

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 |
| | | | |
| | | | |
| | | | |

| | |
|-----------------------------------|----|
| 版本管理工具小乌龟 Git 的安装和使用..... | 1 |
| 修改记录..... | 1 |
| 1、软件的安装：..... | 3 |
| 1、安装 Git..... | 3 |
| 2、安装小乌龟 Git 的配置..... | 3 |
| 生成公钥/私钥..... | 5 |
| 2、添加公钥到 Github..... | 7 |
| 3、将本地文件纳入版本管理..... | 8 |
| 1、在 Github 新建代码仓库..... | 8 |
| 2、在计算机上创建代码仓库..... | 9 |
| 3、本地仓库连接 Github 仓库..... | 11 |
| 4、文件前标识符的含义..... | 13 |
| 5、提交修改和推送到 Github..... | 14 |
| 4、其它..... | 21 |
| 1、Github 可以提交和推送各种文件，可作免费网盘用..... | 21 |
| 2、克隆 Github 的仓库到本地..... | 21 |
| 3、提交之前检查修改的内容..... | 22 |
| 4、解决提交冲突..... | 23 |
| 5、删除本地仓库..... | 26 |
| 6、不要使用加密的文件..... | 27 |

1、软件的安装：

1、安装 Git

使用软件管理工具搜索 Git：

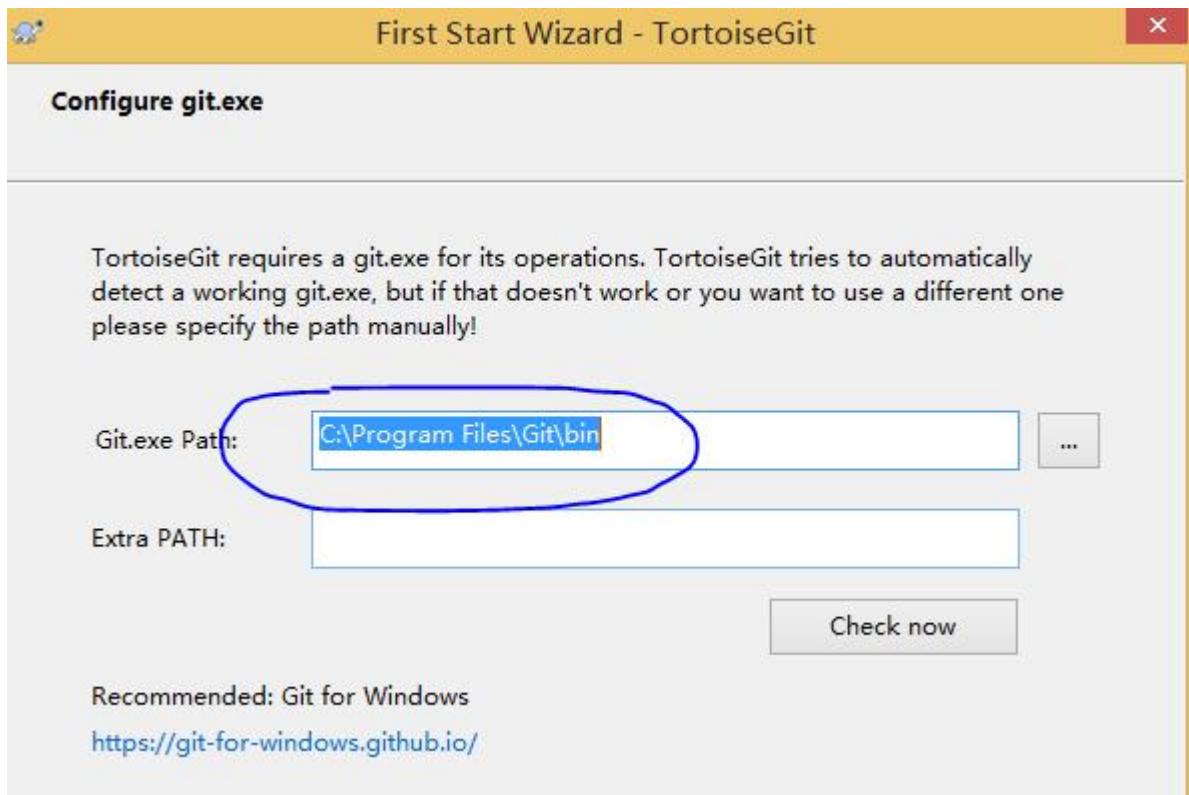


先安装 Git（也可以在 <https://git-for-windows.github.io/> 找到最新版），全部选择默认即可；

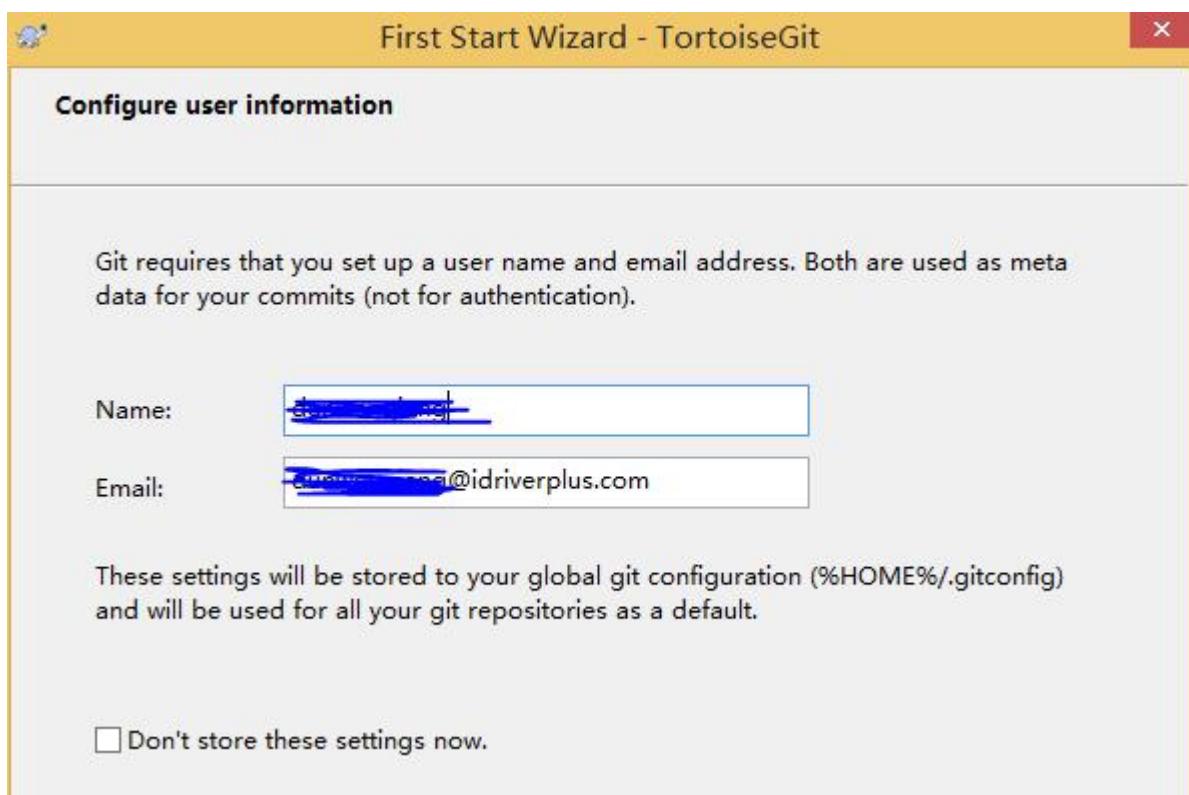
再安装小乌龟（也可以在 <https://tortoisegit.org/download/> 下载最新版）；

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：

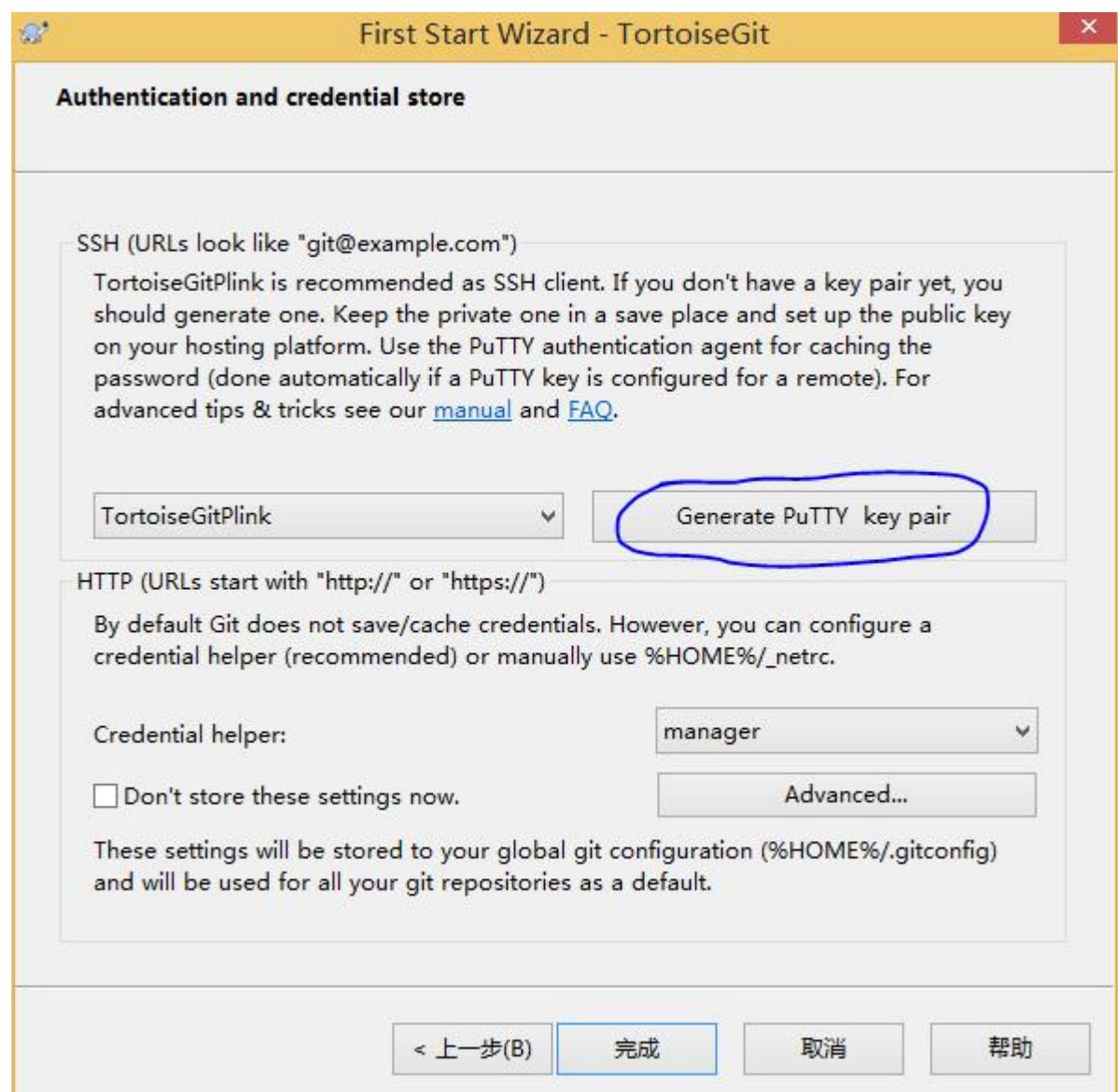


下一步，填写姓名和邮件地址，不是 Github 的注册邮箱，只作为提交时的记录；

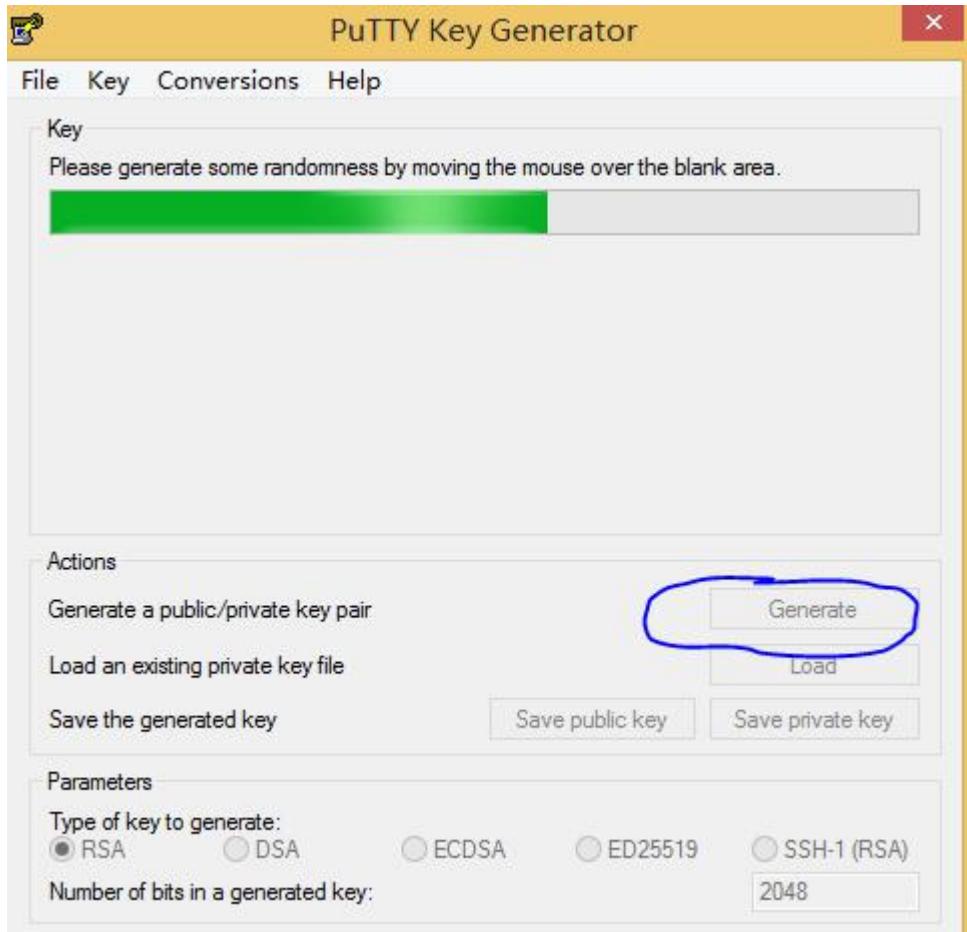


生成公钥/私钥

生成公、私钥对，用来将代码提交到 Github：

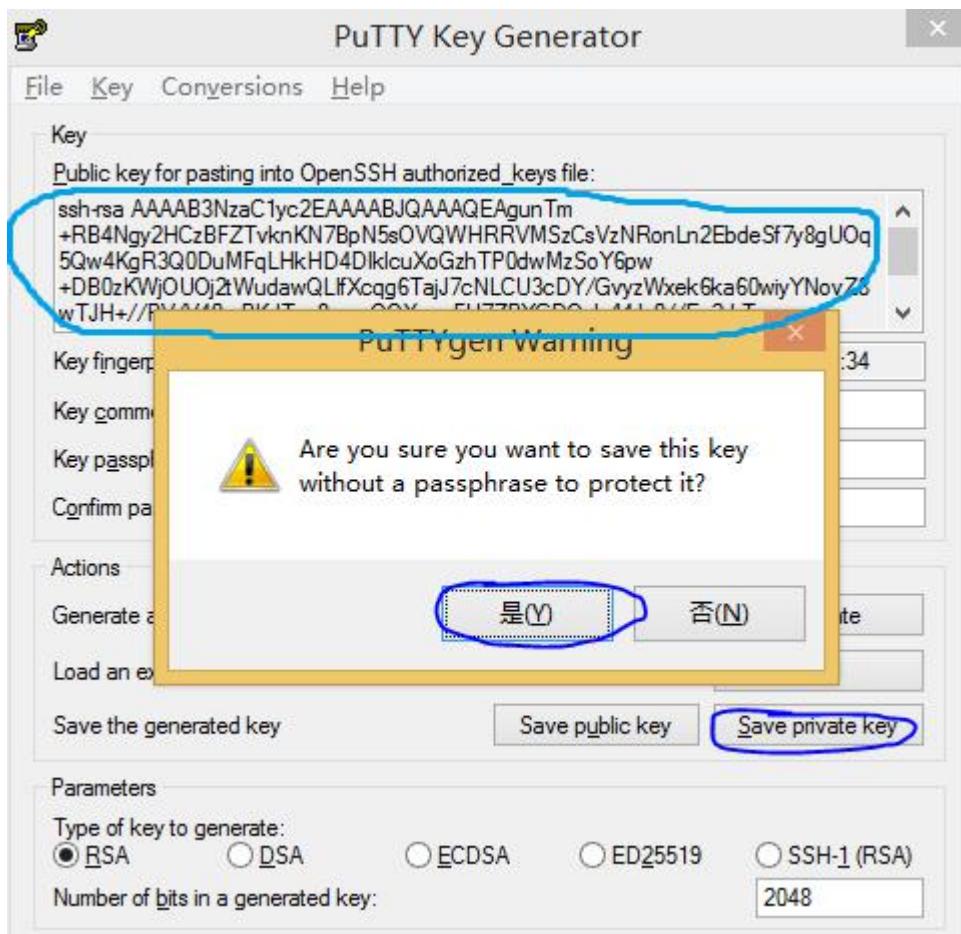


点击 Generate，并不断移动鼠标在空白处；



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，

A screenshot of the GitHub user settings page, specifically the "SSH keys" section. On the left, there's a sidebar with "Personal settings", "Profile", and "Account". On the right, the "SSH keys" list is shown with a green "New SSH key" button at the top right. A note below the list says, "This is a list of SSH keys associated with your account. Remove any keys that you do not recognize." The "New SSH key" button is circled in green.

名字任意；Key 中粘贴上一步中生成的公钥（共一行）；

Title

Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAQEAiMLQkR2HGVsJFj7eTQZM/6XJbtbjA3v192n1oiWUR3wB9Gqf1BomnqKB35PFVcJ4h5Epy/Y6rcML6/AQ+K1CKVTB37Og3SufolmcGpY1jdB9/jDGDK9RUwjqYY2va6l5PguuHY7nX+Vdnz/AOMNMIm9xcCw0SYIYO2RBtkDRQzaslmi7Zv3DR9q0AkW5ZO1EVt9OjztIdMx5f3FoQMgEEaCibqfygGYD9M9c8weC7dLpP+a/59DyC8aBo3wCKsvYt0Csz/gwS2FMkgwUarL1hY8hJGJ1s3sP9oBnPjRdgzgZTA4XjrOpnJKQoYgbLuKHuwczgjkFAJYKQ/Hxf5UKw==rsa-key-20170502
```

Add SSH key

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：

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name
/

Great repository names are short and memorable. Need inspiration? How about [glowing-winner](#).

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None

Create repository

进入该仓库，点击 SSH，复制后面的 SSH 地址（后面要用）

[/ testGit](#)

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Quick setup — if you've done this kind of thing before
 or git@github.com:[/testGit_.git](#)

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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：

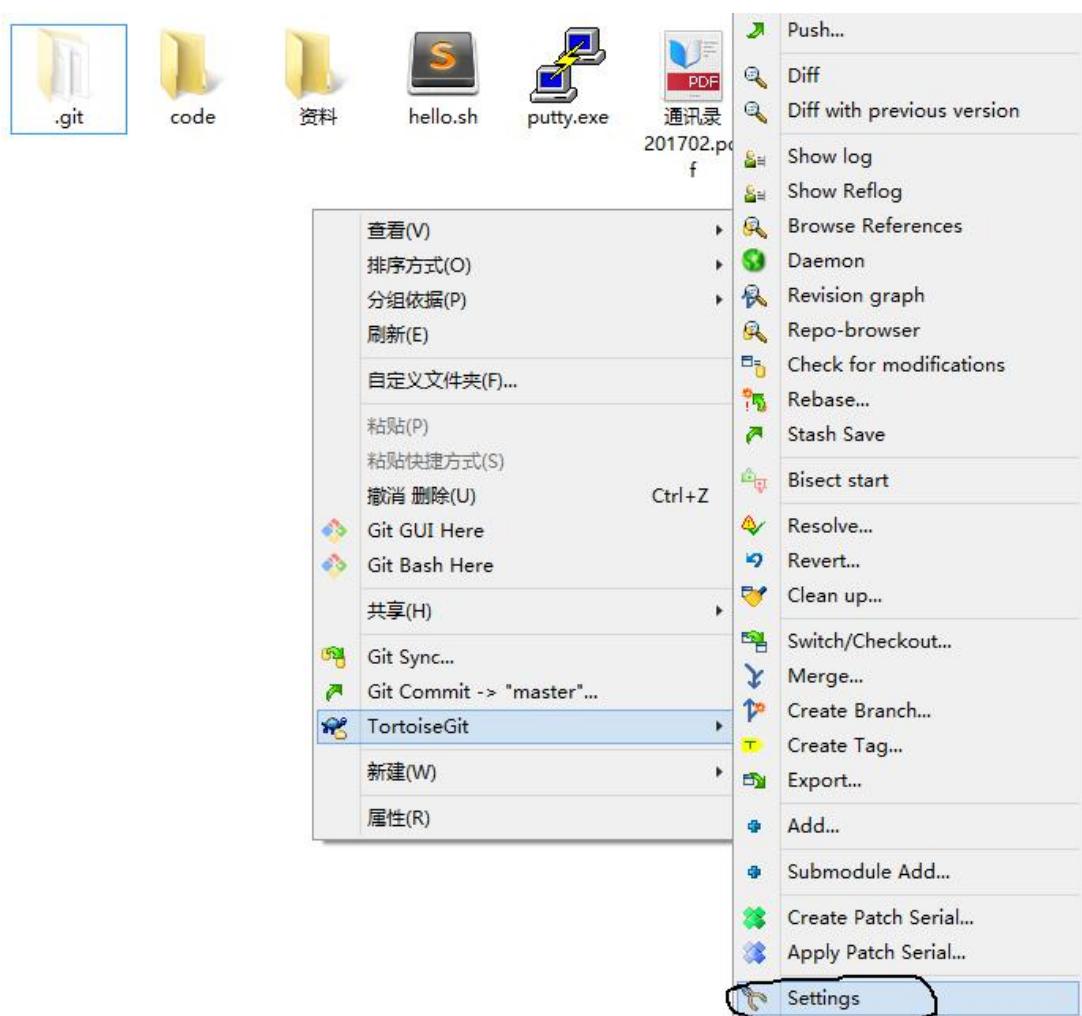


成功后，该文件夹下生成.git 文件夹：

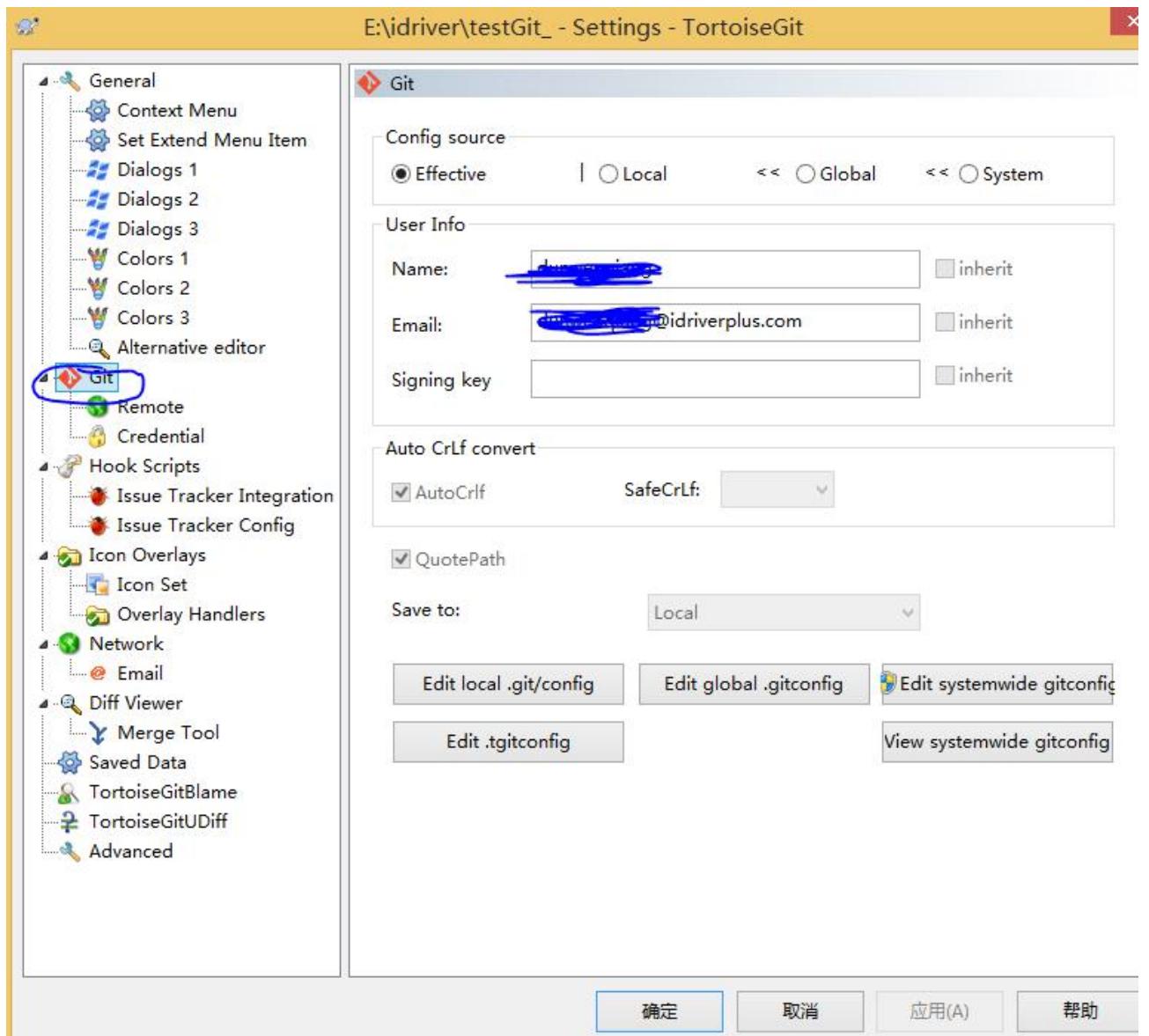


3、本地仓库连接 Github 仓库

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）：



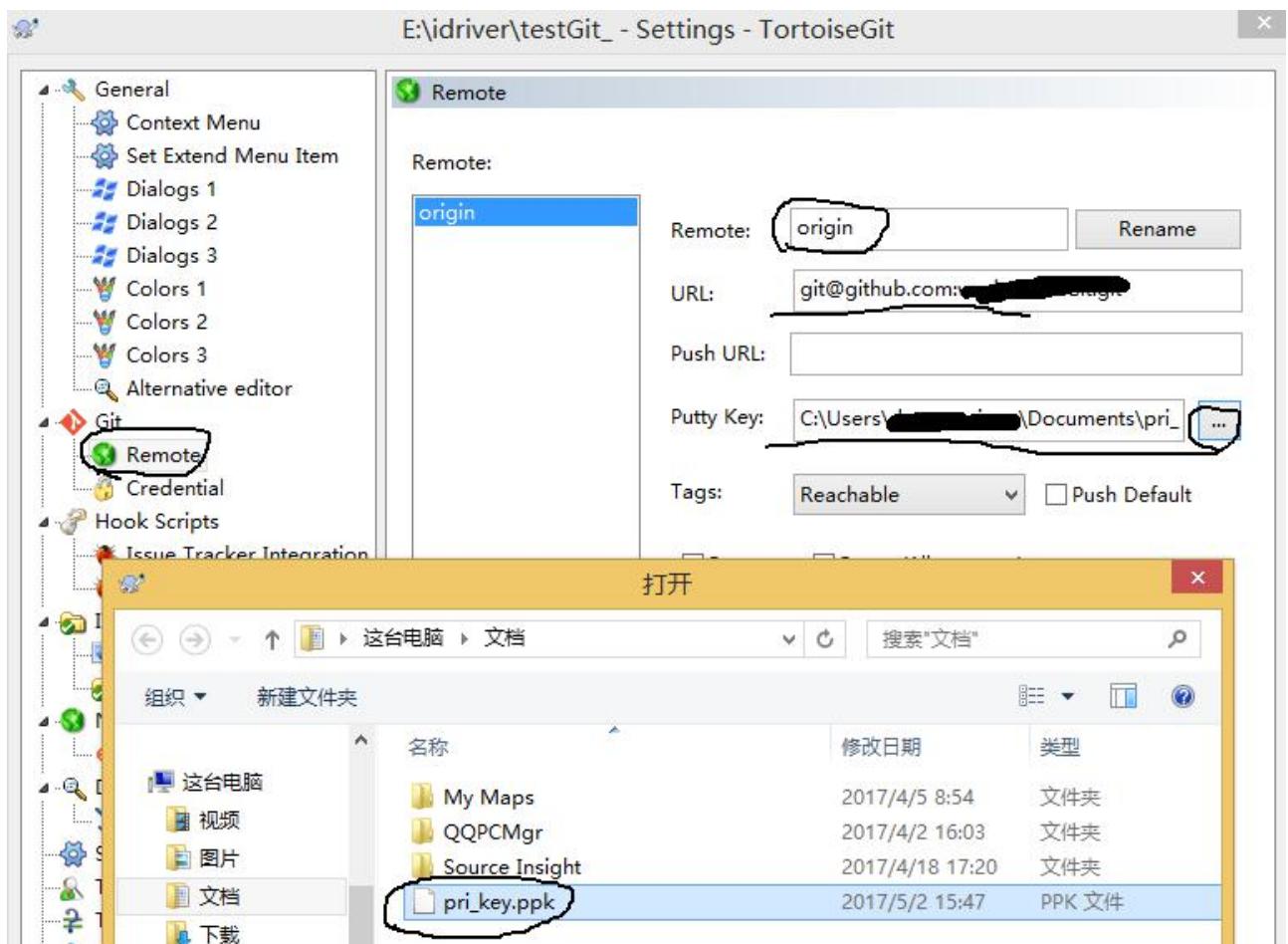
点击左侧 Remote :

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。

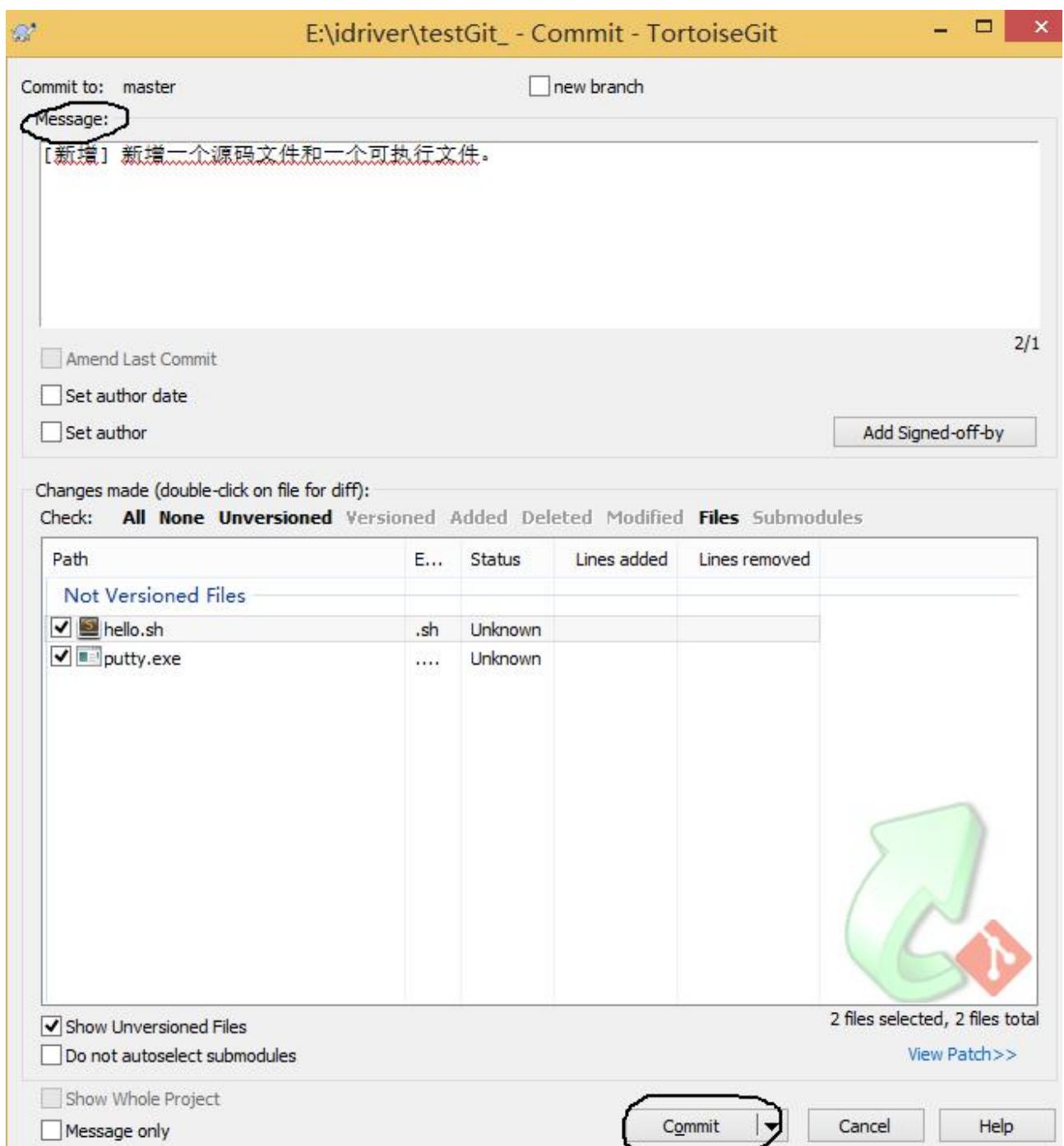
4、文件前面有黄色的 !，表示提交冲突，自动合并修改（Merge）失败了。

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；

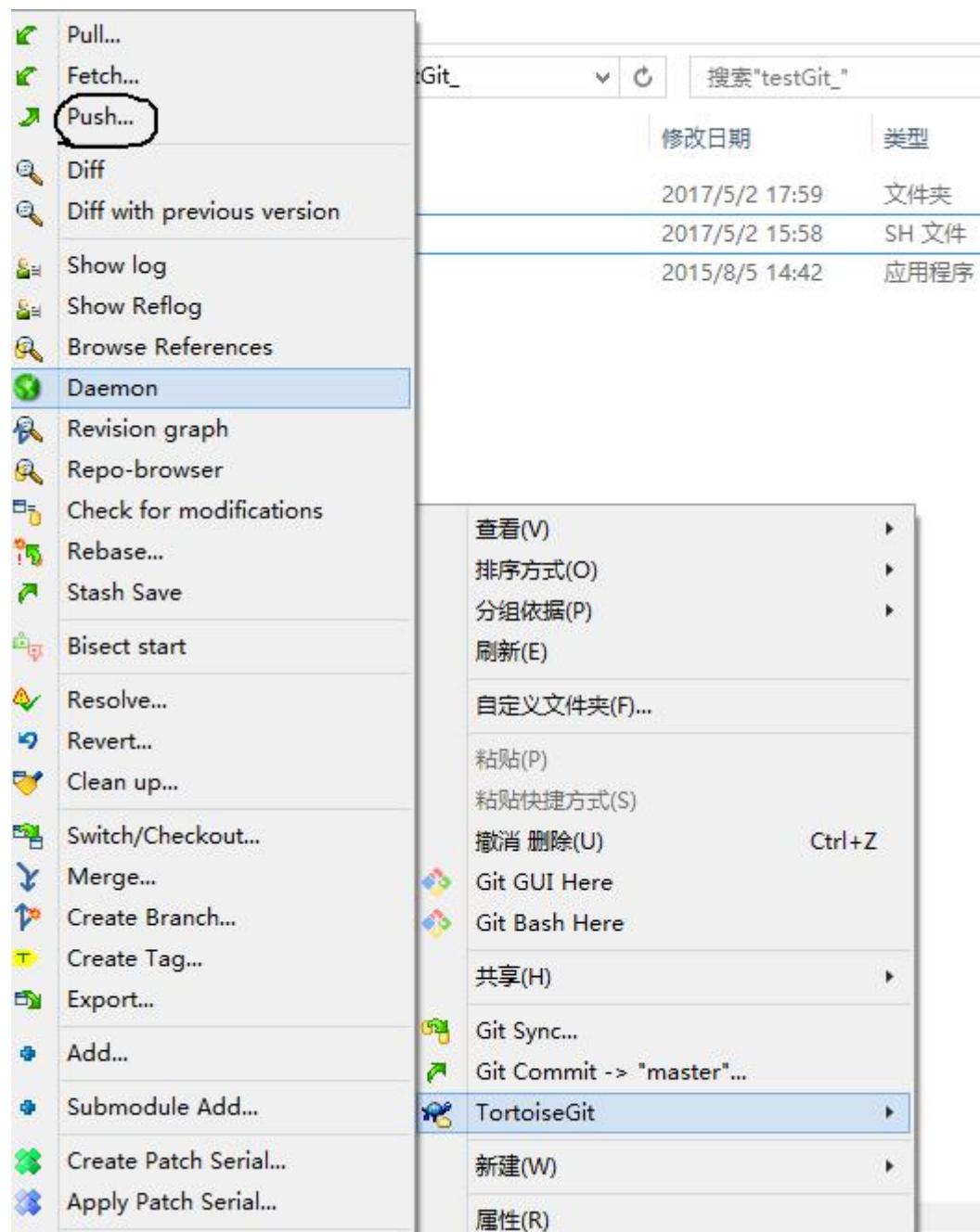


然后，! 变成了✓，表示修改已提交；

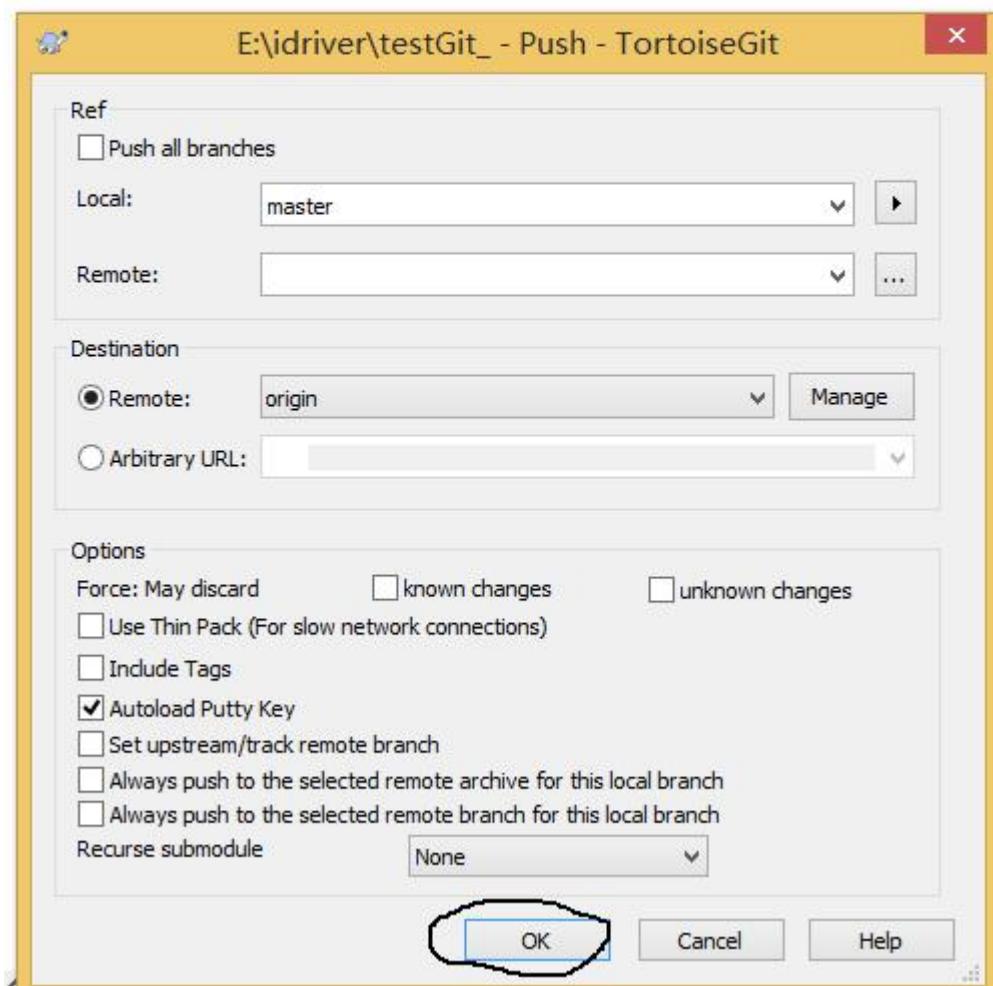
| 名称 | 修改日期 | 类型 |
|-----------|----------------|-------|
| .git | 2017/5/2 17:48 | 文件夹 |
| hello.sh | 2017/5/2 15:58 | SH 文件 |
| putty.exe | 2015/8/5 14:42 | 应用程序 |

2、将文件推送到 Github :

右键 Git->Push



| 名称 | 修改日期 | 类型 |
|-----------|----------------|-------|
| .git | 2017/5/2 18:02 | 文件夹 |
| hello.sh | 2017/5/2 15:58 | SH 文件 |
| putty.exe | 2015/8/5 14:42 | 应用程序 |

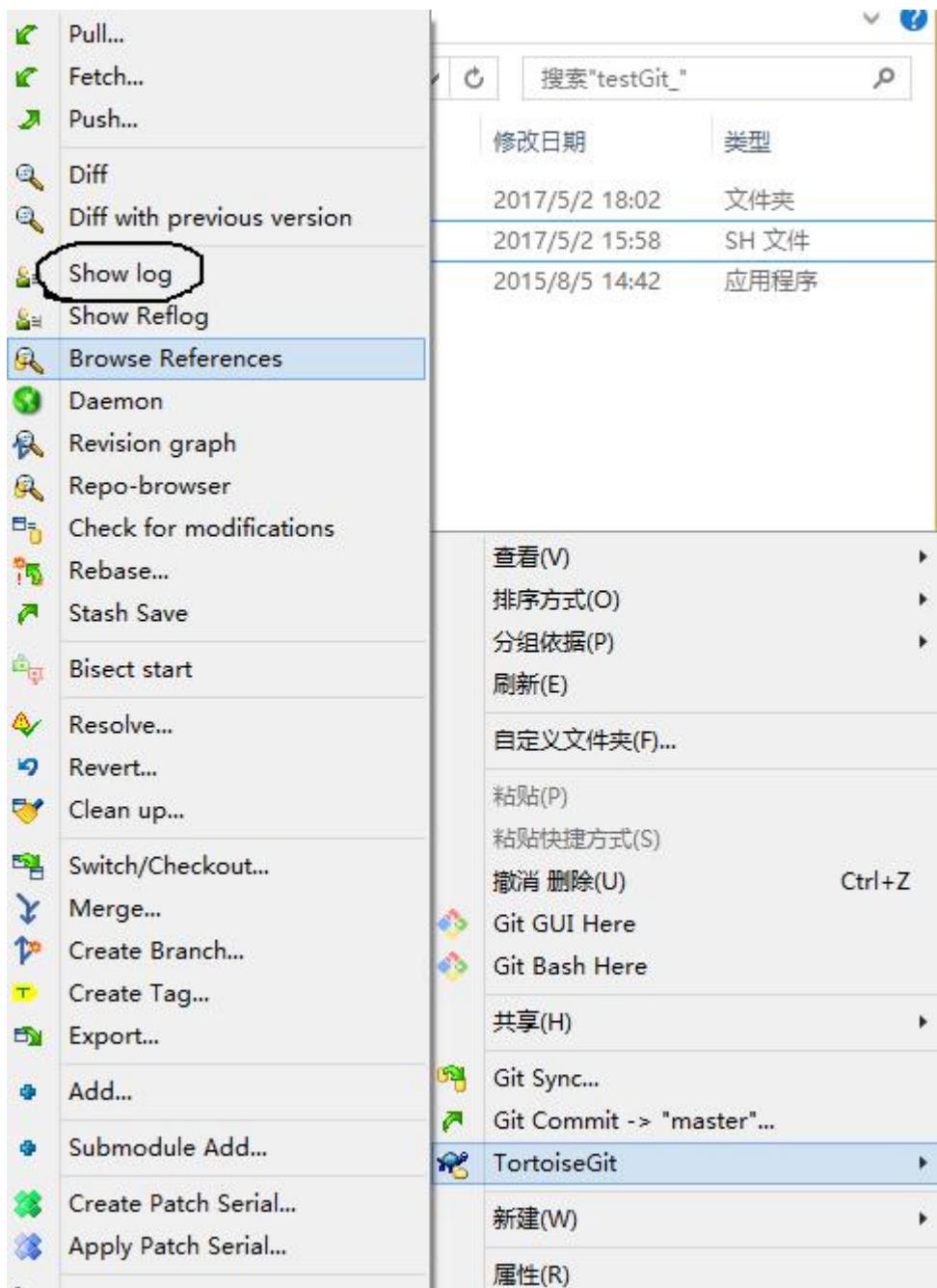


推送成功后，在 Github 上可以看到提交记录和更新的文件：

The screenshot shows a GitHub repository page for a user named 'dunwenqiang'. The repository name is 'testGit_'. The top navigation bar includes links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. There are also buttons for 'Watch', 'Star', 'Fork', and 'Edit'. Below the navigation, a message says 'No description, website, or topics provided.' with a 'Add topics' link. A green bar displays summary statistics: '1 commit', '1 branch', '0 releases', and '0 contributors'. A dropdown menu shows 'Branch: master'. Buttons for 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download' are available. The commit history lists two commits by 'dunwenqiang': one for 'hello.sh' added 16 minutes ago and one for 'putty.exe' added 16 minutes ago. Both commits include the message '[新增] 新增一个源码文件和一个可执行文件.' (Added a source code file and an executable file.). At the bottom, there's a note to 'Help people interested in this repository understand your project by adding a README.' with a 'Add a README' button.

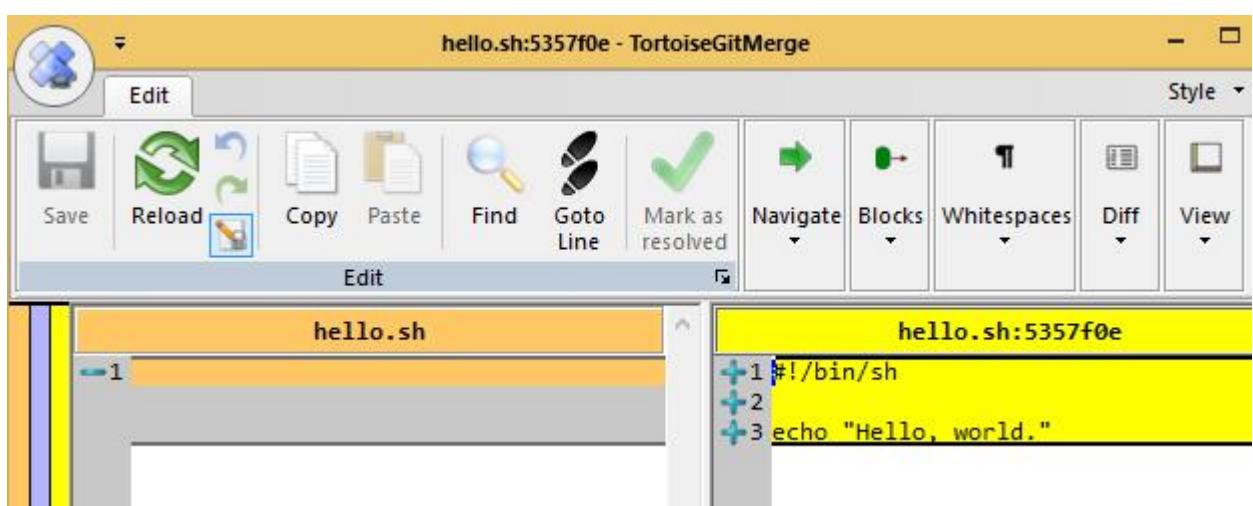
3、查看代码提交记录：

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

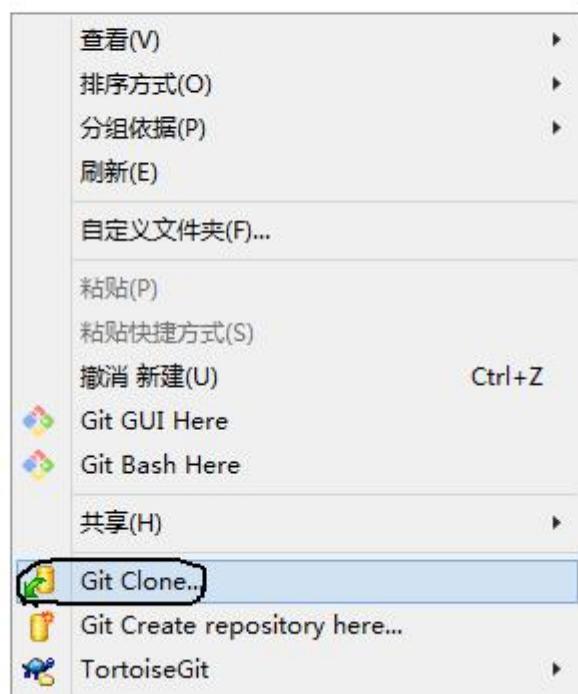
1、Github 可以提交和推送各种文件，可作免费网盘用

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）。

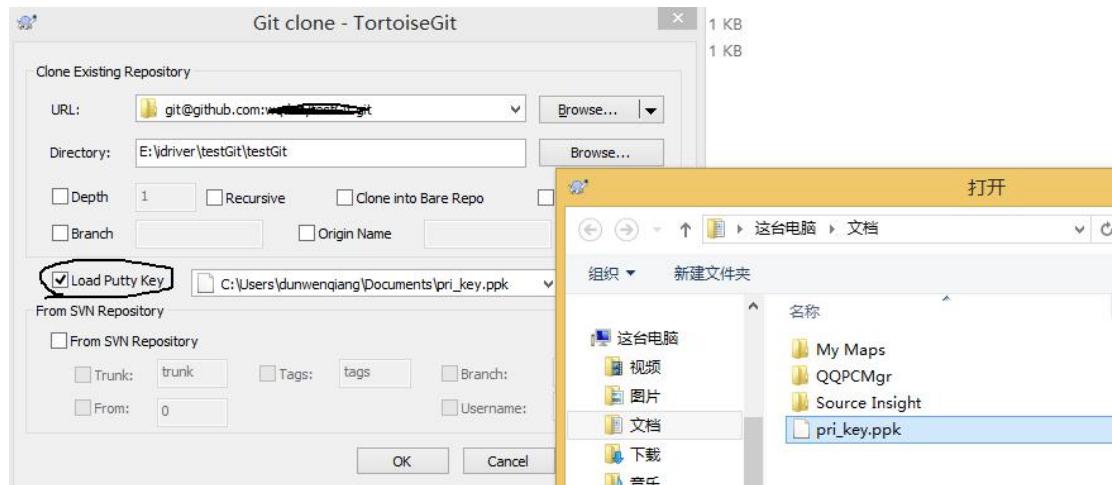
但是对它们的比较功能不好用。

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

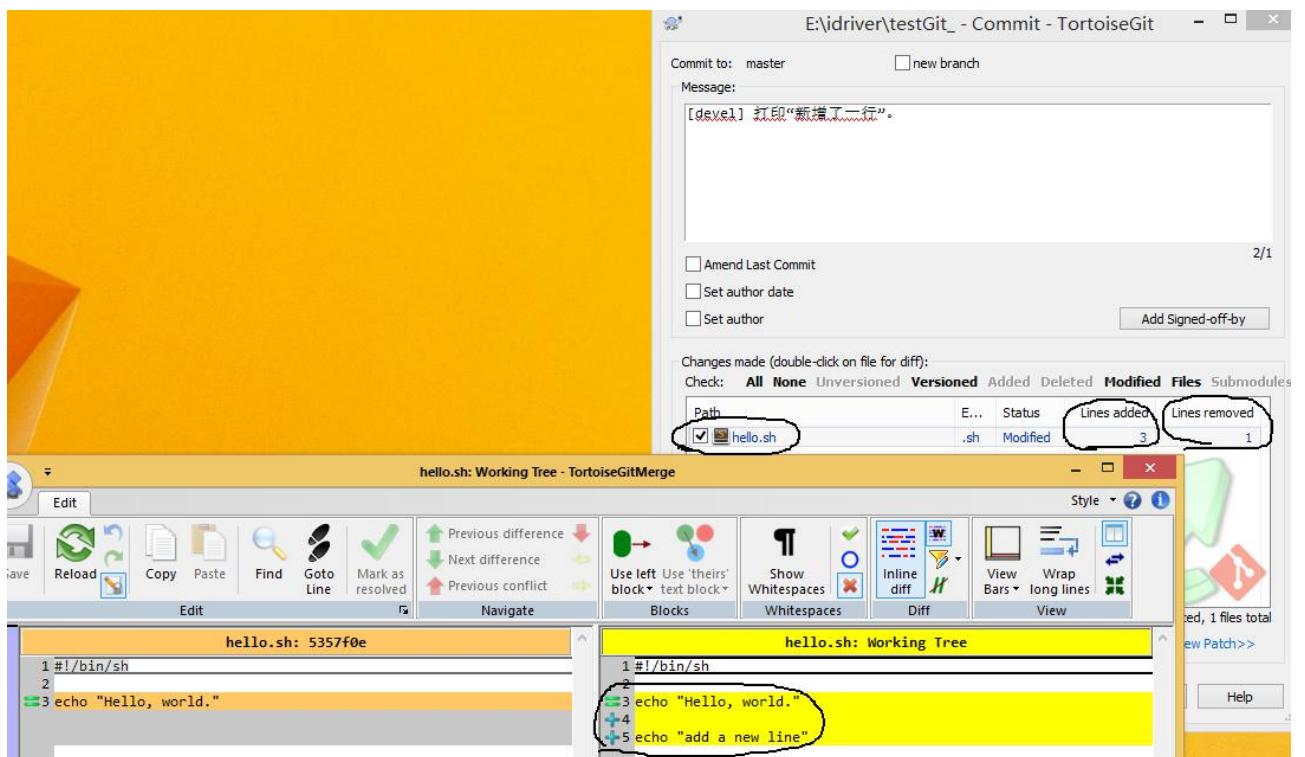
1、提交 Commit 不必保证编译通过，只作为本地的代码修改日志；但是 Push 时，要保证编译通过，不能影响别的模块测试；

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。

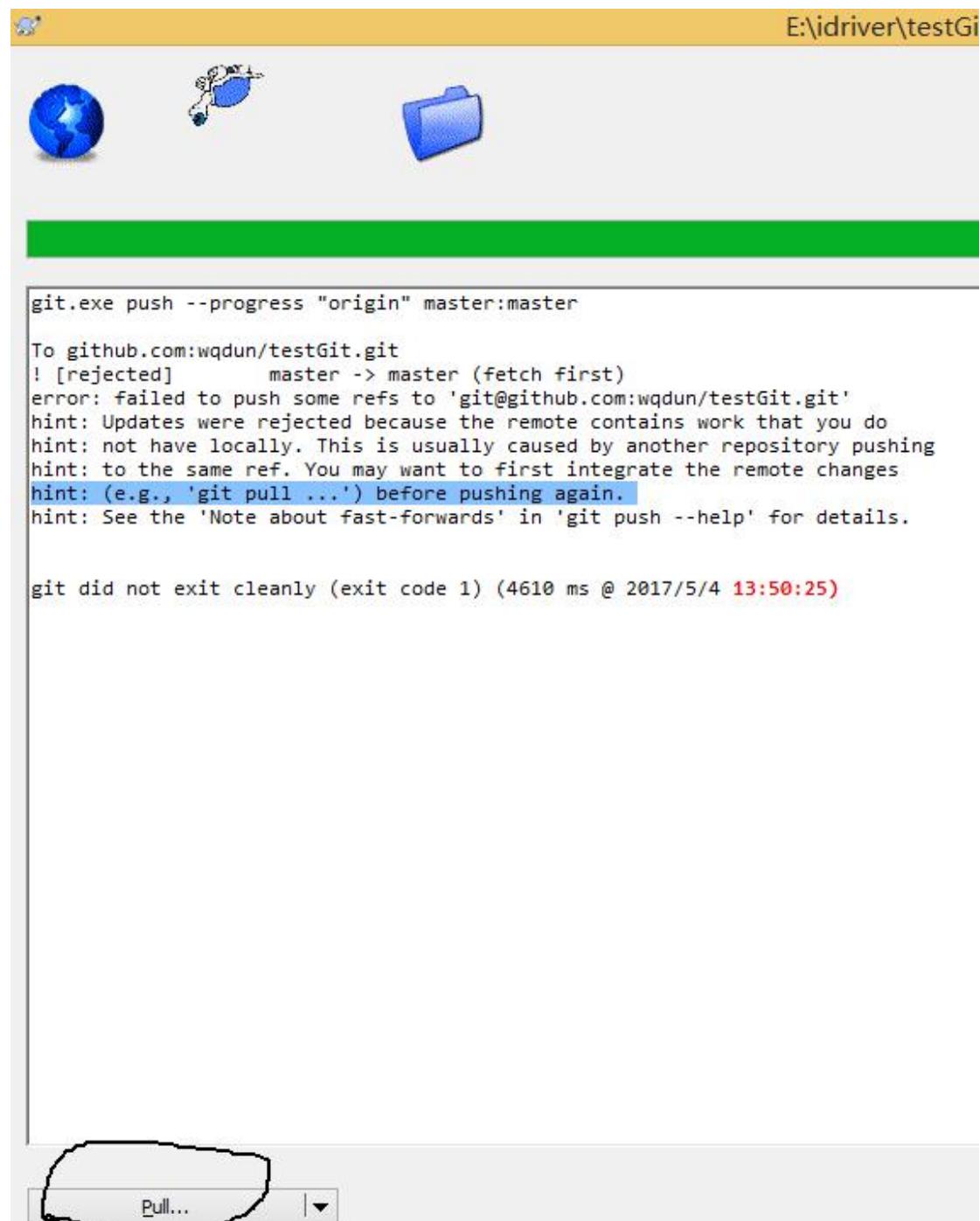
4、解决提交冲突

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。



The screenshot shows the GitHub desktop application interface. At the top, there's a toolbar with icons for committing, pushing, pulling, and creating branches. The main area has a title bar "E:\idriver\testGi". Below the title bar is a toolbar with icons for globe, user, and folder. A large green progress bar is visible. The main content area is a terminal window displaying the following text:

```
git.exe push --progress "origin" master:master
To github.com:wqdun/testGit.git
! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'git@github.com:wqdun/testGit.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

git did not exit cleanly (exit code 1) (4610 ms @ 2017/5/4 13:50:25)
```

At the bottom of the terminal window, there's a button labeled "Pull..." with a dropdown arrow next to it.

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）：

| File | Action | Lines added | Lines removed |
|----------|----------|-------------|---------------|
| hello.sh | Modified | 2 | 0 |

2、如果 Pull，自动合并失败，小乌龟会提示解决冲突后再提交：

```
git.exe pull --progress -v --no-rebase "origin"
remote: Compressing objects...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
From github.com:[REDACTED]
 * [new branch] master      -> origin/master
Auto-merging hello.sh
CONFLICT (content): Merge conflict in hello.sh
Automatic merge failed; fix conflicts and then commit the result.

git did not exit cleanly (exit code 1) (10406 ms @ 2017/5/4 14:34:49)
```

此时，重新 Commit，打开冲突的文件，手动解决所有冲突：

```
1#!/bin/sh
2
3echo "Hello, world."
4
5
6# use Github edit
7
8# locally edit
9# locally edit
```

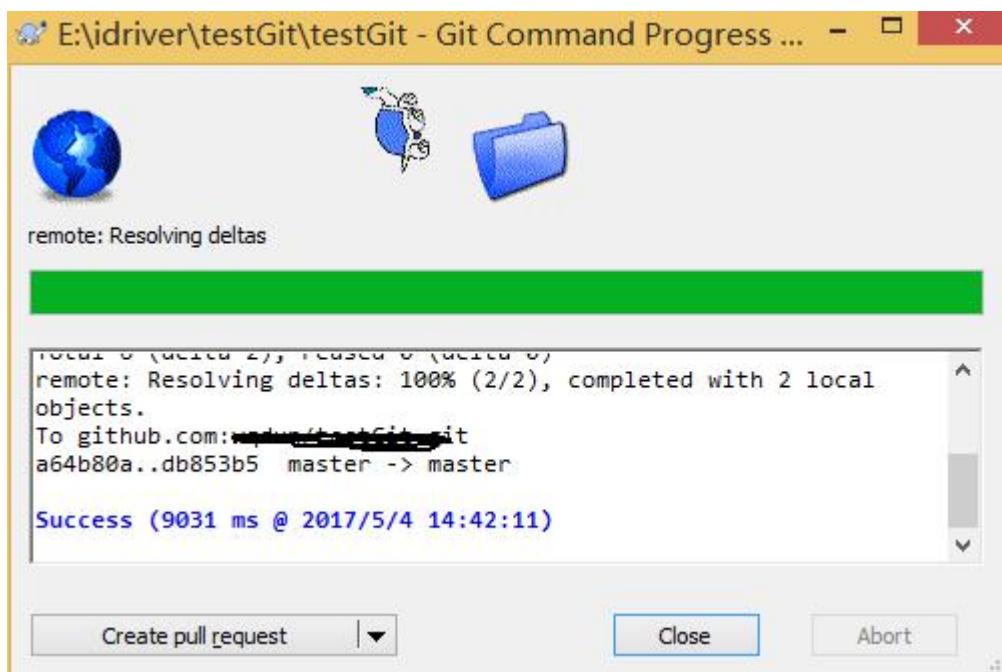
Use this text block
Use this whole file
Use text block from 'mine' before 'theirs'
Use text block from 'theirs' before 'mine'

Copy

* Merged - hello.sh

```
1#!/bin/sh
2
3echo "Hello, world."
4
5
6# use Github edit
7
8# locally edit
```

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，

The screenshot shows a command-line interface window titled "E:\idriver\PID_speed&dis - Git Command Progress...". The window contains a green progress bar at the top. Below it is a terminal window displaying the following error message:

```
error: short read: No such file or directory
error: drivercontrol.cpp: failed to insert into database
error: unable to index file drivercontrol.cpp
fatal: adding files failed

git did not exit cleanly (exit code 128)
```

At the bottom of the window are two buttons: "Close" and "Abort".