版本管理工具小乌龟 Git 的安装和使用

修改记录

A 类文档			
撰写人	时间	版本	修改内容

1、软件的安装：

1、安装 Git

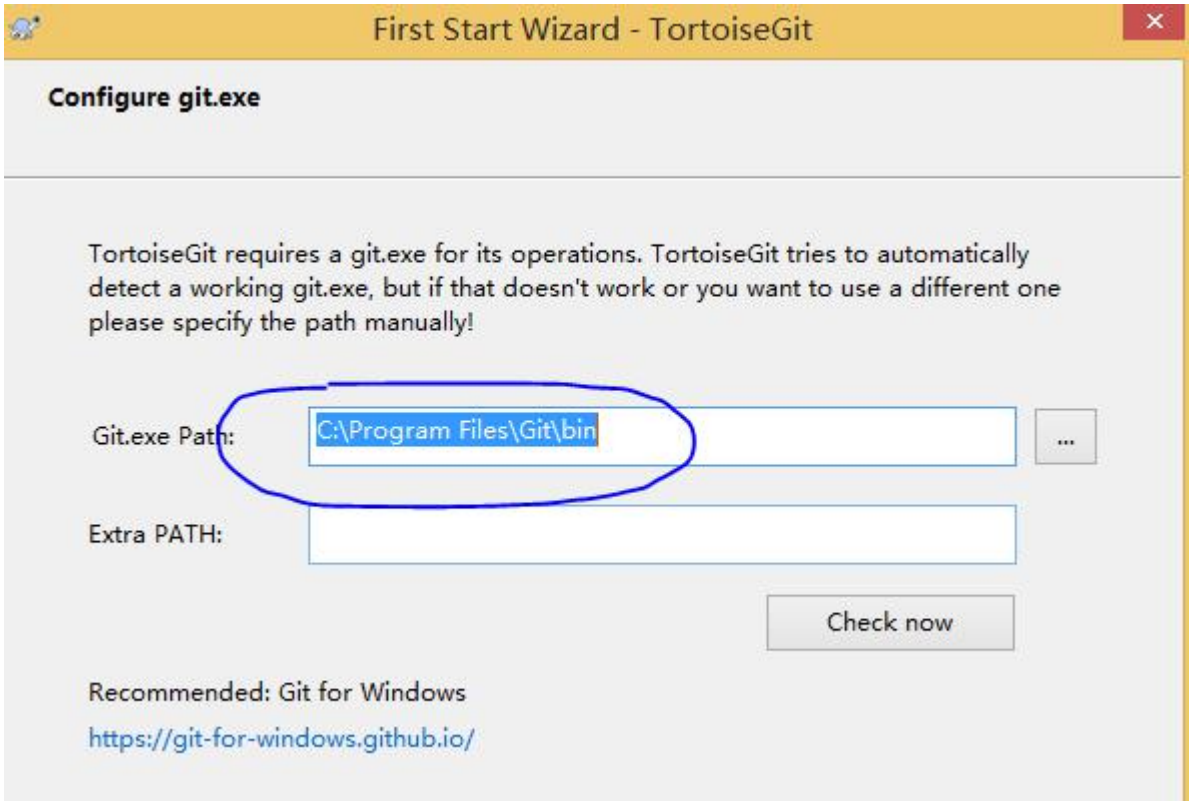
使用软件管理工具搜索 Git：

先安装 **Git**（也可以在 **https://git-for-windows.github.io/** 找到最新版），全部选择默认即可；

再安装小乌龟（也可以在 **https://tortoisegit.org/download/** 下载最新版）；

2、安装小乌龟 Git 的配置

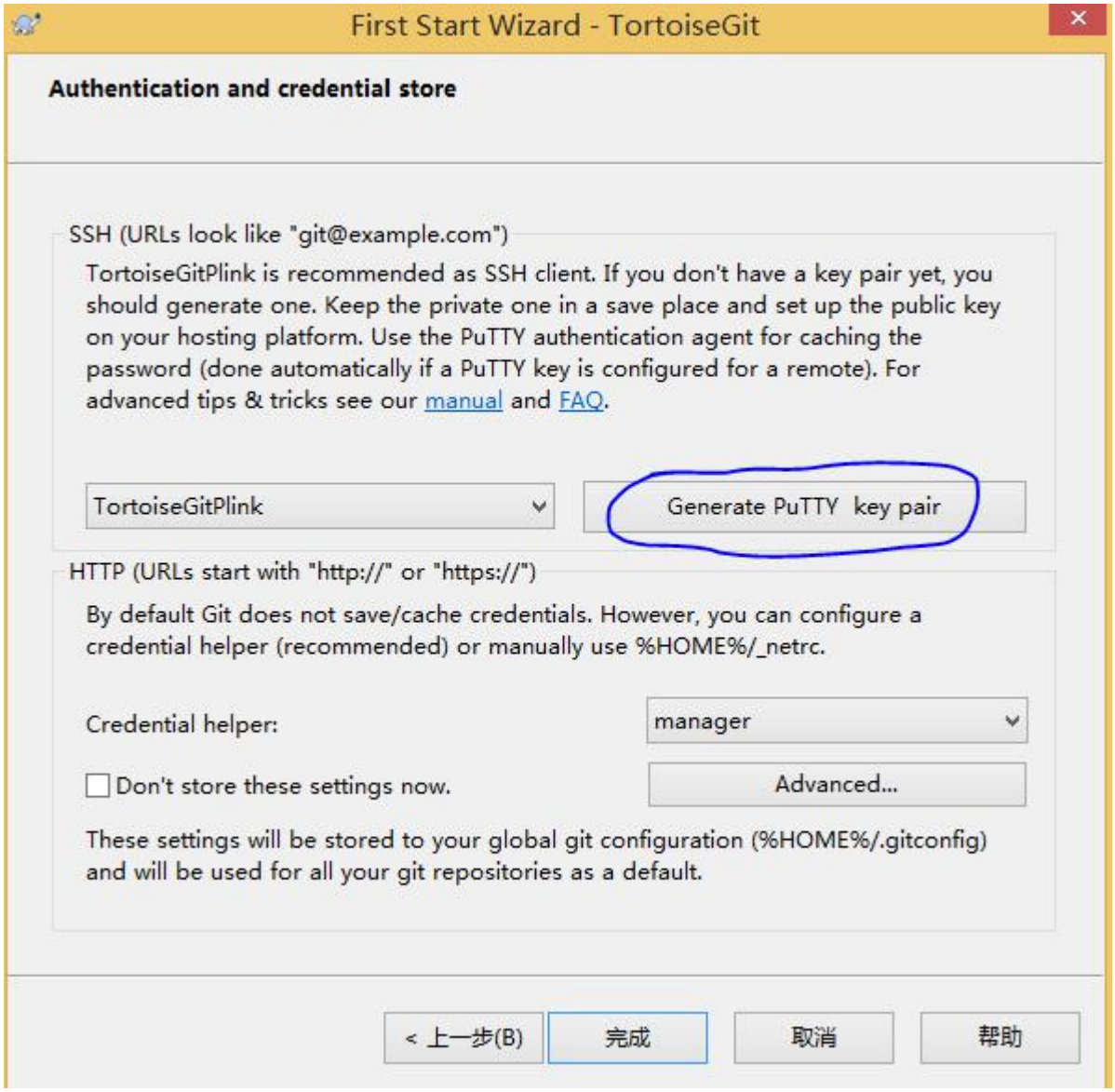
小乌龟 Git 的安装向导会自动检测 Git 的安装路径（一般是 C:\Program Files\Git\bin），如果没有，点击 Check now 或者手动填写：



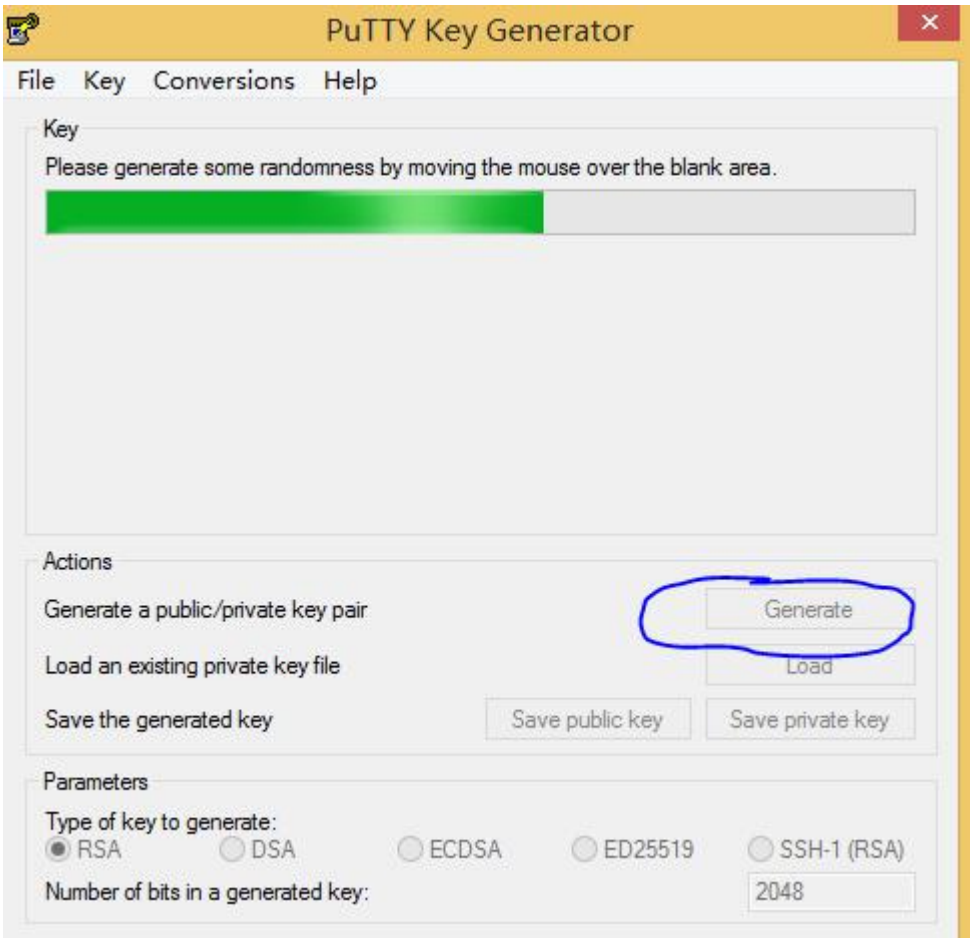
下一步，填写姓名和邮件地址，不是 Github 的注册邮箱，只作为提交时的记录；

生成公钥/私钥

生成公、私钥对，用来将代码提交到 Github：

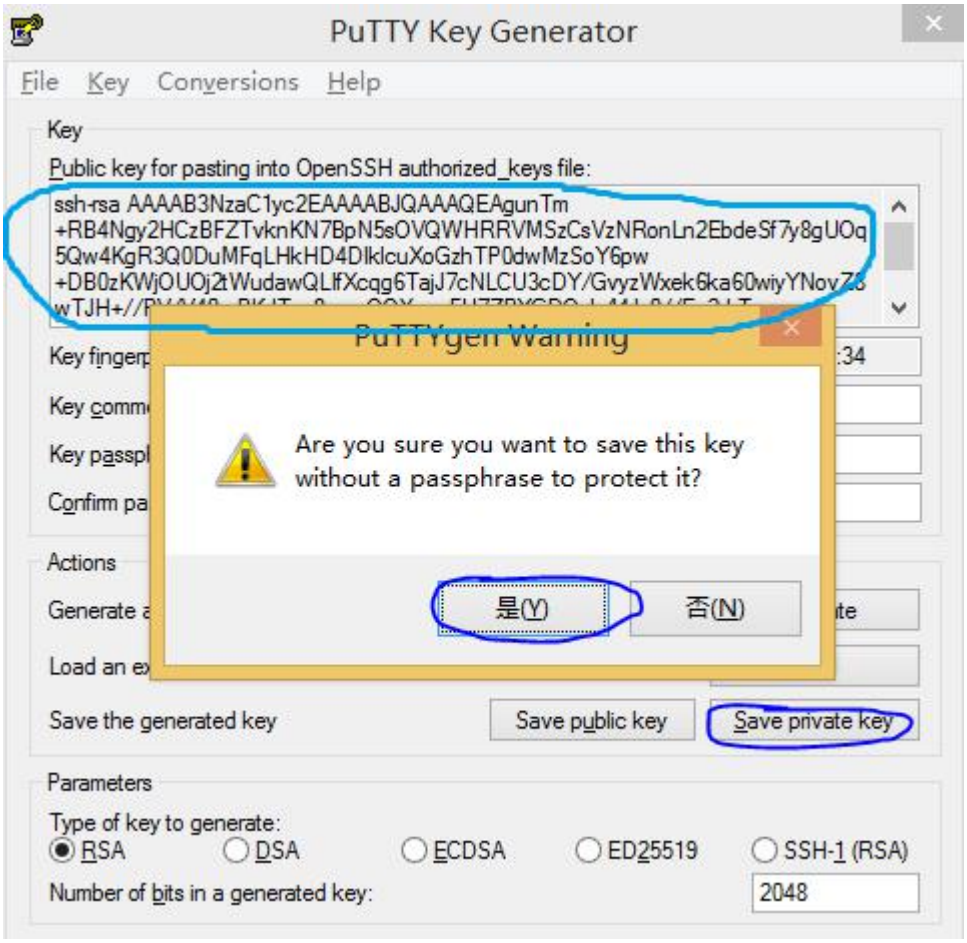


点击 Generate，并不断移动鼠标在空白处；



待 Generate 进度完成后，复制公钥信息（可以保存到文本文件里，以免丢失）；

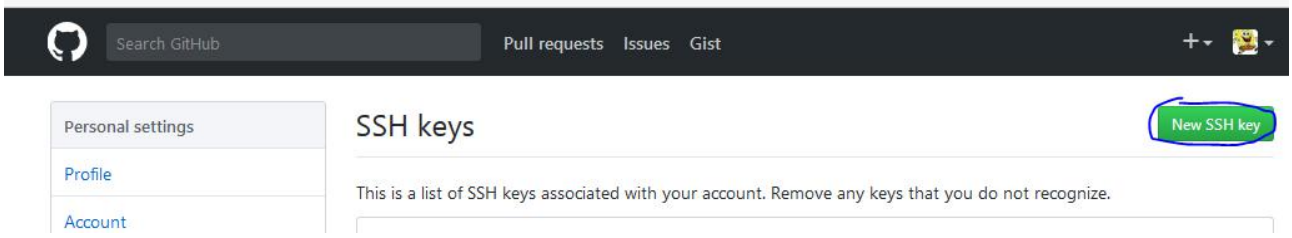
点击 Save private key 保存密钥，不必添加密码短语：



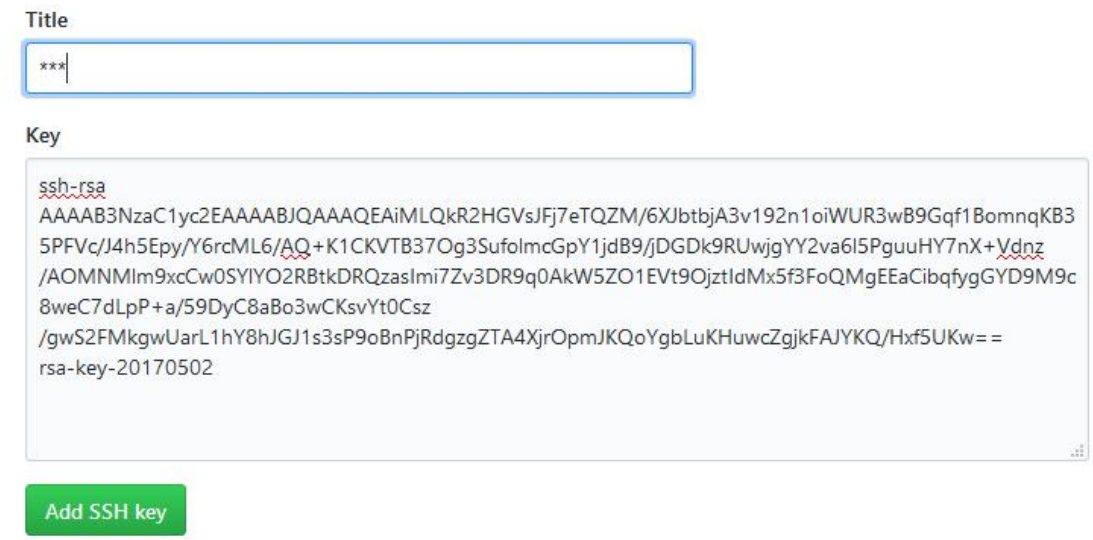
2、添加公钥到 Github

登录：https://github.com/settings/keys

添加新的 SSH 公钥，



名字任意；Key 中粘贴上一步中生成的公钥（共一行）；

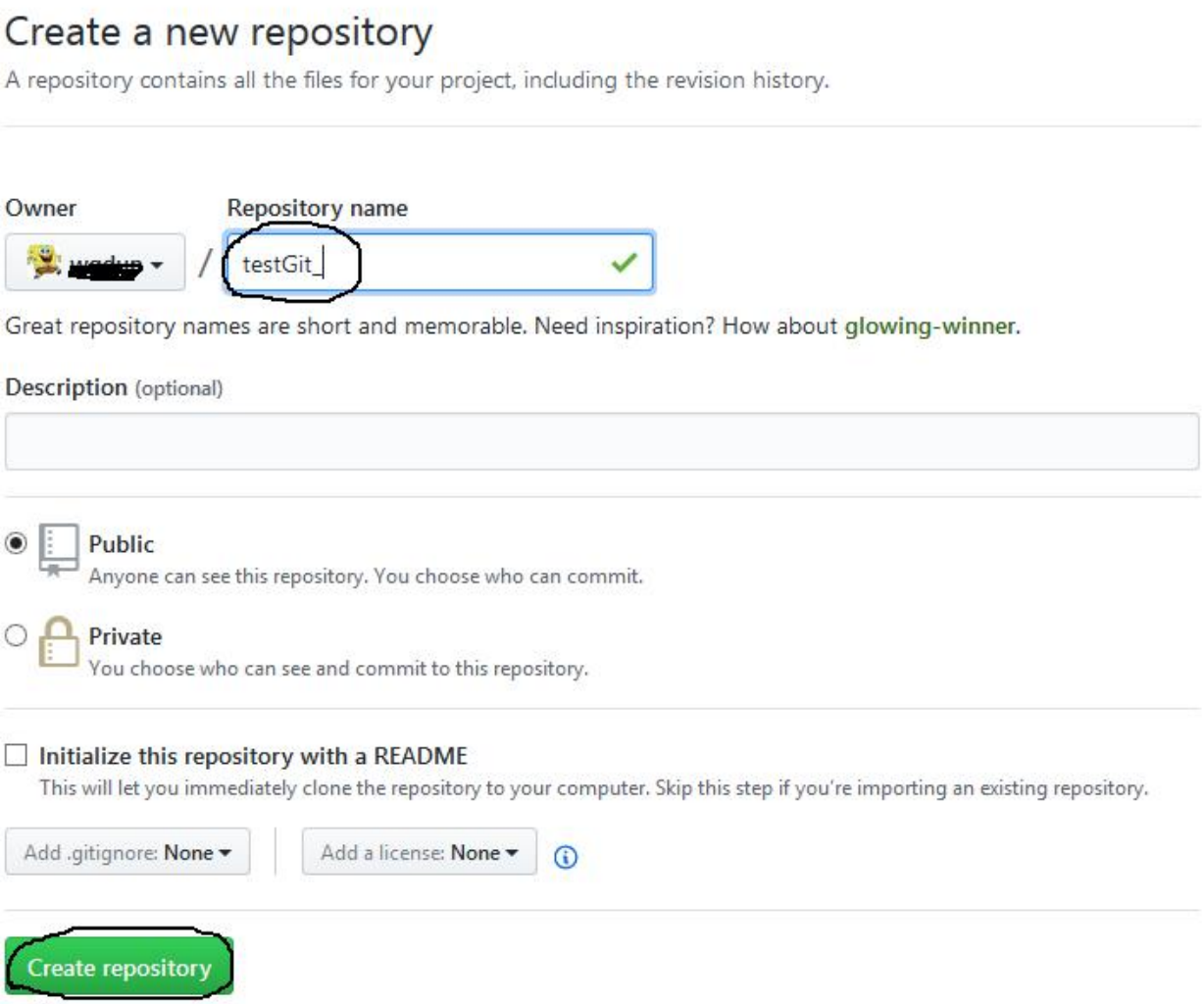


3、将本地文件纳入版本管理

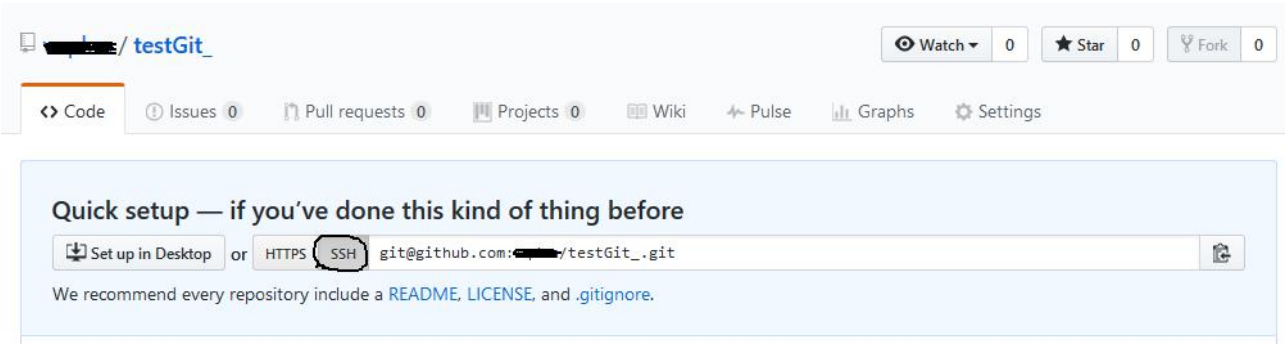
1、在 Github 新建代码仓库

登入：**https://github.com/new**

新建代码仓库：

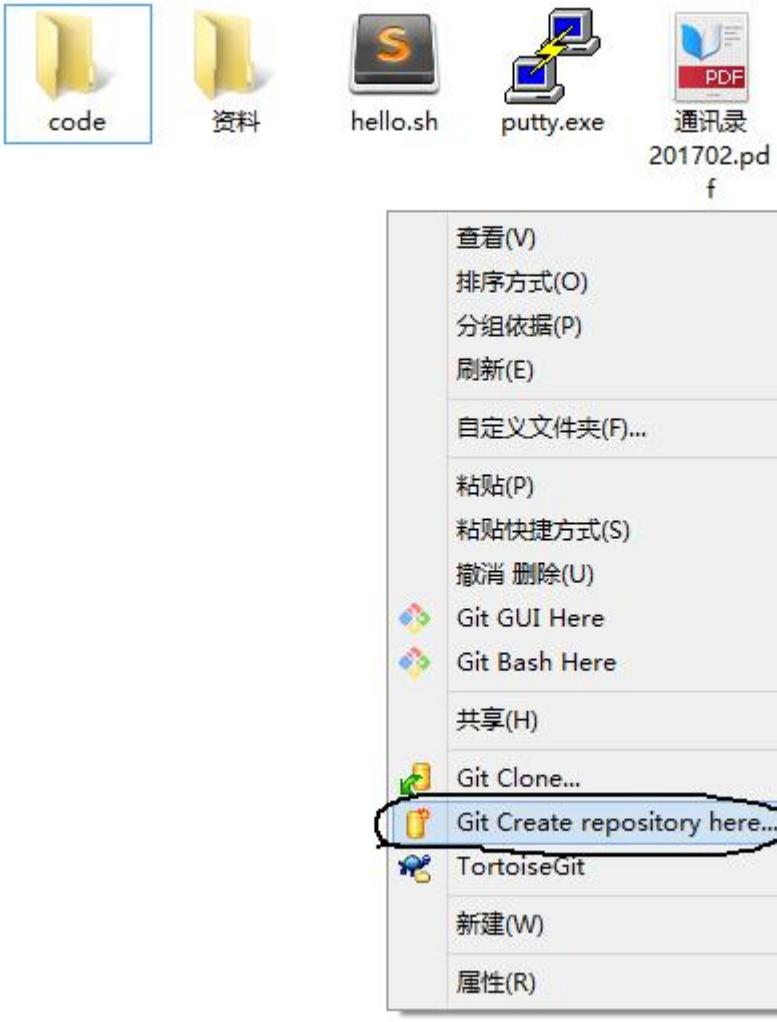


进入该仓库，点击 **SSH**，复制后面的 **SSH** 地址（后面要用）



2、在计算机上创建代码仓库

进入要纳入版本管理的文件夹，右键鼠标**-Git** 在这里创建版本库：

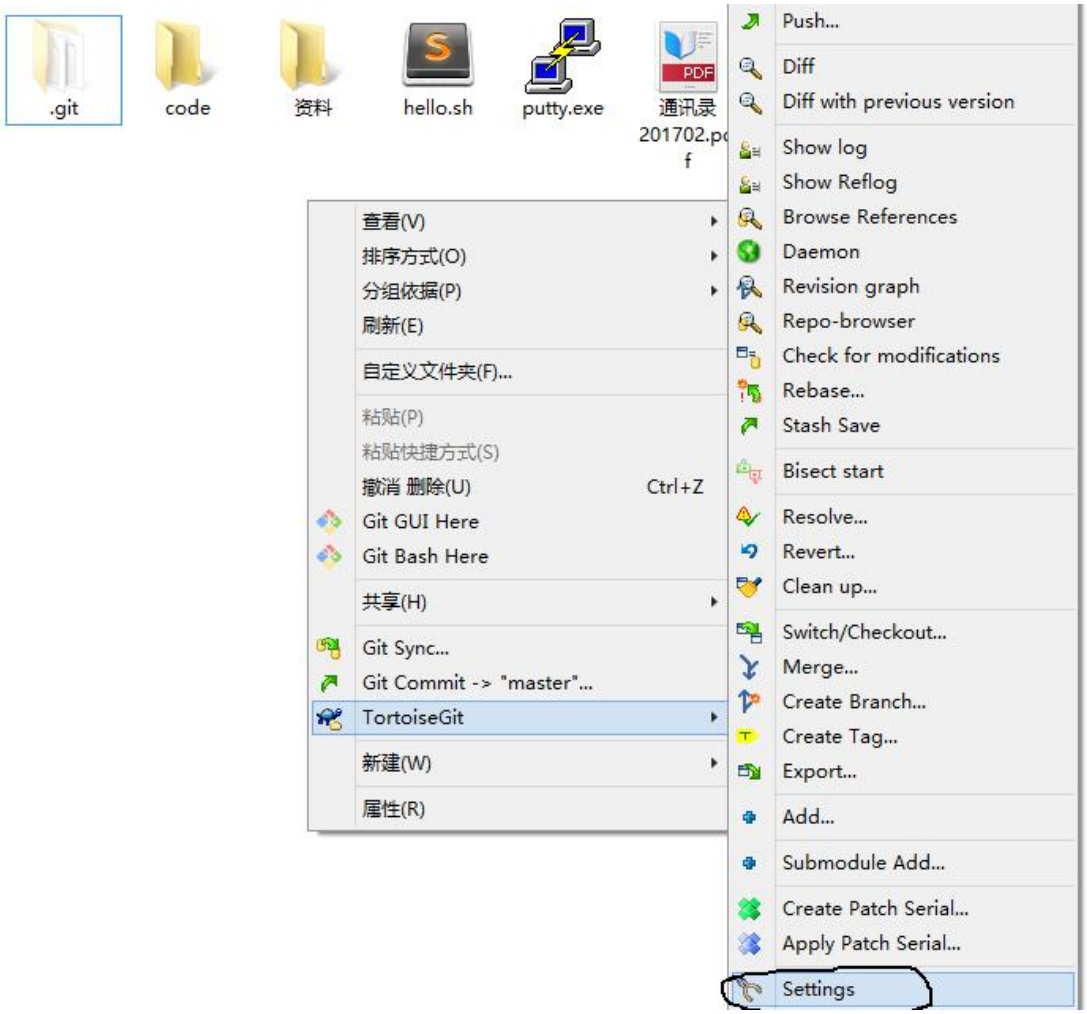

code
资料
hello.sh
putty.exe
通讯录
201702.pdf
查看(V)
排序方式(O)
分组依据(P)
刷新(E)
自定义文件夹(F)...
粘贴(P)
粘贴快捷方式(S)
撤消 删除(U) Ctrl+Z
Git GUI Here
Git Bash Here
共享(H)
Git Clone...
Git Create repository here...
TortoiseGit
新建(W)
属性(R)


E:\idriver\testGit_ - Git Init - TortoiseGit
Make it Bare (No working directories)
If you plan to work inside this folder, leave this unchecked. Typically a bare repo can only have changes 'pushed' to it. (By convention, a bare repo folder should have a name that ends with '.git')
OK
Cancel
Help

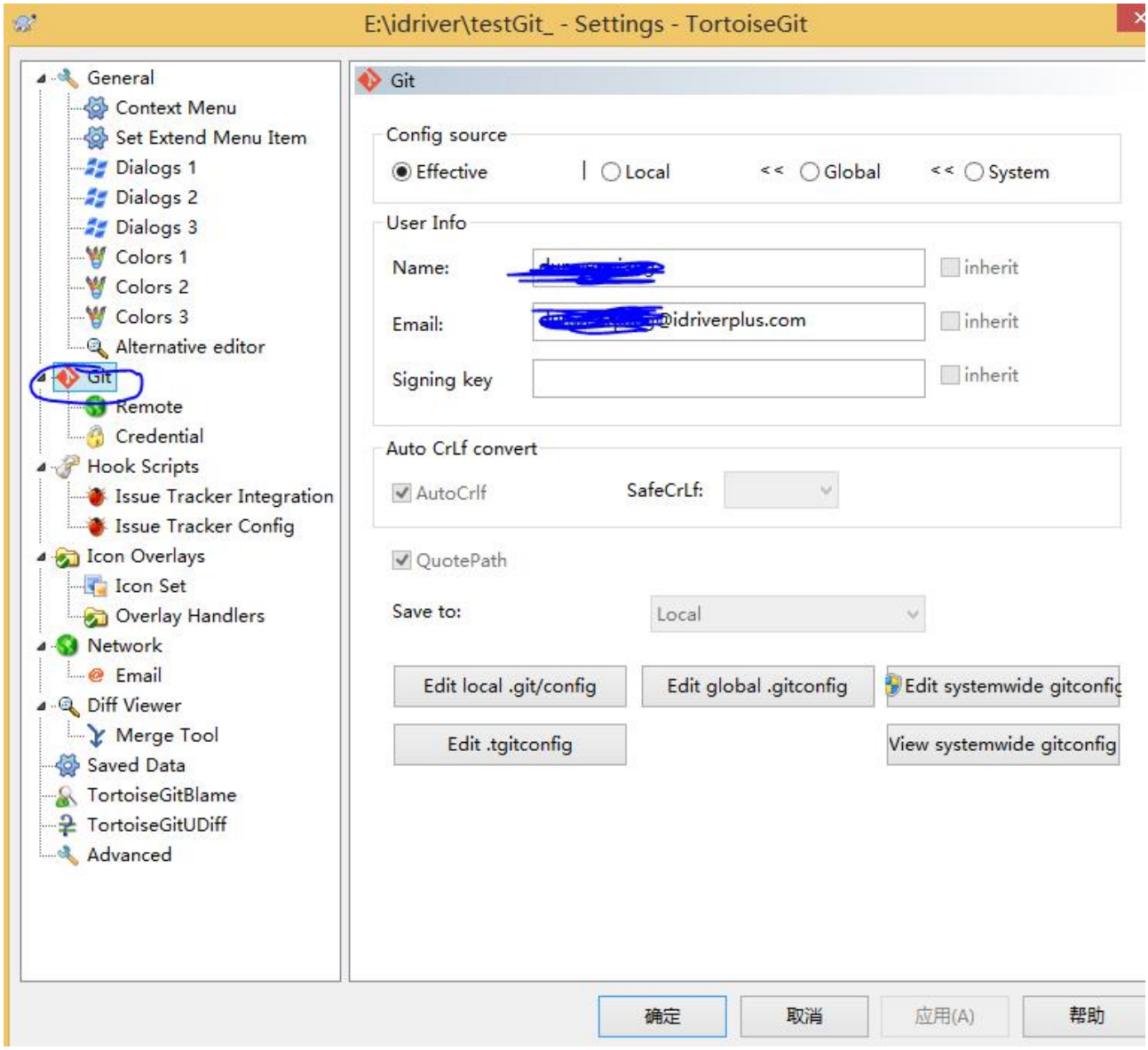
成功后，该文件夹下生成.git 文件夹：

3、本地仓库连接 Github 仓库

继续右键，**TortoiseGit->**设置：



点击左侧 **Git**，设置名字和邮件地址（该信息与 **Github** 无关，只用来记录提交日志）：



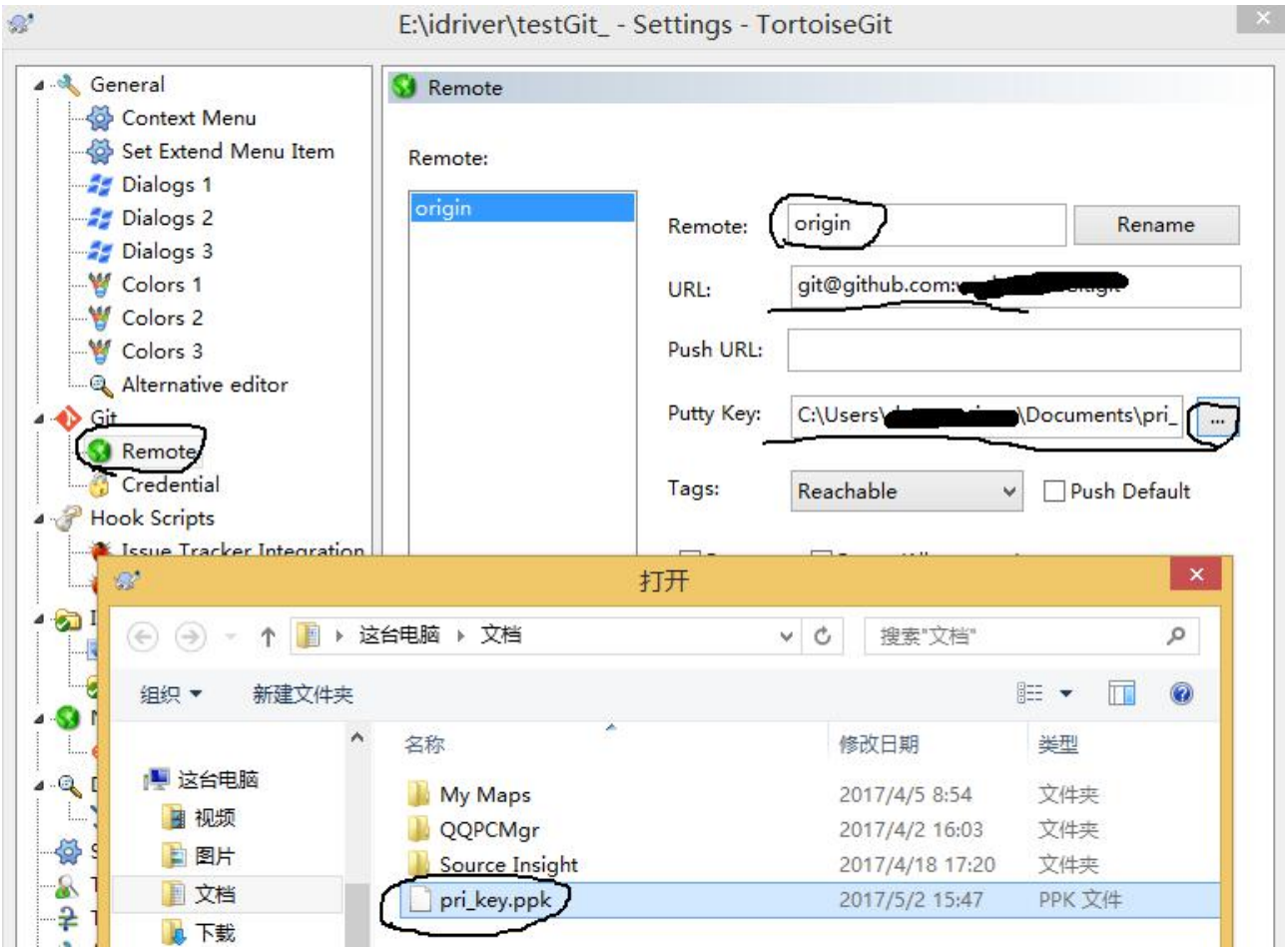
点击左侧 **Remote**：

在右边 Remote 中填入 origin，

url 填刚才 Github 中代码仓库的 SSH 地址，

Putty Key 导入第一步中生成的私钥文件；

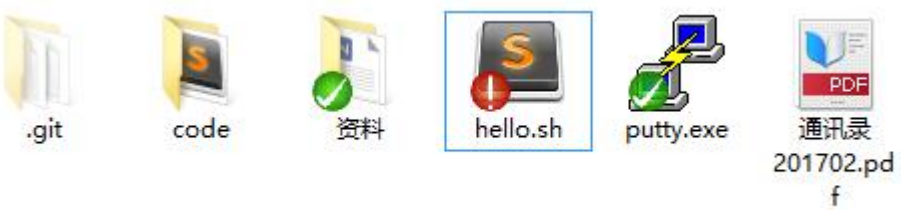
点击添加/保存：



做完以上步骤，就可以将代码进行版本管理，也可以 **push** 到 **Github** 的仓库上了。

4、文件前标识符的含义

1、文件前面没有任何标志：



表示没有 track 该文件（即没有 git add）；

2、文件前面有红色的！，表示该文件有未提交（git commit）的修改；

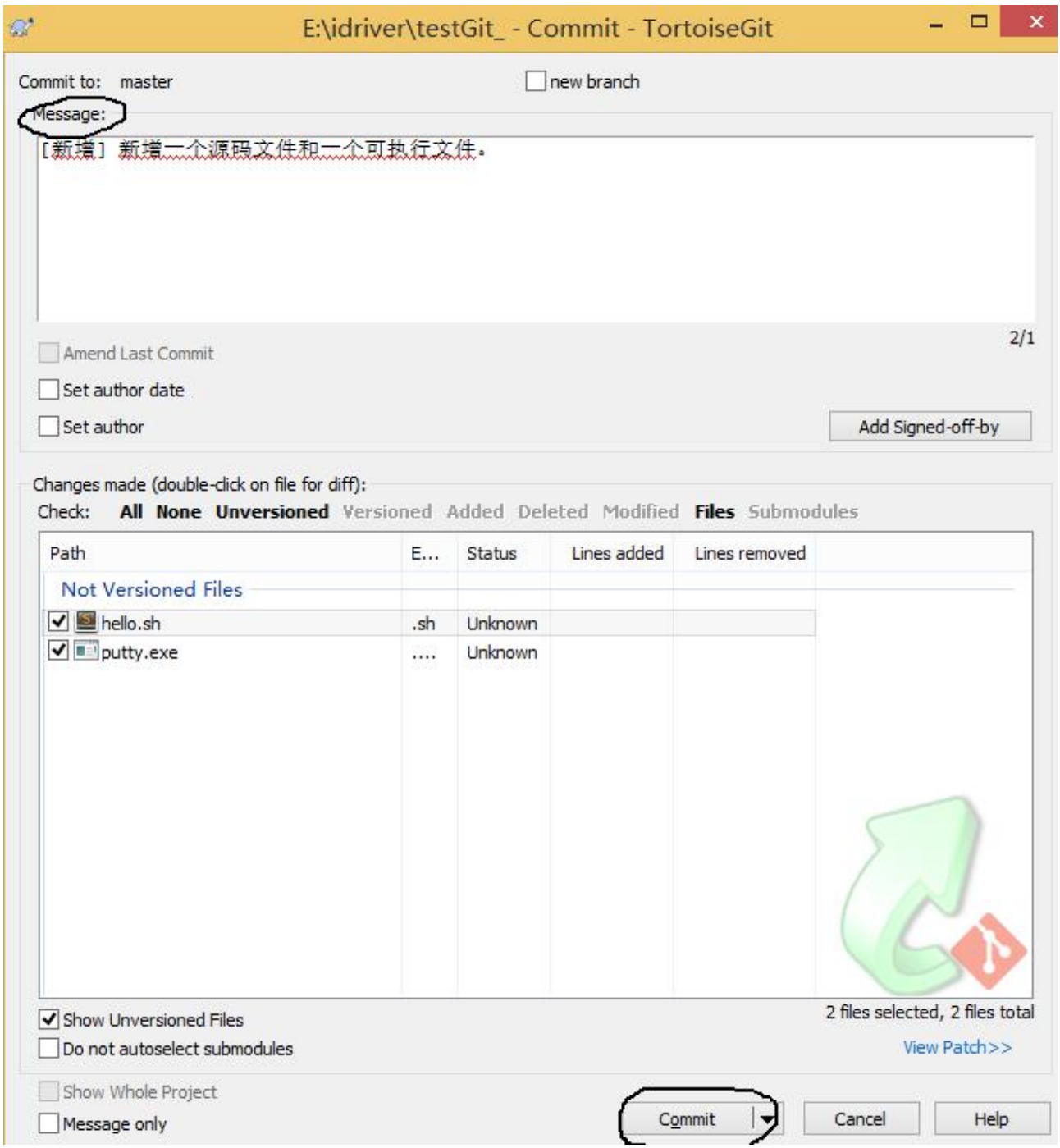
3、文件前面有绿色的√，表示该文件所有修改已被提交。

4、文件前面有黄色的！，表示提交冲突，自动合并修改（Merge）失败了。

此时需要手动合并修改。

5、提交修改和推送到 Github

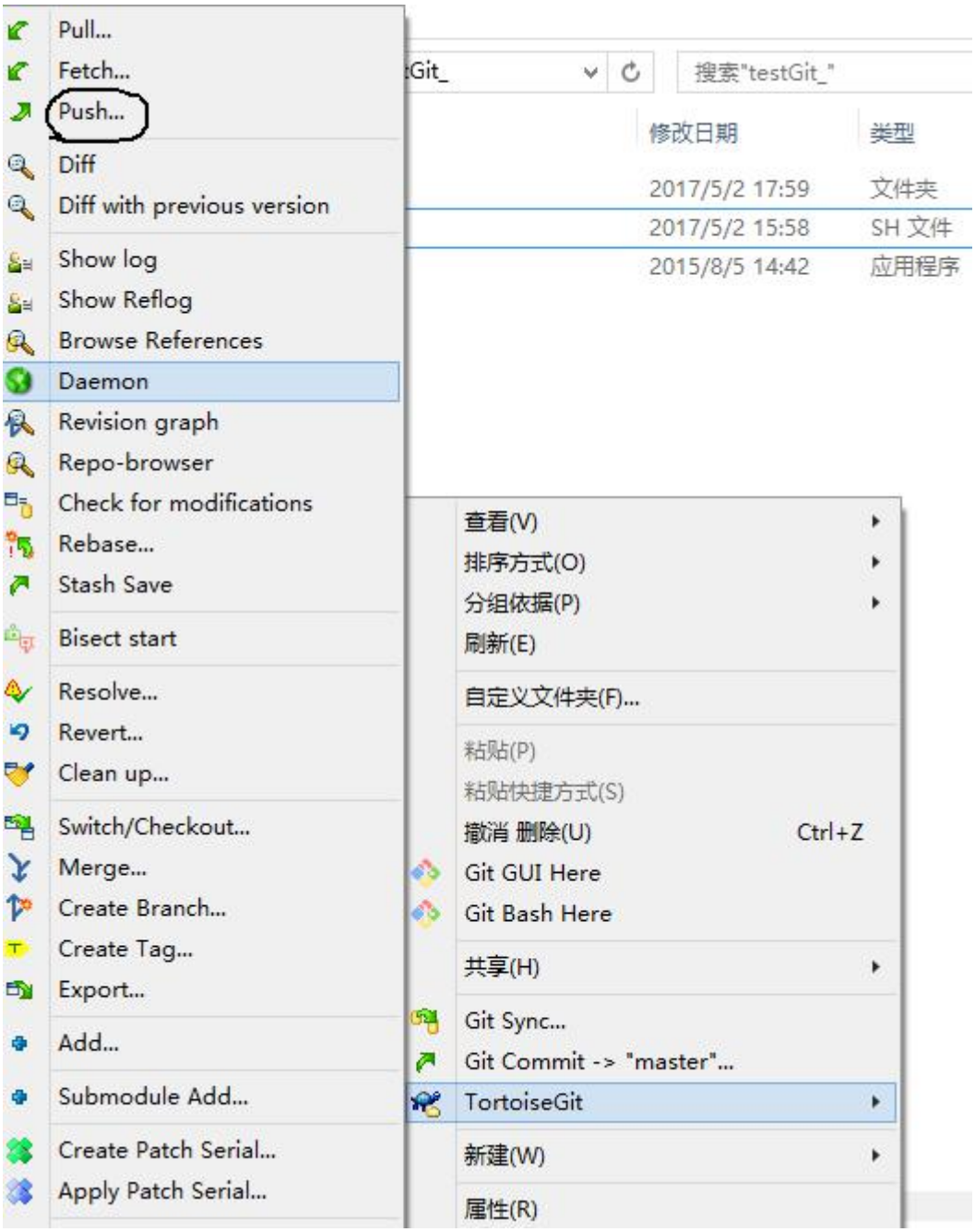
1、提交文件：填写修改记录，点击提交；

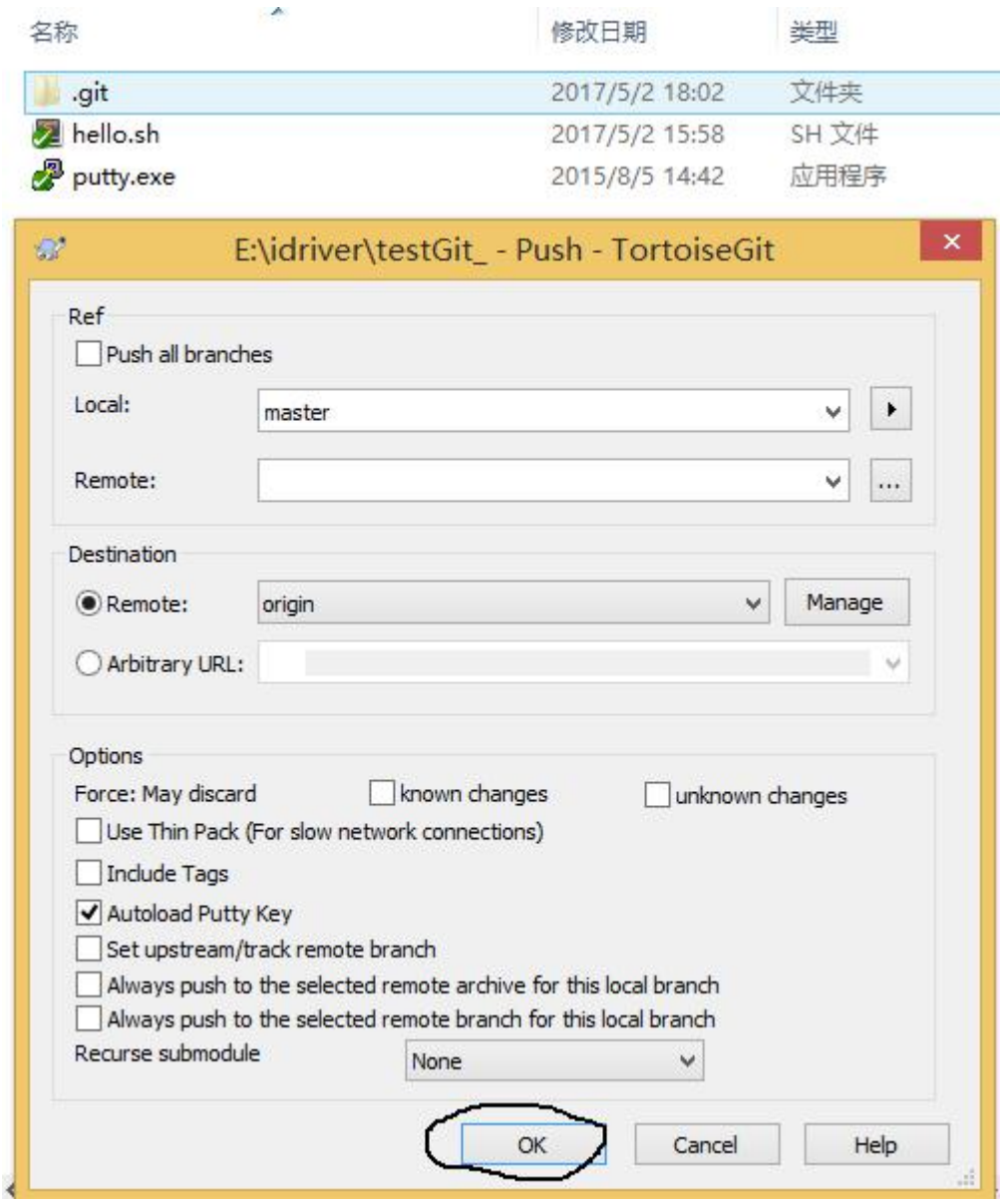


然后，！变成了√，表示修改已提交；

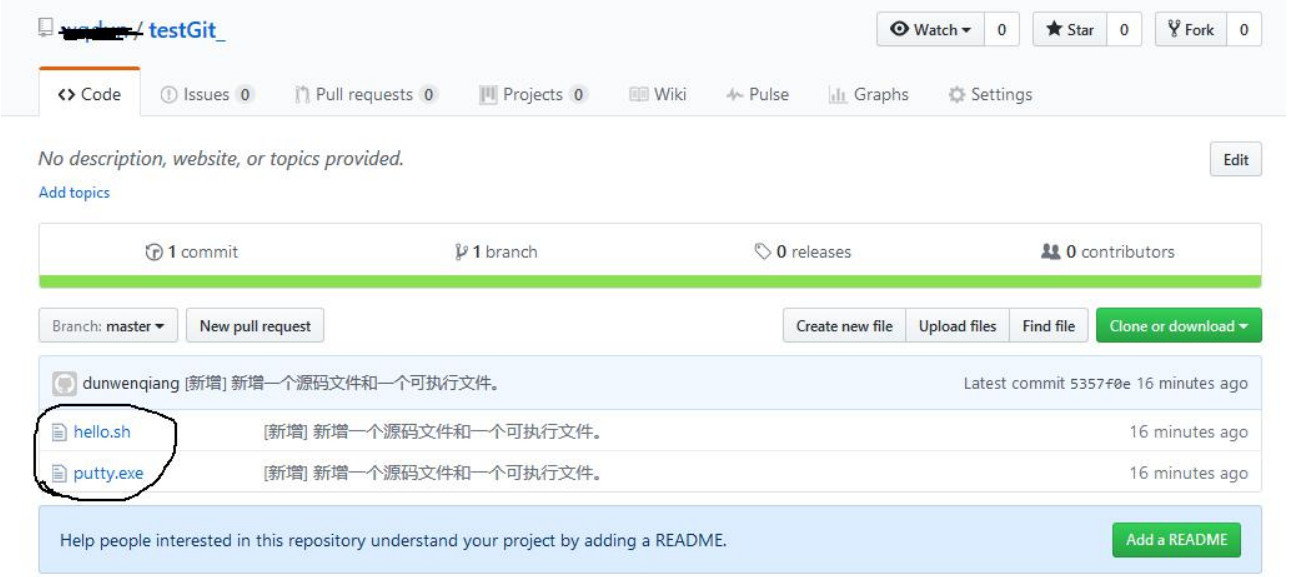
2、将文件推送到 Github：

右键 Git->Push



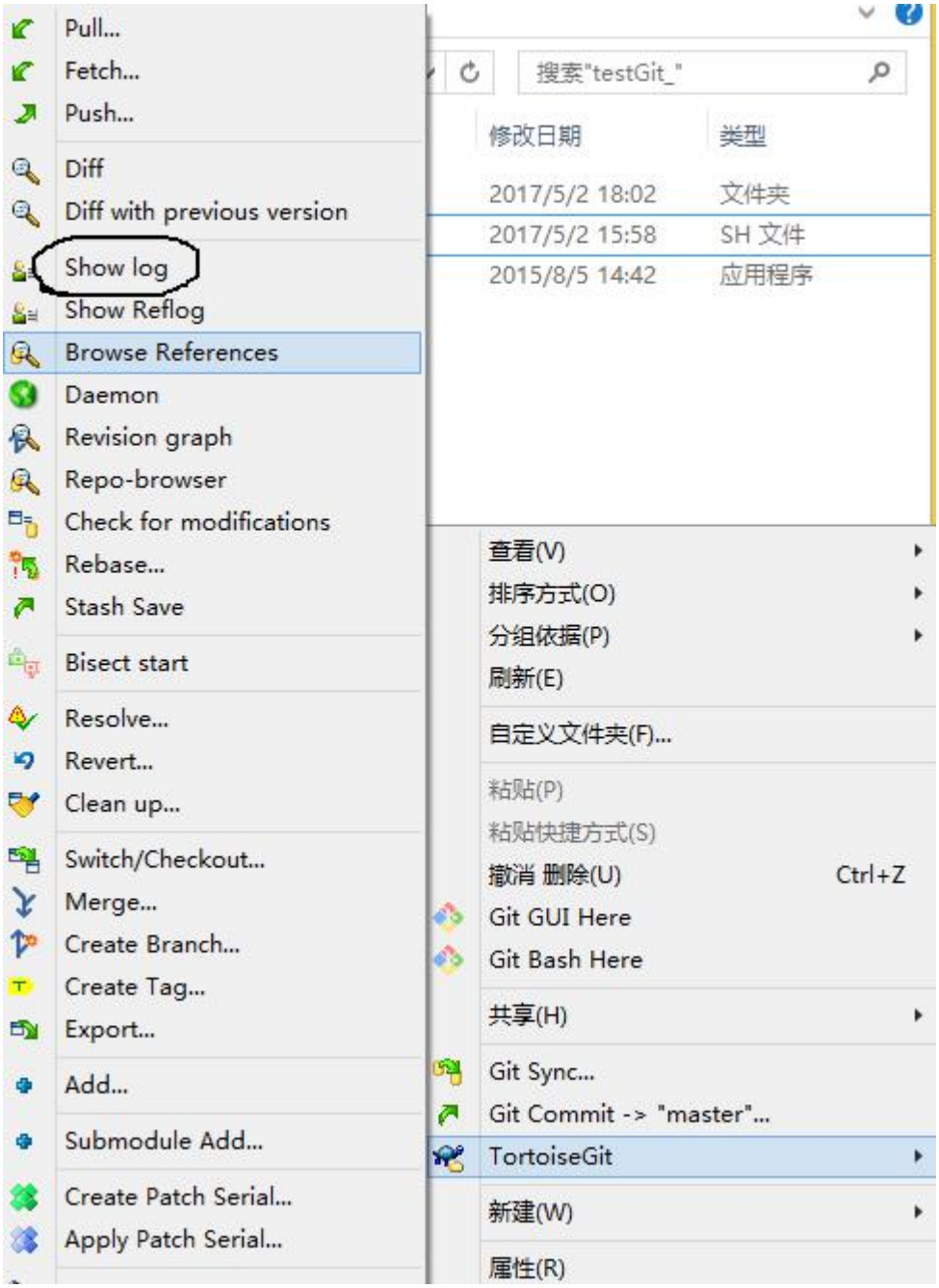


推送成功后，在 Github 上可以看到提交记录和更新的文件：

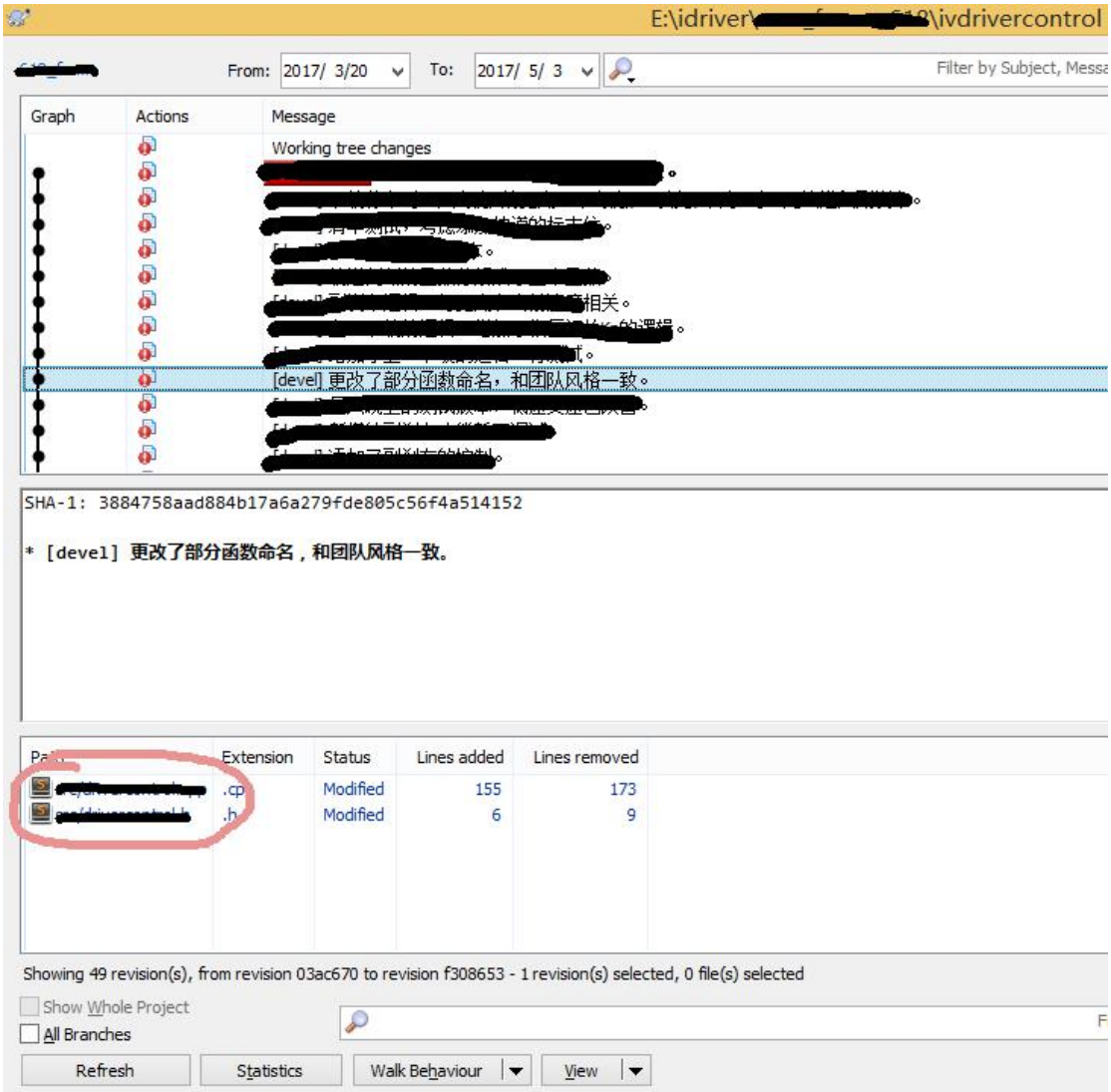


3、查看代码提交记录：

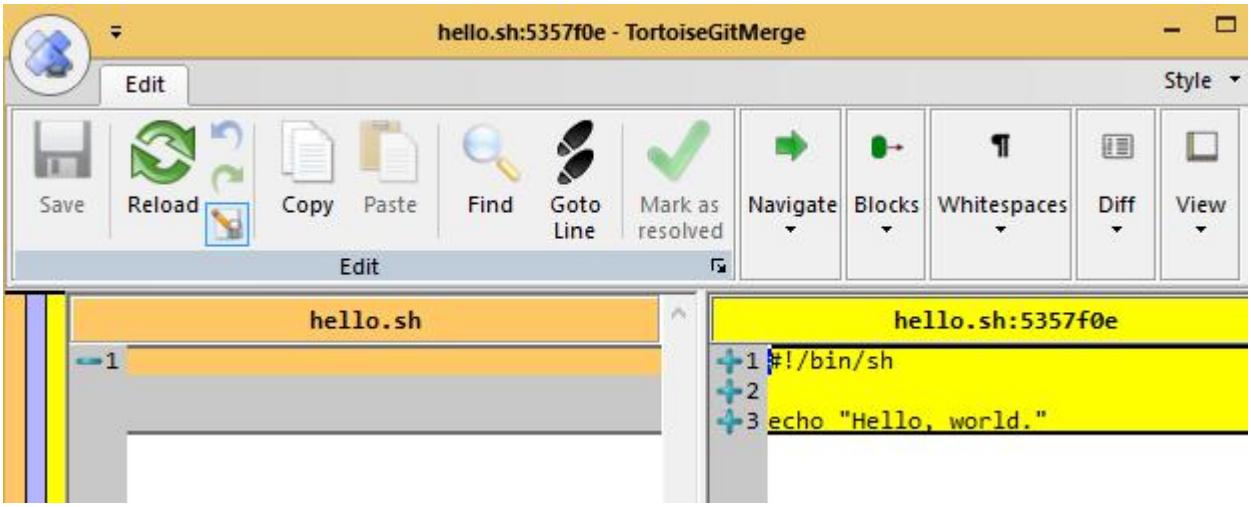
Git->Show log



然后下方显示该次提交涉及修改的代码，双击某个文件，显示该次提交修改的内容：



显示 **hello.sh** 文件新增了 **3** 行内容：



4、其它

1、Github 可以提交和推送各种文件，可作免费网盘用

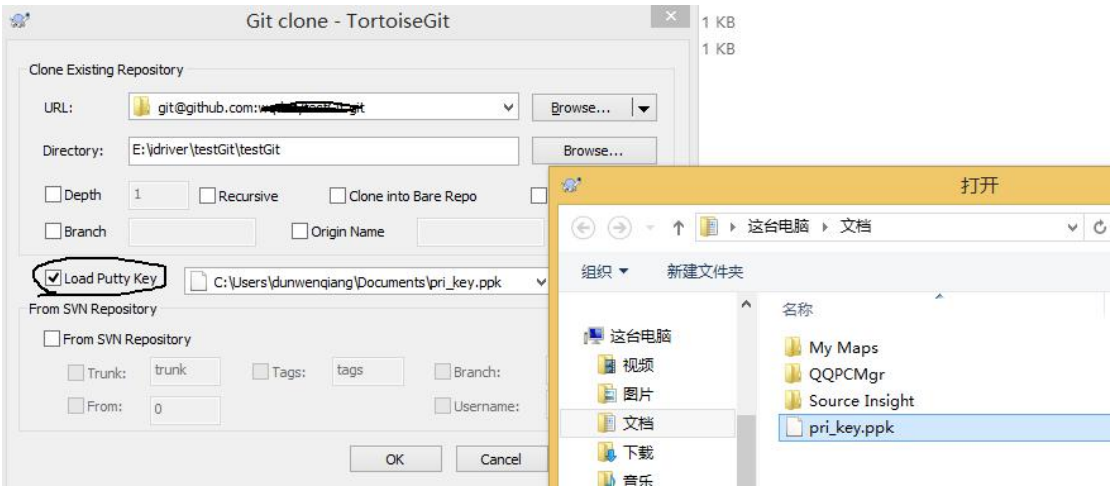
Github 可以提交和推送二进制文件（exe，doc 等），所以 Github 也可以作为网盘用，但是内容是公开的（对免费用户）。

但是对它们的比较功能不好用。

2、克隆 Github 的仓库到本地

到任意文件夹下，点击克隆，

载入第一步中生成的私钥文件：



克隆完成后，会在该文件夹生成仓库文件夹；

因为克隆时已经导入私钥，所以修改文件后，可以直接 **Push** 到 **Github** 了。

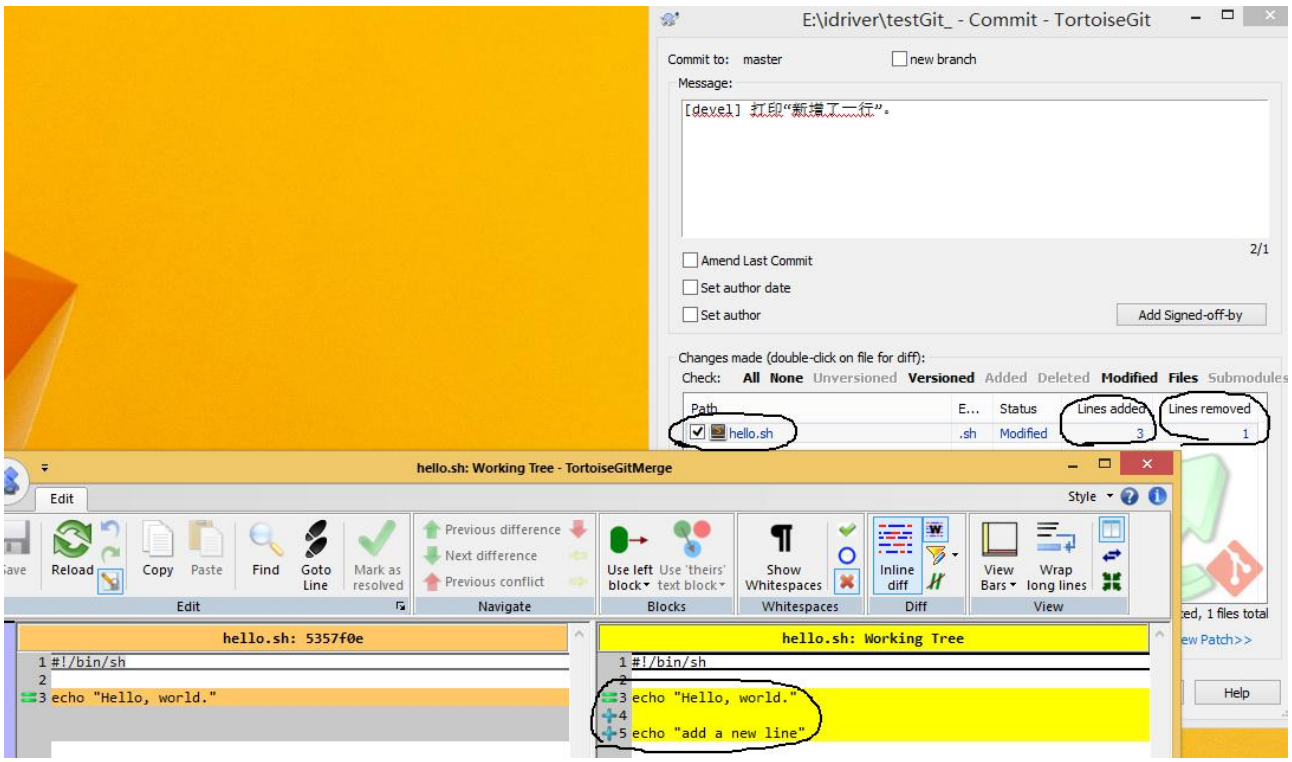
3、提交之前检查修改的内容

1、提交 **Commit** 不必保证编译通过，只作为本地的代码修改日志；但是 **Push** 时，要保证编译通过，不能影响别的模块测试；

小乌龟是 **Windows** 软件，不能编译 **Linux** 软件的问题：

可以尝试 **winscp + putty**，登录 **Linux** 服务器解决；

2、双击本次提交涉及的文件，会以 **Beyond Compare** 的方式，显示修改的内容：



在文件内容上右键，可以进行左<->右双向的复制。

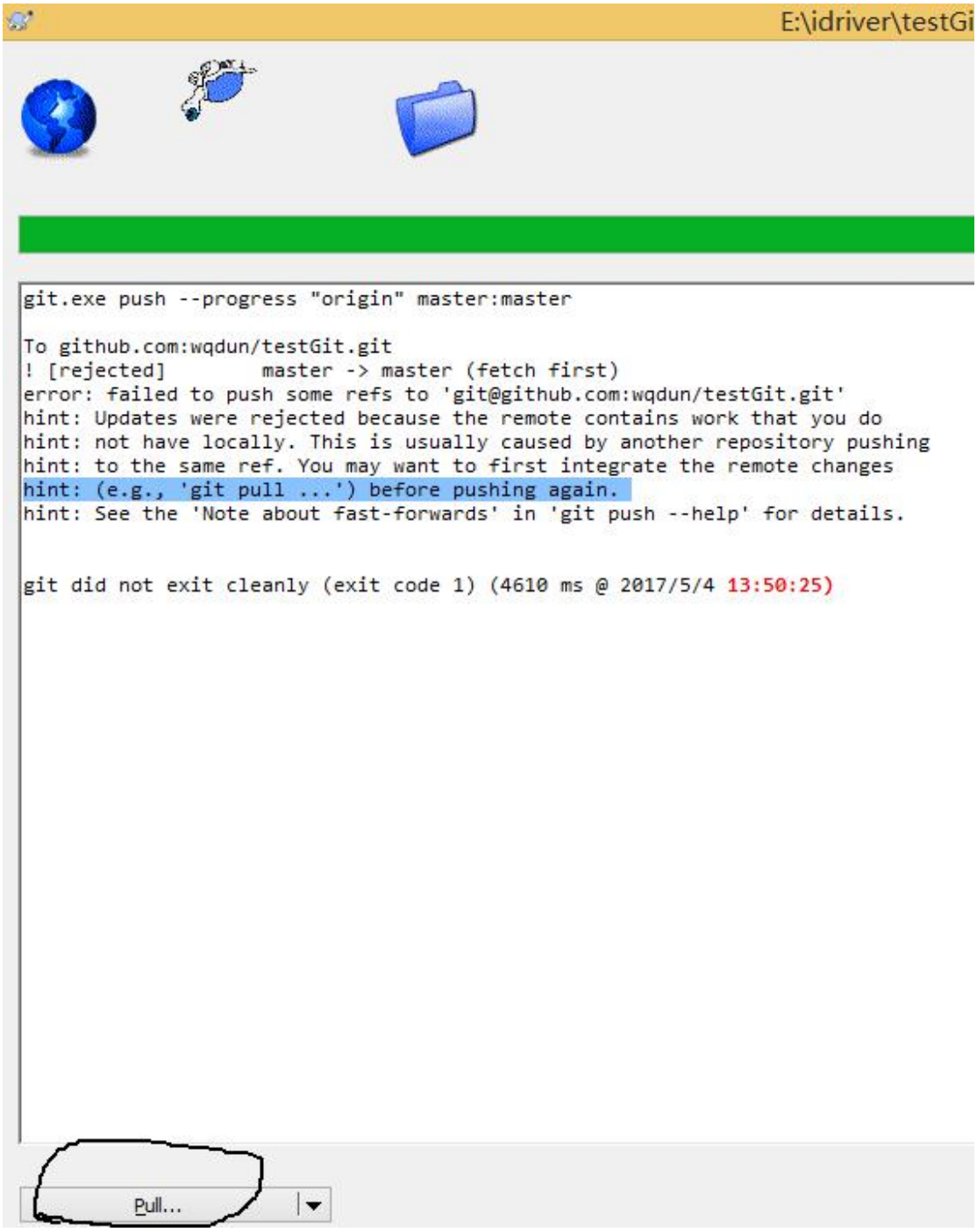
4、解决提交冲突

当远端仓库（此处即 Github）有人推送（Push）了修改，而且本地也要推送修改时：

此时不影响本地的提交（Commit 与远端仓库无关，是本地的代码修改记录），

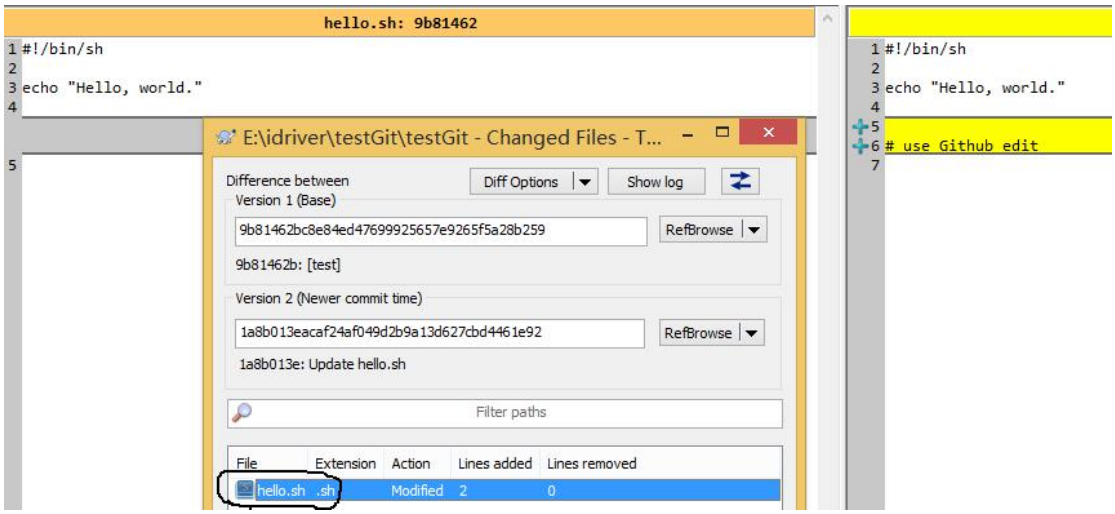
但是推送（Push）报错：远端有较新的修改，而本地没有；

同时小乌龟给出解决方案：先拉取（Pull）远端的修改到本地，再推送。

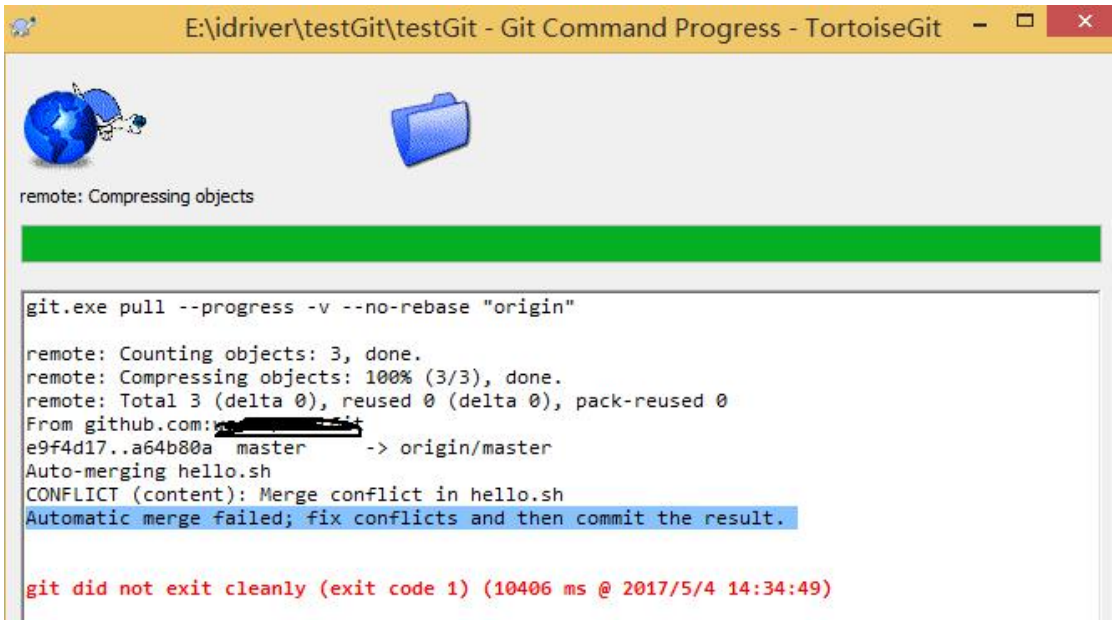


直接点击 Pull：

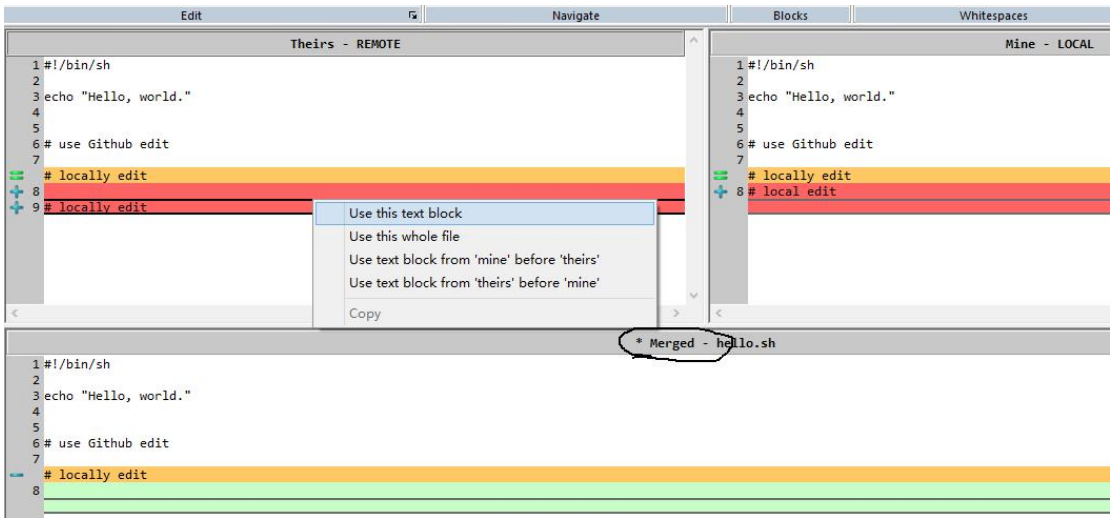
1、如果 Pull，自动合并成功后，小乌龟会提示使用 diff 工具查看拉取的修改内容（双击文件名）：



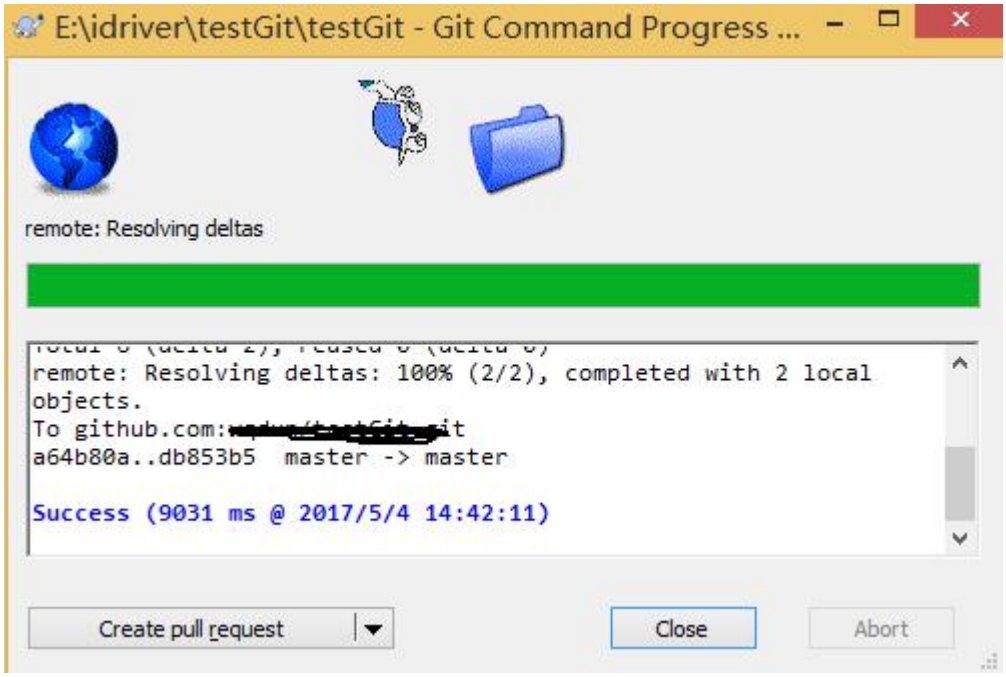
2、如果 Pull，自动合并失败，小乌龟会提示解决冲突后再提交：



此时，重新 **Commit**，打开冲突的文件，手动解决所有冲突：



重新 **Commit**，**Push**，推送成功：



5、删除本地仓库

删除.git 文件夹即可：

6、不要使用加密的文件

操作加密的文件时，不能保证以下操作成功：

1、创建本地仓库

2、提交

提交加密文件，报错：无法为文件建立索引，