# 实验一：Git和Github的使用

## 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 开发工具中使用git和github

## 实验要求：

1. 掌握github的使用
2. 注册github个人账户
3. 在github个人账户中创建仓库
4. 掌握git的安装
5. 在git官网上下载相应的安装包
6. 通过安装包进行安装，并检查是否安装成功
7. 环境配置，设置用户名，邮箱。
8. Git创建本地化仓库，并推送到github
9. 初始化本地仓库
10. 添加文件到版本库
11. 把添加的文件提交到版本库，并填写提交备注
12. 把本地库与远程库关联
13. 推送本地文件到远程库
14. （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 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](http://lib.csdn.net/base/git" \o "Git知识库" \t "https://www.cnblogs.com/javabg/p/_blank)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

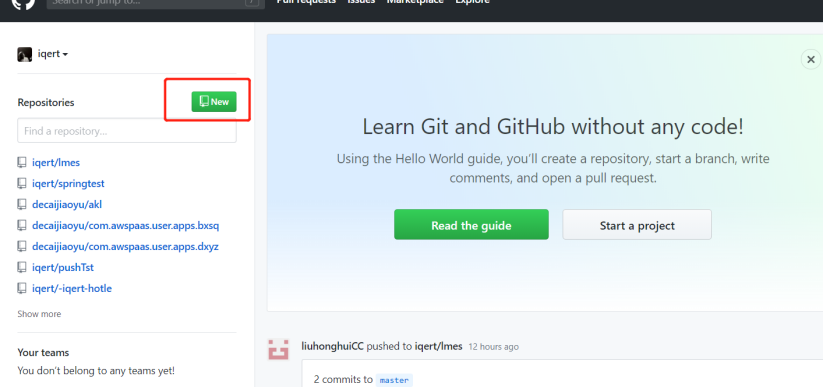
场景八：小张把分支合并到主干

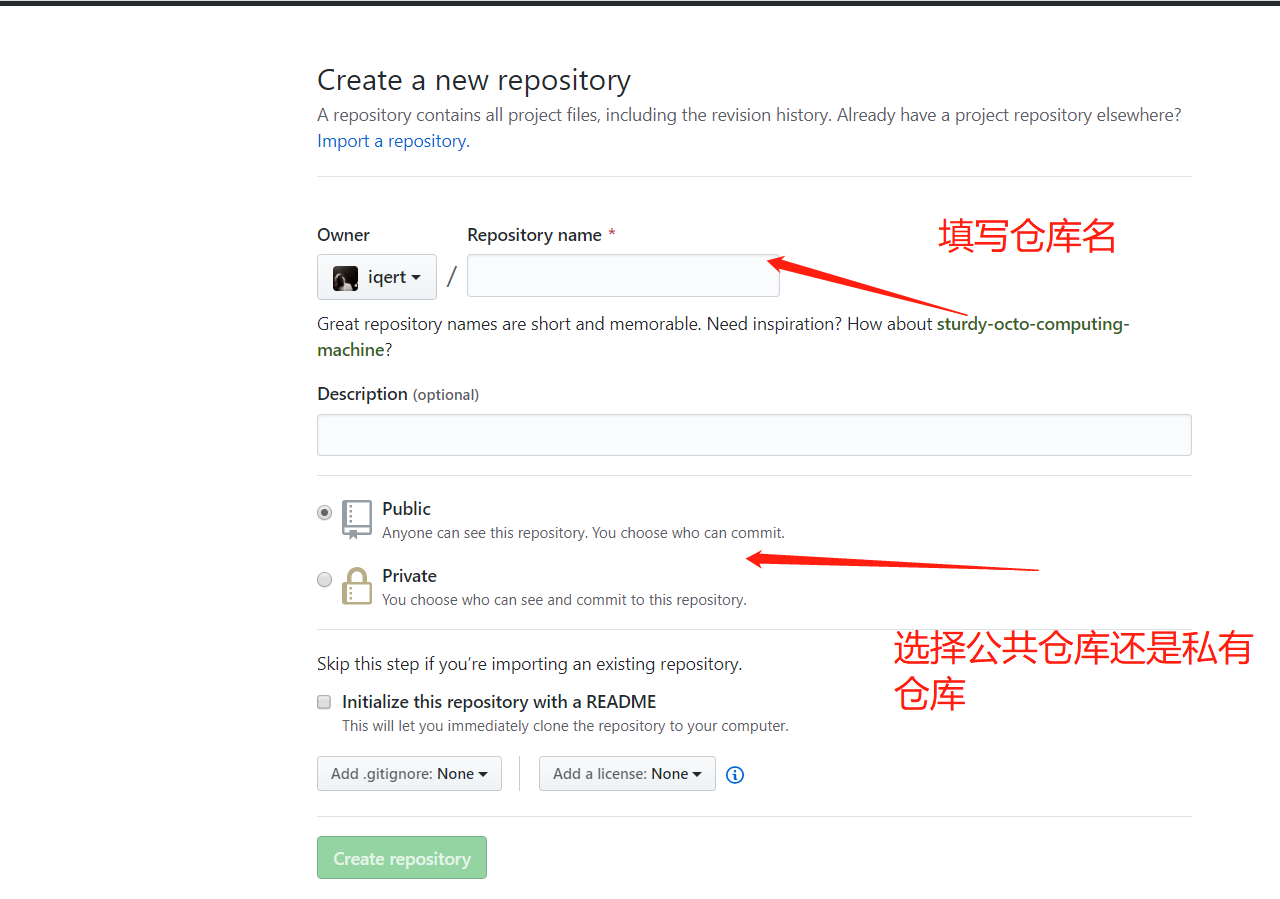
## 实验过程及结果

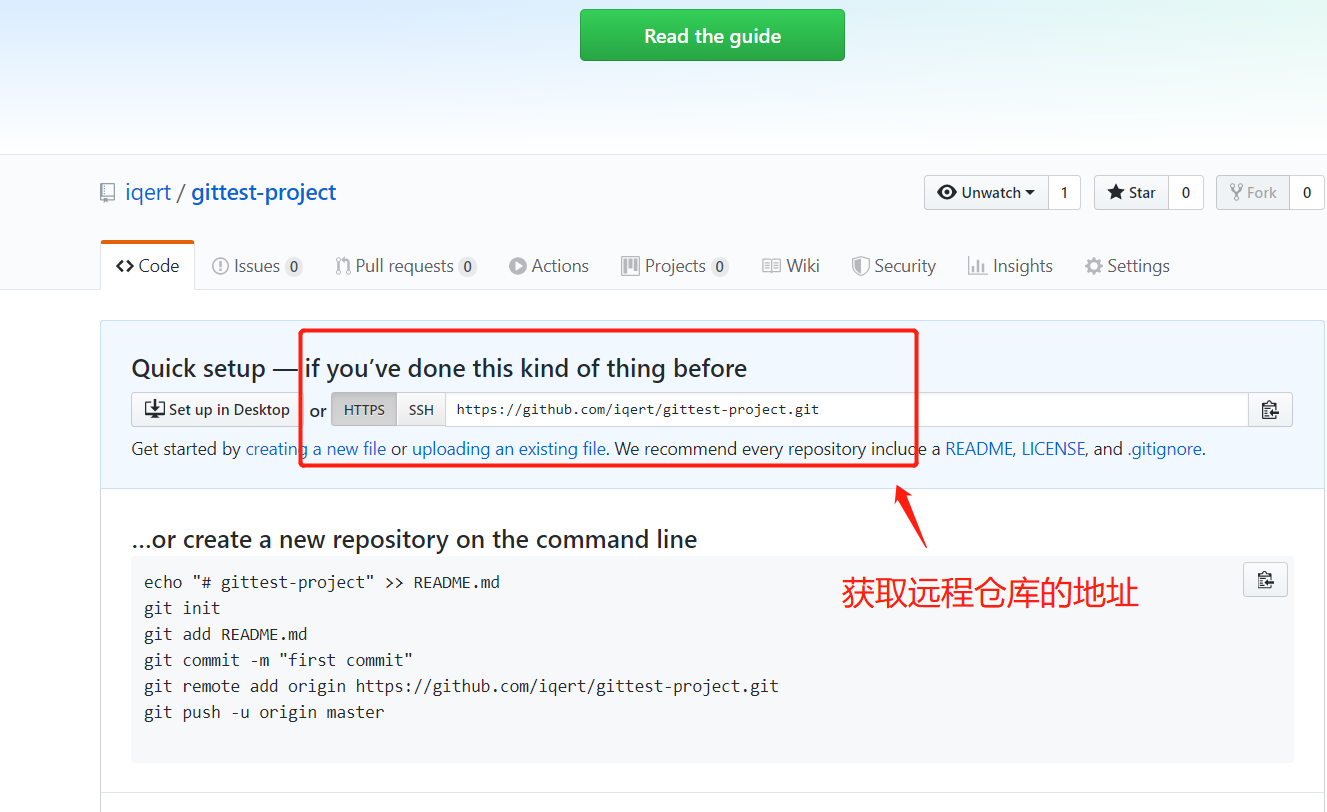
### 掌握github的使用

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

2.在github个人账户中创建仓库



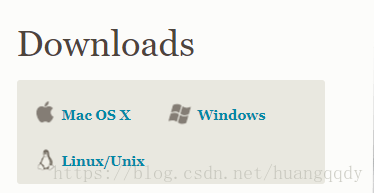


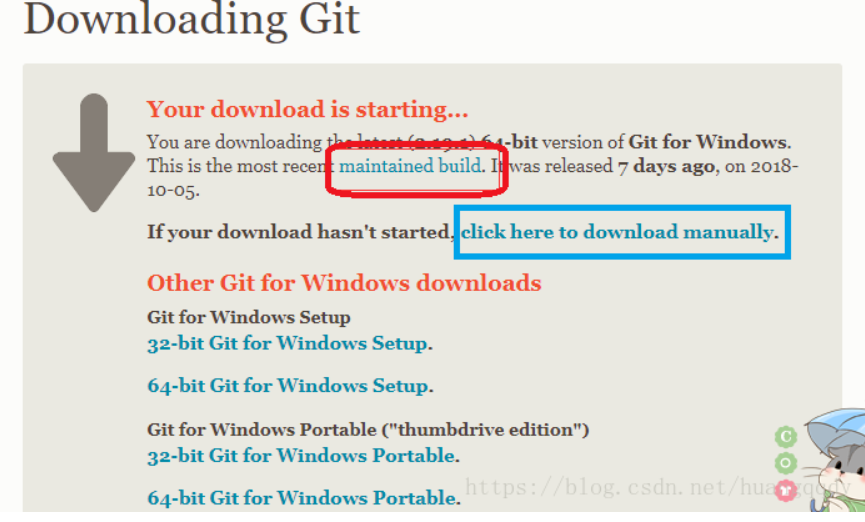


### 掌握git的安装

到git官网地址：<https://git-scm.com/download> 下载相应的安装包

点击windows部分

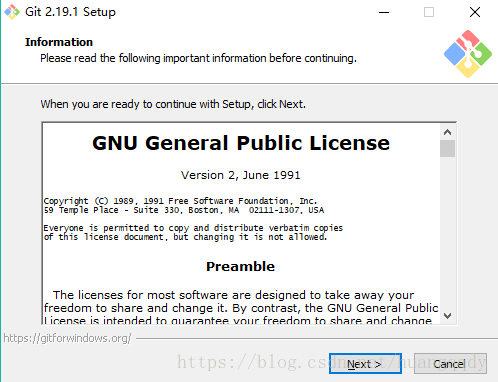


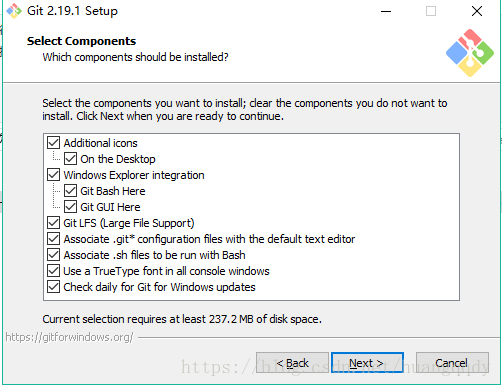


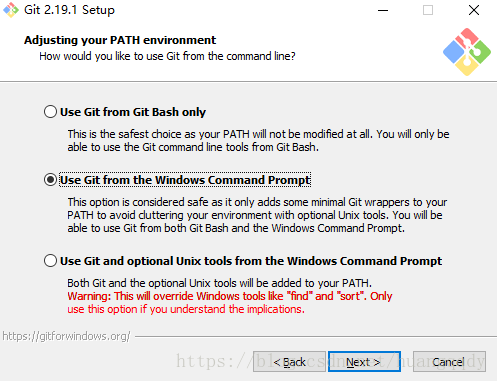
点击下载好的安装包安装这个软件

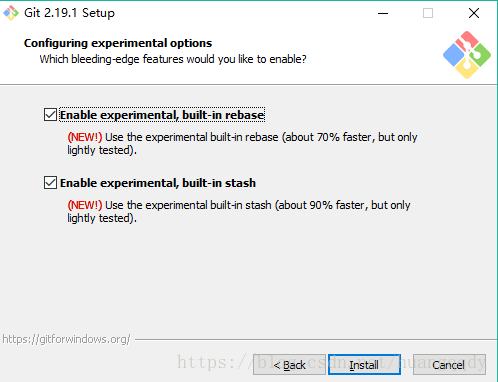
IMG_256

一直点击next，直到出现install，点击install，安装完成后点击finish：







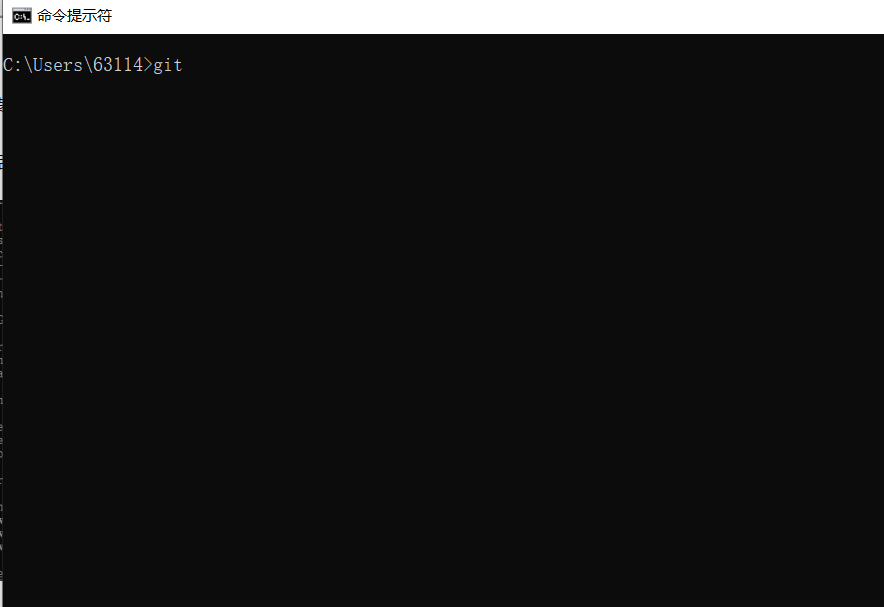


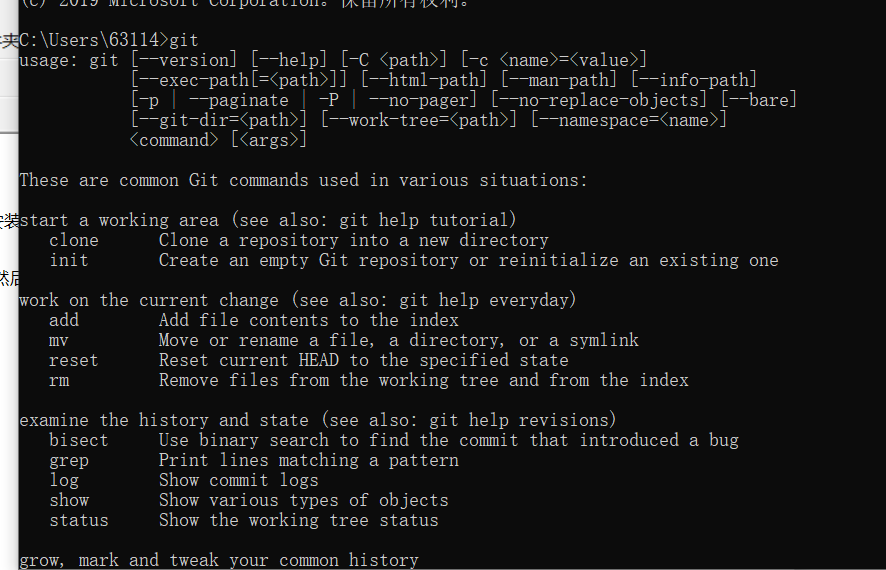
安装好后在桌面界面点击鼠标右键，会出现如下界面



检查git是否安装OK

键盘Ctrl+r，然后在弹出框中出入cdm，弹出如下界面，输入git，回车





配置用户名和邮箱

# 配置用户名

git config --global user.name "username" //（ "username"是自己的账户名，）

# 配置邮箱

git config --global user.email ["username@email.com"](mailto:\"username@email.com\")

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

### Git创建本地化仓库，并推送到github

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



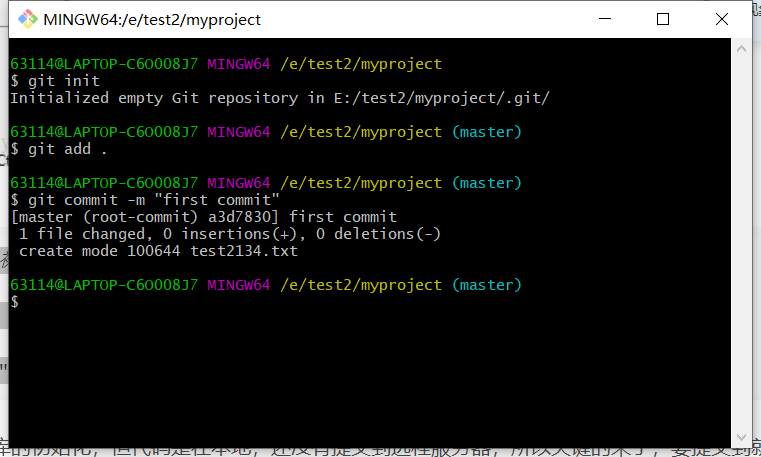
接下来依次执行命令：

git init // 初始化版本库

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（只是添加到缓存区），.代表添加文件夹下所有文件

git commit -m "first commit" // 把添加的文件提交到版本库，并填写提交备注

效果如下



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

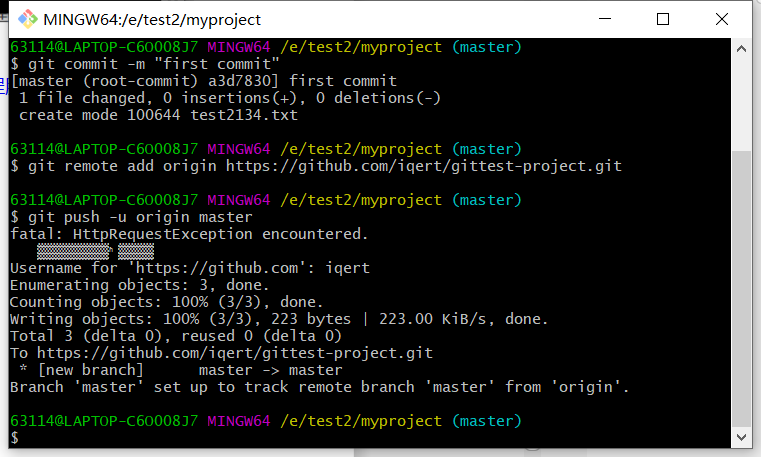
git remote add origin 你的远程库地址 // 把本地库与远程库关联

git push -u origin master // 第一次推送时

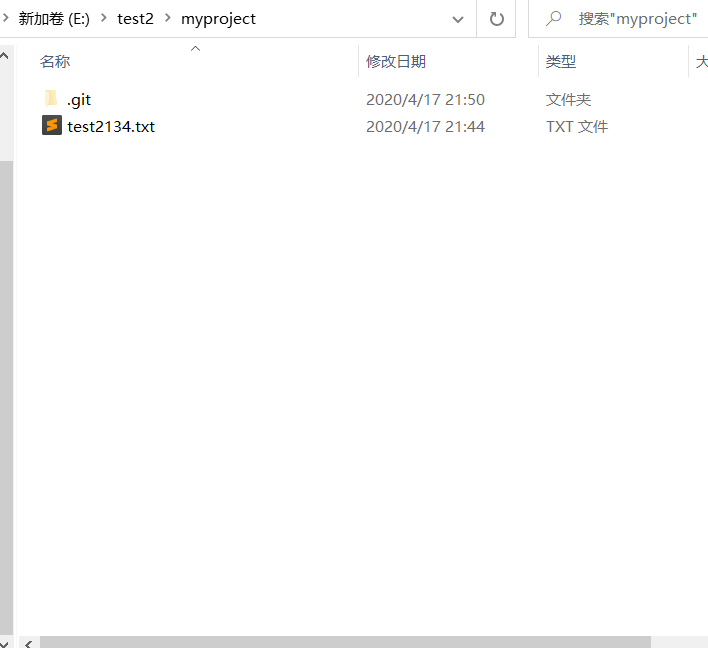
推送时会要求你输入github的账户密码，正确输入即可

git push origin master // 第一次推送后，直接使用该命令即可推送修改

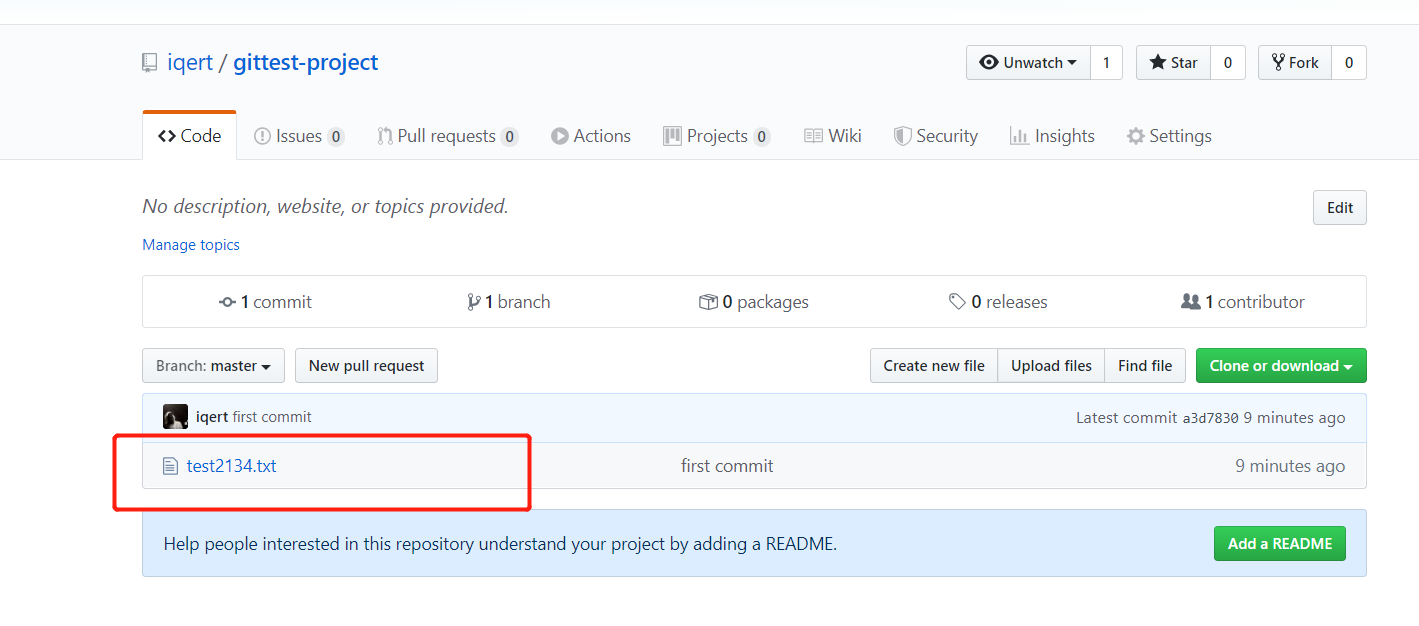
推送成功：



此时我的本地文件夹：



github仓库：



**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

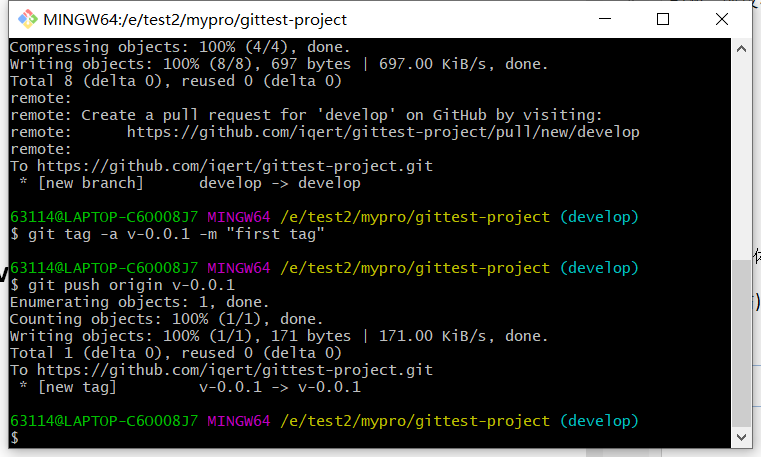
1.添加标签

$ git tag -a v-0.0.1 -m "first tag" //标记一个版本号

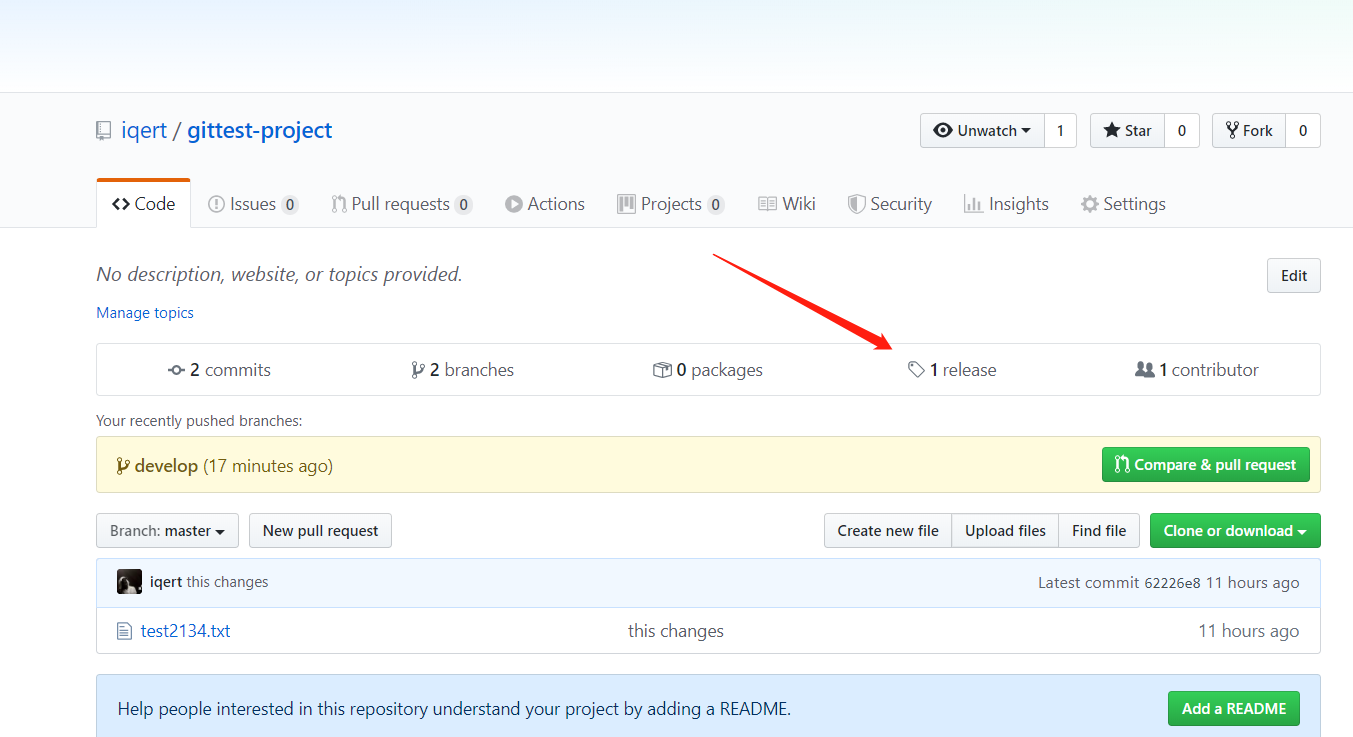
**注解：git tag 是打标签的命令，-a 是添加标签，其后要跟新标签号，-m 及后面的字符串是对该标签的注释。**

2.推送标签到远程仓库

git push origin v-0.0.1



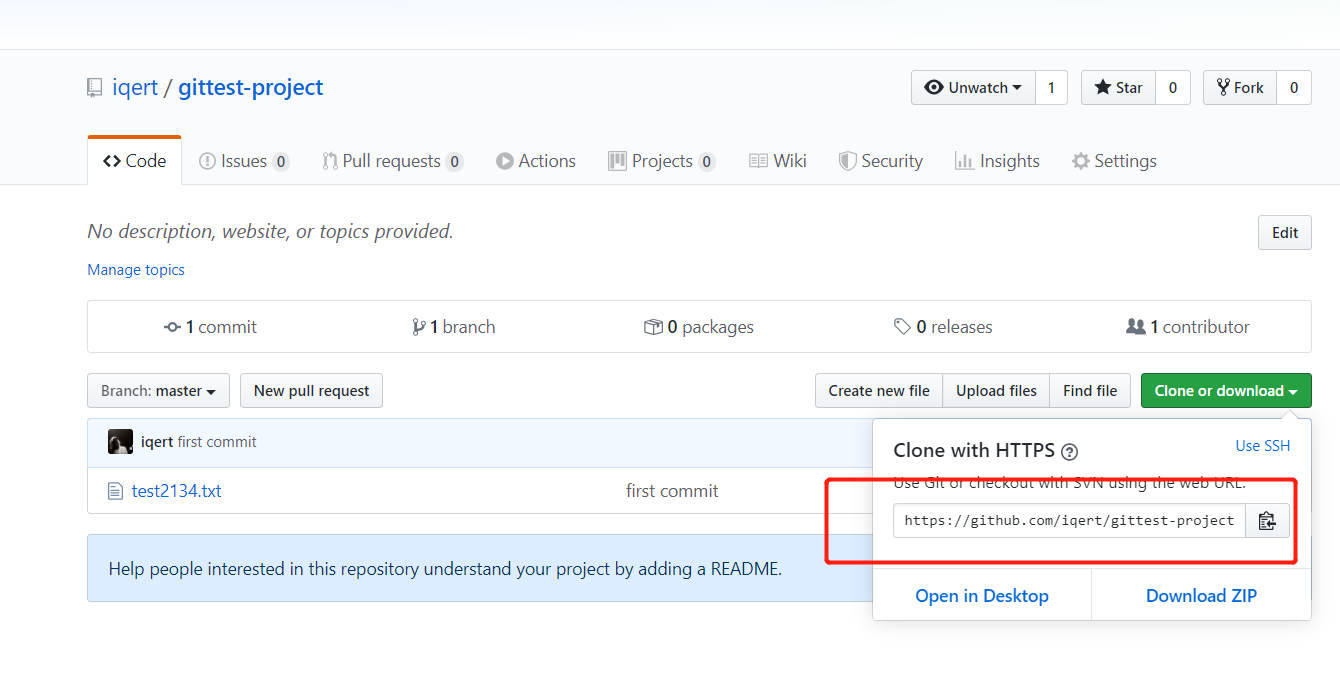
查看github上的标签





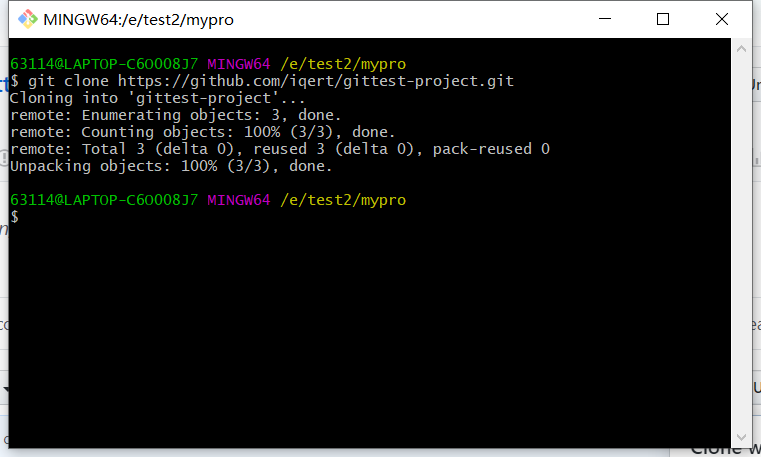
### 通过git拉取github的文件，修改并推送至github仓库

获取github上的项目，需找到项目地址,复制https 路径

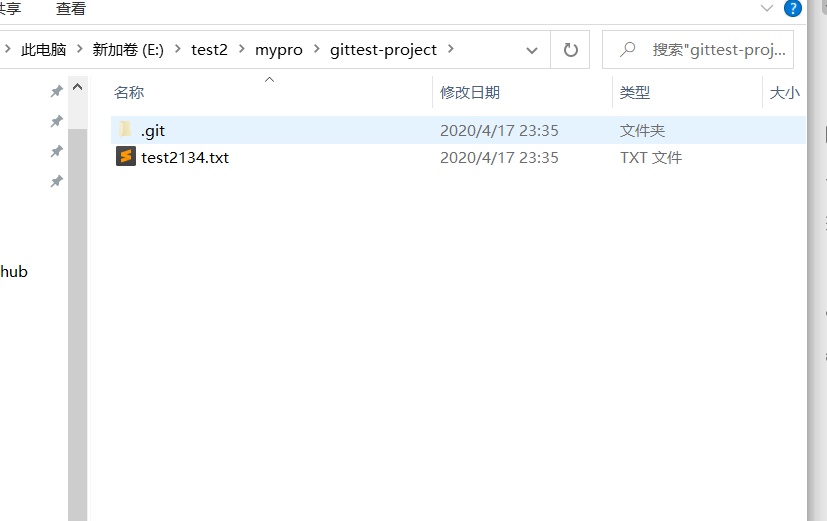


在本地新创建一个目录(mypro),右键点击Git Bash Hera

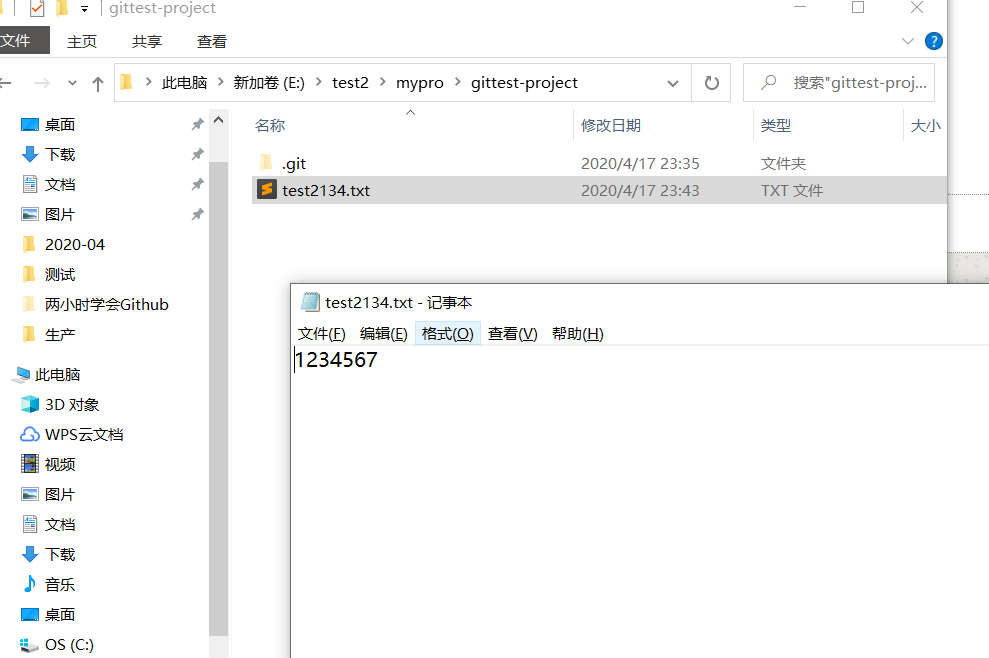
执行$ git clone 项目地址



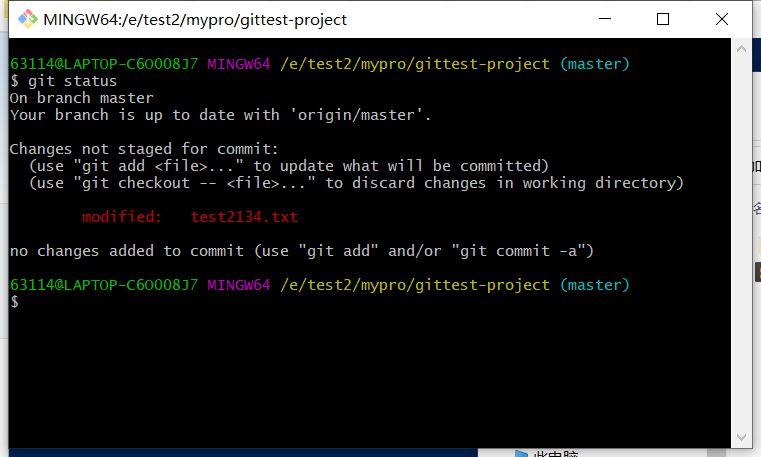
拉取项目成功



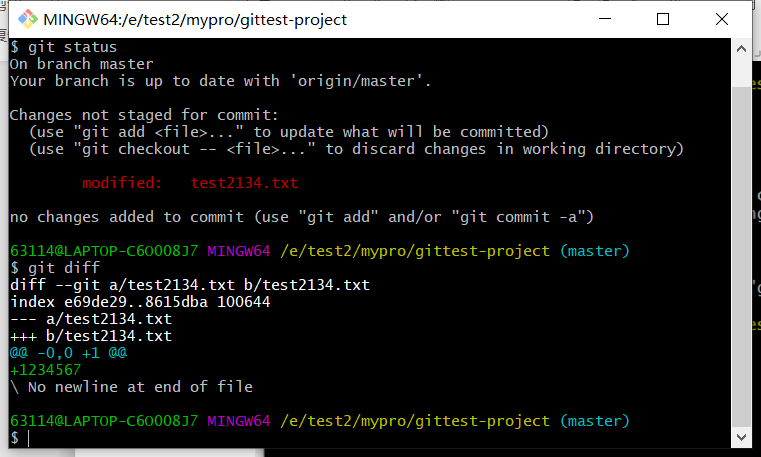
编辑文件夹中的文件，保存



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



输入 git diff 可以查看改动过的内容

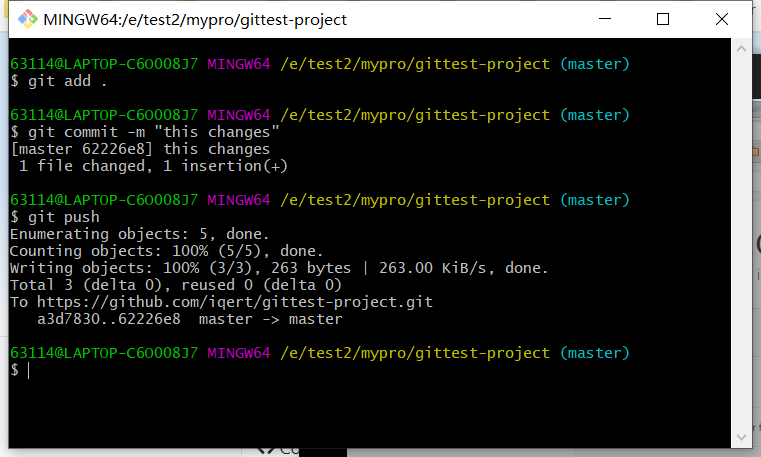


最后输入以下命令，推送到github仓库

Git add .

Git commit -m “this massage”

Git push

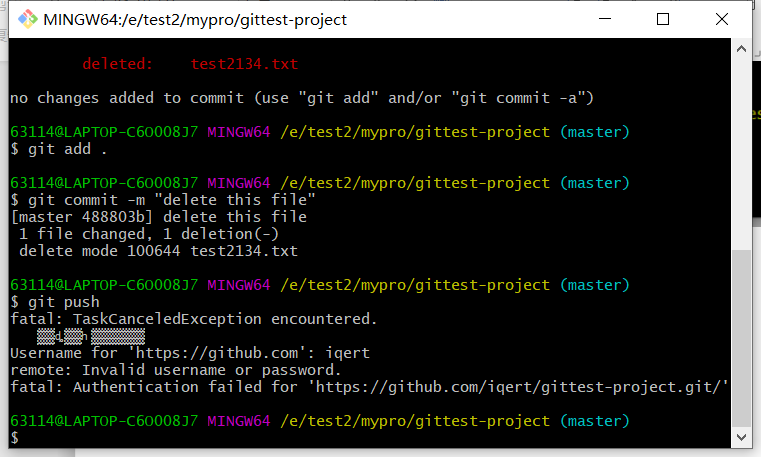


删除文件也一样，删除文件后，执行下面的命令

Git add .

Git commit -m “this massage”

Git push



同样推送到github仓库中，会发现github文件也被删除了。

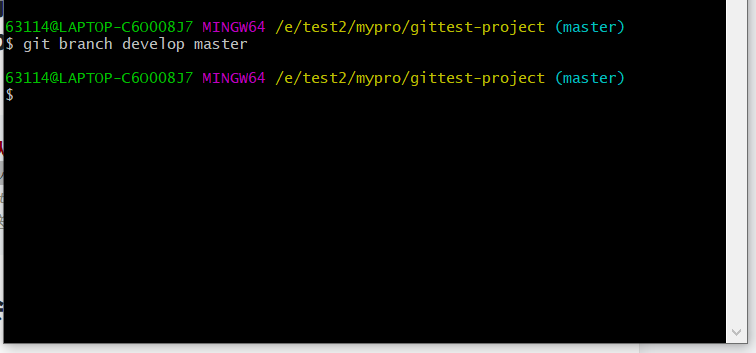
### 基于master主分支上新建develop分支

首先我们先了解以下分支的概念



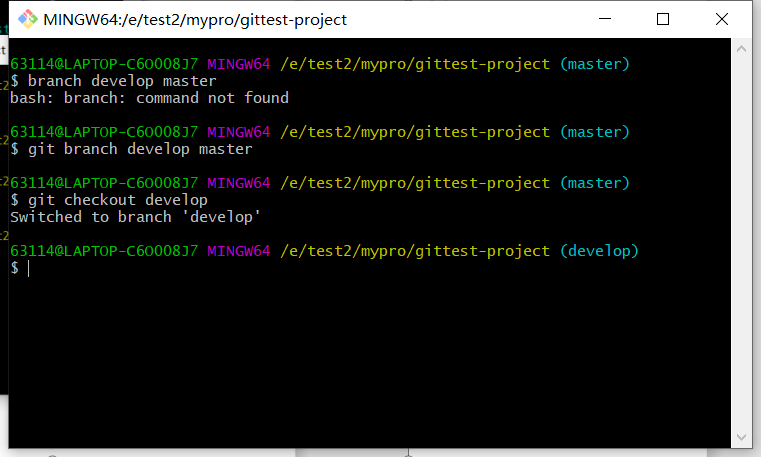
1. 此时本地仓库是再master分支上，从master分支上新建develop分支

Git branch develop master

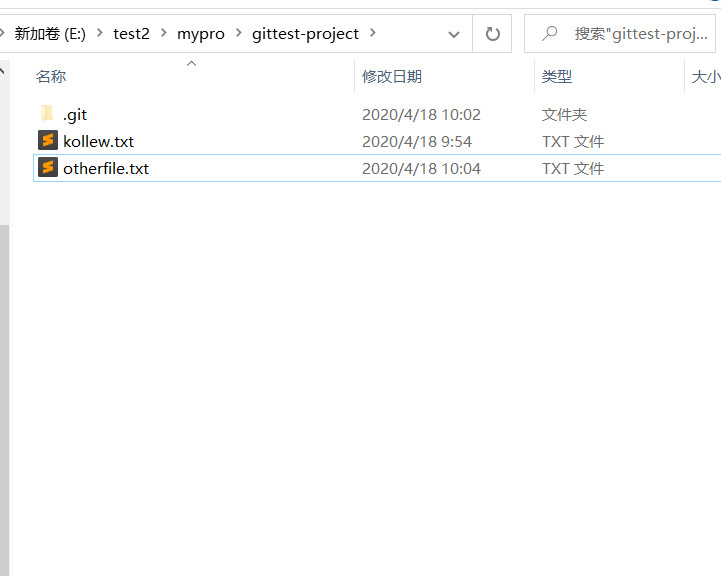


1. 检出develop分支，切换到develop分支上

Git checkout develop



1. 此时我们再develop分支上新建一个文件 otherfile.txt ,然后推送到github的origin/develop分支

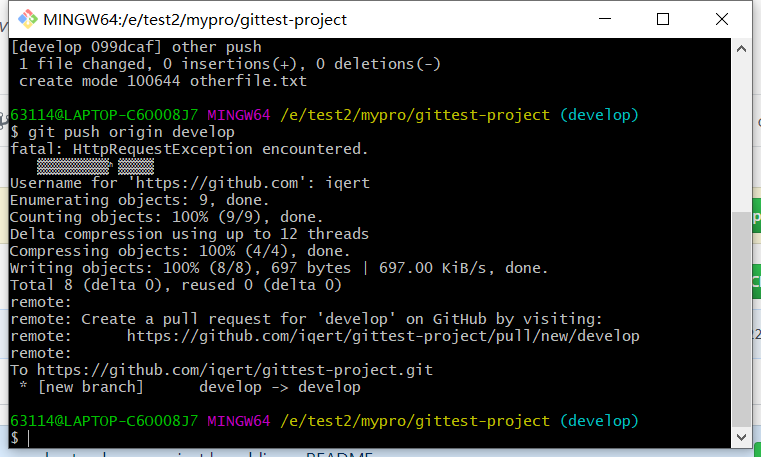


执行以下命令

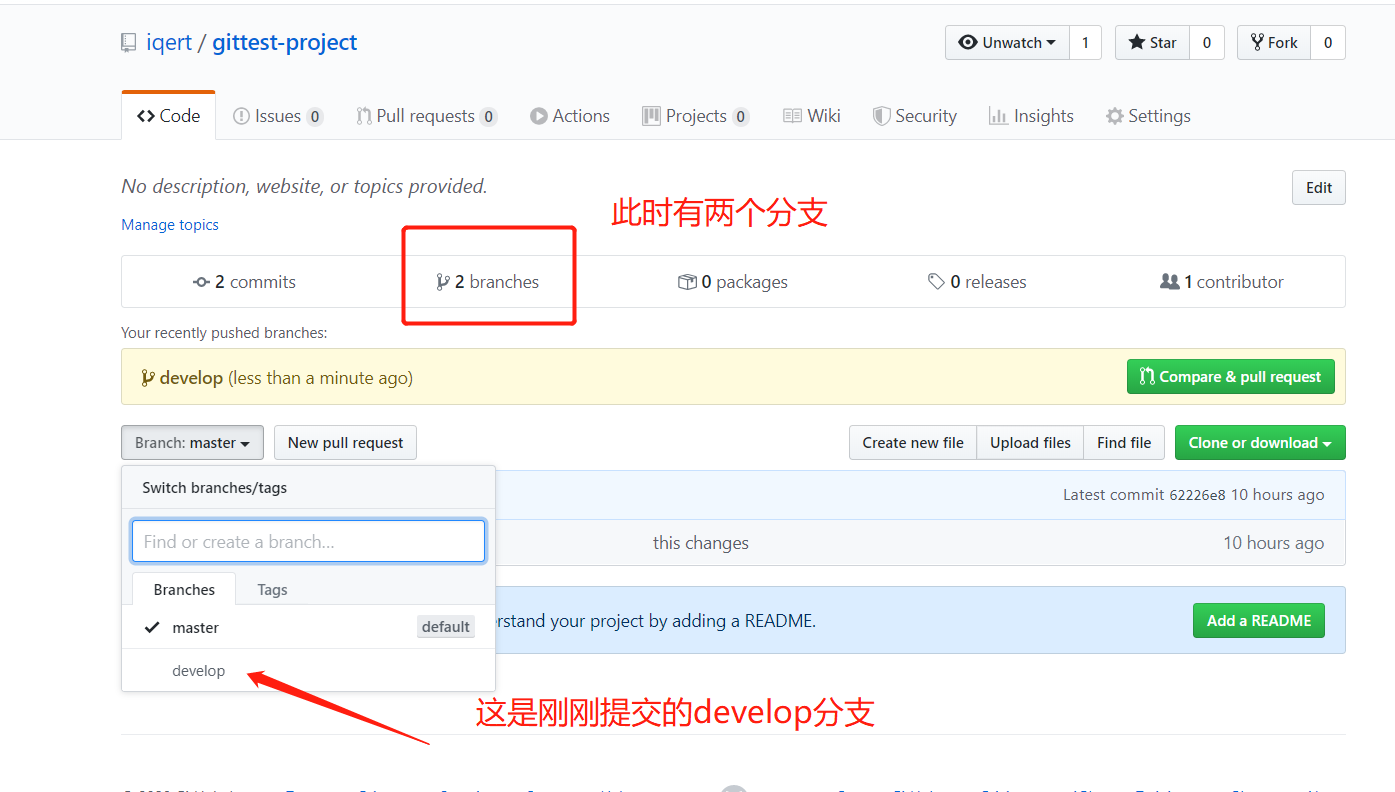
Git add .

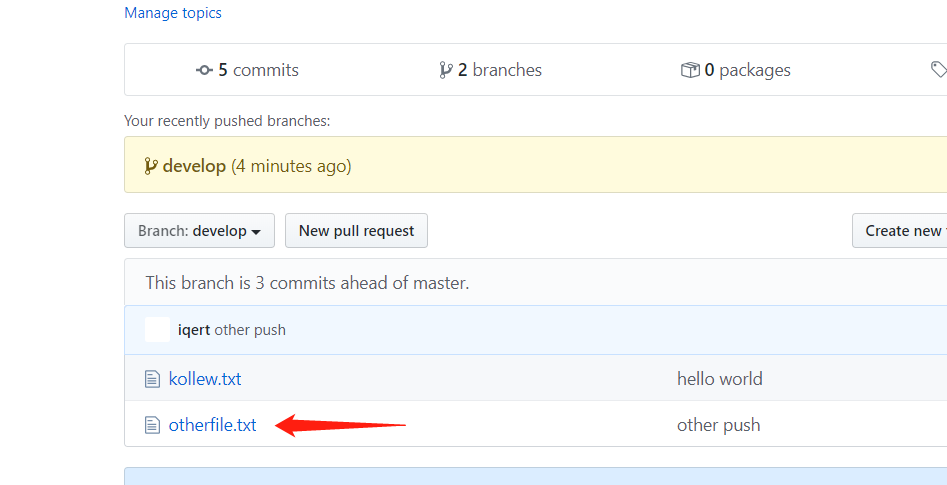
Git commit -m ”other push”

Git push origin develop



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otherfile.txt



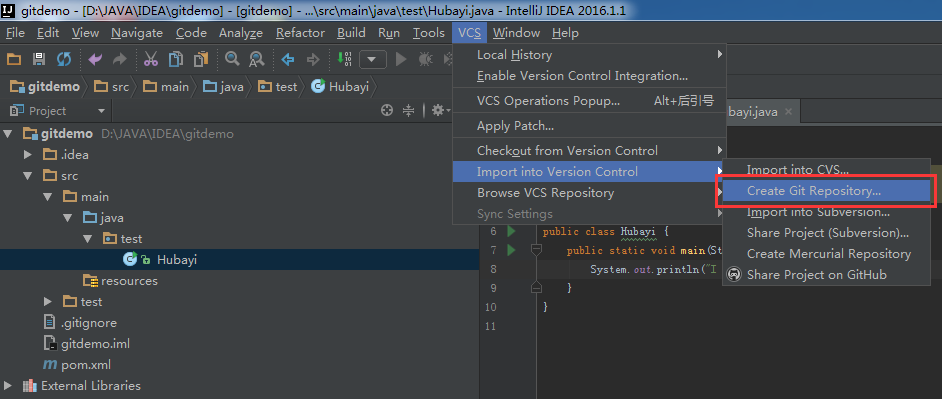


### 在IntelliJ IDEA 开发工具中使用 git的常见的应用场景

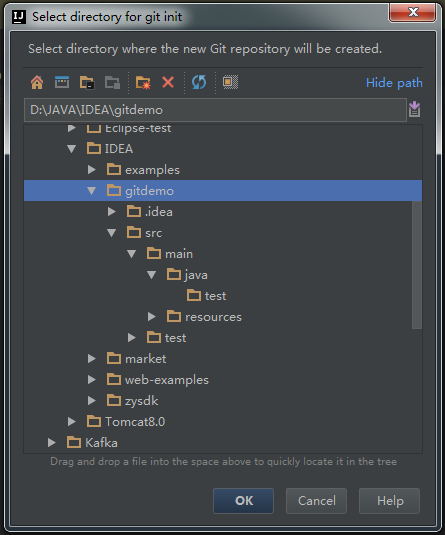
下面来看以上各场景在IDEA中对应的操作。

#### 场景一：小张创建项目并提交到远程Git仓库

创建好项目，选择VCS - > Import into Version Control -> Create Git Repository

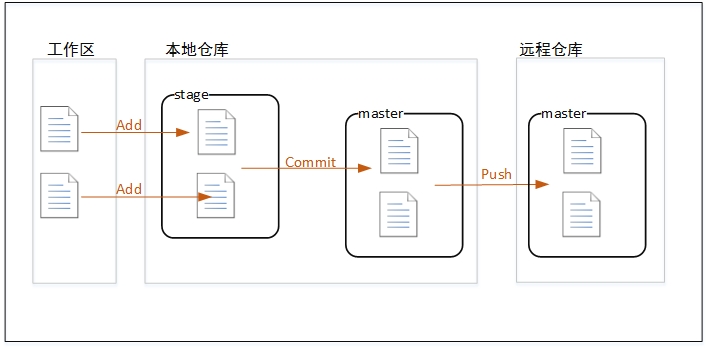


接下来指定本地仓库的位置，按个人习惯指定即可，例如这里选择了项目源代码同目录



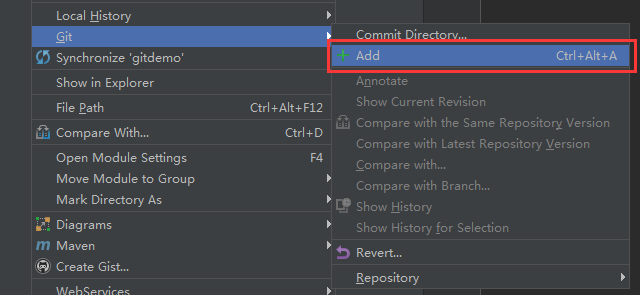
点击OK后创建完成本地仓库，注意，这里仅仅是本地的。下面把项目源码添加到本地仓库。

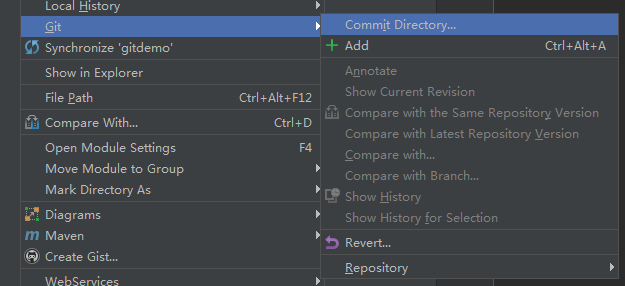
下图是Git与提交有关的三个命令对应的操作，Add命令是把文件从IDE的工作目录添加到本地仓库的stage区，Commit命令把stage区的暂存文件提交到当前分支的仓库，并清空stage区。Push命令把本地仓库的提交同步到远程仓库。

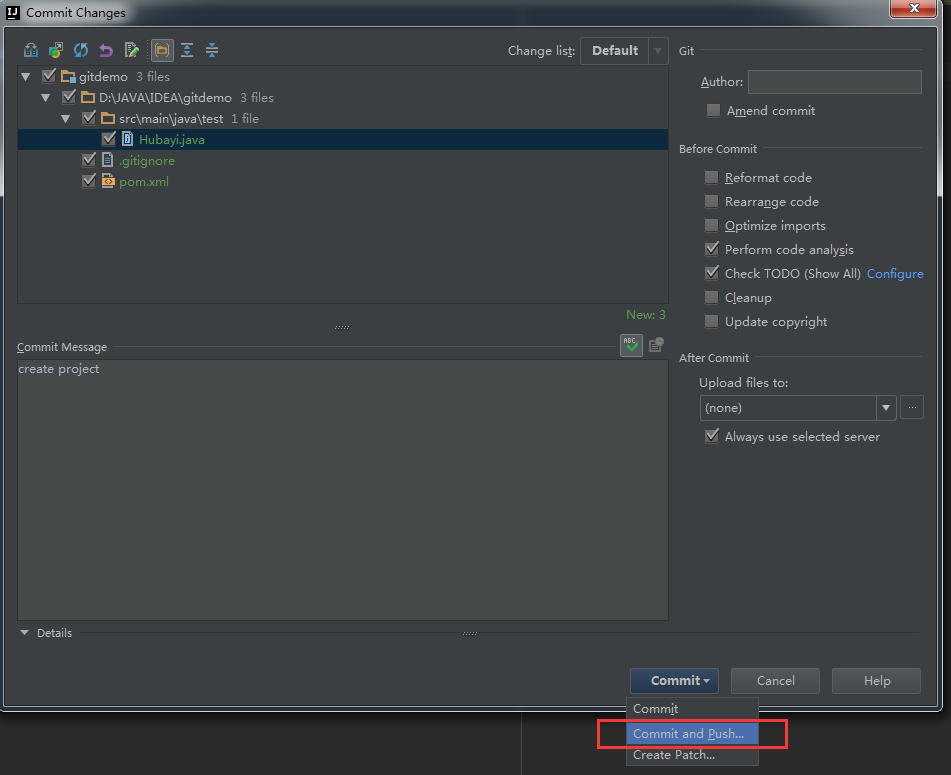


IDEA中对操作做了一定的简化，Commit和Push可以在一步中完成。

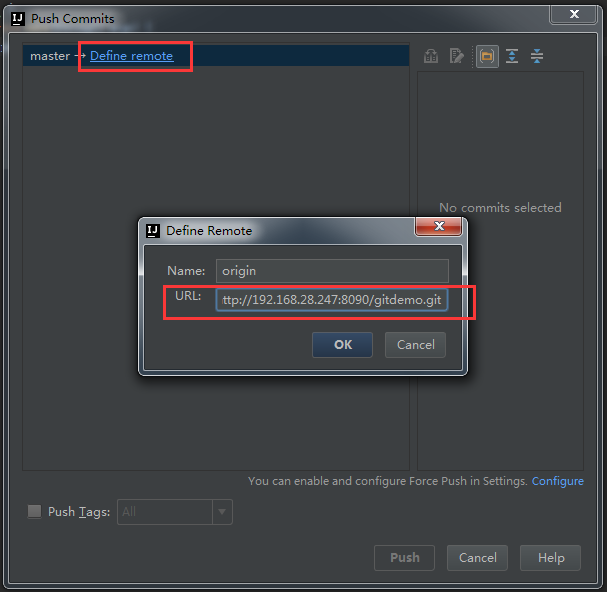
具体操作，在项目上点击右键，选择Git菜单





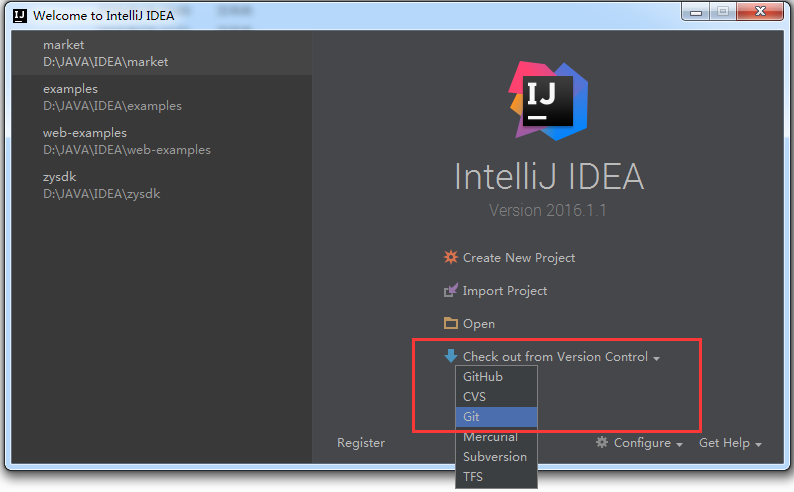


因为是第一次提交，Push前需要指定远程仓库的地址。如下图，点击Define remote后，在弹出的窗口中输入远程仓库地址。

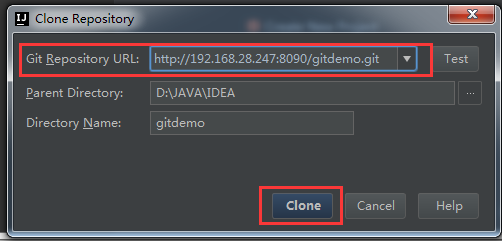


#### **场景二：小袁从远程Git仓库上获取项目源码**

即克隆项目，操作如下：



输入小张Push时填写的远程仓库地址



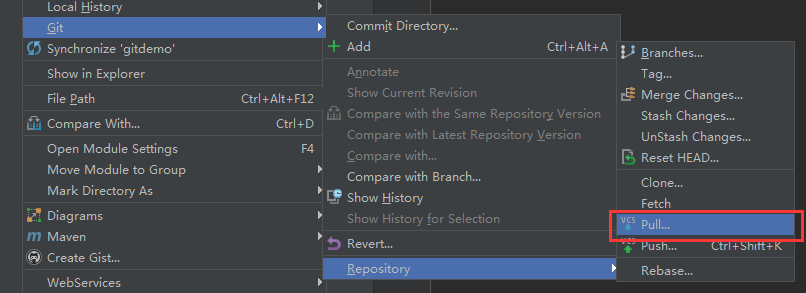
接下来按向导操作，即可把项目从远程仓库克隆到本地仓库和IDE工作区。

#### **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**

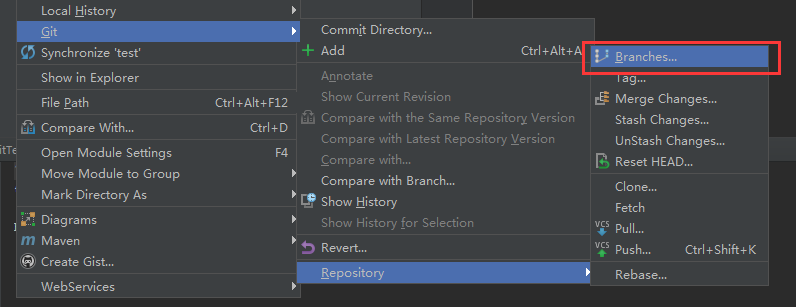
获取更新有两个命令：Fetch和Pull，Fetch是从远程仓库下载文件到本地的origin/master，然后可以手动对比修改决定是否合并到本地的master库。Pull则是直接下载并合并。如果各成员在工作中都执行修改前先更新的规范，则可以直接使用Pull方式以简化操作。



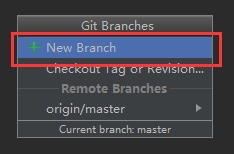
#### **场景五：小袁接受了一个新功能的任务，创建了一个分支并在分支上开发**

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：

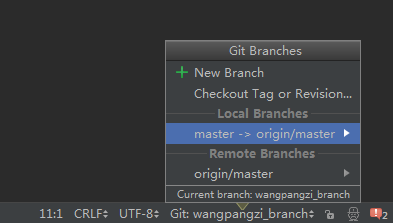


选择New Branch并输入一个分支的名称

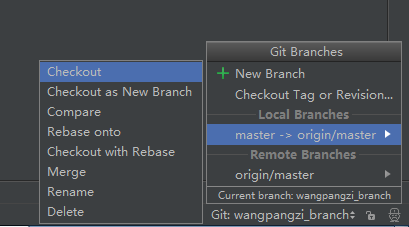


创建完成后注意IDEA的右下角，如下图，Git: wangpangzi\_branch表示已经自动切换到wangpangzi\_branch分支，当前工作在这个分支上。

点击后弹出一个小窗口，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)。



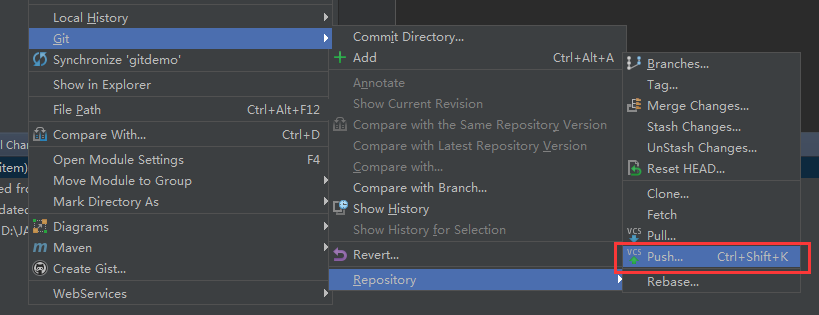
如下图，点击Checkout

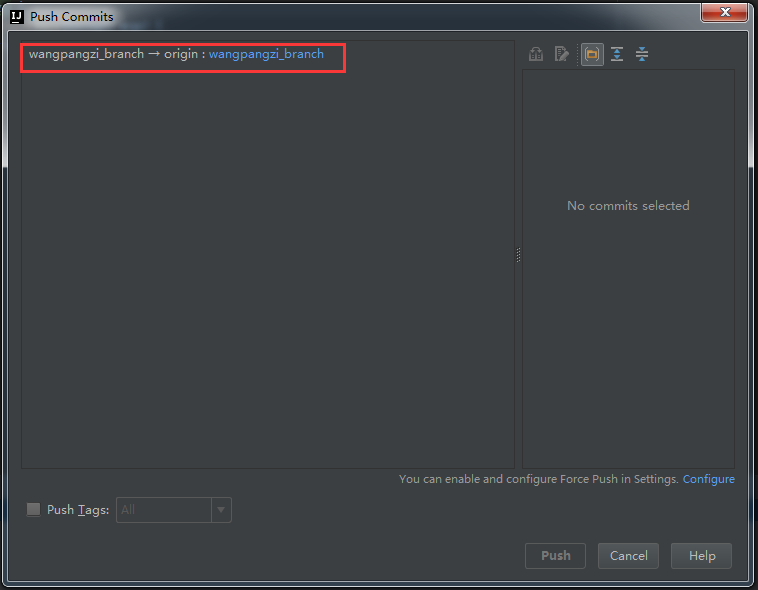


注意，这里创建的分支仅仅在本地仓库，如果想让组长小张获取到这个分支，还需要提交到远程仓库。

#### **场景六：小袁把分支提交到远程Git仓库**

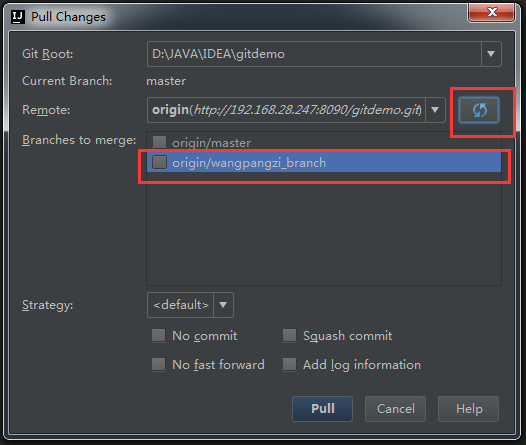
切换到新建的分支，使用Push功能



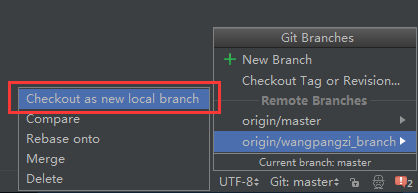


#### **场景七：小张获取小袁提交的分支**

使用Pull功能打开更新窗口，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。



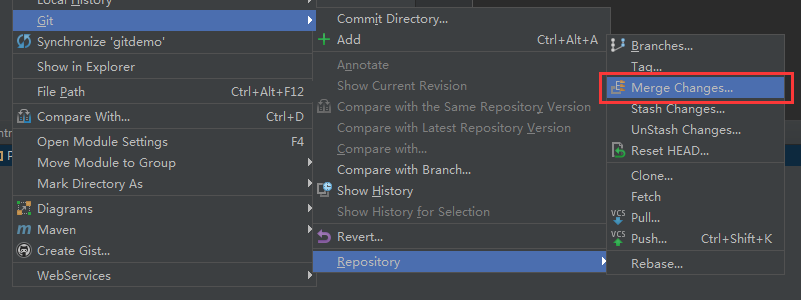
更新后，再点击右下角，可以看到在Remote Branches区已经有了新的分支，点击后在弹出的子菜单中选择Checkout as new local branch，在本地仓库中创建该分支。完成后在Local Branches区也会出现该分支的选项，可以按上面的方法，点击后选择Checkout切换。



#### **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

