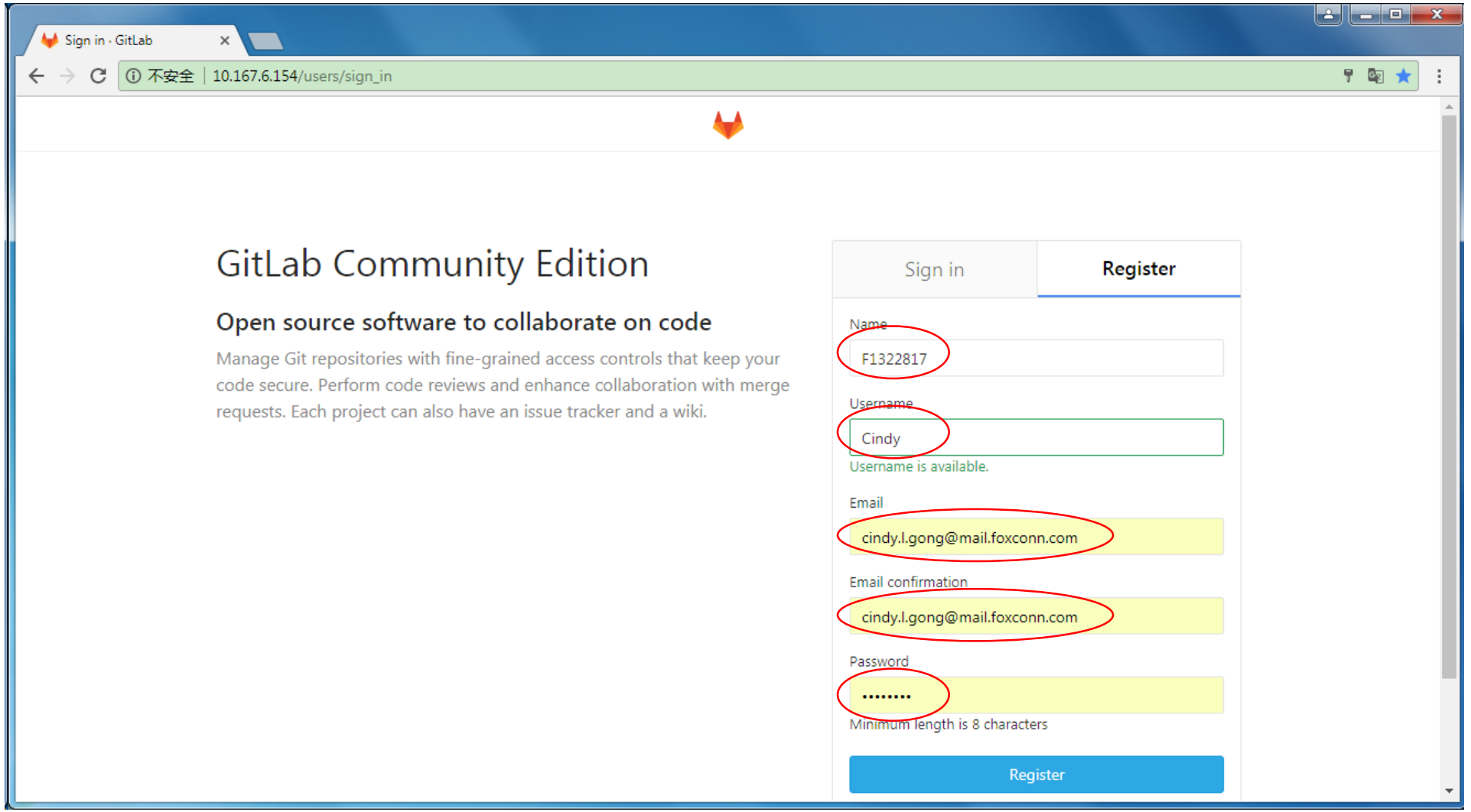


附录

Gitlab基本操作

1.用户注册

按要求填写完所有项目之后单击Register提交。



2.添加个人SSH KEY

在自己电脑上新开Git Bash窗口，用命令ssh-keygen.exe –t rsa生成SSH公钥。在隐形文件夹.ssh下生成的id_rsa.pub即为SSH公钥文件。

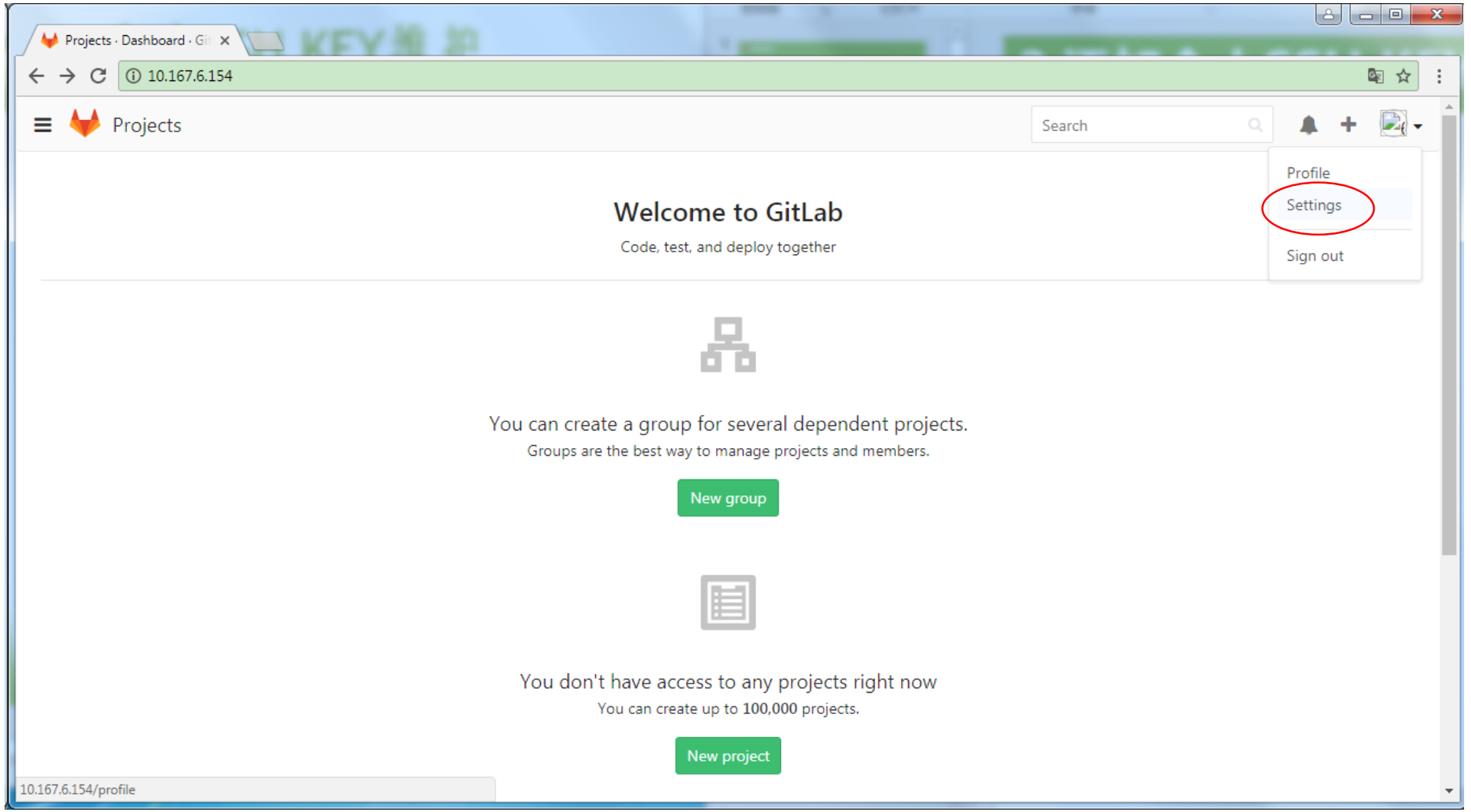
```
Cindy Gong@CINDY-PC MINGW64 ~
$ ssh-keygen.exe -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Cindy Gong/.ssh/id_rsa):
Created directory '/c/Users/Cindy Gong/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Cindy Gong/.ssh/id_rsa.
Your public key has been saved in /c/Users/Cindy Gong/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:4TdO2JZxUqqPQzczSOXm/qxr2azYcZKXgPV5LIu6o98 Cindy Gong@CINDY-PC
The key's randomart image is:
+---[RSA 2048]----+
|          . .    |
|         o +     |
|        o O +    |
|       o % X     |
|        S ^ .    |
|       o X O .   |
|      . o B++    |
|      .o +oBo    |
|     .oo.E.=+o   |
+----[SHA256]-----+
```

```
Cindy Gong@CINDY-PC MINGW64 ~/.ssh
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDC3wD8nF3iflnT+XRfGNNnlH3dJcDAZw48uRbzTNO7fZkJO1rZ/Y4XWN1O3
PBhXgbZ8V/WKFZGeET2xwqWhtrsoTnOfqZLuKiutvR2quKWT8t7QOFipLNGZuxwK6XjhnypTyddzD8fUccDDRo/X/MJ4S/E8Z
jtqmm+SXzcB9Ovd/PvmeVviUQg2r3zRksOgwN2iiXNCw8nPIJx/9kl3ASEPmSqLSZvOYvZf++I6kBO54o2RvlccCKiGWLdomD
sbw22JS7mZoSiM7fbu3AZ/NmyhJkr4L3AZUamrTYL1STTSjzD8dsHbwRXU6HklrRqKucKLG1vesX6PDP3Io3KOnlf Cindy G
ong@CINDY-PC

Cindy Gong@CINDY-PC MINGW64 ~/.ssh
```

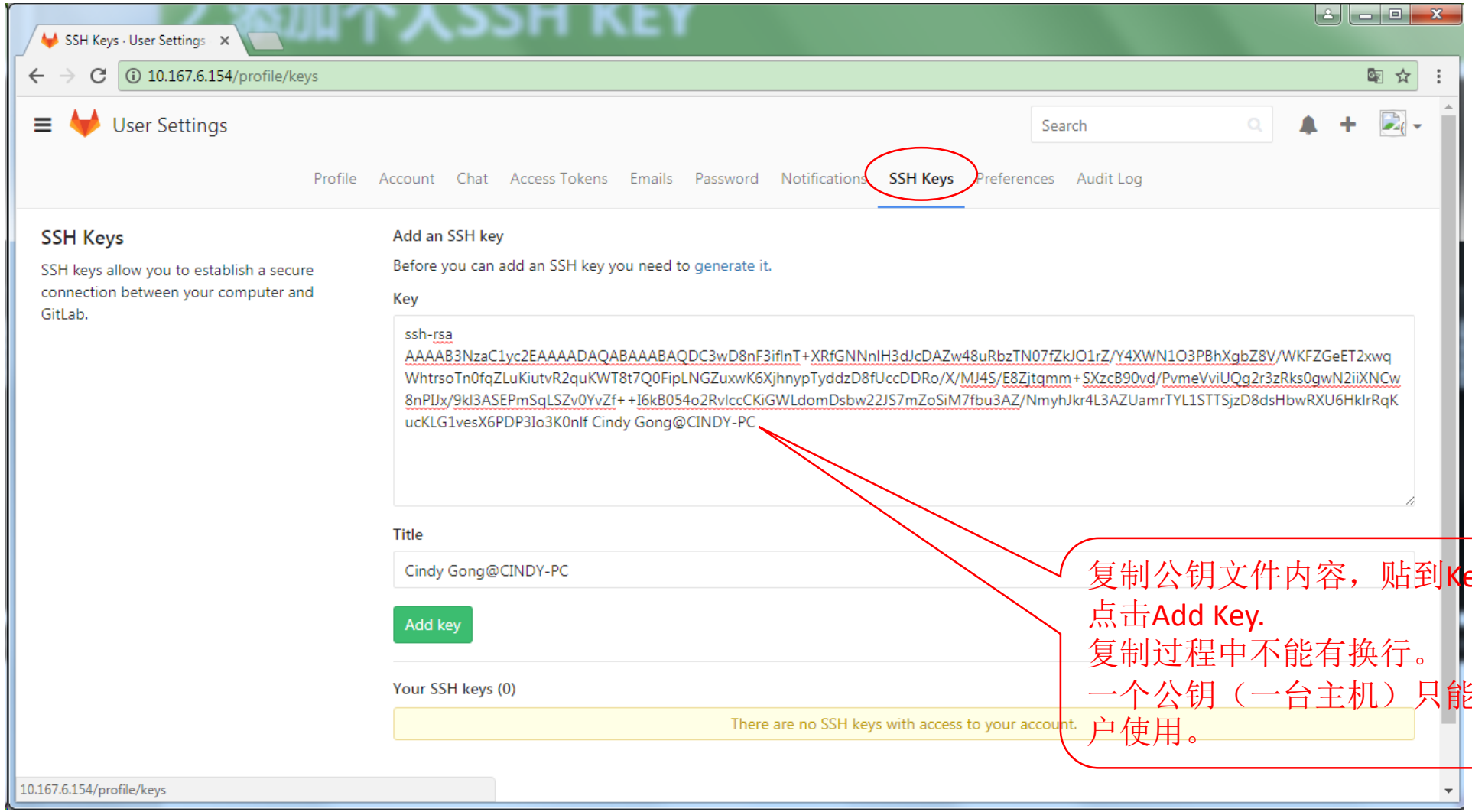
2.添加个人SSH KEY

单击用户头像，选择Settings。



2.添加个人SSH KEY

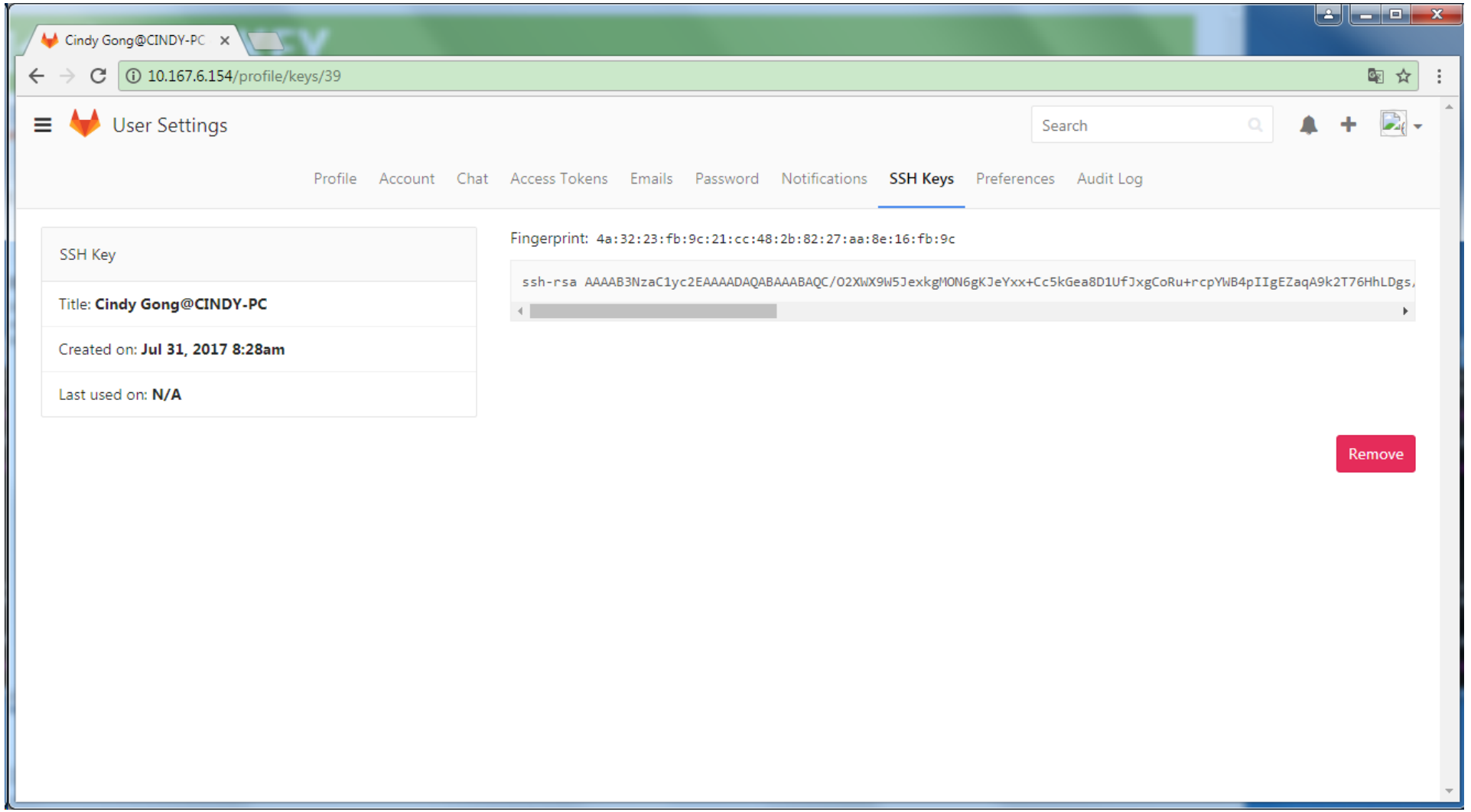
单击SSH Keys切换到SSH Keys配置页面。复制刚才本地生成的公钥文件内容，贴到Key的框框中，点击Add Key添加。



BEACON

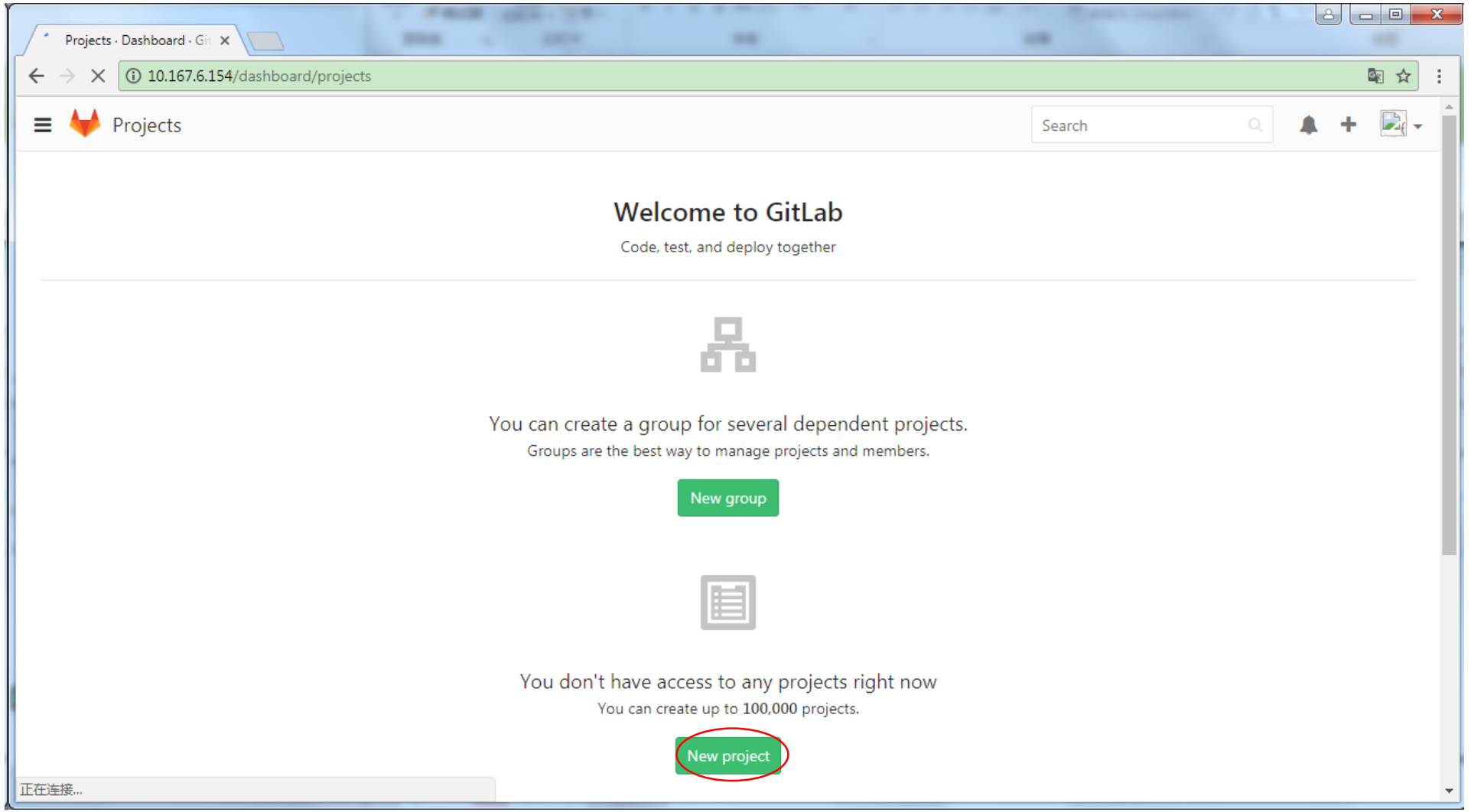
2.添加个人SSH KEY

至此，个人SSH KEY添加完成。



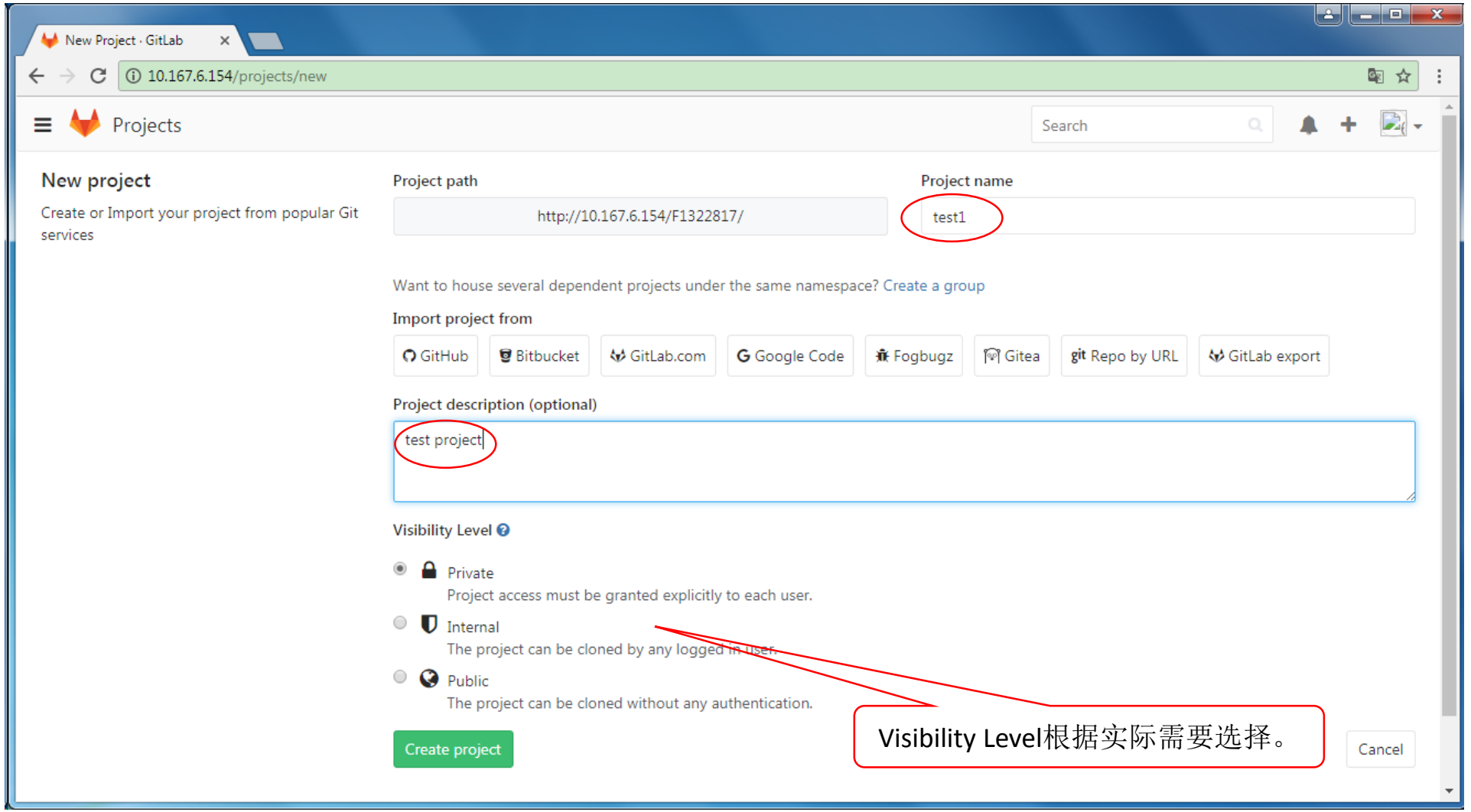
3.新建project

单击左边菜单栏，选择Projects -> New Project



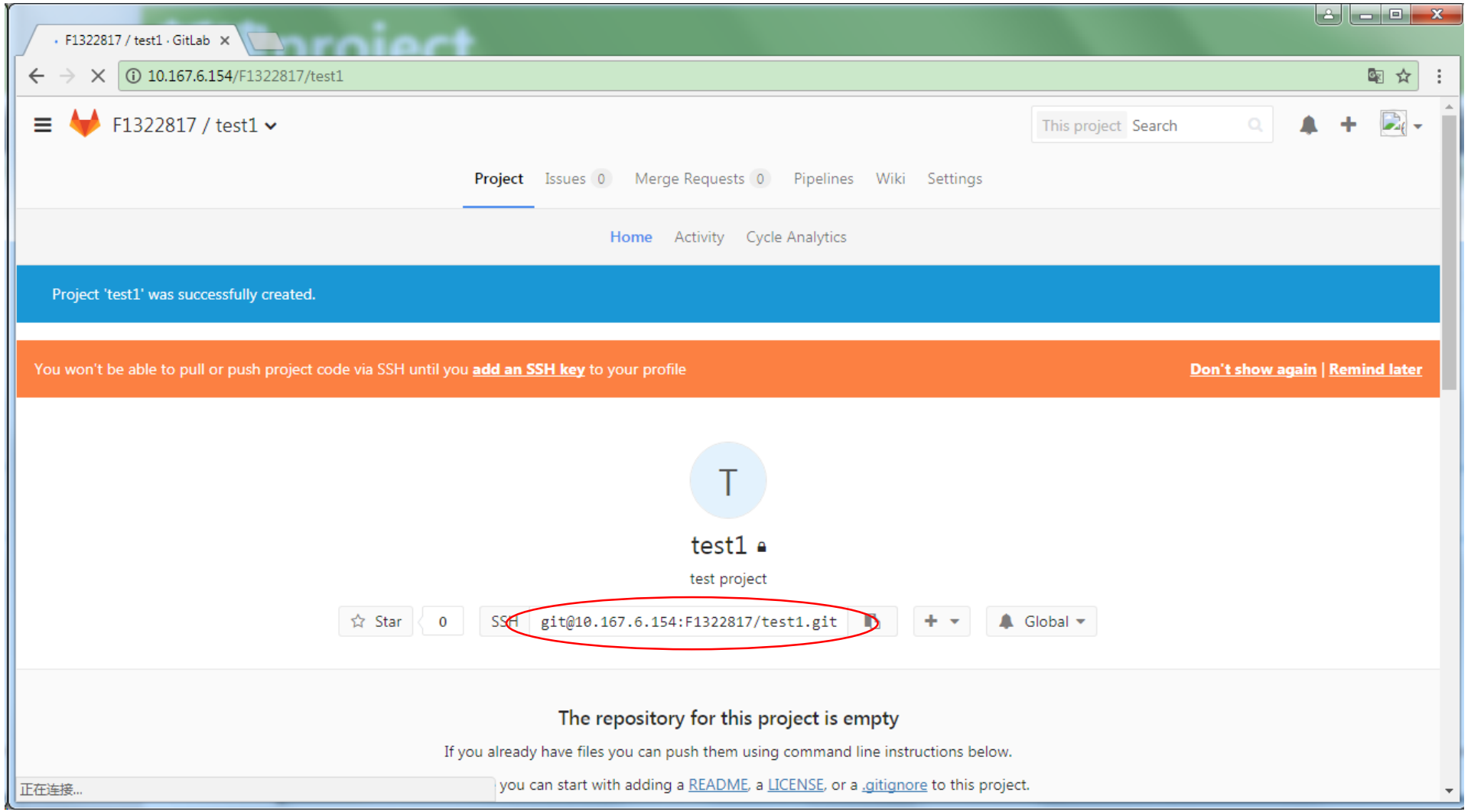
3.新建project

写入项目名称，和项目描述，点击Create project。



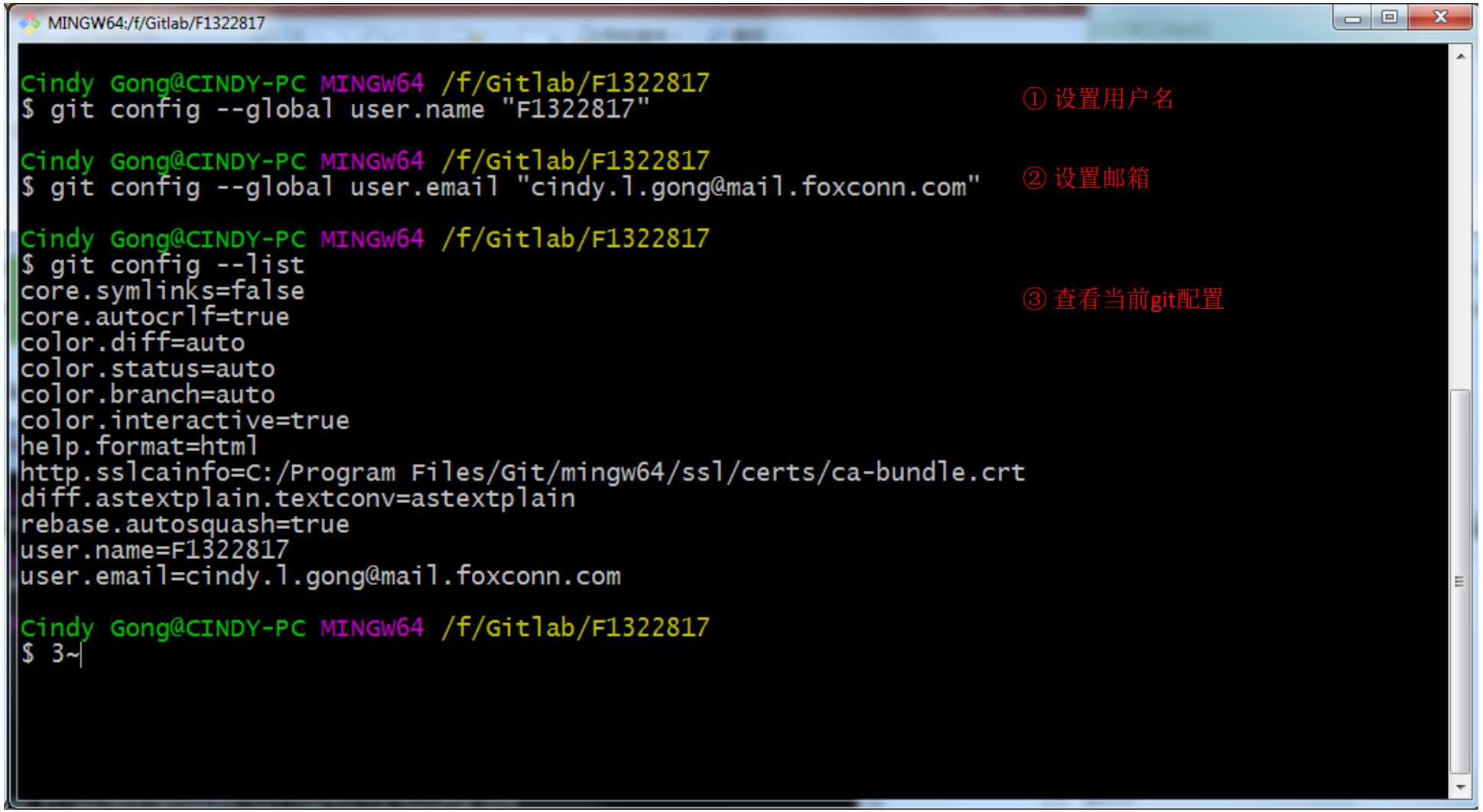
3.新建project

Project创建完成之后，会产生一个以project.git命名的仓库。剩下就是用git上传下载代码，管理自己的项目。



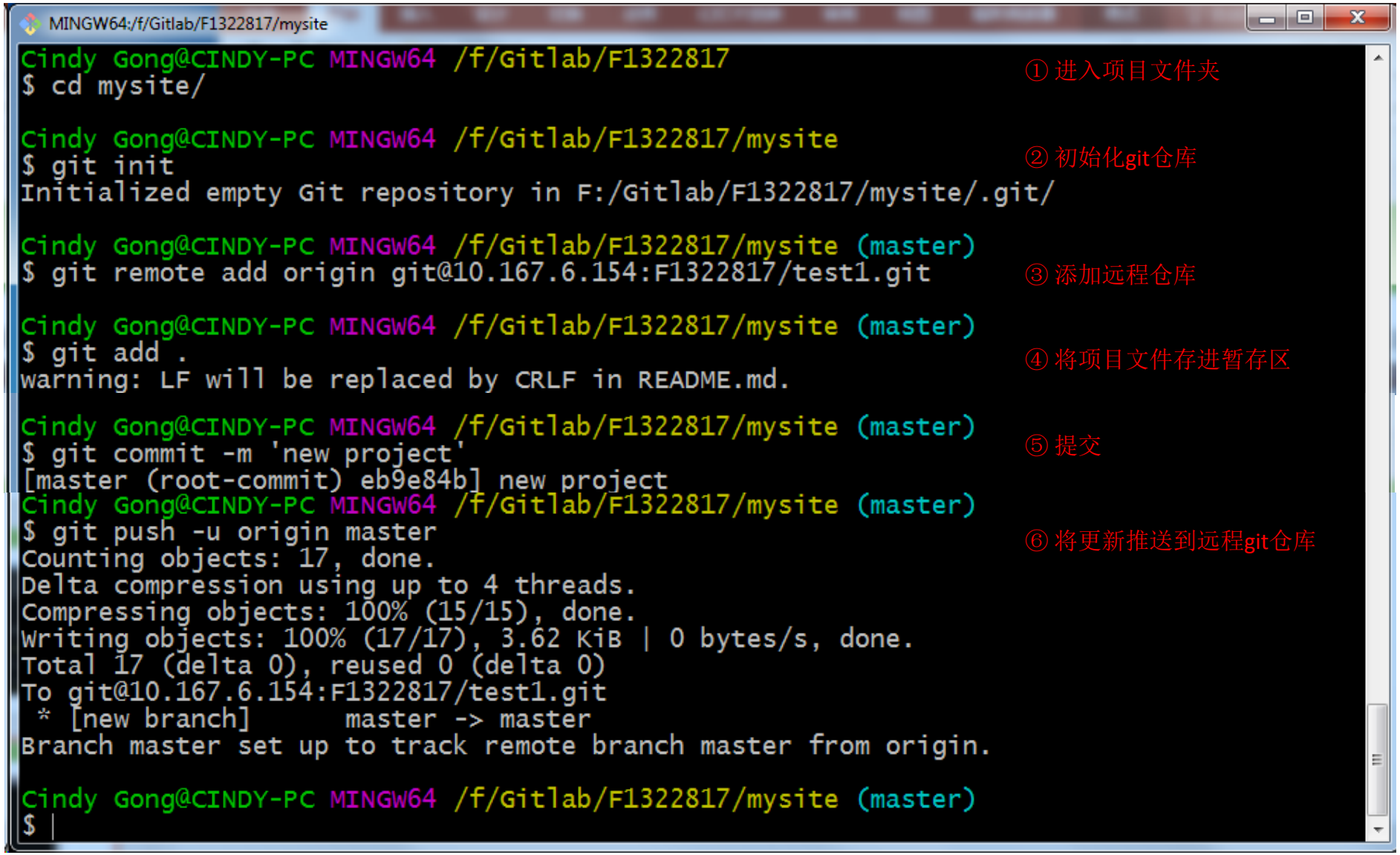
4.导入项目

在本地主机并设置下git的用户名和邮箱。



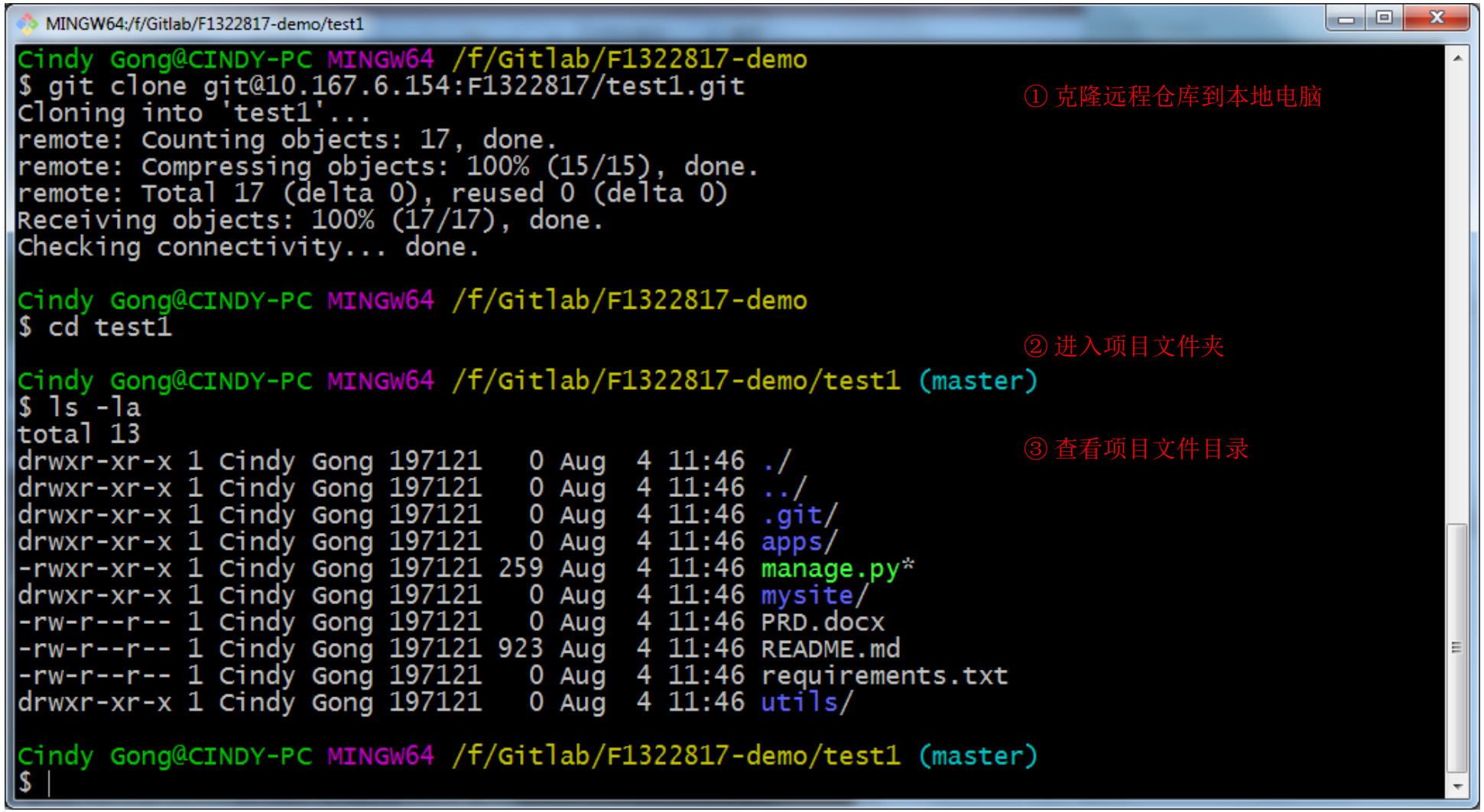
4.导入项目

如果项目在本地已经存在，需要导入到gitlab，可以通过命令直接将项目导入上去。



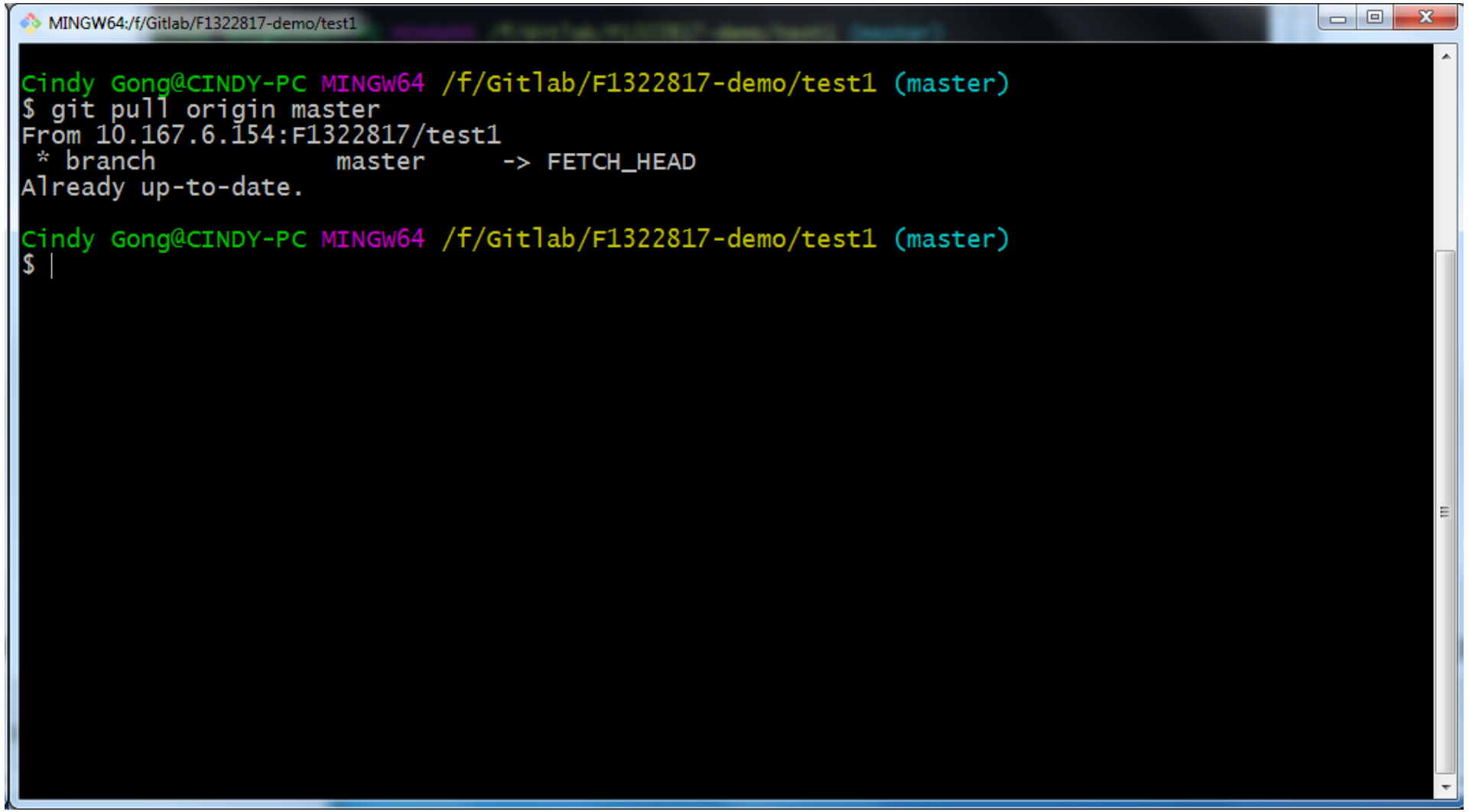
4.导入项目

如果本地没有项目仓库，可以从gitlab上把项目克隆下来。



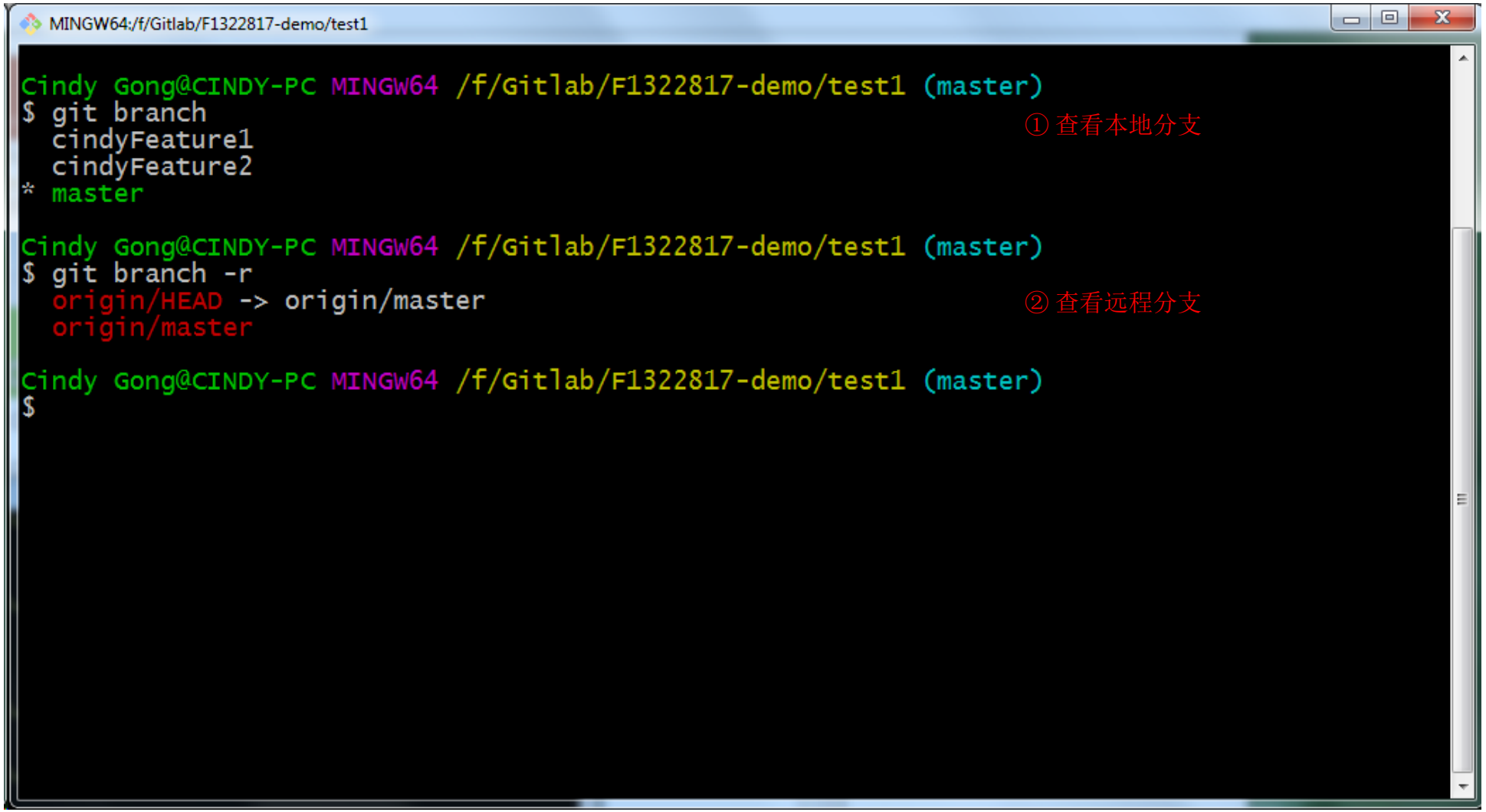
4.导入项目

如果已经拉取过远程仓库，每次在本地编辑代码之前，应该从远程获取最新版本并merge到本地。



5.分支(branch)操作

查看分支：git branch



5.分支(branch)操作

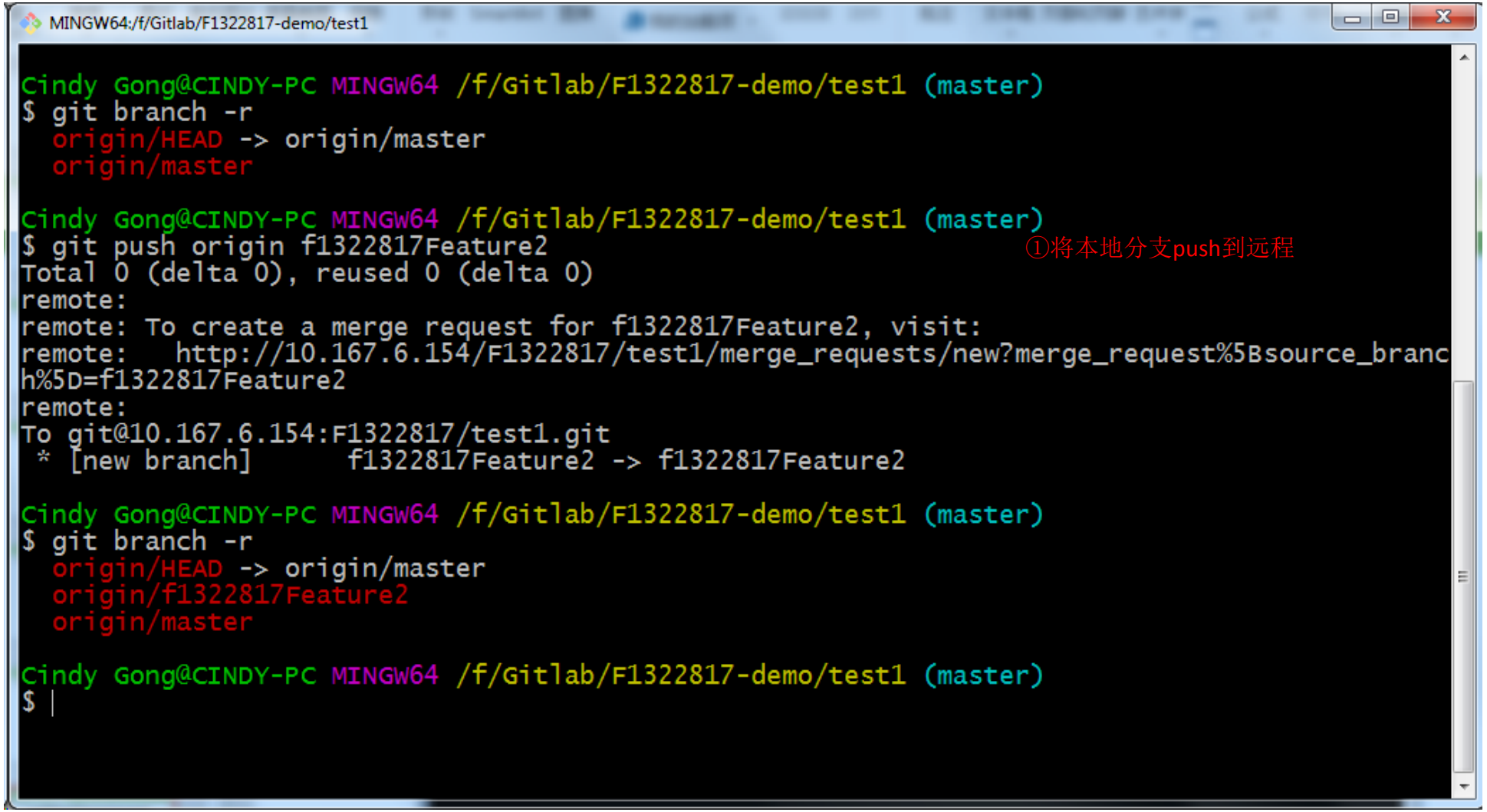
创建本地分支并切换到分支：git branch [name] , git checkout [name]

5.分支(branch)操作

删除本地分支：git branch -d [name]。注意删除分支时该分支不能存在还未合并的代码。

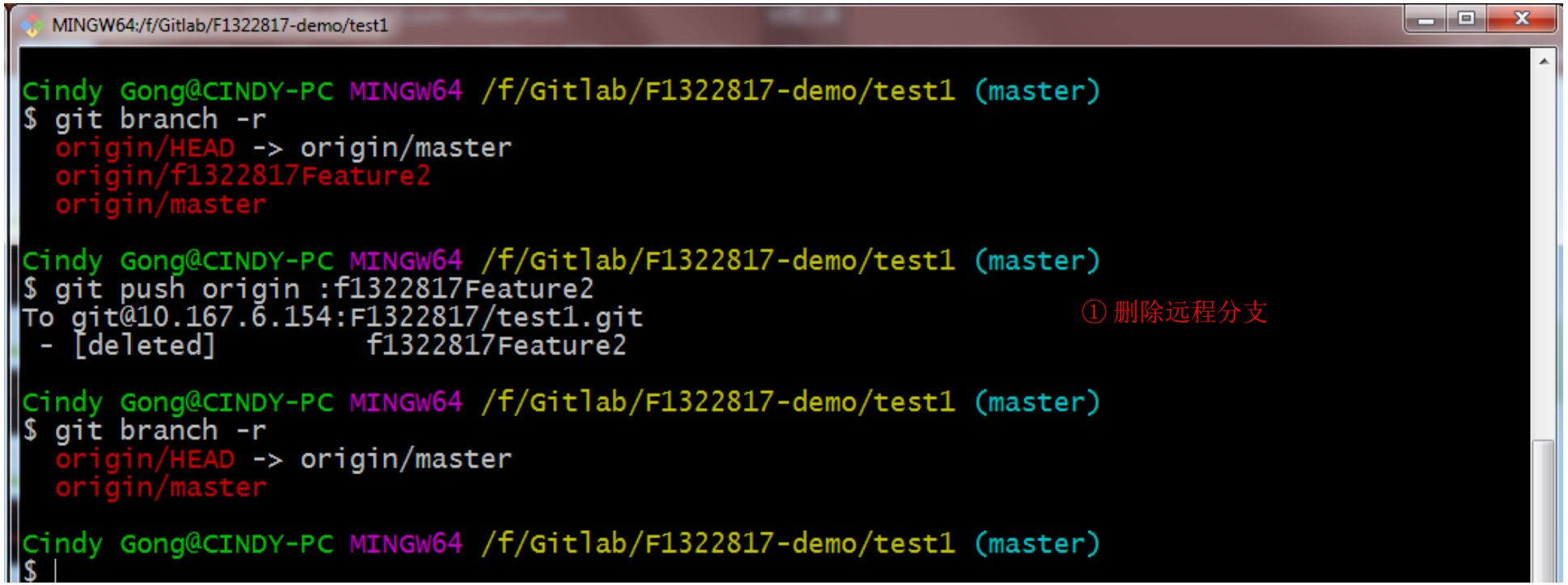
5.分支(branch)操作

创建远程分支：（创建本地分支，并push到远程）git push origin [name]



5.分支(branch)操作

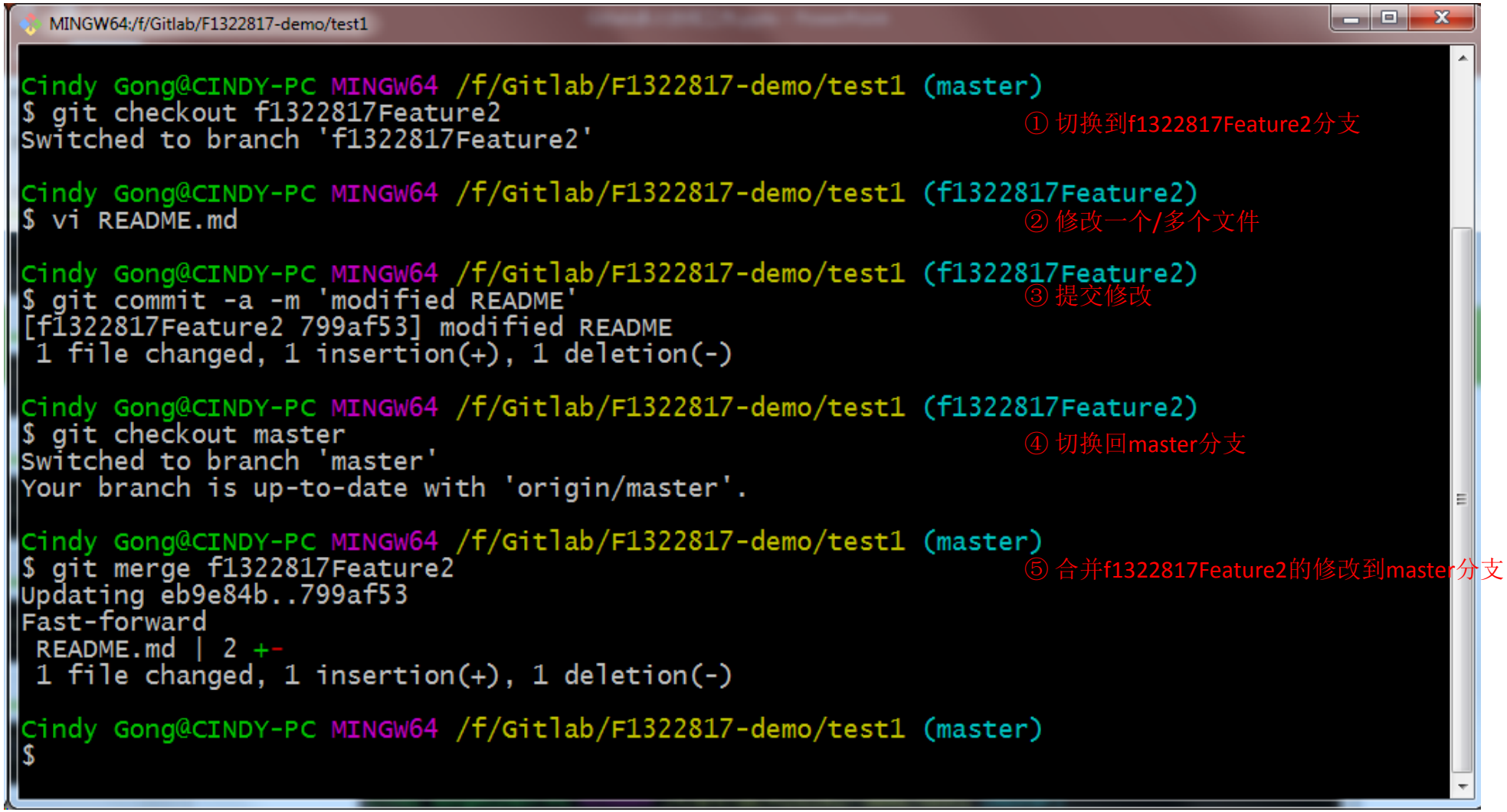
删除远程分支：git push origin :[name]



把本地某个分支提交到远程仓库，并作为远程仓库的分支，完整命令是：git push origin [name]:[name]。
删除远程分支的命令其实就是让左边的分支为空，其作用是删除远程的分支。但是该分支在本地还是会保存的。

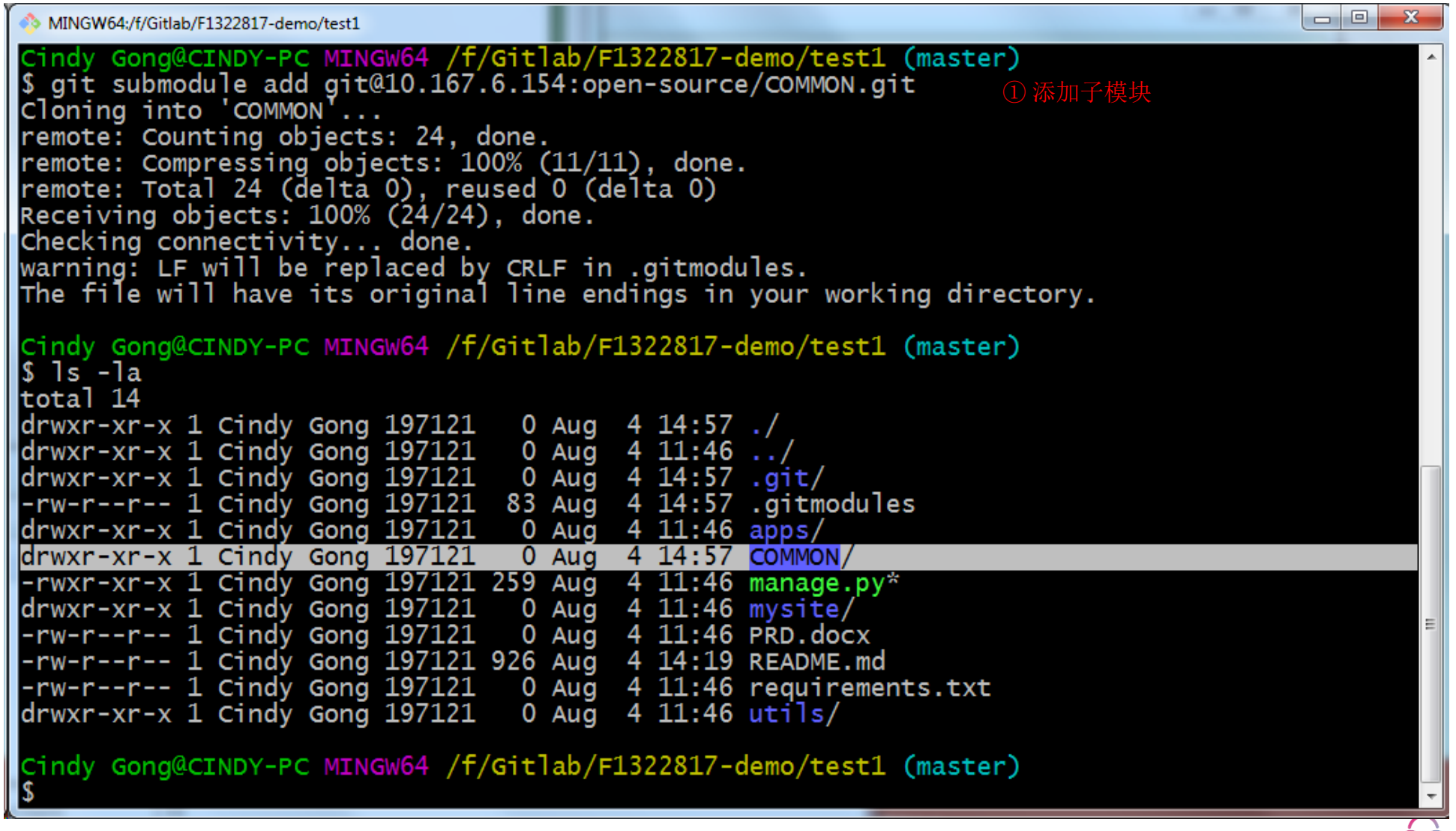
5.分支(branch)操作

合并分支：git merge [name]



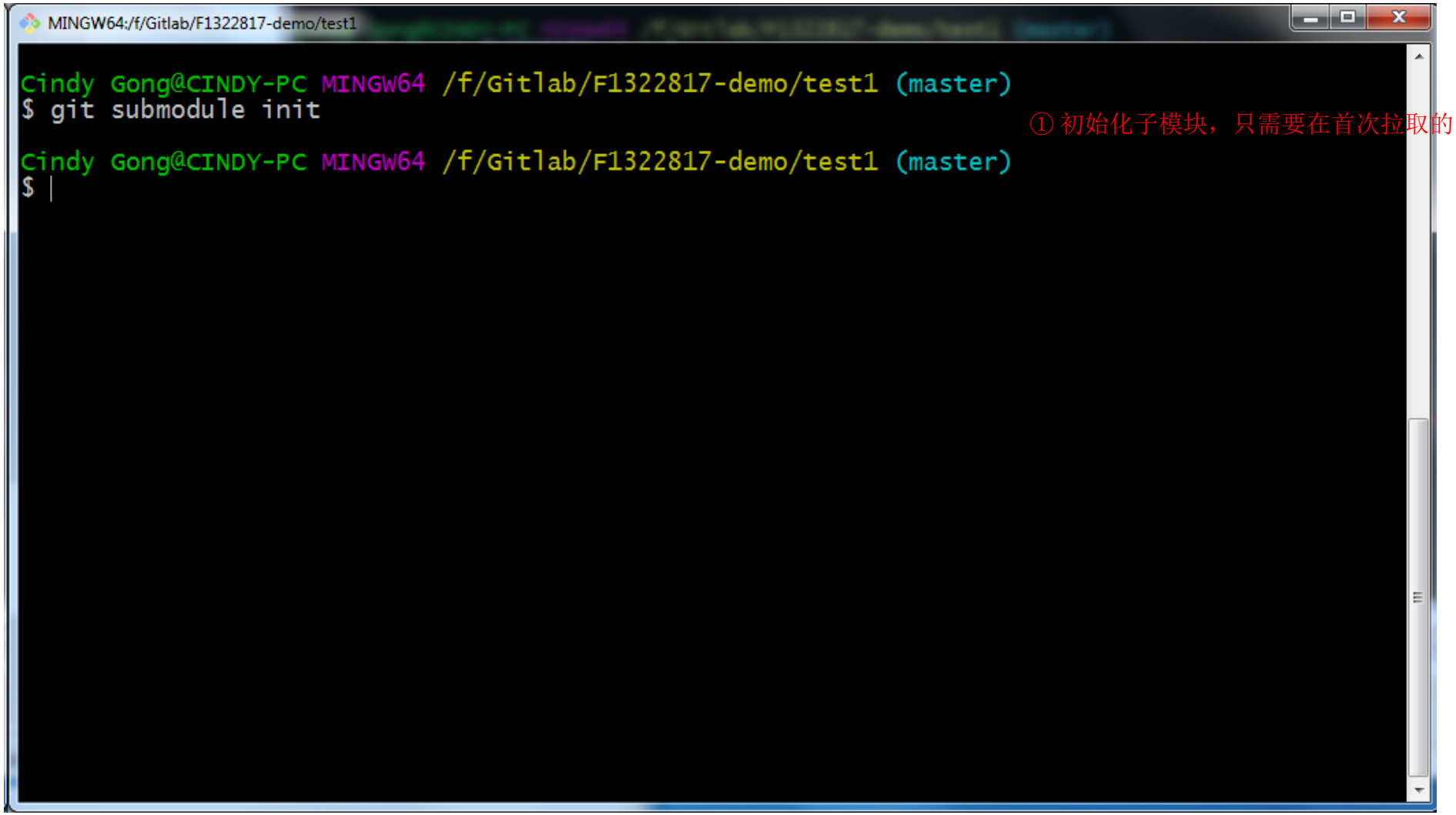
6.子模块(submodule)操作

添加子模块：git submodule add [url][path]



6.子模块(submodule)操作

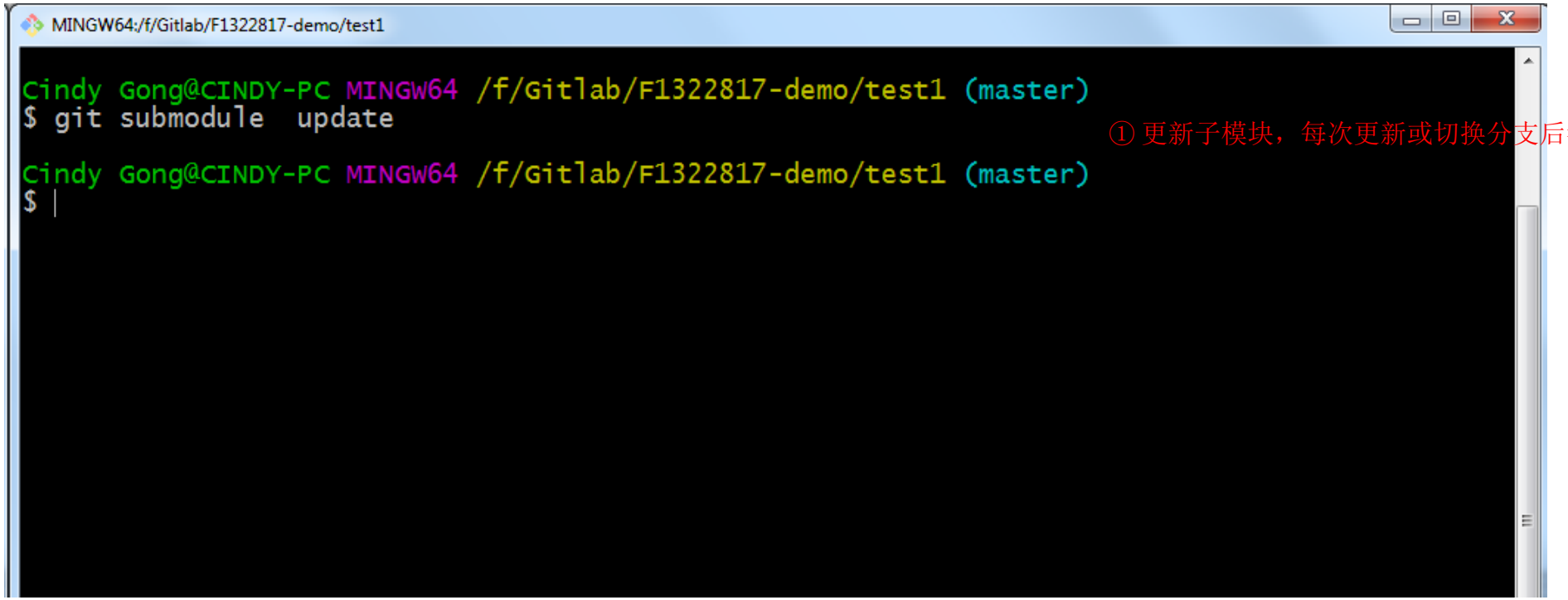
初始化子模块：git submodule init



① 初始化子模块，只需要在首次拉取的时候运行一次。

6.子模块(submodule)操作

更新子模块：git submodule update

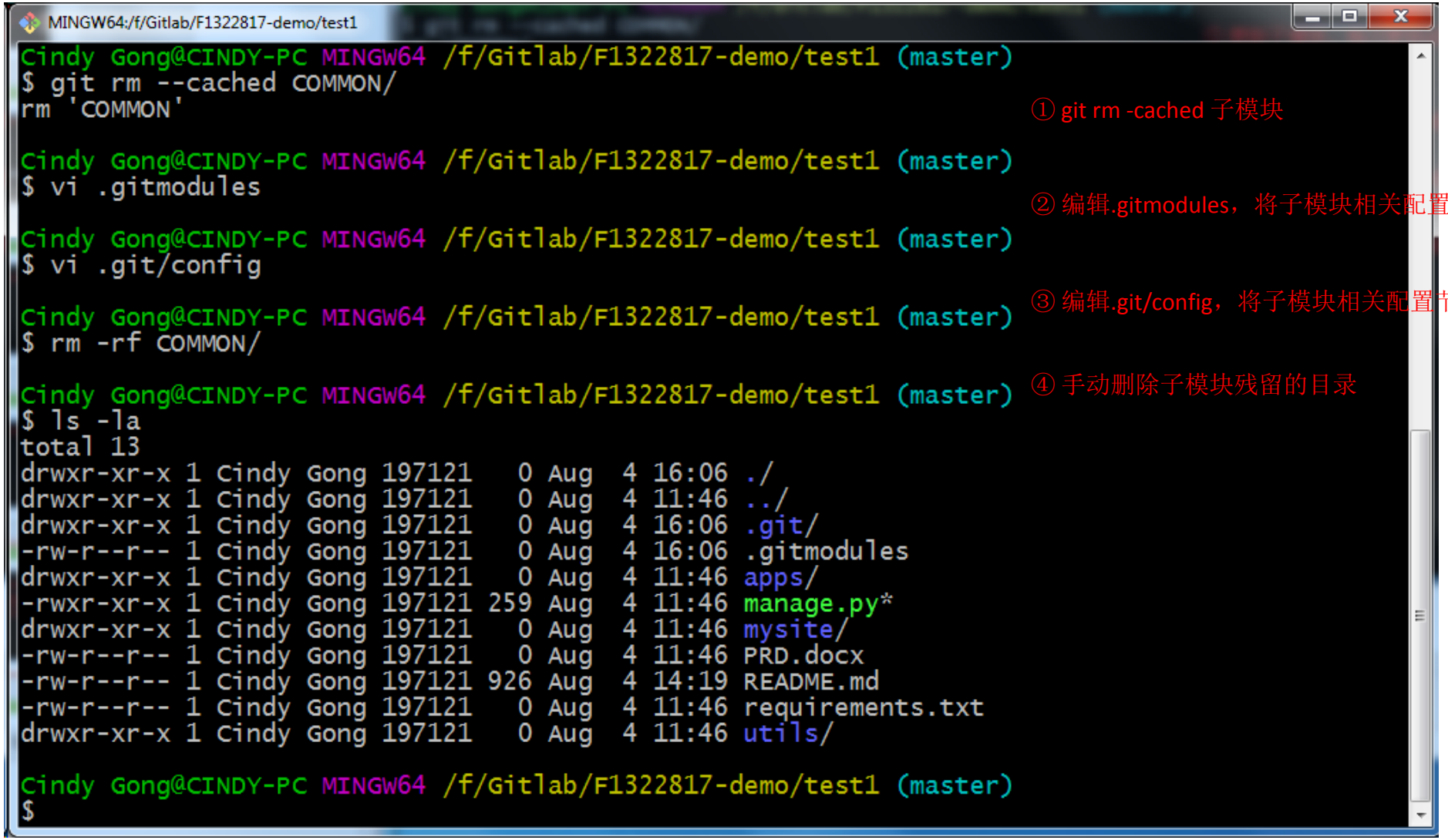


① 更新子模块，每次更新或切换分支后都需要运行一下。

submodule项目和它的父项目本质上是2个独立的git仓库。只是父项目储存了它依赖的submodule项目的版本号而已。如果同组的同事更新了submodule，然后更新了父项目中依赖的版本号，你需要在git pull之后，调用git submodule update来更新submodule的信息。

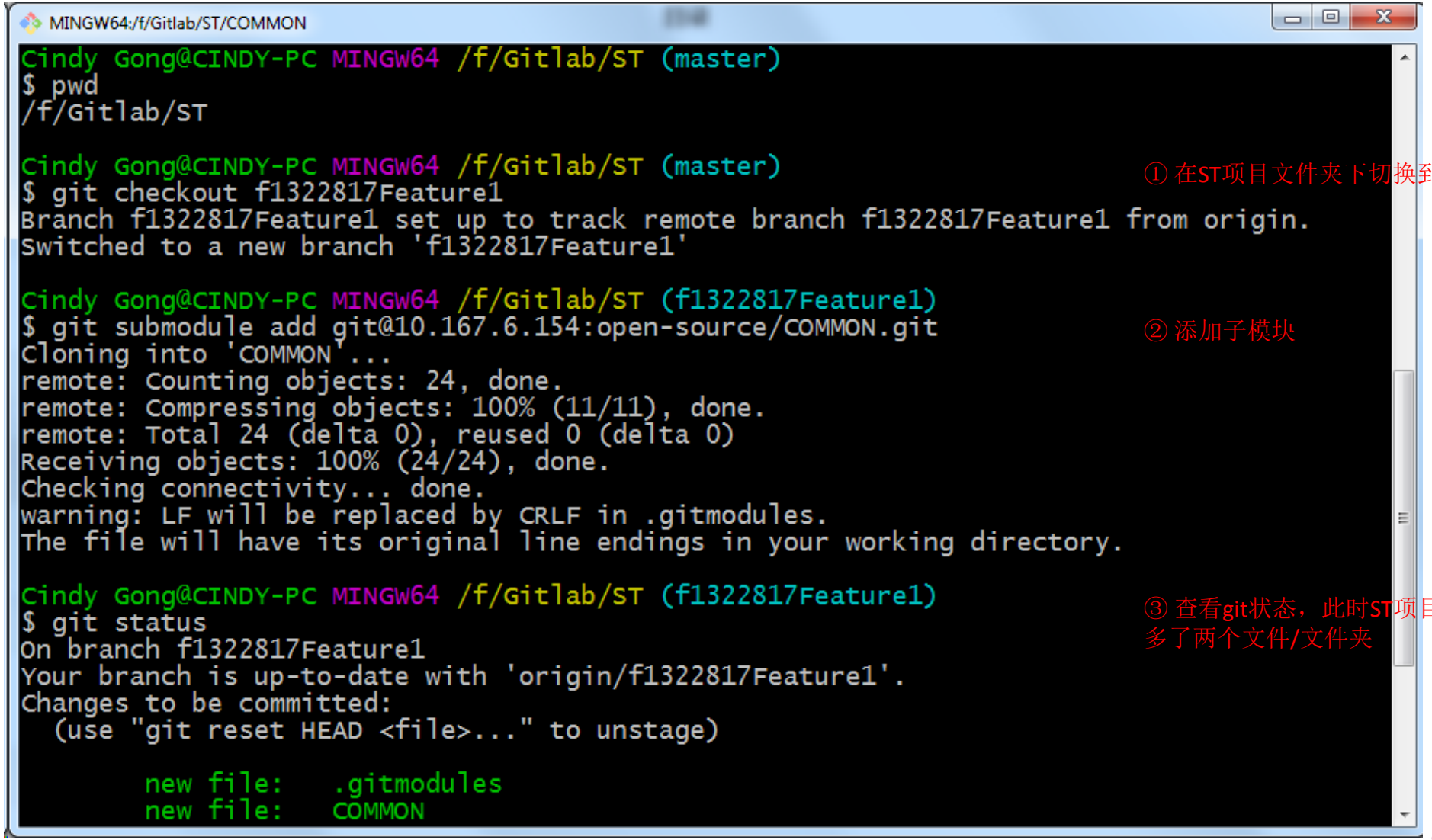
6.子模块(submodule)操作

删除子模块，有4个步骤



6.子模块(submodule)操作 - 子模块实例

场景：本地已有ST项目的仓库，从远程拉取COMMON作为ST项目的子模块，更新子模块并push到远程。



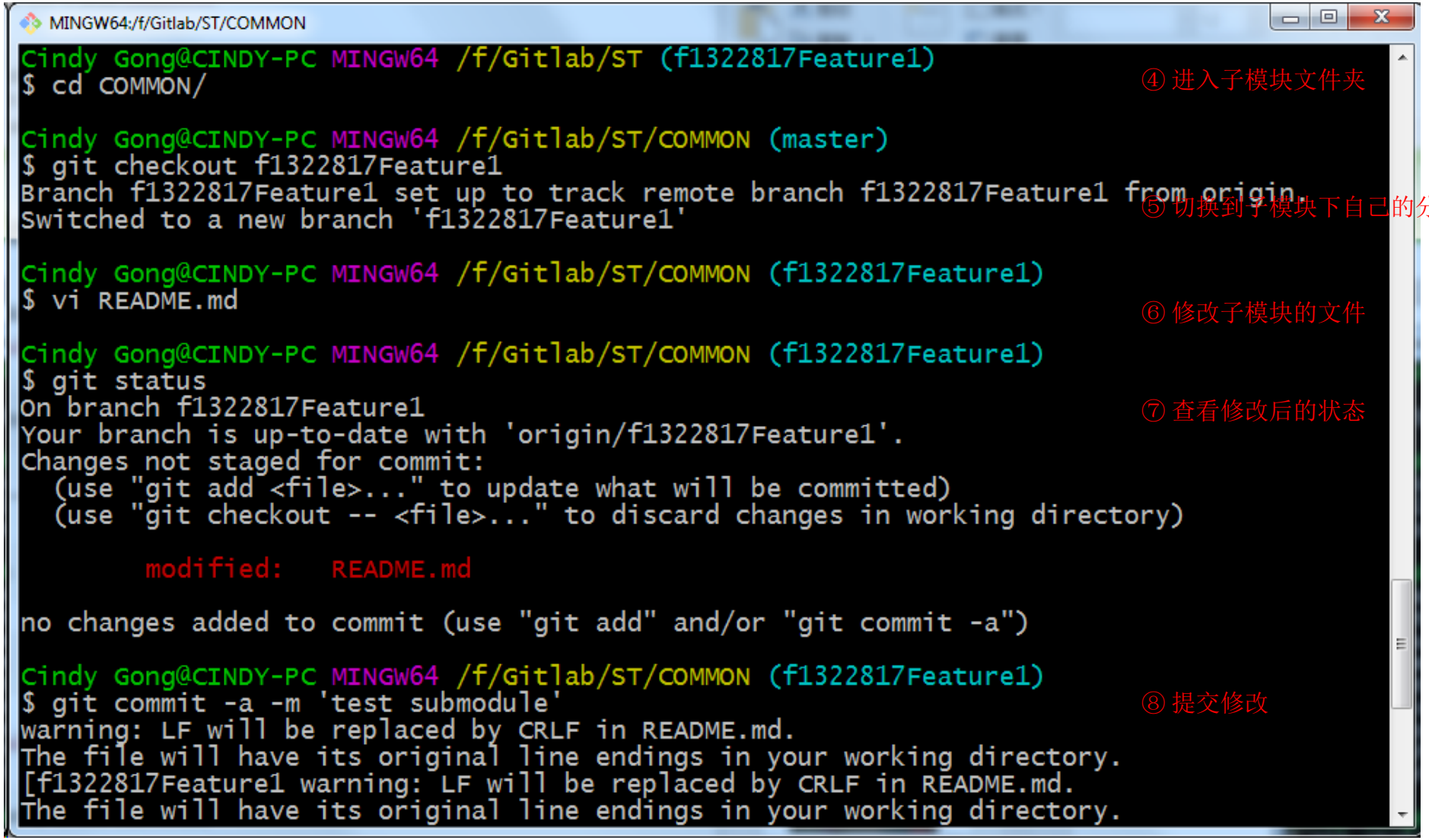
① 在ST项目文件夹下切换到自己的分支

② 添加子模块

③ 查看git状态，此时ST项目文件夹下多了两个文件/文件夹

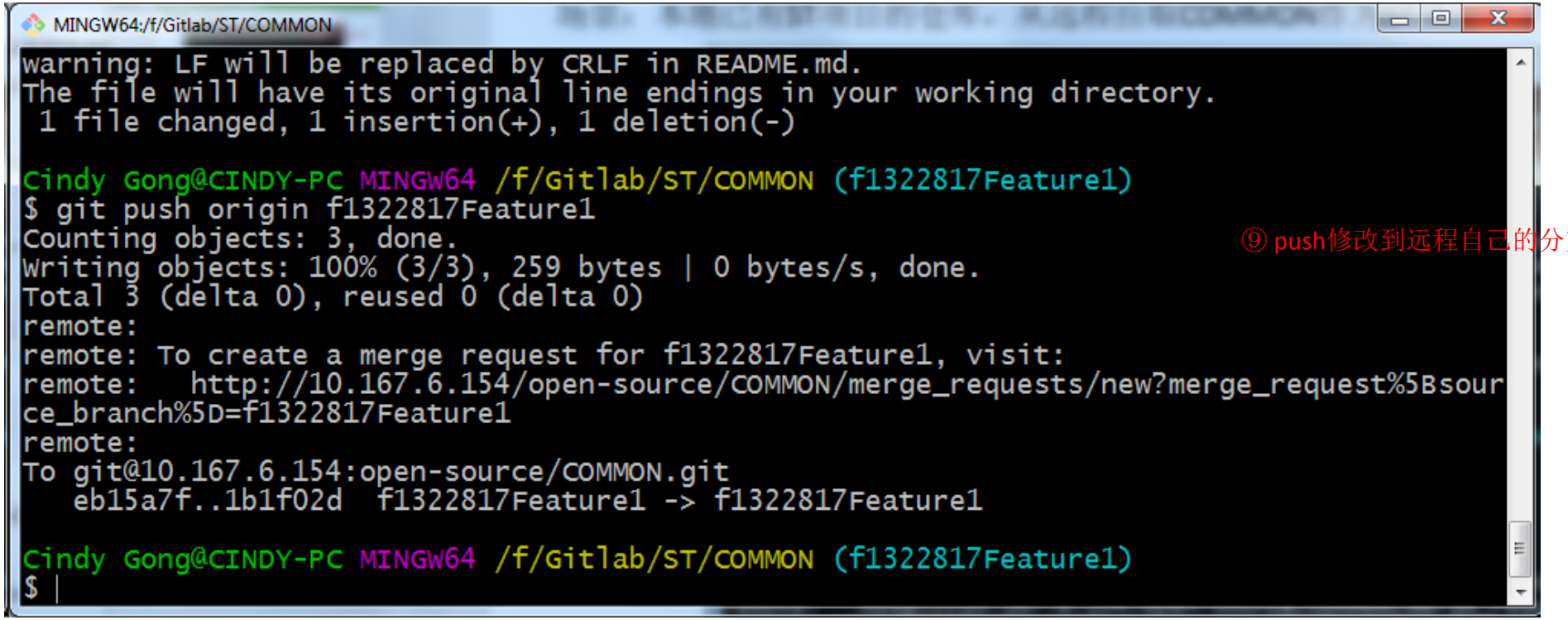
6.子模块(submodule)操作 - 子模块实例

场景：本地已有ST项目的仓库，从远程拉取COMMON作为ST项目的子模块，更新子模块并push到远程仓库。



6.子模块(submodule)操作 - 子模块实例

场景：本地已有ST项目的仓库，从远程拉取COMMON作为ST项目的子模块，更新子模块并push到远程仓库。



⑨ push修改到远程自己的分支

之后的操作就是去Gitlab网页上发起合并分支申请，等待管理员合并子模块的代码去master分支了。

特别注意：

在主项目(ST)和子模块（COMMON）之间切换目录的时候，切换目录的同时特别要注意切换自己的分支，否则所有的操作都跟自己的用户无关，最后的修改无效无法提交。

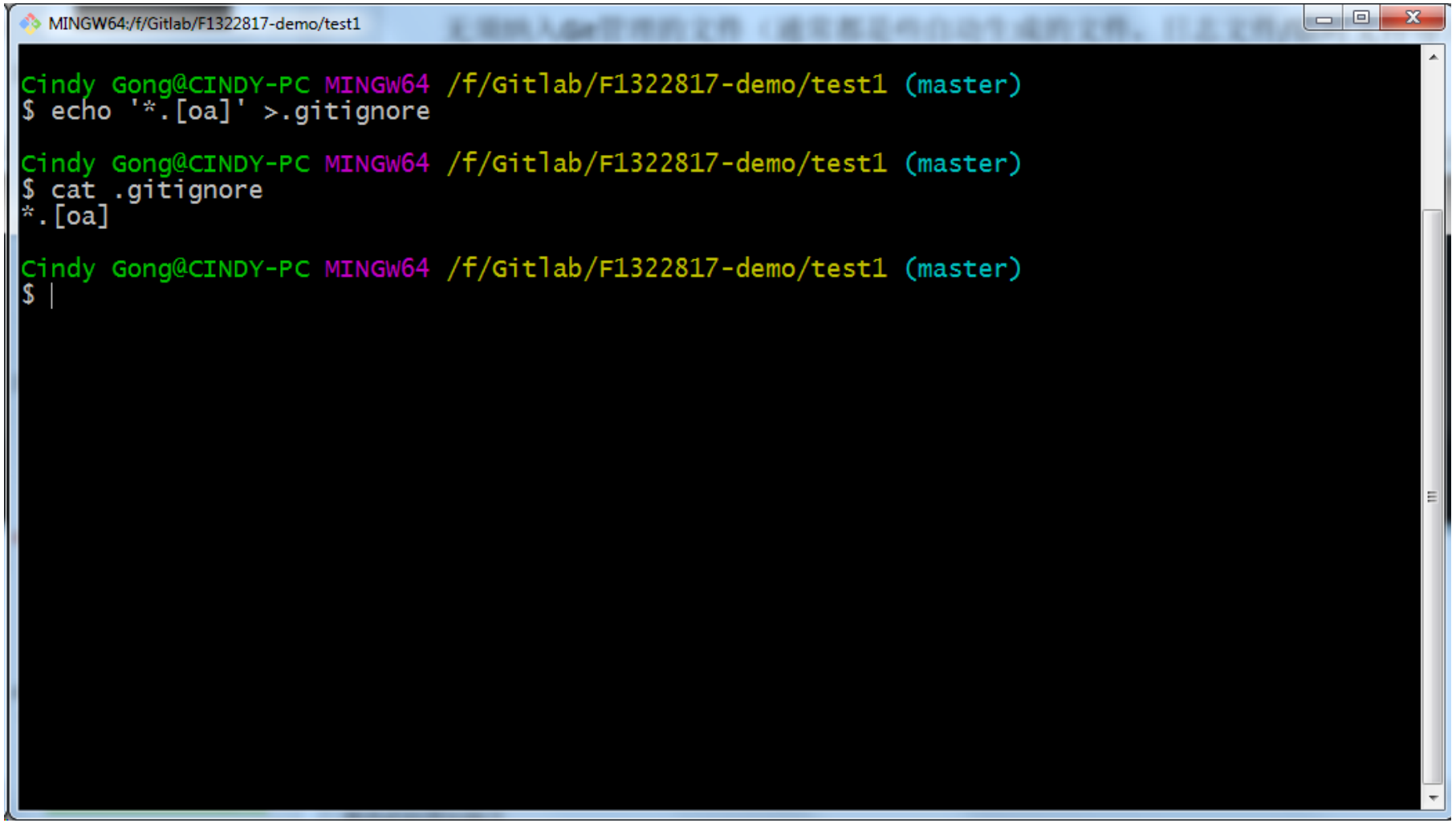
6.子模块(submodule)操作

使用submodule注意事项：

1. 对submodule做修改操作前，确保切换到正确的submodule项目分支。因为进入submodule文件夹，分支状态还保留在父项目的分支下。对submodule做修改前，要立即切换到submodule下你的本地分支。
2. git pull 之后，立即执行git status，如果发现submodule有修改，立即执行git submodule update。
3. 尽量不要使用git commit -a，git add存在的意义就是让你对加入缓存区的文件做第二次确认，而git commit -a相当于跳过了这个确认过程。

7.忽略的文件/文件夹

无须纳入Git管理的文件（通常都是些自动生成的文件，日志文件/临时文件等），可以通过创建一个名为.gitignore的文件来管理。



Git官方文档（中文）

https://git-scm.com/book/zh/v2

FOXCONN®

The End

BEACON