实验一：Git 和 Github 的使用

一、实验目的：

1.掌握 github 的使用。

2.掌握在 windows 环境下 git 的安装。

3.学会通过 git 创建本地仓库，并在本地仓库创建文件夹，创建文件，然后推送到 github。

4.学会通过 git 拉取 github 的文件，修改并推送至 github 仓库。

扩展内容：

1.了解基于 master 主分支上新建 develop 分支，并在上面进行文件的增删改操作并推送到 github。

2.了解学习通过 Tag 标记版本号，并提交推送到分支。

3.学会在 IntelliJ IDEA 等开发工具中使用 git 和 github。

二、实验要求：

（一）掌握 github 的使用

1. 注册 github 个人账户
2. 在 github 个人账户中创建仓库

（二）掌握 git 的安装

1. 在 git 官网上下载相应的安装包
2. 通过安装包进行安装，并检查是否安装成功
3. 环境配置，设置用户名，邮箱。

（三）Git 创建本地化仓库，并推送到 github

1. 初始化本地仓库
2. 添加文件到版本库
3. 把添加的文件提交到版本库，并填写提交备注
4. 把本地库与远程库关联
5. 推送本地文件到远程库
6. （扩展内容）创建一个 tag 标签，推送到 github 仓库

（四）通过 git 拉取 github 的文件，修改并推送至 github 仓库

1. 获取 github 仓库的地址，通过 gitl 拉取该地址的文件到本地
2. 新增或者修改本地文件
3. 推送到 github 远程仓库，到 github 仓库中检查是否推送成功

（五）（扩展内容）基于 master 主分支上新建 develop 分支

1.master 分支上新建 develop 分支

2.检出 develop 分支

3.进行文件的操作，开发等，然后推送到 github 的 origin/develop

（六）（补充内容）在 IntelliJ IDEA 等开发工具中使用 git 的常见的应用场景

假设小组中有两个人，组长小张，组员小袁

场景一：小张创建项目并提交到远程 GitHub 仓库

场景二：小袁从远程 Git 仓库上获取项目源码

场景三：小袁修改了部分源码，提交到远程仓库

场景四：小张从远程仓库获取小袁的提交

场景五：小袁接受了一个新功能的任务，创建了一个分支并在分支上开发

场景六：小袁把分支提交到远程 Git 仓库

场景七：小张获取小袁提交的分支

场景八：小张把分支合并到主干

三、参考资料：

视频：两小时学会 Github（若干视频）

网页：https://boxueio.com/series/git-essential 在开发工作中引入 git 版本控制

四、实验过程：

（一）掌握 github 的使用

1. 基本概念：

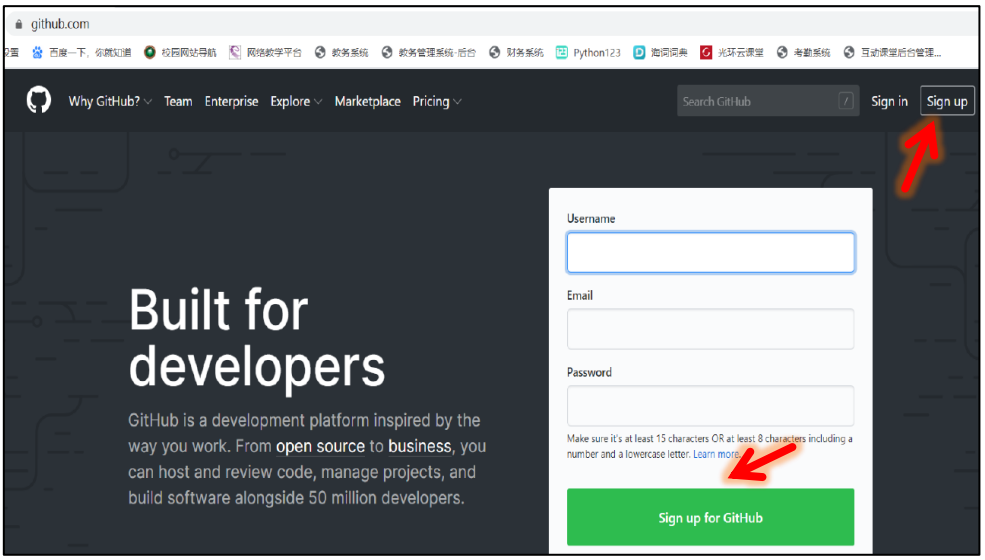
参见视频“01-使用 Github（目的、基本你概念、注册账号）.wmv”

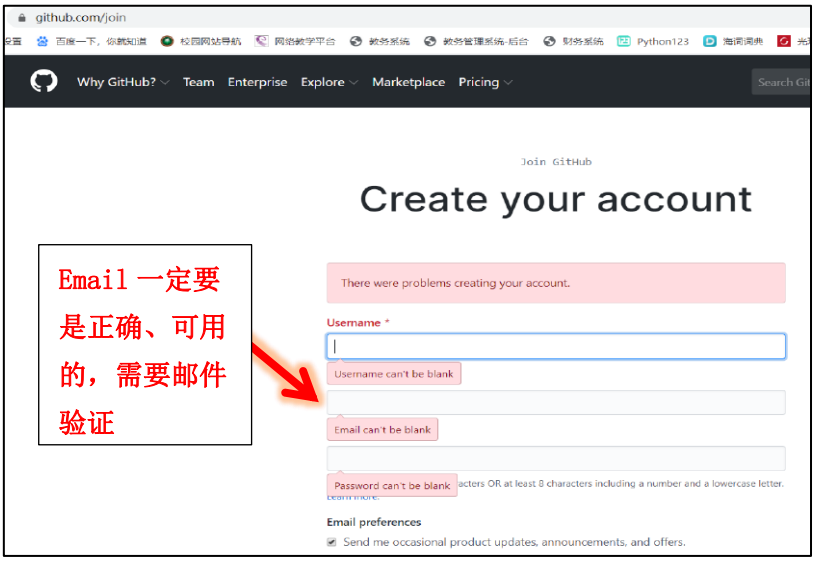
注意其中“仓库 Repository”、“克隆 Fork”、“Github 主页”、“仓库主页”等概念

2. 注册 github 账户：

参见视频“01-使用 Github（目的、基本概念、注册账号）.wmv”

自行到 github 官网首页去注册，https://github.com/。





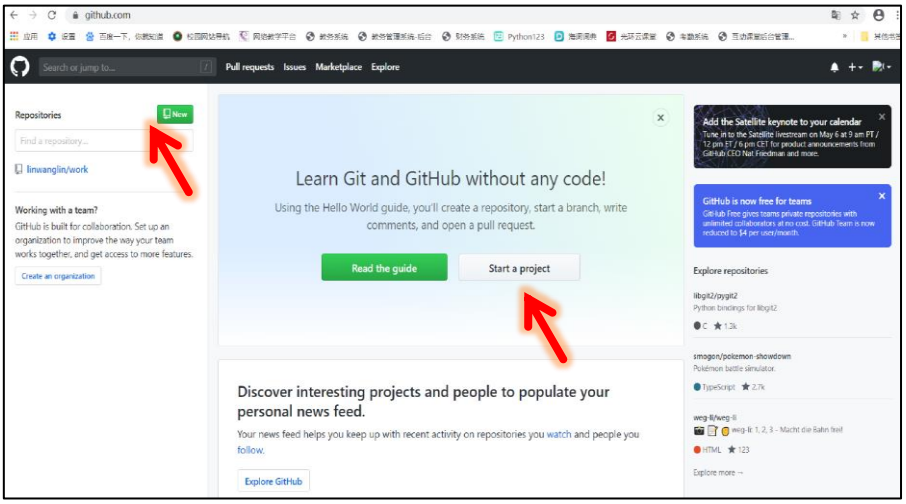
注意事项：

1、因为 github 在国外服务器所以访问较慢或者无法访问，需要翻墙（Shadowsocks）
2、私有仓库只能自己或者指定的朋友才有权限操作（私有仓库是收费的）
3、新注册的用户必须验证邮箱后才可以创建 git 库仓库（）

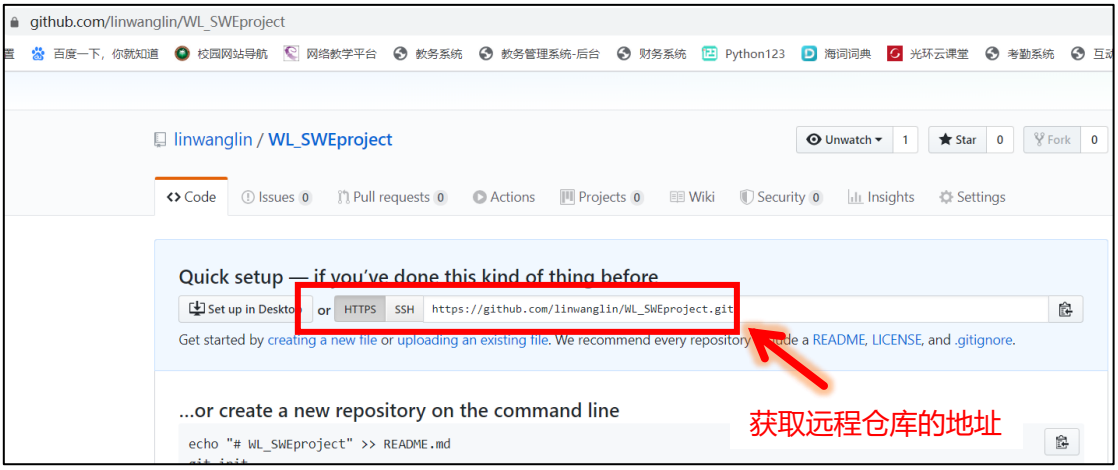
验证邮箱的内容：参见视频“02-使用 Github（创建仓库、仓库主页说明）.wmv”

3. 在 github 个人账户中创建仓库

参见视频“02-使用 Github（创建仓库、仓库主页说明）.wmv”（从 10:07 开始）



创建后进入项目主页：

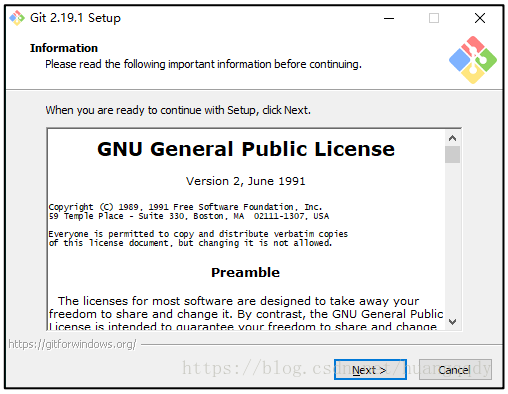


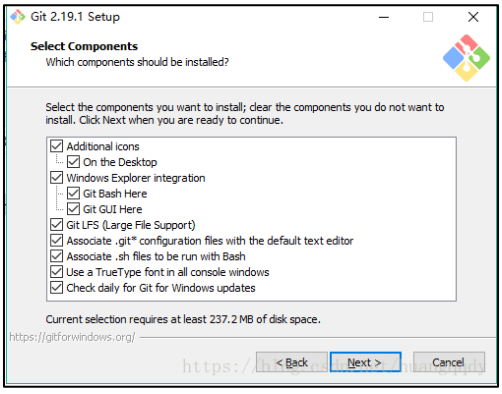
（二）掌握 git 的安装

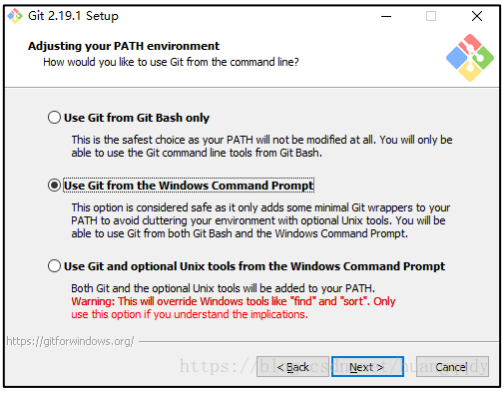
1.下载相应的安装包

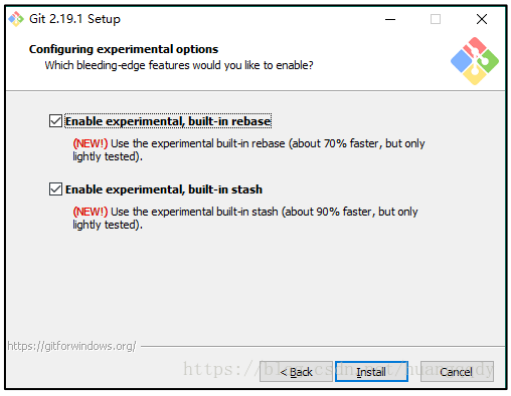
到 git 官网地址：https://git-scm.com/download

点击下载好的安装包安装这个软件，一直点击 next，直到出现 install，点击 install，安装完成后点击 finish：

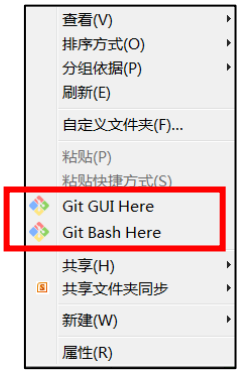








安装好后，在电脑桌面界面点击鼠标右键，会出现如下界面：



2. 检查 git 是否安装 OK：

打开 cmd 界面，输入 git，回车。

```
命令提示符
Microsoft Windows [版本 10.0.18362.778]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\linwa>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status
```

3. 配置用户名和邮箱：

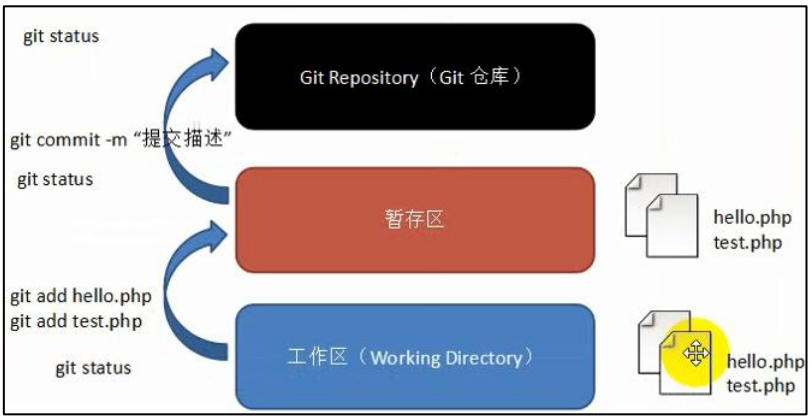
```
# 配置用户名
git config --global user.name "username"
# 配置邮箱
git config --global user.email "username@email.com"
```

注意：上面的"username"和"username@email.com"是用户自己的账户名和邮箱，可以和 github 一致，可以不一致。

4. 查看配置是否 OK

以上命令执行结束后，可用 **git config --global --list** 命令查看配置是否 OK

（三）Git 创建本地化仓库，并推送到 github



Git 工作流程（参见视频“07-Git 基本工作流程.wmv”）

1. 打开命令行窗口

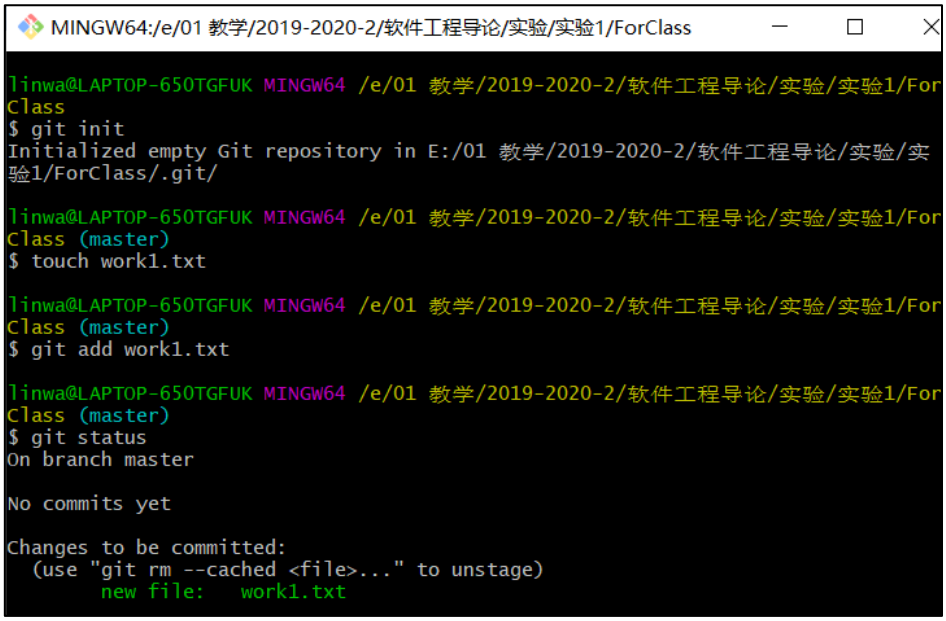
首先找到你项目的文件夹，比如项目名称为 ForClass，进入到这个文件夹，右键，选择 Git Bash Here，打开模拟 linux 风格的命令窗口。

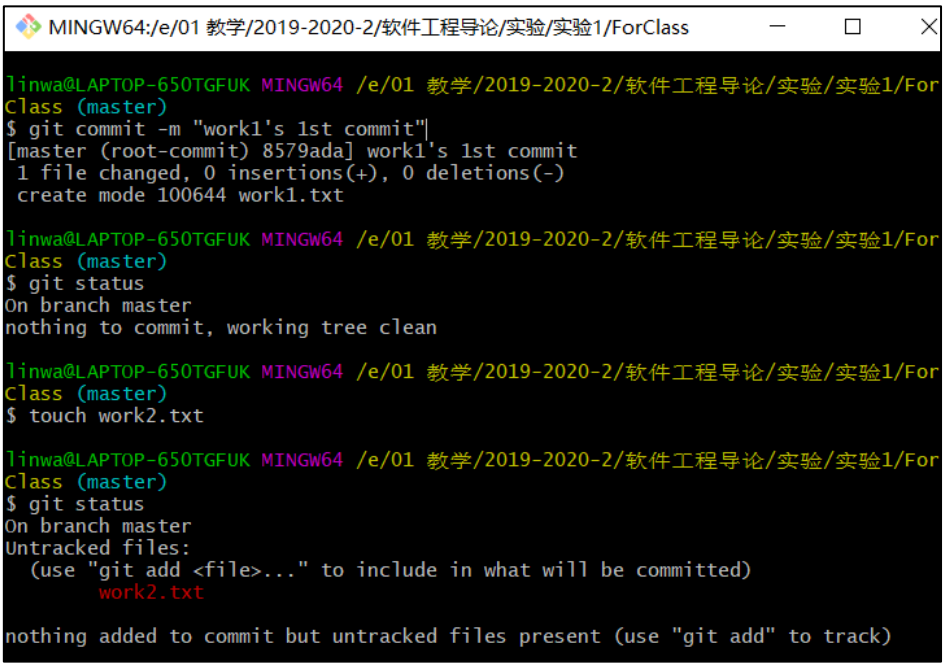
2. 执行命令

接下来依次执行命令：

- **git init** // 初始化版本库，创建空白的 git 项目，指定文件夹下面会出现一个“.git”的隐藏文件夹
- **touch 文件名** //在 git 项目的目录下创建一个指定文件名的文件，例如 touch test.txt
- **git add .** // 注意：后面有个点号，代表当前目录所有文件，添加文件到版本库（只是添加到暂存区），“.”代表添加文件夹下所有文件
- 或者 **git add 文件名** // 添加指定文件到版本库（只是添加到暂存区）
- **git status** //查看 git 仓库当前状态，可以随时使用。文件名为绿色表示文件在暂存区；如果是红色表示未提交到暂存区。
- **git commit -m "备注信息"** // 把添加的文件提交到版本库，并填写提交备注。例如：

效果如下：



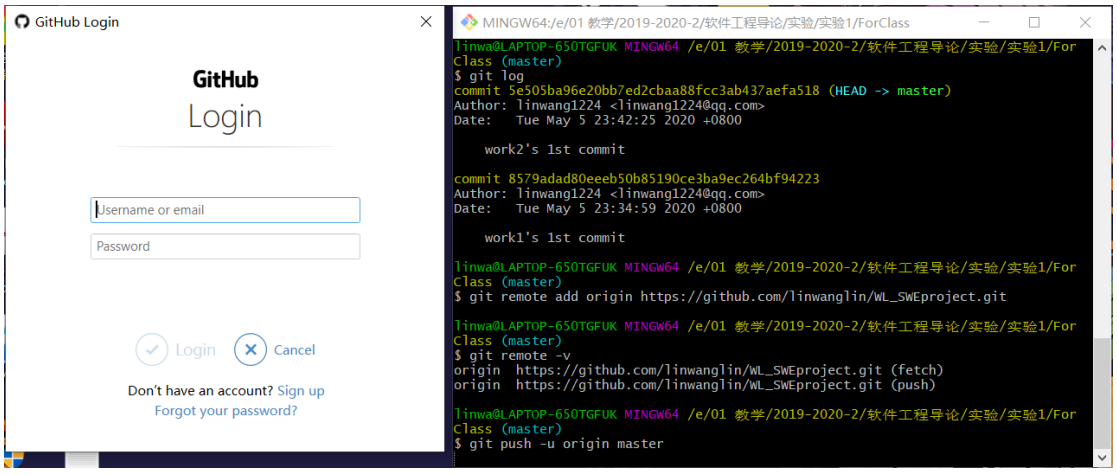


3. 提交到 github 远程代码服务器

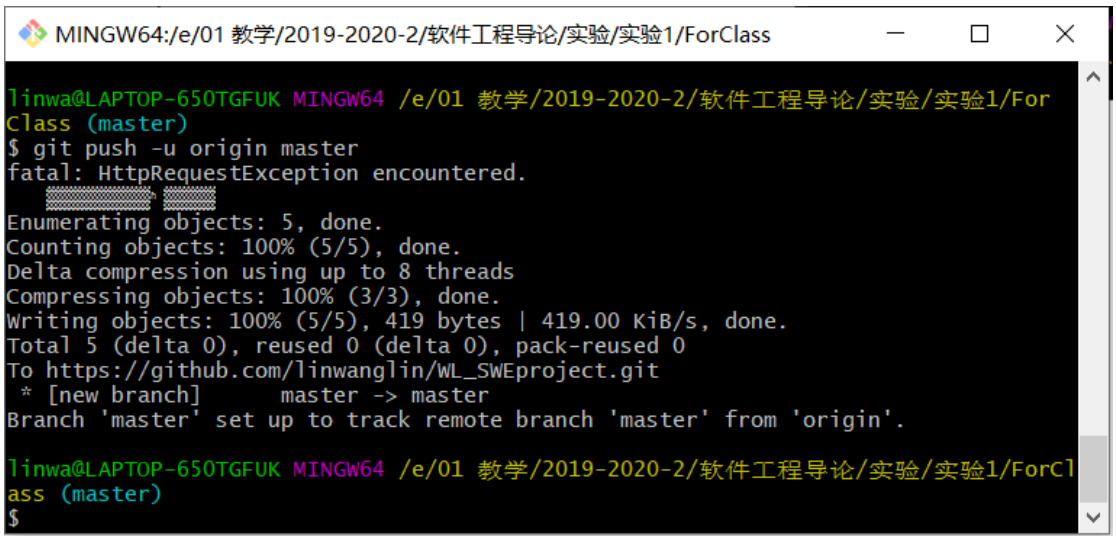
到目前为止，我们完成了代码库的初始化，但代码是在本地，还没有提交到远程服务器，所以关键的来了，要提交到就远程代码服务器，进行以下两步：

- **git remote add origin 你的远程库地址** // 把本地库与远程库关联，远程库地址即为 github 上新建库之后的地址，例如：https://github.com/linwanglin/WL_SWEproject.git
- **git remote -v** //查看远程仓库地址信息，可以用来验证上述操作是否关联成功
- **git push -u origin master** // 第一次推送时使用；推送时会要求你输入 github 的账户密码，正确输入即可（参见下图）
- **git push origin master** // 第一次推送后，直接使用该命令即可推送修改
- **git log** //查看提交记录历史

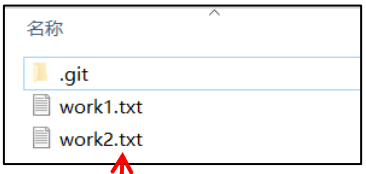
（1）第一次推送，要求输入 github 的账户和密码信息：



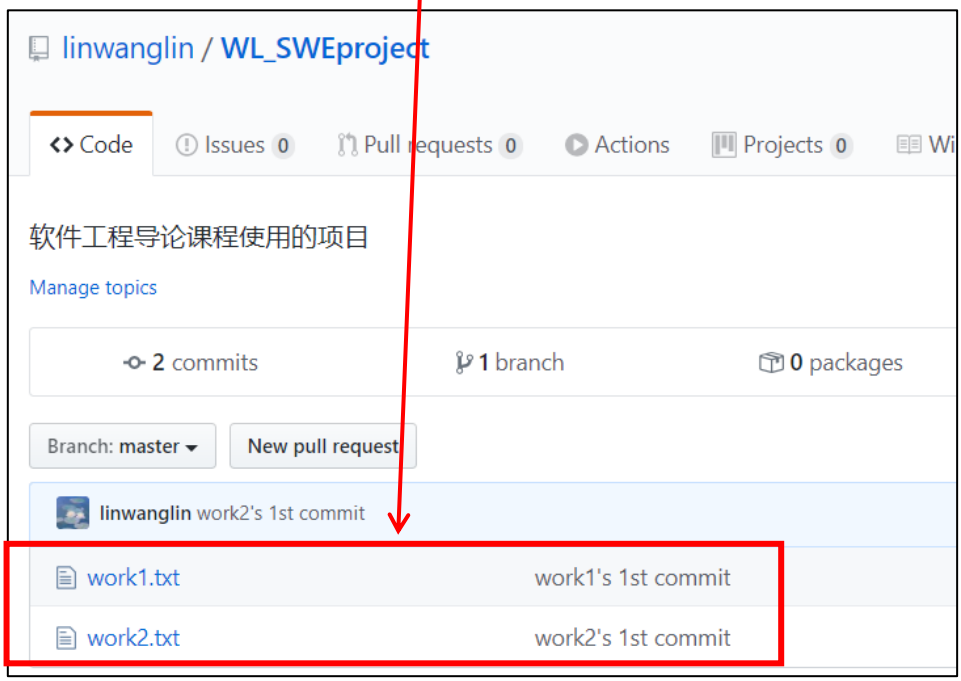
第一次推送成功：



此时查看本地文件夹：

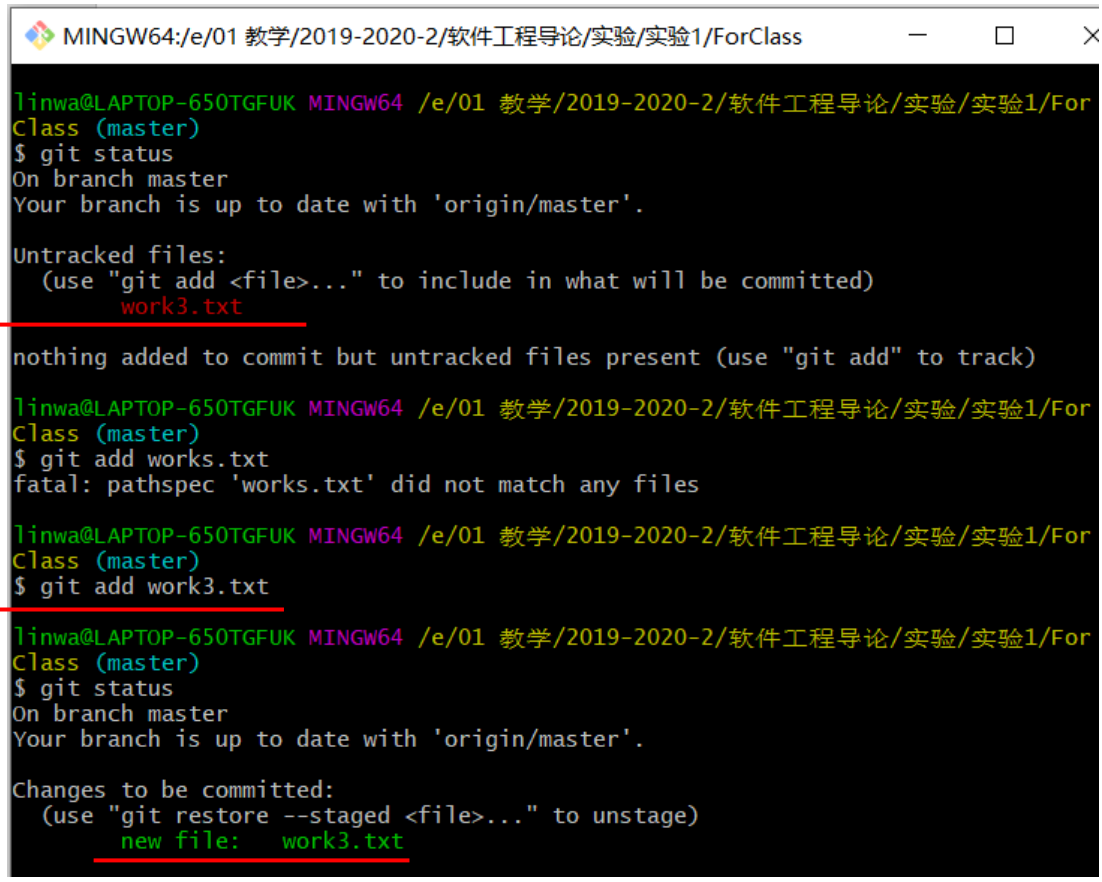


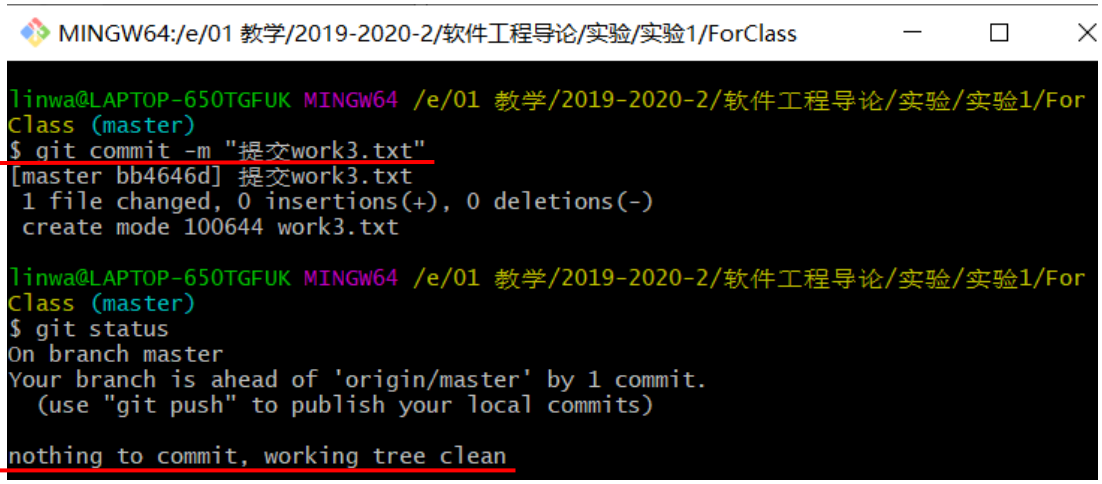
进入 github 仓库查看：

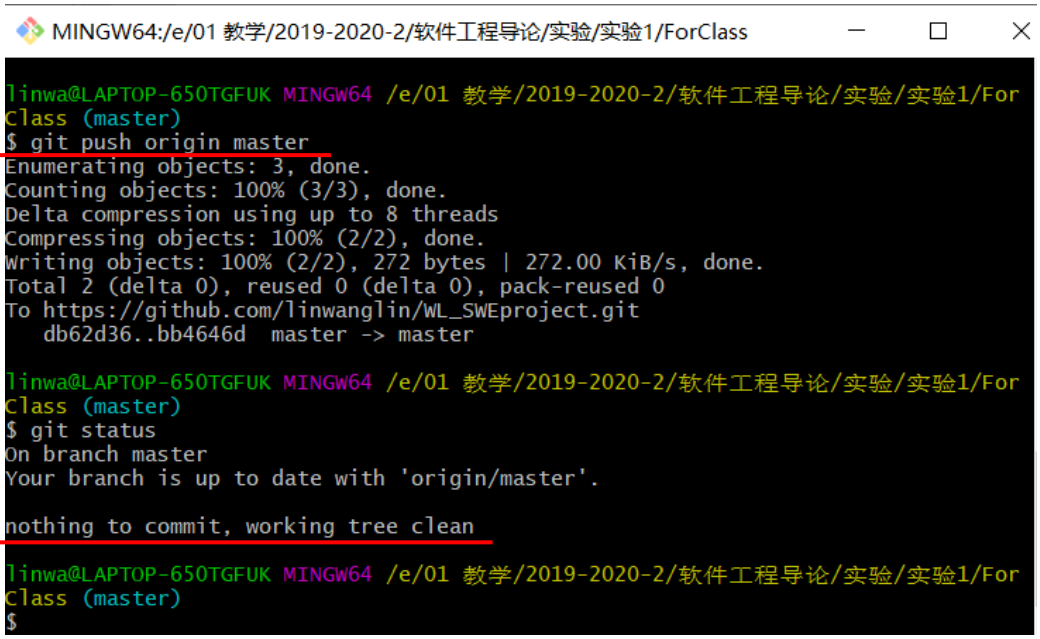


（2）之后的推送

在本地文件夹增加一个文件 work3.txt，然后推送到 github 上。此时已经不是第一次推送，不需要输入账号和密码信息。操作如下：


MINGW64:/e/01 教学/2019-2020-2/软件工程导论/实验/实验1/ForClass
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
Untracked files:
(use "git add <file>..." to include in what will be committed)
work3.txt
nothing added to commit but untracked files present (use "git add" to track)
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git add works.txt
fatal: pathspec 'works.txt' did not match any files
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git add work3.txt
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: work3.txt


MINGW64:/e/01 教学/2019-2020-2/软件工程导论/实验/实验1/ForClass
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git commit -m "提交work3.txt"
[master bb4646d] 提交work3.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 work3.txt
linwa@LAPTOP-65OTGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Class (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
nothing to commit, working tree clean



4. Git 的 tag 的用法

Git 中的 tag 指向一次 commit 的 id，用来给开发分支做一个标记，如标记一个版本号

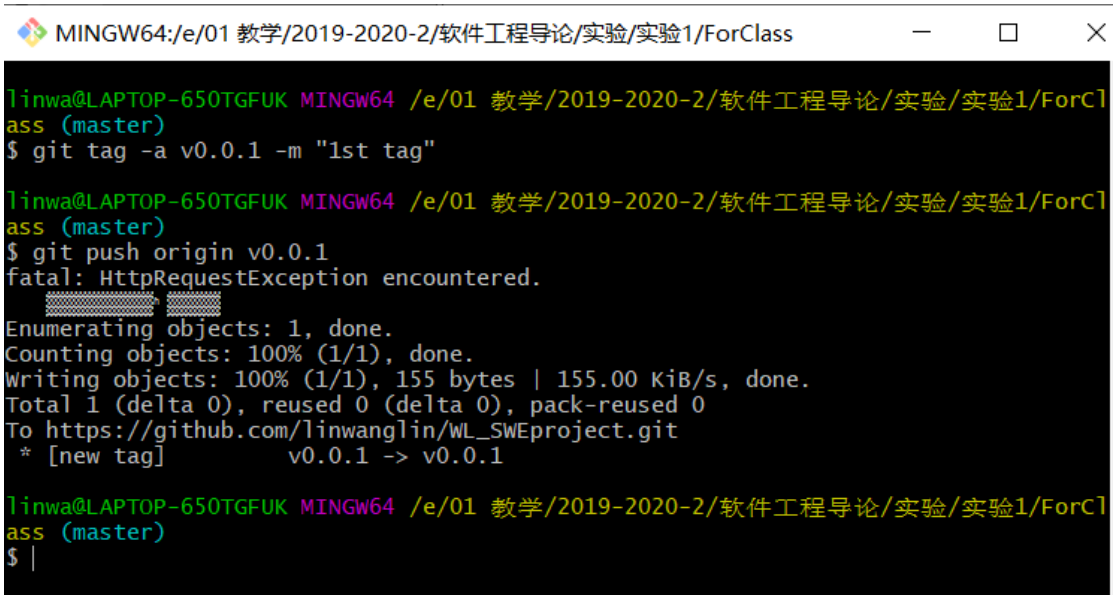
（1）添加标签

- **git tag -a v0.0.1 -m "first tag"** //标记一个版本号。

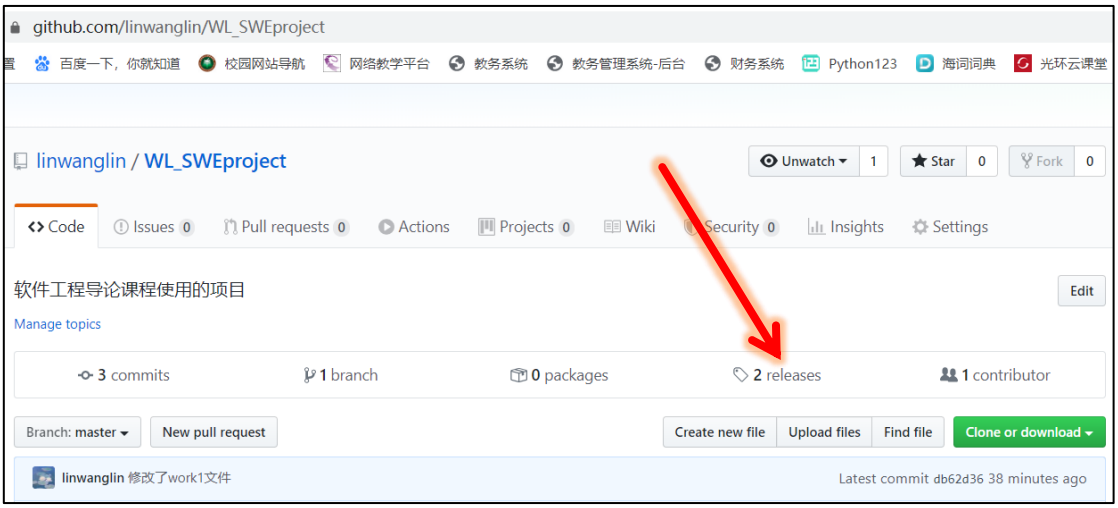
注解：**git tag** 是打标签的命令，**-a** 是添加标签，其后要跟新标签号（例如：v0.0.1），**-m** 及后面的字符串是对该标签的注释（例如：备注信息“first tag”）。

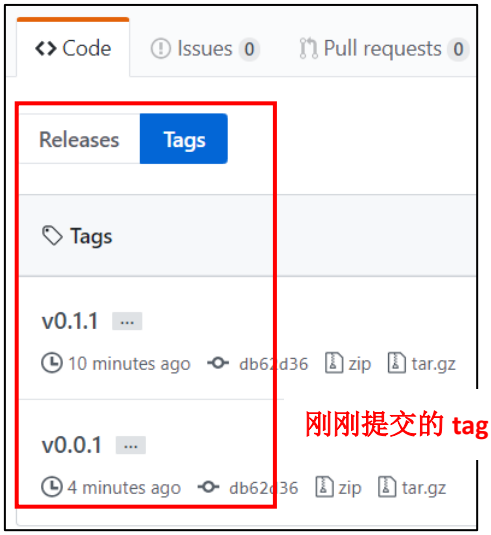
（2）推送指定标签到远程仓库

- **git push origin v0.0.1 //推送指定标签 v0.0.1 到远程仓库**



进入 github，查看 github 上的标签



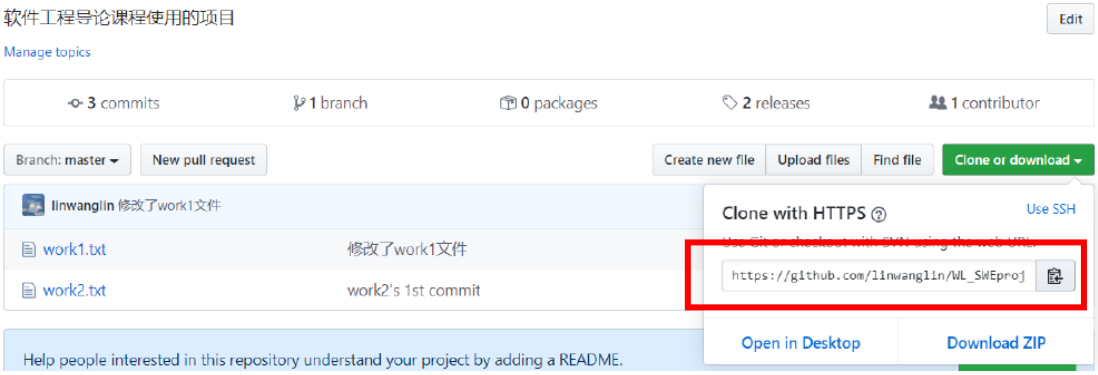


刚刚提交的 tag 信息

（四）通过 git 拉取 github 的文件，修改并推送至 github 仓库

1. 获取 github 上的项目

在 github 上找到要获取的项目的地址，复制 https 路径。



2. 拉取 github 上的整个仓库（本地没有对应仓库）

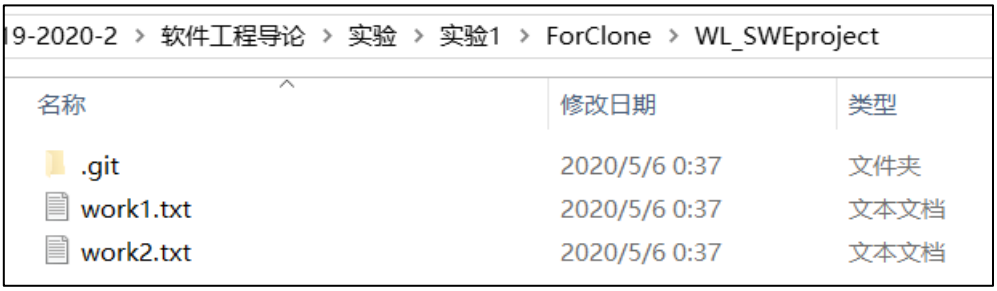
在本地新创建一个目录（ForClone），右键点击 Git Bash Here，打开命令行窗口。执行：

- **git clone 远程库上项目地址 //拉取 github 上指定仓库里的文件**

```
MINGW64:/e/01 教学/2019-2020-2/软件工程导论/实验/实验1/ForClone

linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Clone
$ git clone https://github.com/linwanglin/WL_SWEproject.git
Cloning into 'WL_SWEproject'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 1), reused 9 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.

linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1/For
Clone
$
```

3. 检查拉取情况

拉取项目成功后，进入本地电脑的文件夹（ForClone）下面查看：



4. 从 github 上拉取文件，更新本地文件夹（本地已有的仓库）

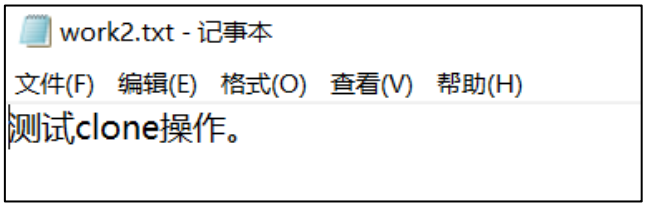
如果之前本地版本库已经和 github 绑定过了，本地已经有项目对应的文件夹了，但是在 github 上项目文件有变化。这时候需要重新从 github 上拉取文件，以便更新本地版本库，保持和 github 的统一。命令如下：

- **git pull --rebase origin master**

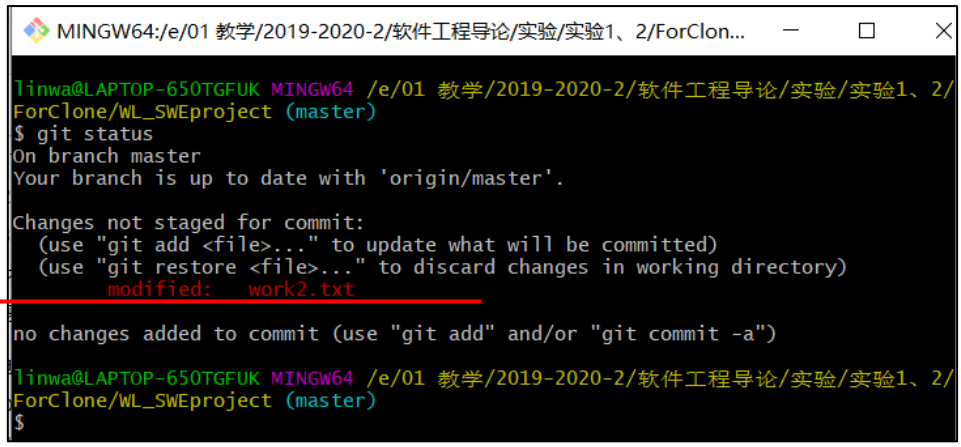
```
linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (master)
$ git pull --rebase origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 428 bytes | 38.00 KiB/s, done.
From https://github.com/linwanglin/WL_SWEproject
 * branch            master     -> FETCH_HEAD
   db62d36..8c8f893  master     -> origin/master
Successfully rebased and updated refs/heads/master.

linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (master)
$
```

5. 在本地文件夹中编辑文件

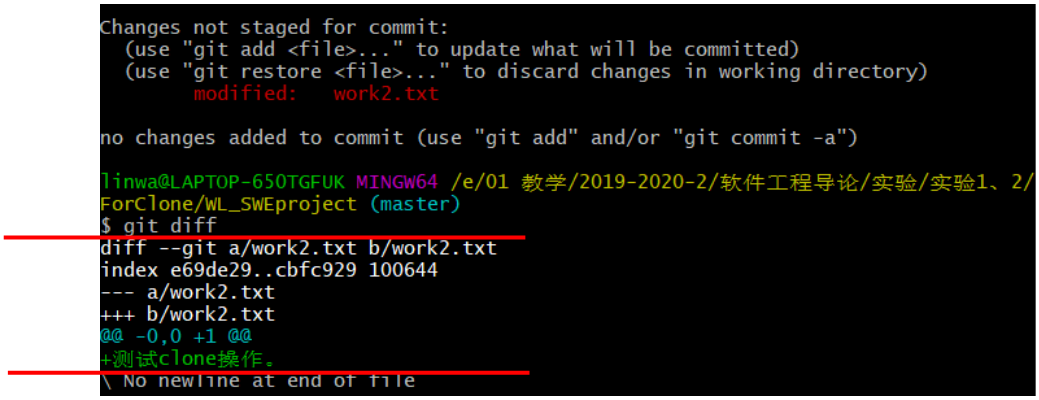
（1）编辑文件夹（ForClone）中的文件，保存。



（2）再打开命令工具，输入 git status 查看状态，会发现红色的就是改动过的文件。



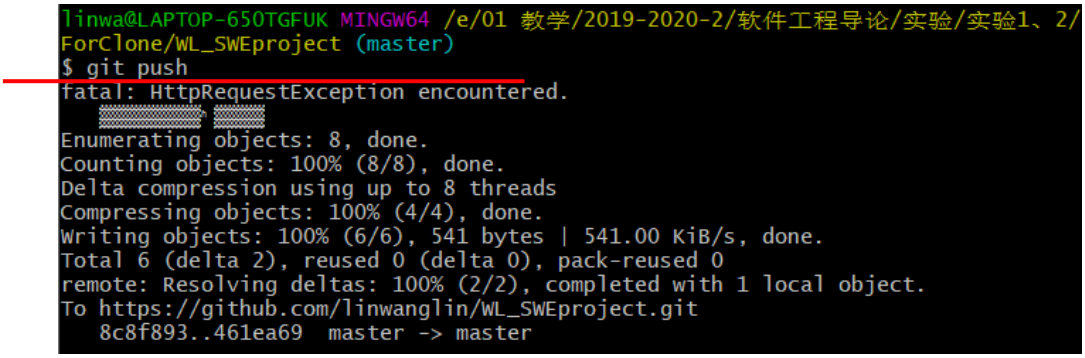
（3）输入 git diff 可以查看改动过的内容



（4）将修改后的内容推送到 github

最后输入以下命令，推送到 github 仓库

- git add . 或者 git add 文件名
- git commit -m “备注信息”
- git push 或者 git push origin master

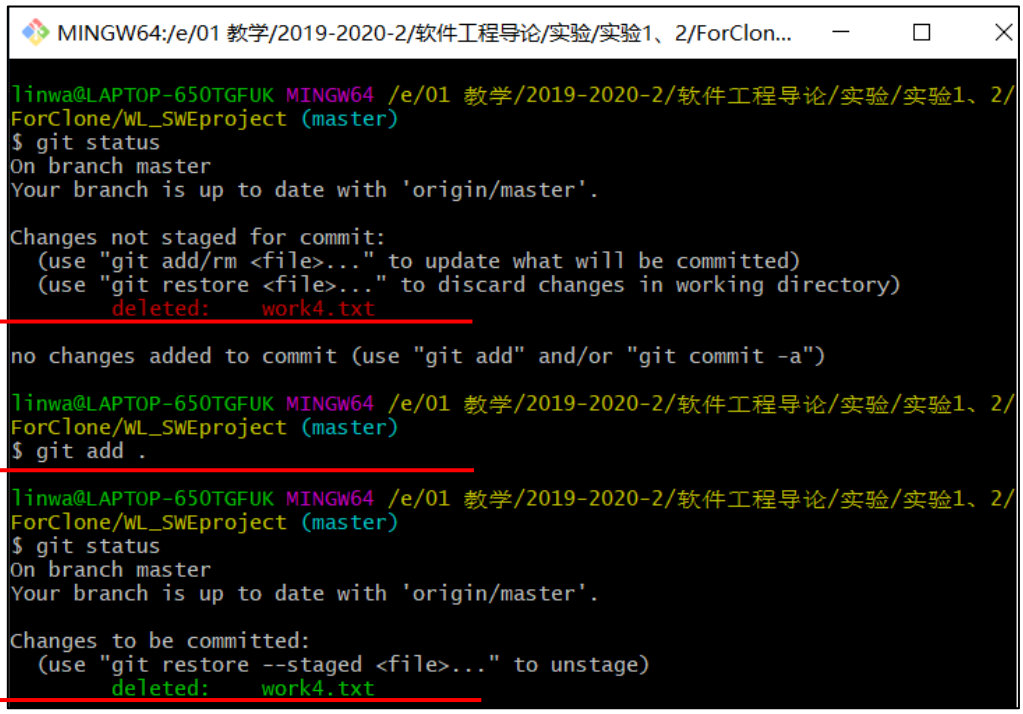


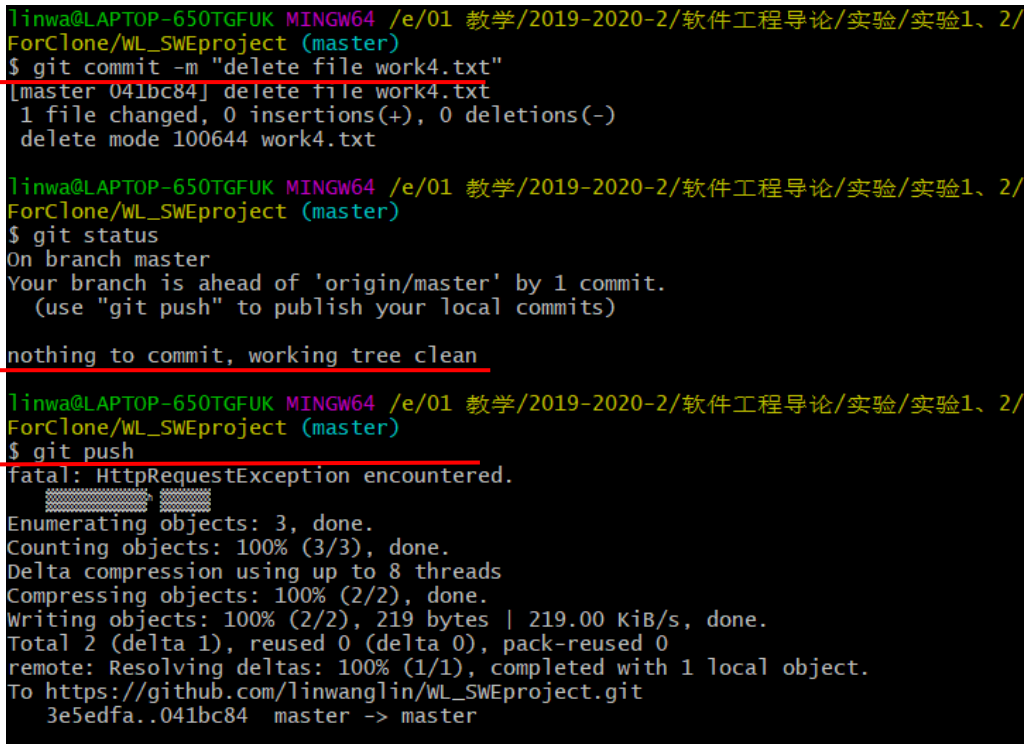
5. 删除文件

删除文件也一样。删除文件后（例如删除文件 work4.txt），执行下面的命令：

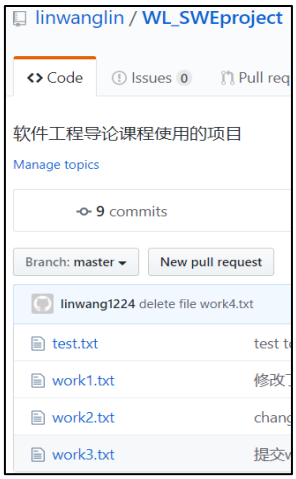
- **git add . 或者 git add 文件名**
- **git commit -m “备注信息”**
- **git push origin master //推送文件到 master 分支**
  或者 git push //推送文件到当前分支

效果如下：





进入到 github 对应的仓库，会发现 github 中该文件也被删除了。



（五）基于 master 主分支上新建 develop 分支

1. 基本概念：

首先我们先了解以下分支的概念：

名称解释

- **Github**，是代码托管平台
- **Git**，是版本控制工具
- **master分支**，即主分支。任何项目都必须有个这个分支。对项目进行tag或发布版本等操作，都必须在该分支上进行。
- **develop分支**，即开发分支，从master分支上检出。团队成员一般不会直接更改该分支，而是分别从该分支检出自己的feature分支，开发完成后将feature分支上的改动merge回develop分支。同时release分支由此分支检出。
- **release分支**，即发布分支（测试分支），从develop分支上检出。该分支用作发版前的测试，可进行简单的bug修复。如果bug修复比较复杂，可merge回develop分支后由其他分支进行bug修复。此分支测试完成后，需要同时merge到master和develop分支上。
- **feature分支**，即功能分支，从develop分支上检出。团队成员中每个人都维护一个自己的feature分支，并进行开发工作，开发完成后将此分支merge回develop分支。此分支一般用来开发新功能或进行项目维护等。
- **fix分支**，即补丁分支，由develop分支检出，用作bug修复，bug修复完成需merge回develop分支，并将其删除。所以该分支属于临时性分支。
- **hotfix分支**，即热补丁分支。和fix分支的区别在于，该分支由master分支检出，进行线上版本的bug修复，修复完成后merge回master分支，并merge到develop分支上，merge完成后也可以将其删除，也属于临时性分支。

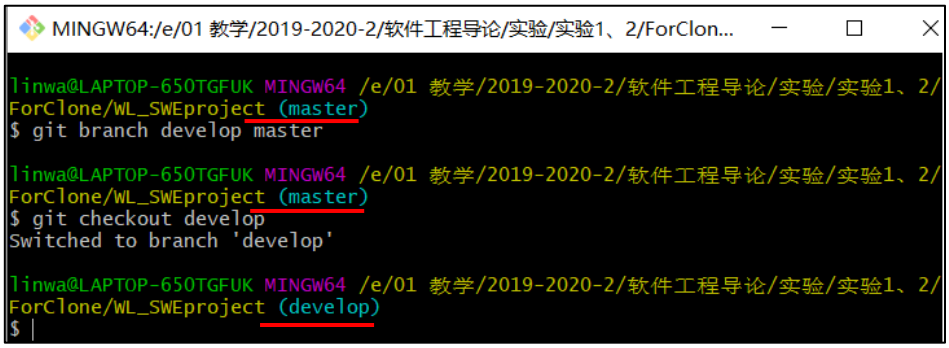
2. 新建 develop 分支（方法 1）

此时本地仓库是在 **master 分支**上，从 master 分支上新建 **develop 分支**

- **git branch develop master　//在 master 分支上新建 develop 分支**

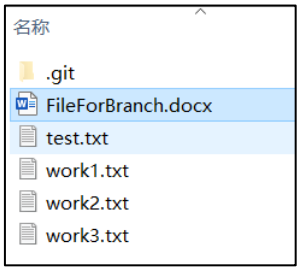
3. 切换到 develop 分支上

- **git checkout develop　//切换到 develop 分支上**



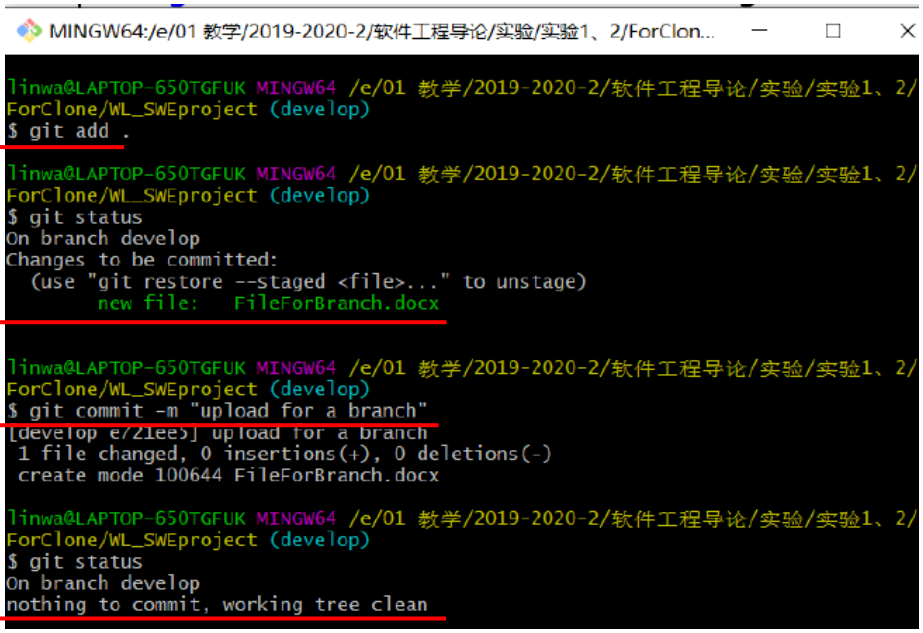
4. 推送文件到分支

我们本地文件夹中新建一个文件 FileForBranch.docx。



执行以下命令，将文件推送到 github 的 origin/develop 分支：

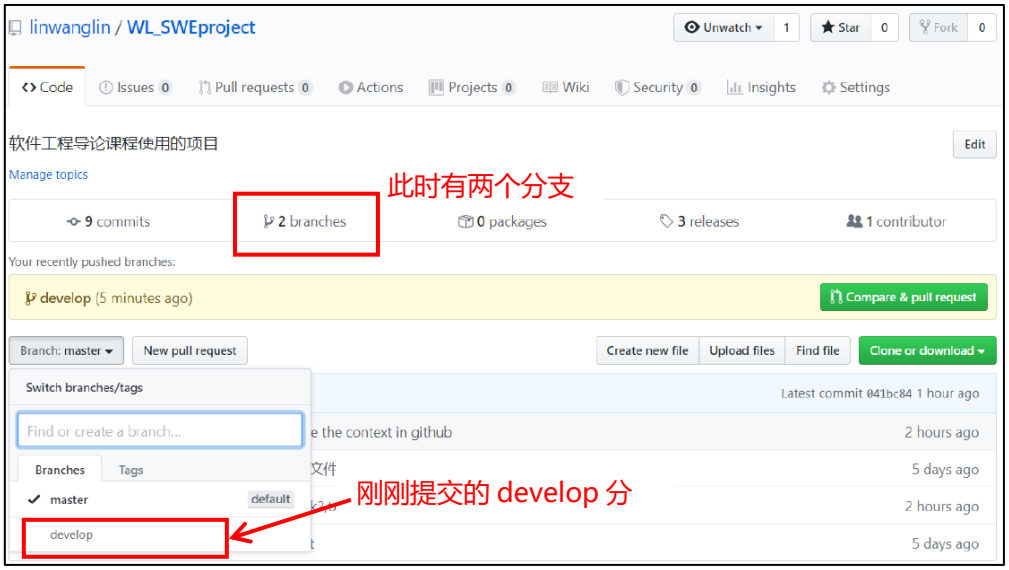
- **git add . 或者 git add 文件名**
- **git commit -m “备注信息” //例如：git commit -m “other push”**
- **git push origin develop**

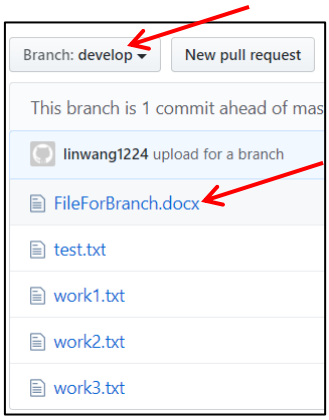


```
linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (develop)
$ git push origin develop
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 269 bytes | 269.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/linwanglin/WL_SWEproject/pull/new/develop
remote:
To https://github.com/linwanglin/WL_SWEproject.git
 * [new branch]      develop -> develop
```

5. 检查推送是否成功

在到 github 上看是否推送成功，可以看到 github 的 develop 分支上有我们新增的文件 FileForBranch.docx。



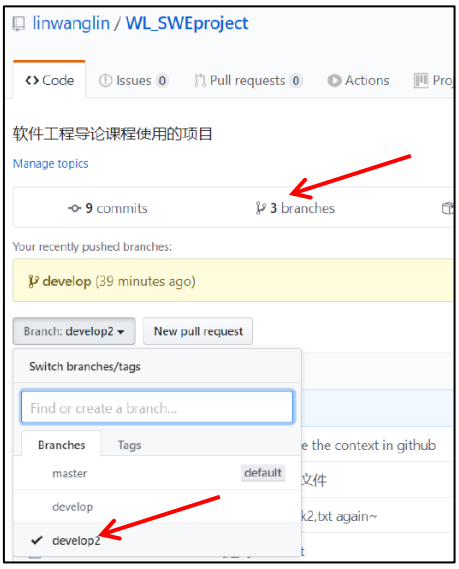


6. 新建 develop 分支（方法 2）

- **git checkout -b xxxx //在当前分支上创建一个分支 xxxx， 并且切换到 xxxx 分支，例如：git checkout -b develop2**
- **git add . 或者 git add 文件名**
- **git commit -m “备注信息” //例如：git commit -m “other push”**
- **git push origin develop2**

```
linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (master)
$ git checkout -b develop2
Switched to a new branch 'develop2'

linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (develop2)
$ git add .

linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (develop2)
$ git commit -m "第二个分支develop2"
On branch develop2
nothing to commit, working tree clean
```

```
linwa@LAPTOP-650TGFUK MINGW64 /e/01 教学/2019-2020-2/软件工程导论/实验/实验1、2/
ForClone/WL_SWEproject (develop2)
$ git push origin develop2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop2' on GitHub by visiting:
remote:      https://github.com/linwanglin/WL_SWEproject/pull/new/develop2
remote:
To https://github.com/linwanglin/WL_SWEproject.git
 * [new branch]      develop2 -> develop2
```



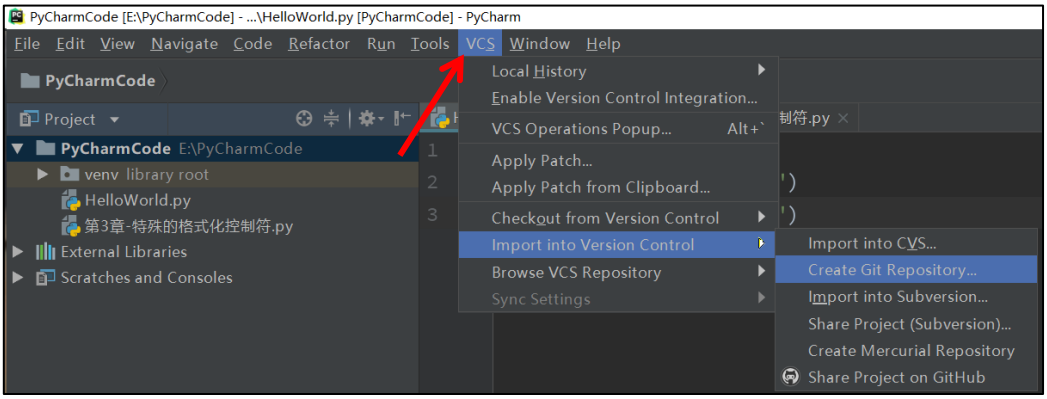
（六）在 Pycharm 等开发工具中使用 git 的常见的应用场景

下面来看以上各场景在 Pycharm 中对应的操作。

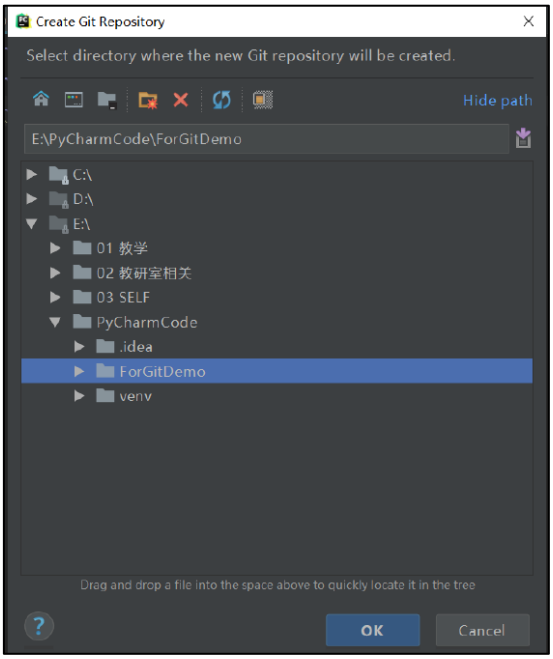
此处：同一个用户在本地使用两个文件夹，分别表示两个用户小张和小袁。

场景一：小张创建项目并提交到远程 Git 仓库

1) 创建好项目，在主菜单上选择 VCS → Import into Version Control → Create Git Repository

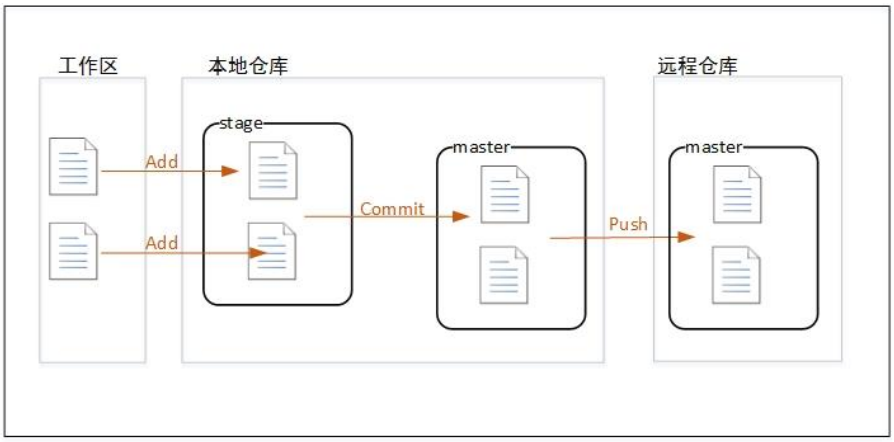


2) **指定本地仓库的位置**，按个人习惯指定即可，例如这里选择了项目源代码同目录。



点击 OK 后创建完成本地仓库，注意，这里仅仅是本地的。

3) 下面把项目源码添加到本地仓库。



上图是 Git 与提交有关的三个命令对应的操作（如下图所示）：

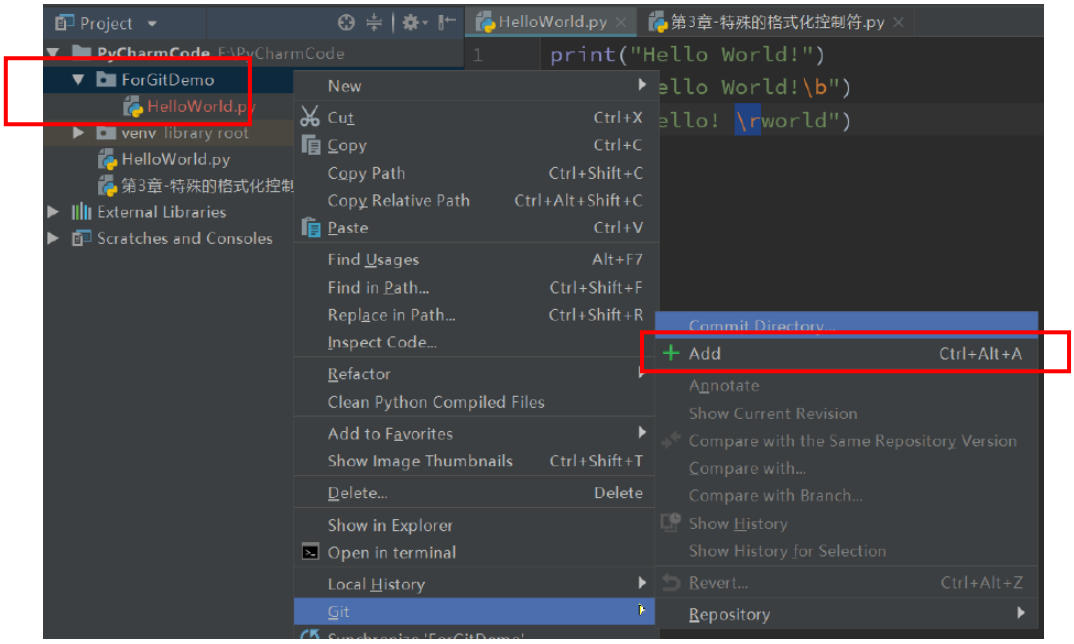
- **Add 命令**是把文件从 Pycharm 的工作目录添加到本地仓库的 **stage 区（暂存区）**，
- **Commit 命令**把 stage 区的暂存文件提交到**当前分支的仓库（本地版本库）**，并清空 stage 区（暂存区）。
- **Push 命令**把本地仓库的提交同步到**远程仓库**。

Pycharm 中对操作做了一定的简化，**Commit 和 Push 可以在一步中完成**。

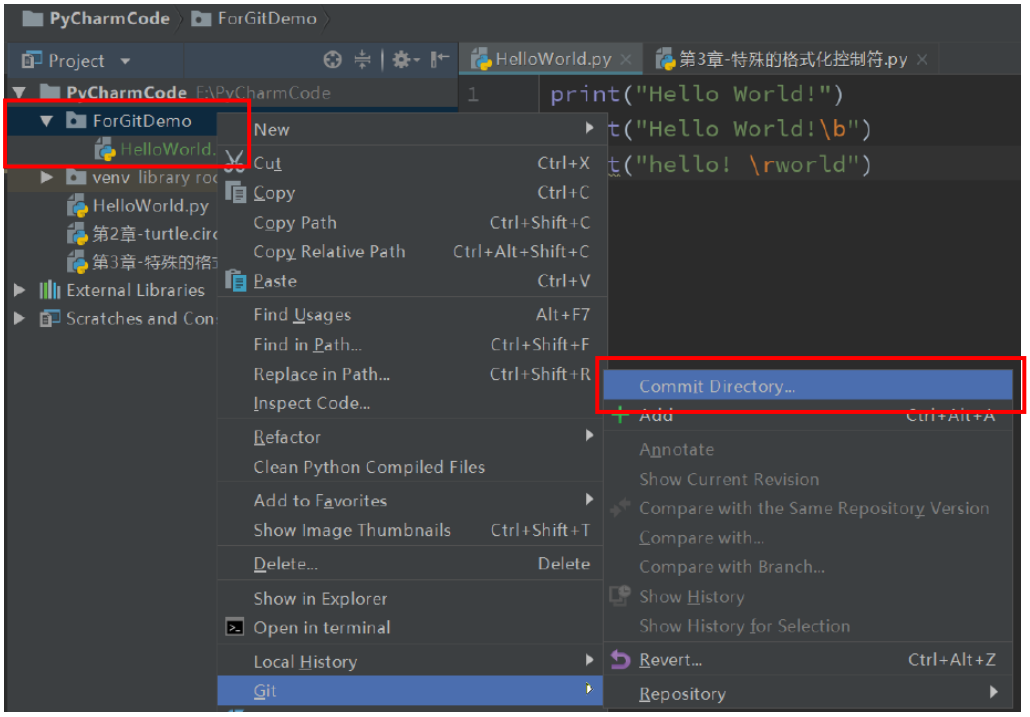
具体操作：

1) 首先在本地项目目录里面添加一个文件，例如：hello.py。此时在 pycharm 中，新添加的文件名为**红色**，参见下图。

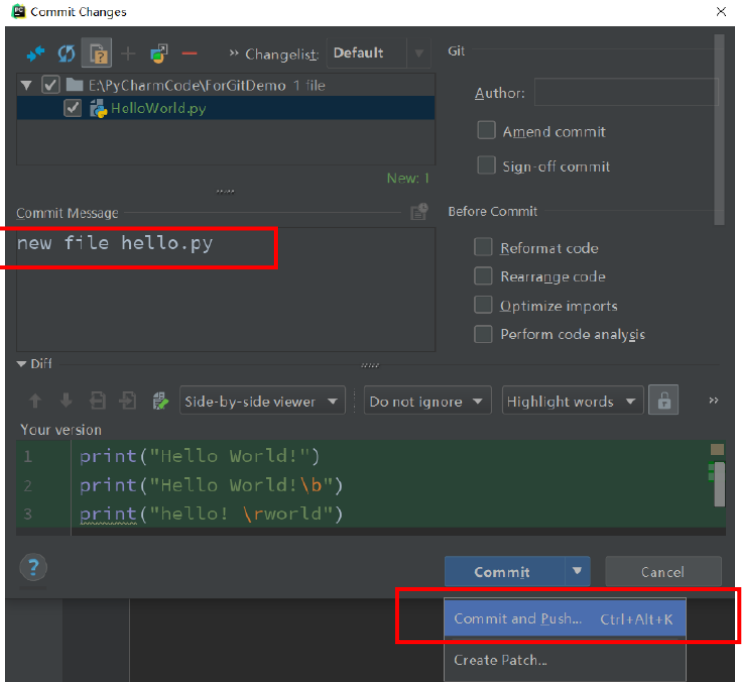
2) 然后在 pycharm 的项目上点击右键，选择“Git→Add”。**将本地文件目录添加到 git 的 stage 取（暂存区）**。操作之后，文件名变为**绿色**。



3) 然后在项目上点击右键，选择“Git→Commit Directory”操作。目的是**将文件推送到 git 的本地版本库中**。

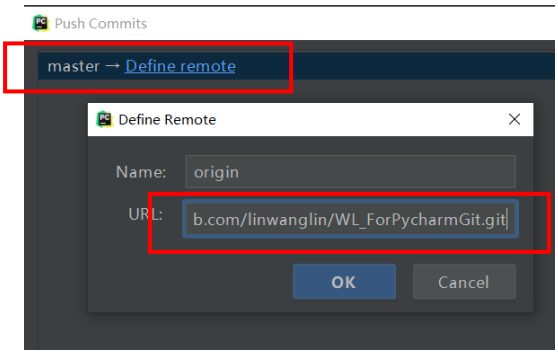


4) 在弹出的【**Commit Changes**】对话框中填写备注信息（Commit Message），并点击右下角的“**Commit and Push**”。

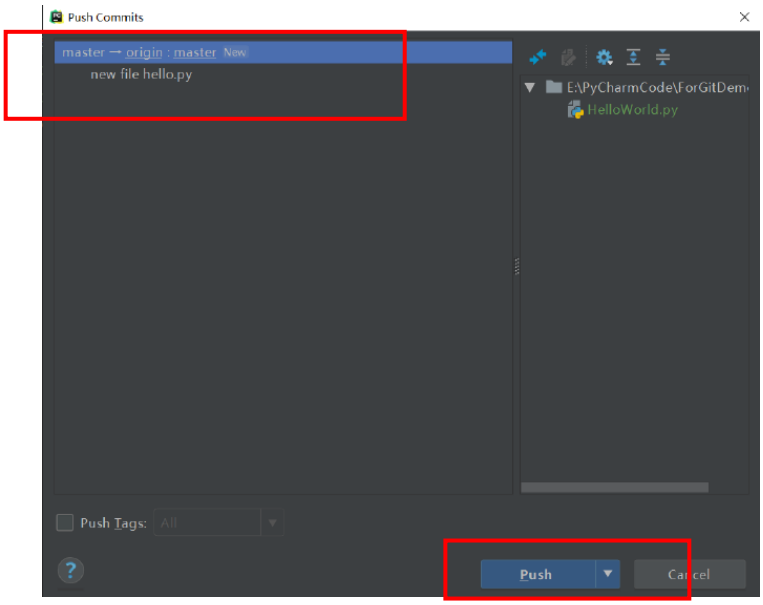


5) 因为是第一次提交，Push 前需要指定远程仓库的地址。

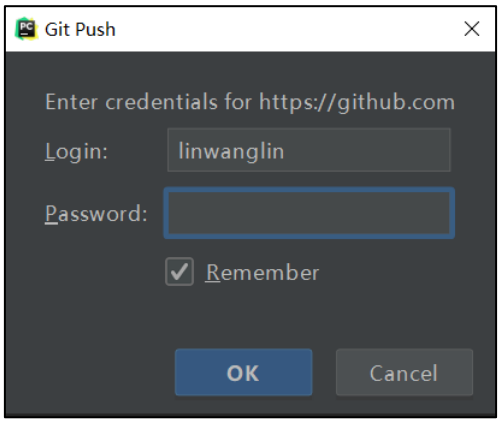
如下图，在【Push Commit】对话框点击左上角的“Define remote”，在弹出的窗口中输入远程仓库地址（github 中对应的仓库）。



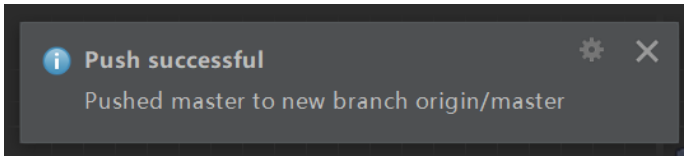
6) 设置完成后，【Push Commit】对话框显示上传文件的备注信息。点击“Push”按钮。



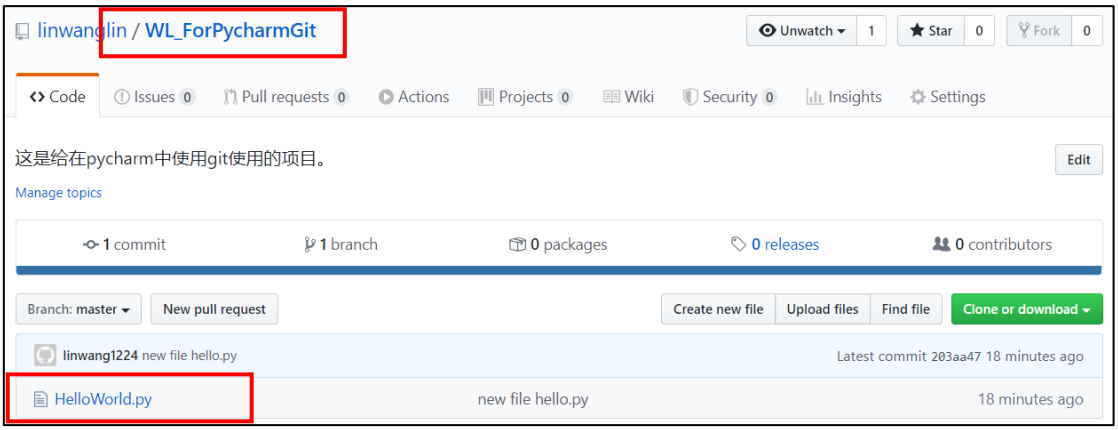
第一次推送的时候，会弹出对话框，让你输入 github 的账号和密码。



7) 推送成功后，系统给出相应提示（有可能不出现，与 pycharm 软件的个人设置有关）。

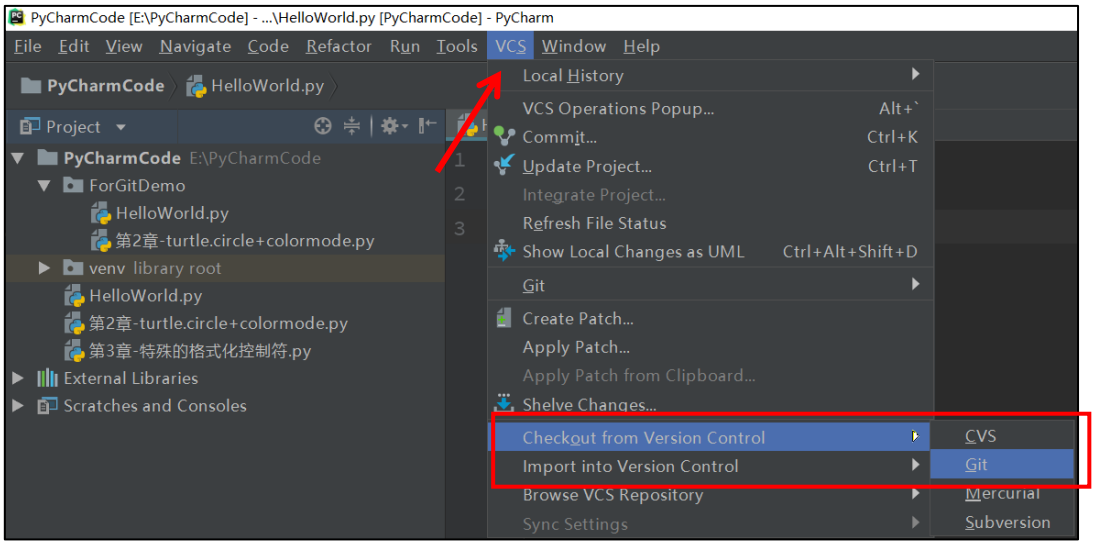


8) 进入到 github 对应仓库，可以查看到推送成功的文件。

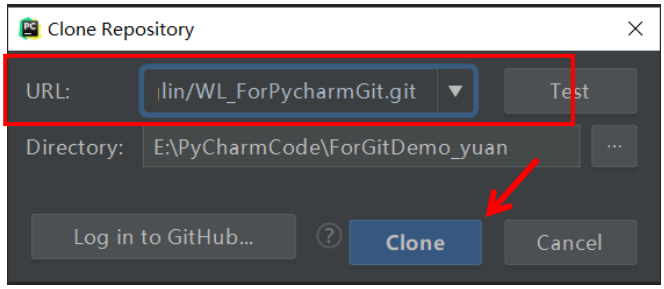


场景二：小袁从远程 Git 仓库上获取项目源码

即**克隆项目**，操作如下：



输入小张 Push 时填写的远程仓库地址，接下来按向导进行操作，即可把项目从远程仓库克隆到本地仓库和本地工作区。



场景三：小袁修改了部分源码，提交到远程仓库

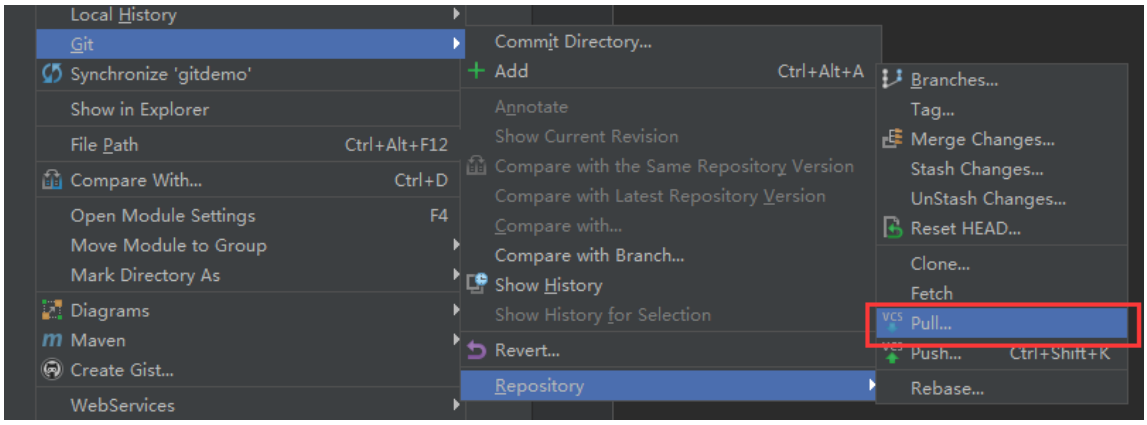
这个操作和首次提交的流程基本一致，分别是 Add -> Commit -> Push。请参考场景一。

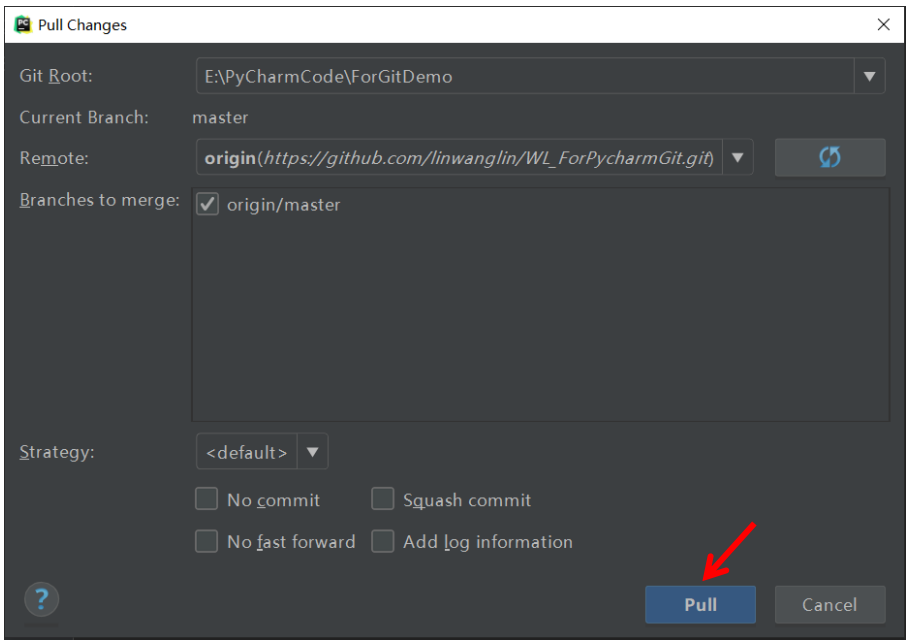
场景四：小张从远程仓库获取小袁的提交

获取更新有两个命令：Fetch 和 Pull。

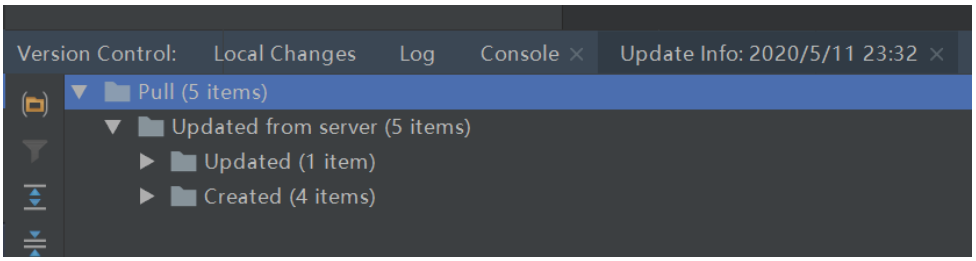
Fetch 是从远程仓库下载文件到本地的 origin/master，然后可以**手动对比修改**决定是否合并到本地的 master 库。

Pull 则是**直接下载并合并**。如果各成员在工作中都执行修改前先更新的规范，则可以直接使用 Pull 方式以简化操作。





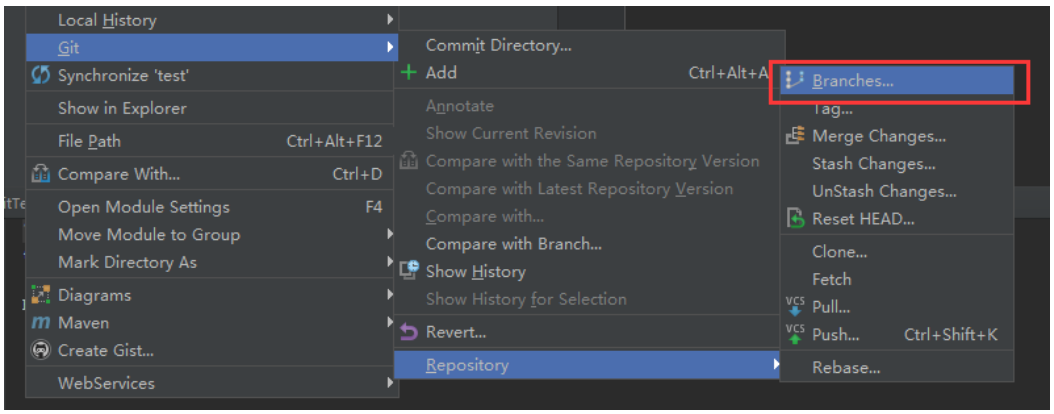
拉取成功后，系统会显示拉取的信息：



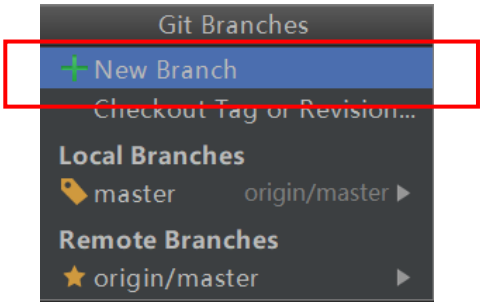
场景五：小袁接受了一个新功能的任务，创建了一个分支并在分支上开发

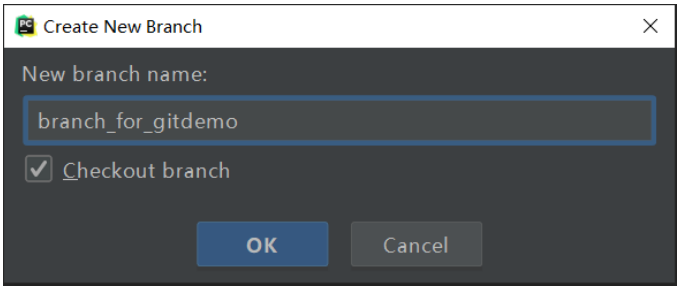
创建分支也是一个常用的操作，例如临时修改 bug、开发不确定是否加入的功能等，都可以创建一个分支，再等待合适的时机合并到主干。创建流程如下：

1. 在项目上，选择“Git→Repository→Branches”

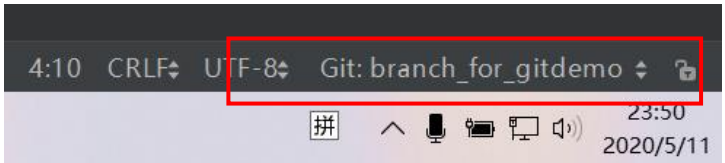


2. 选择“New Branch”，并输入一个分支的名称

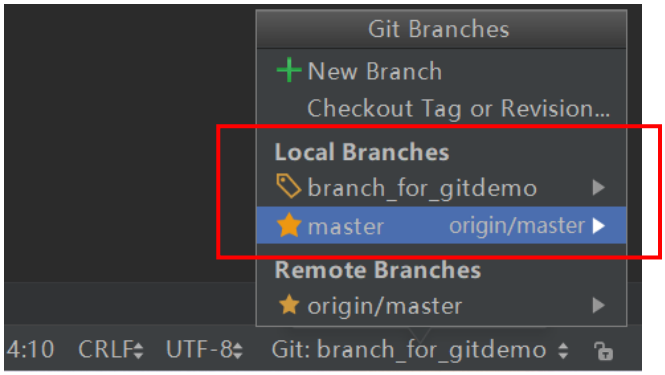




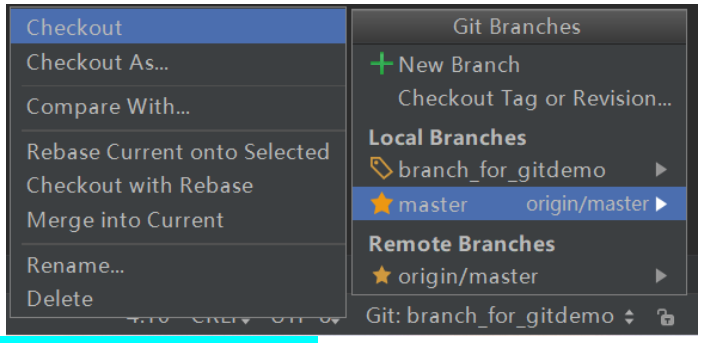
创建完成后注意 pycharm 界面的右下角，如下图，Git: branch_for_gitdemo 表示已经自动切换到 branch_for_gitdemo 分支，当前工作在这个分支上。



点击分支名称后，弹出一个小窗口，在 Local Branches 中有其他可用的本地分支选项，点击后选择 Checkout 即可切换当前工作的分支(见场景 7 操作切换其他分支)。



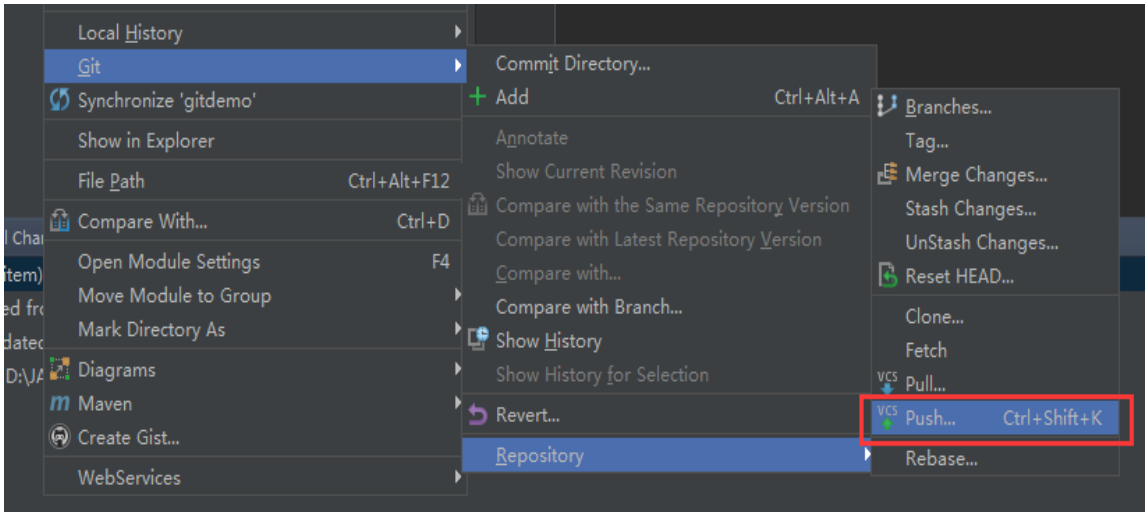
如下图，点击 Checkout，可以在不同分支之间切换。

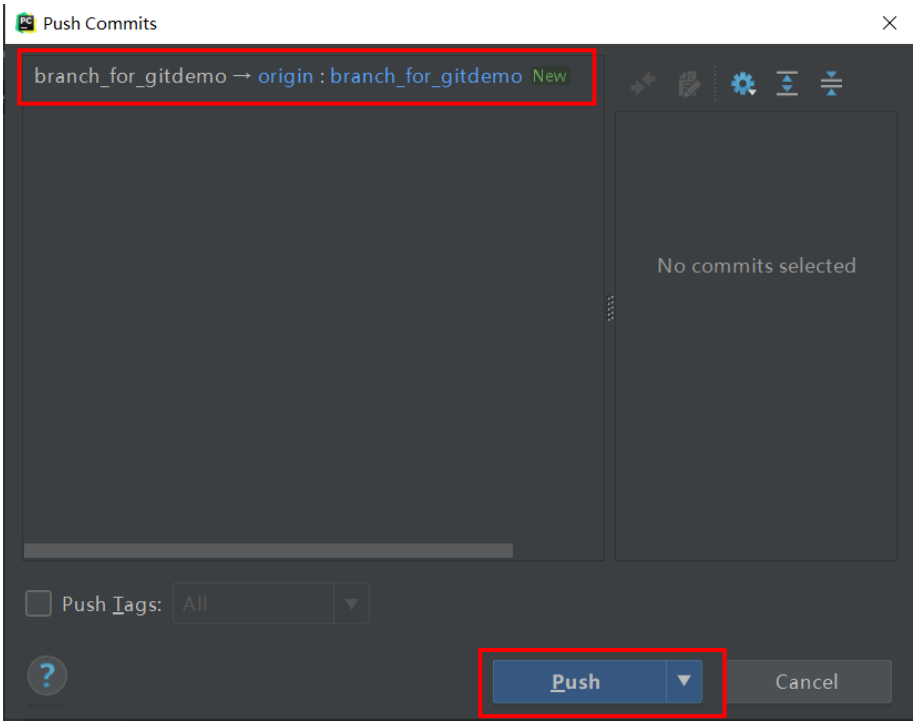


注意，这里**创建的分支仅仅在本地仓库**，如果想让组长小张获取到这个分支，还需要**提交到远程仓库**。

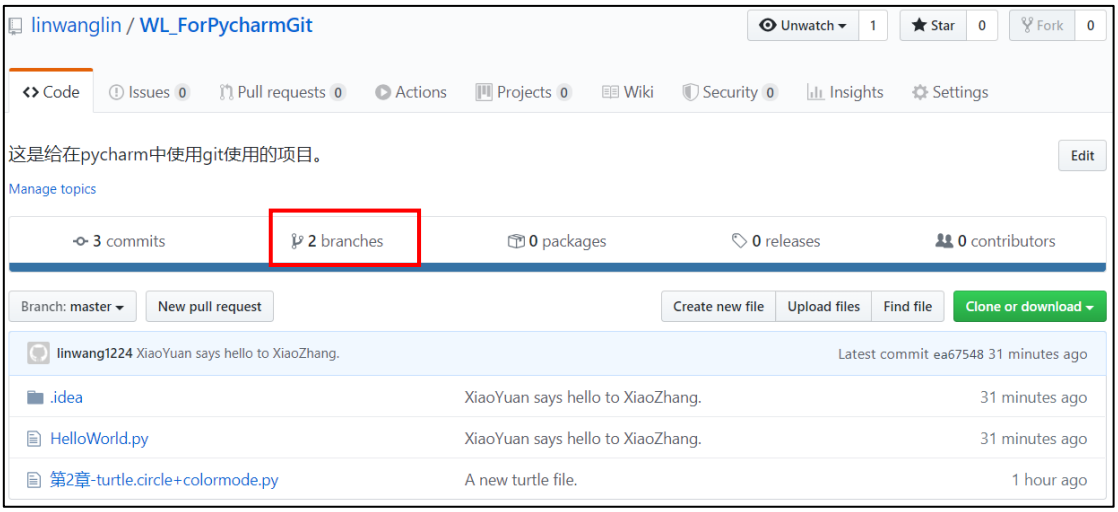
场景六：小袁把分支提交到远程 Git 仓库

切换到新建的分支（checkout 到 XXX 分支），使用 Push 功能。



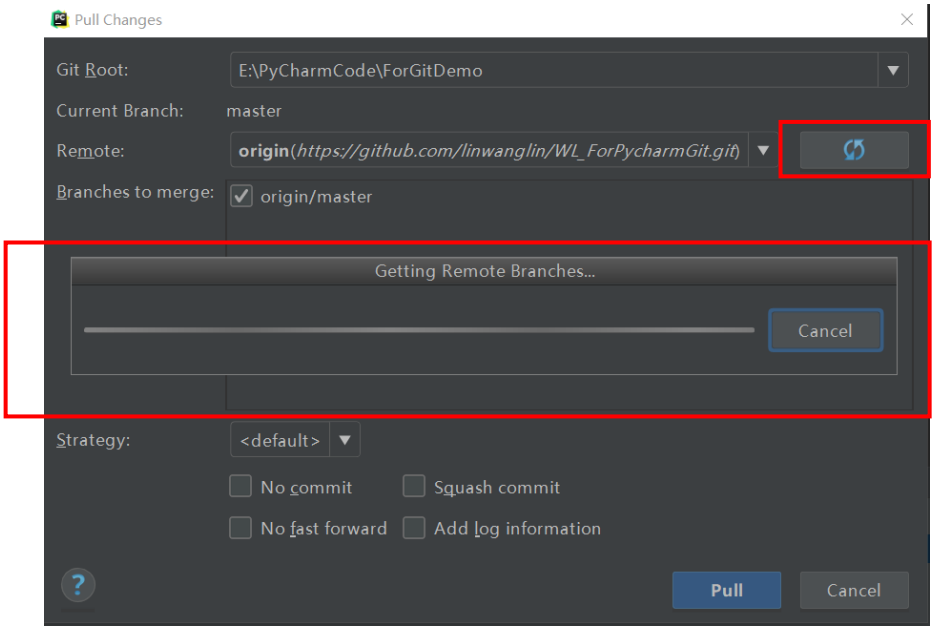


进入 github 查看，可以看到项目的分支数变为了 2。

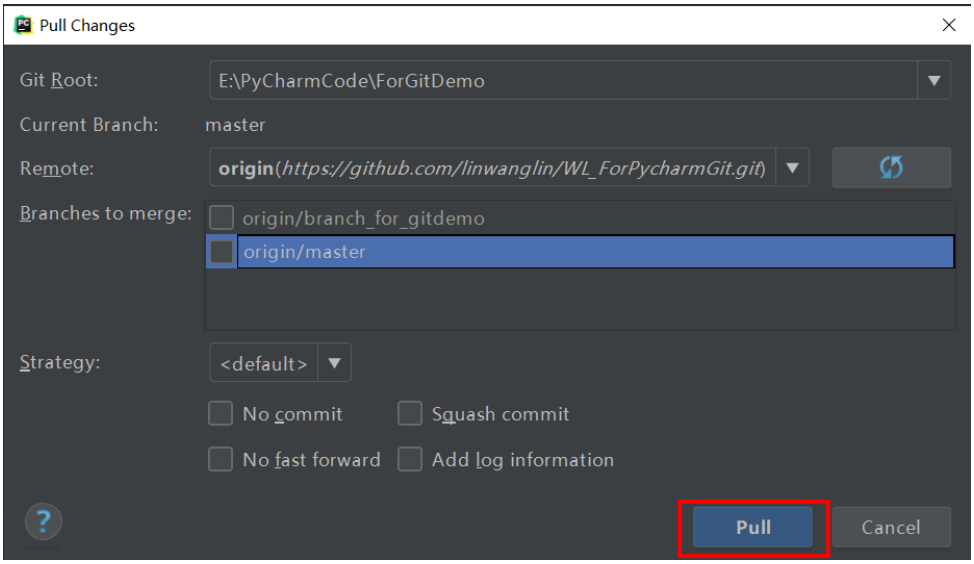


场景七：小张获取小袁提交的分支

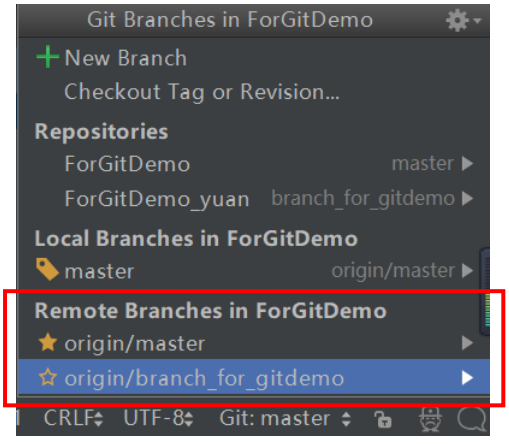
使用 Pull 功能打开更新窗口（参见场景四），点击 Remote 栏后面的刷新按钮，会在 Branches to merge 栏中刷新出新的分支。



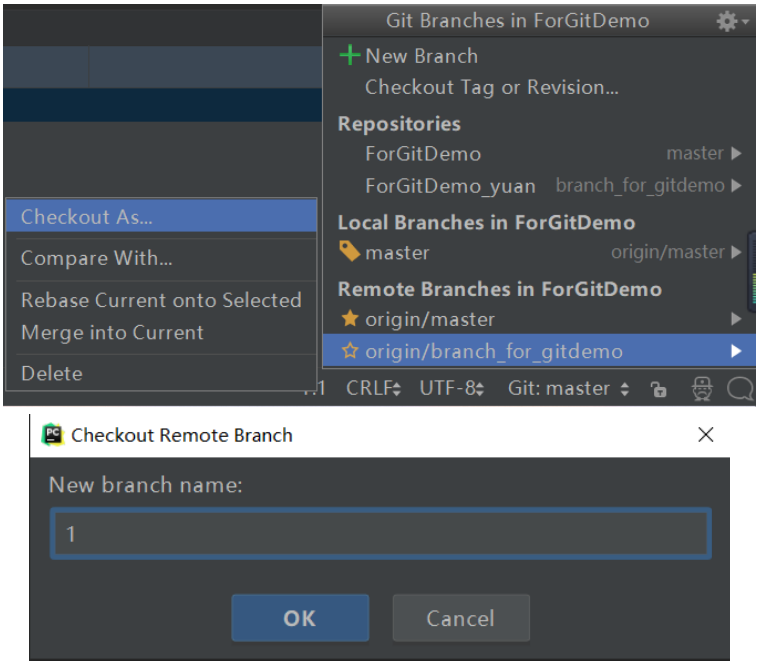
这里并不想做合并，所以不要选中任何分支，直接点击 Pull 按钮完成操作。



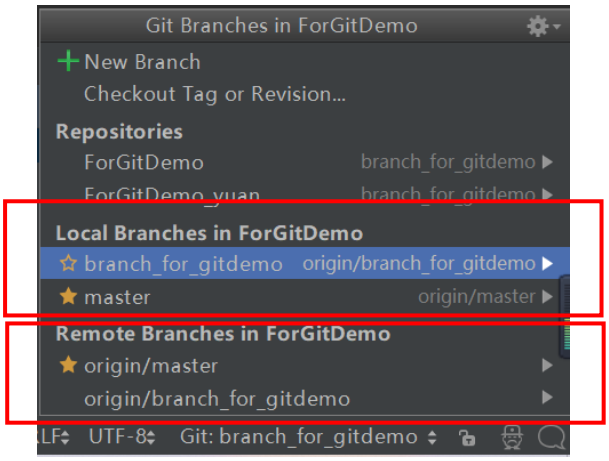
更新后，再点击右下角，可以看到在 Remote Branches 区已经有了新的分支。



选定分支，在弹出的子菜单中选择“Checkout as ...”，输入分支名称，在本地仓库中创建该分支。



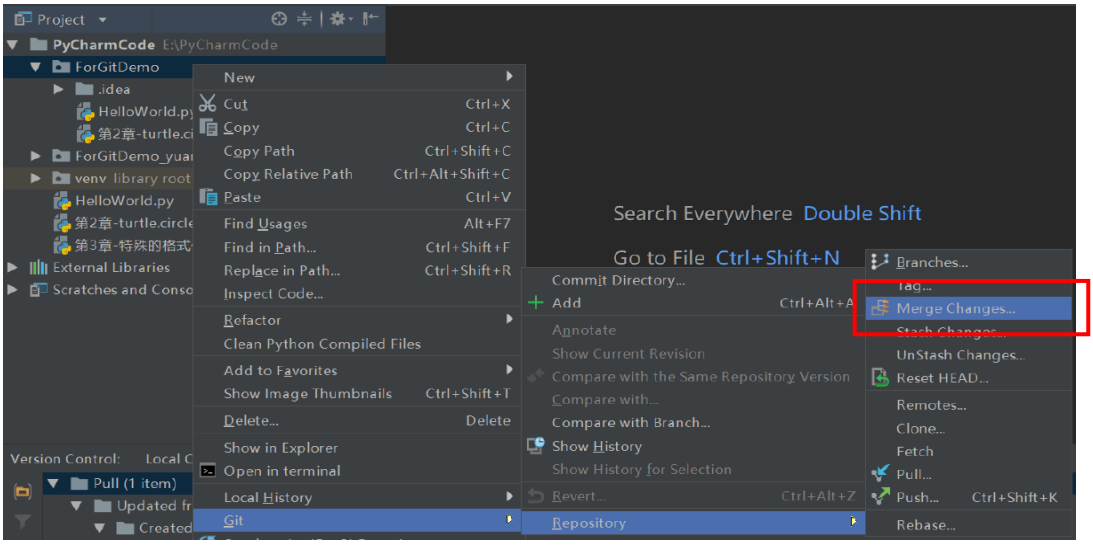
完成后在 Local Branches 区也会出现该分支的选项，点击后选择 Checkout 切换。



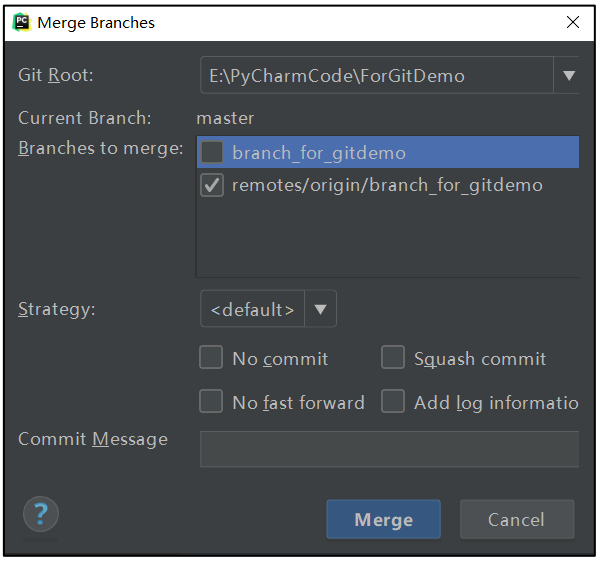
场景八：小张把分支合并到主干

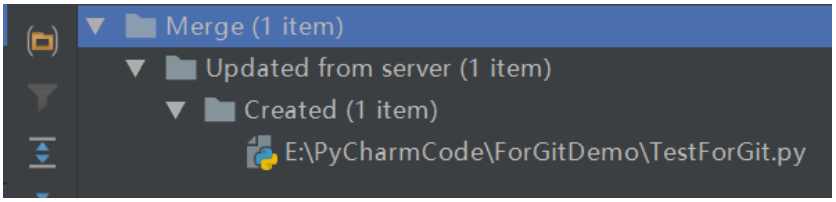
新功能开发完成，体验很好，项目组决定把该功能合并到主干上。

切换到 master 分支，选择 Merge Changes。



选择要合并的分支，点击 Merge 完成。





此时，只是将分支合并到本地版本库，还需要推送到 github 上（add→commit→push）