

Git

1.简介

版本控制的重要性

没有版本控制系统的话，代码可能被别人或自己不小心覆盖或遗失、也不知道是谁因为什么原因改了这段代码、也没办法可以复原回前几天的修改。有了版本控制系统，开发人员只要将每次程式码的变更都纪录（Commit）起来，并且透过版本控制系统中进行更新。

版本控制的目的

1. 将所有东西都放进版本控制系统

包括源代码、测试程序、文件、设定档、各种自动化脚本等。

除了新成员可以很容易拉出最新的版本马上开始工作之外，我们也希望在测试环境、正式环境中，也可以随时更新到我们所指定的版本，因此将所有变更的纪录保存起来，包括数据库的变更和服务器版本和环境的配置

2. 频繁且适当大小的递交

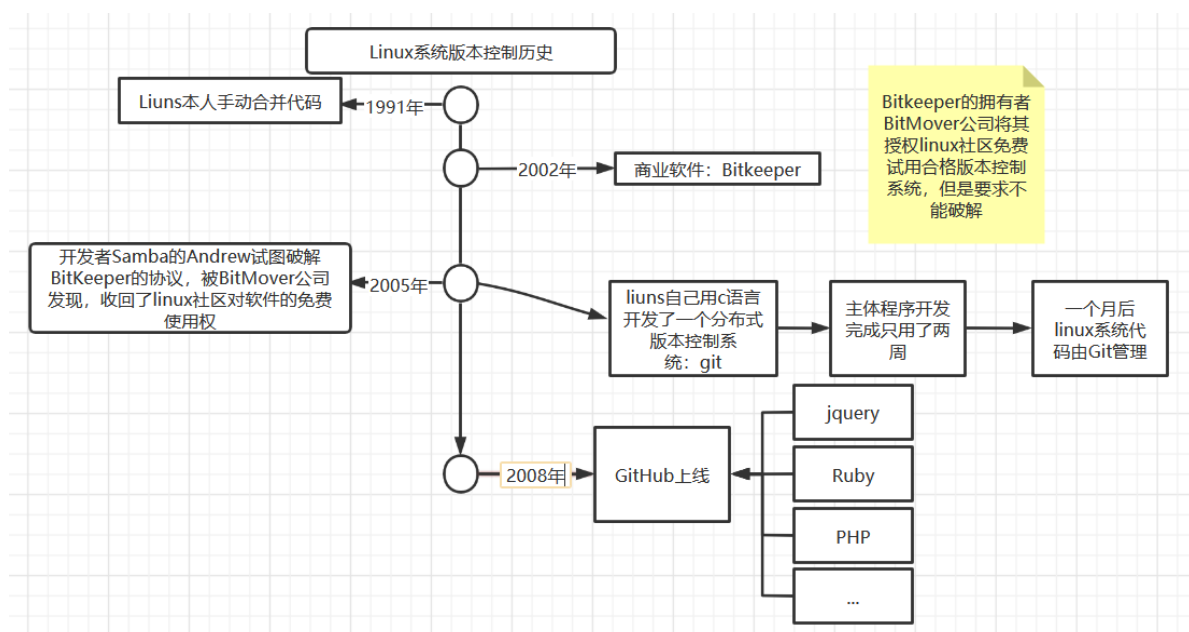
频繁的递交可以帮助团队开发进度的透明化，减少多人开发时的代码冲突。

我们可以非常轻易的做代码的复原或移植，假设上述的新功能A有问题，我们可以只复原A而不影响修好的Bug，或是只挑选修Bug的移植到不同分支去。

3. 良好的递交信息

每一次的递交程序员都必须附上一段解释信息，说明修改的内容和原因。这除了可以帮助团队合作之外，更重要的是让软件有更好的维护性，以便将来备查，以下的场景相信开发者都不陌生：

Git简历



logo：



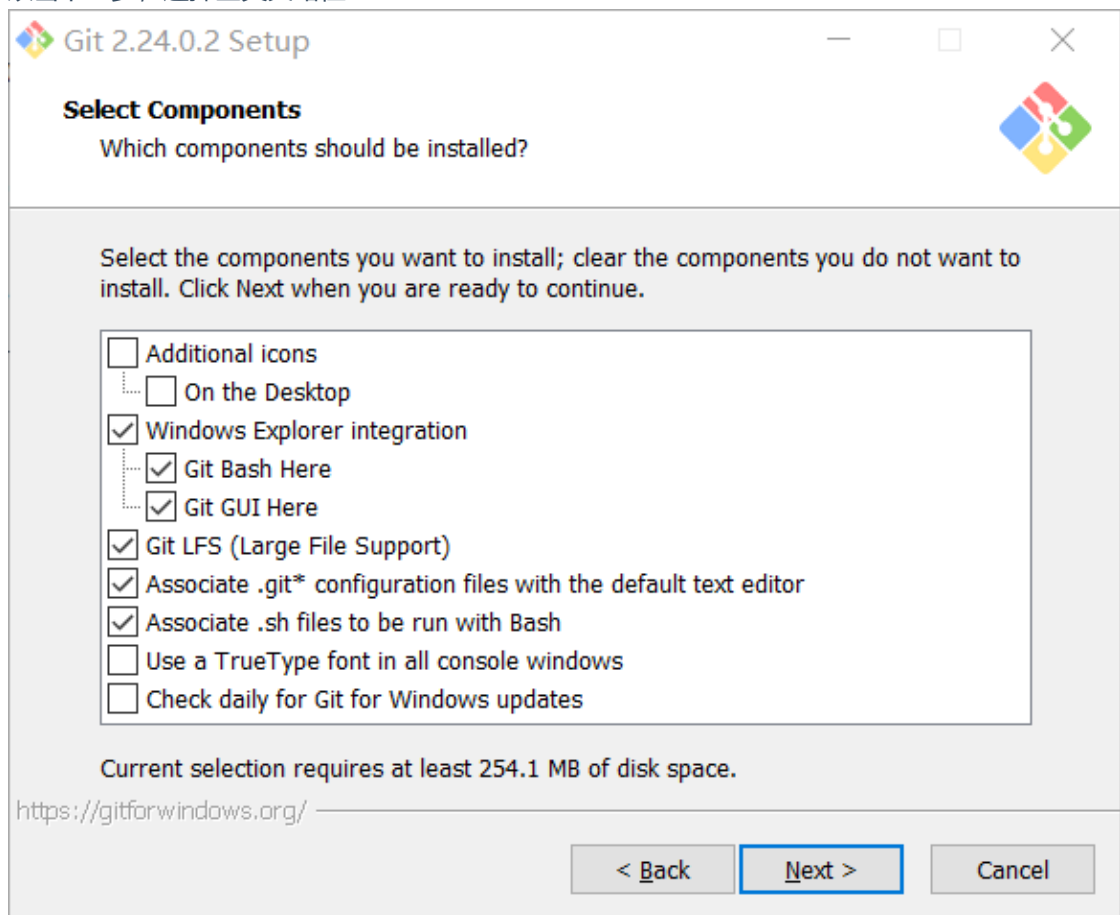
Git的优势

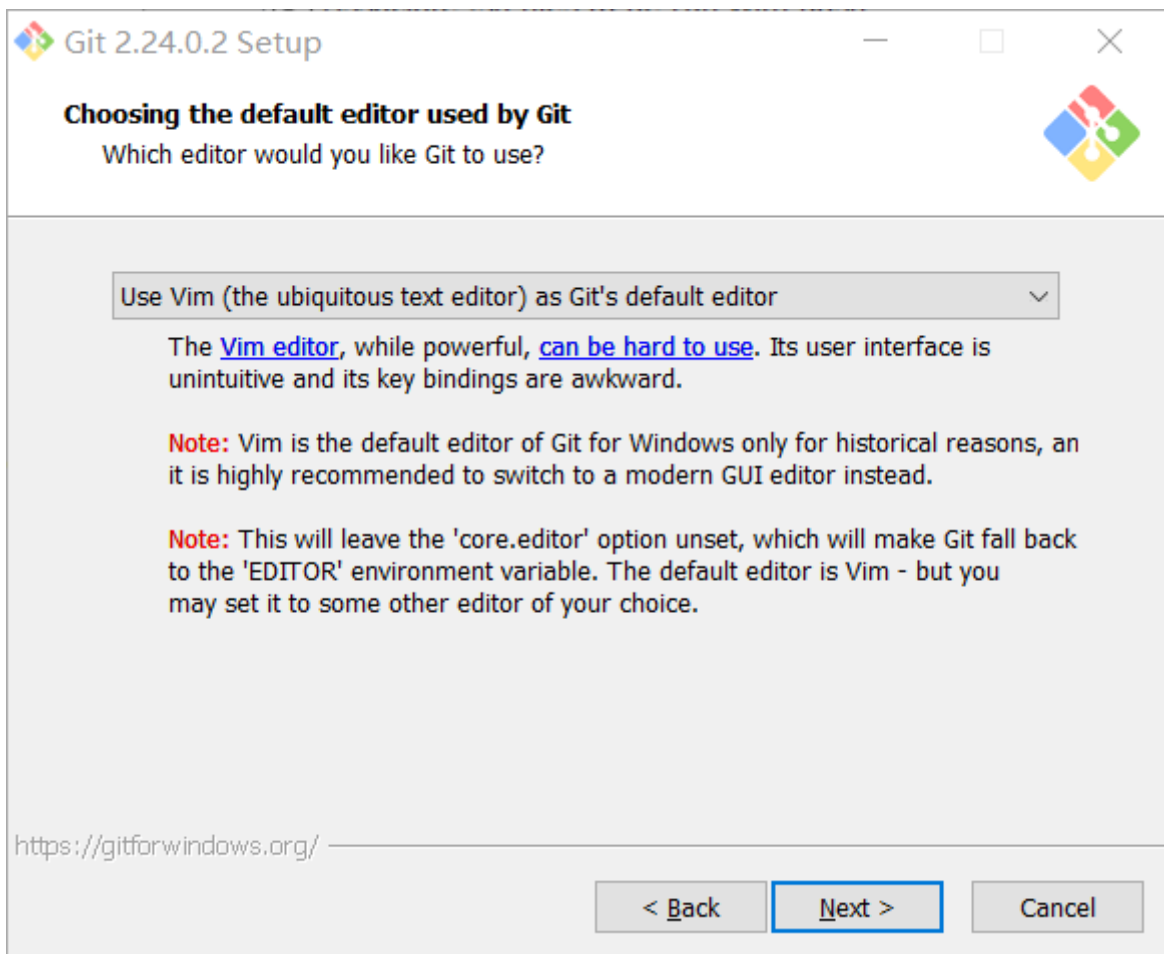
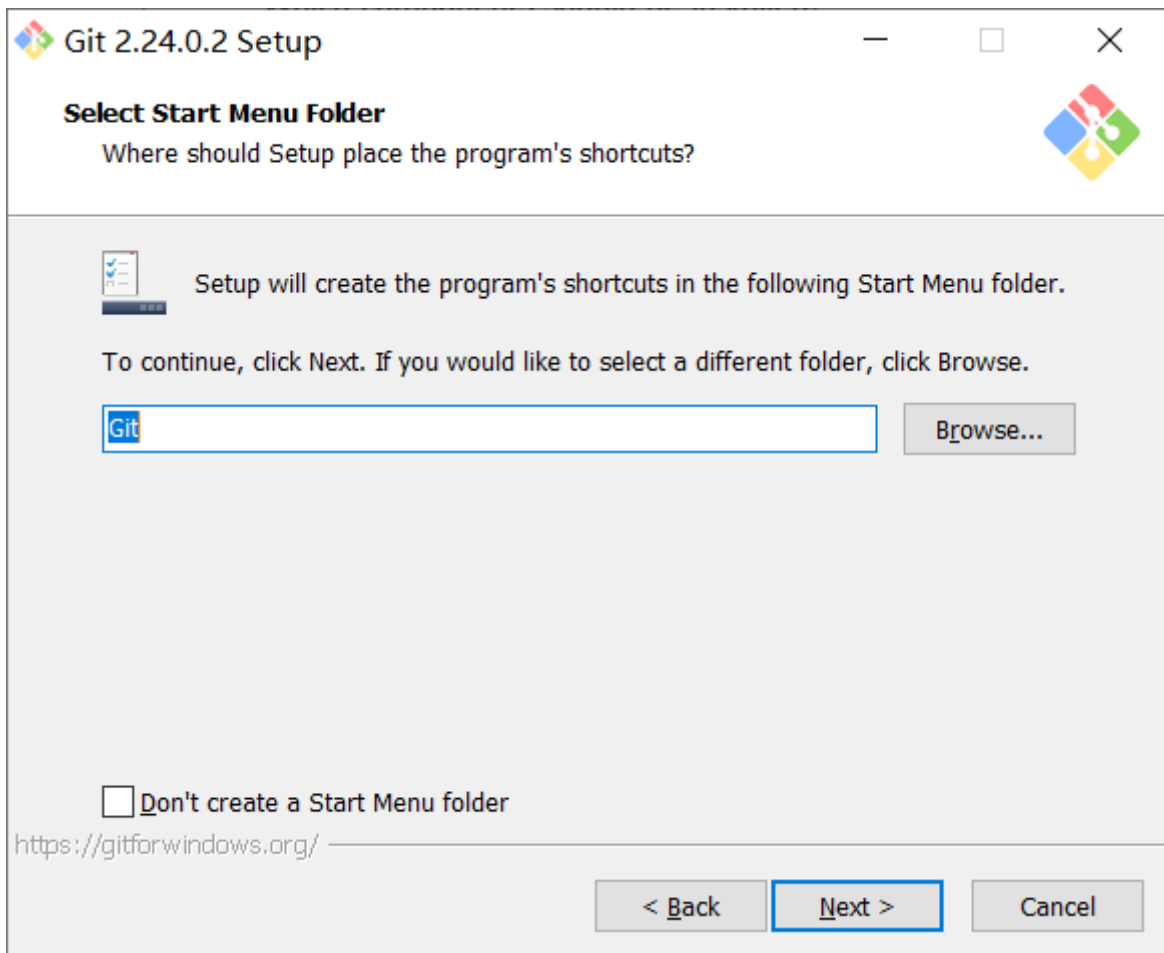
- 大部分操作在本地完成，不需要联网
- 完整性保证 (hash)
- 尽可能添加数据而不是删除或修改数据
- 分支操作非常快捷流畅
- 与linux命令全面兼容

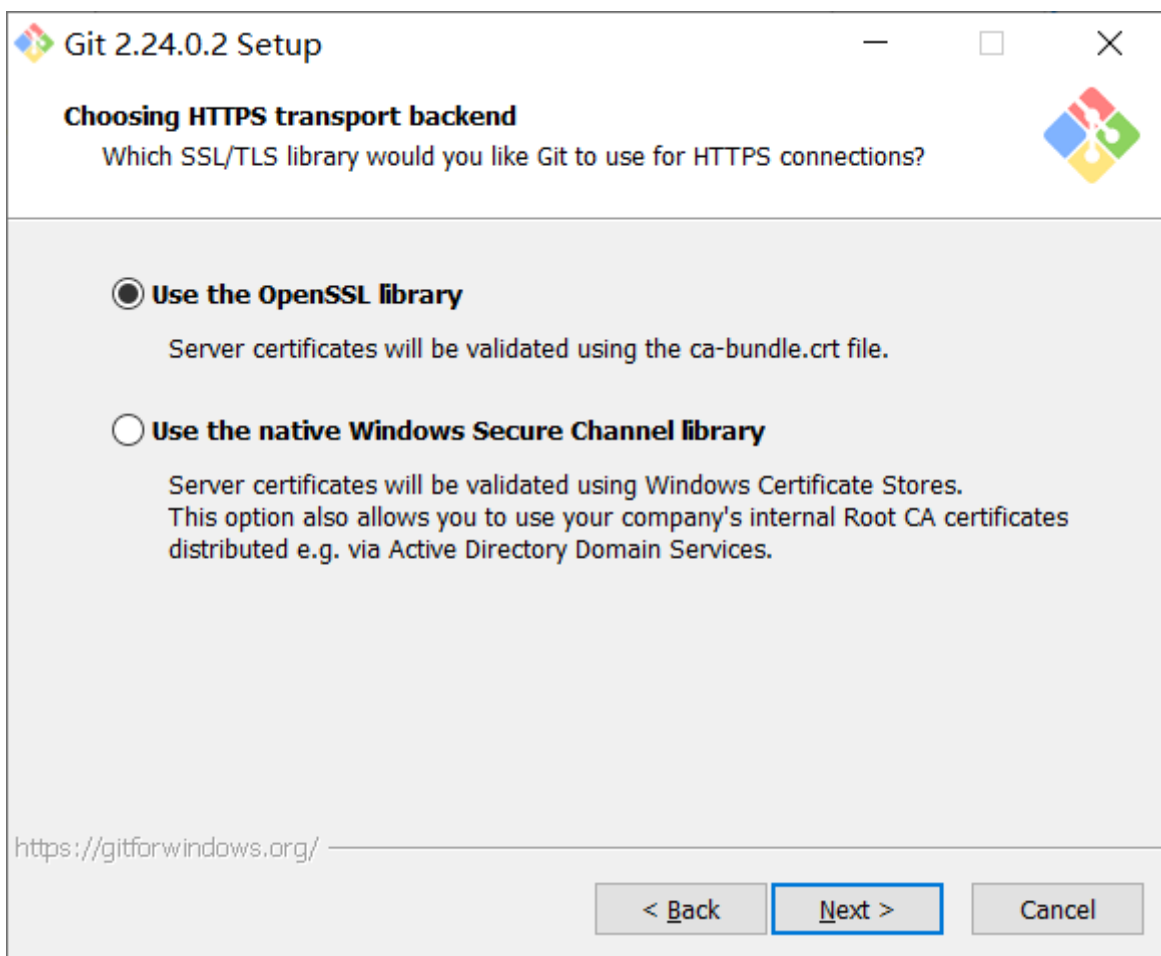
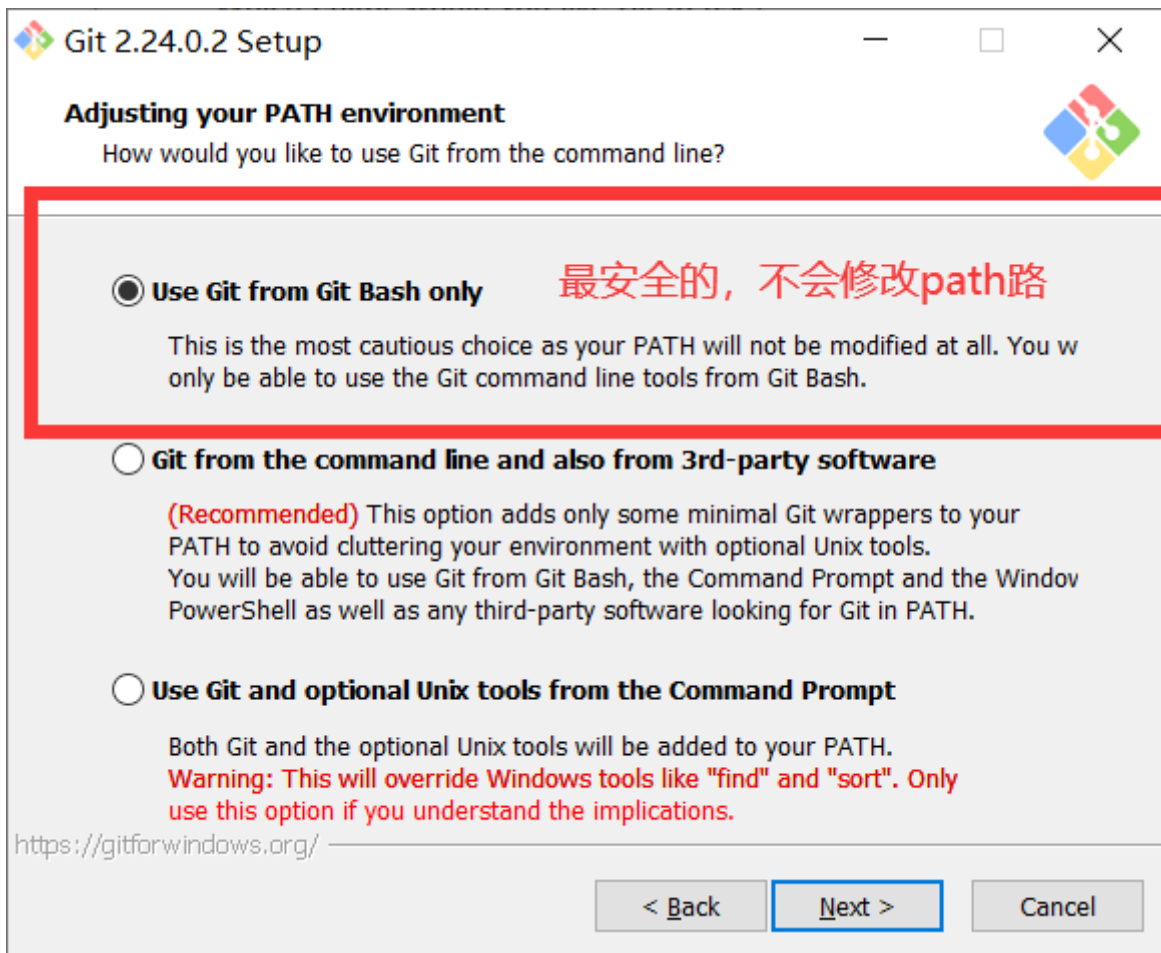
2. Git安装

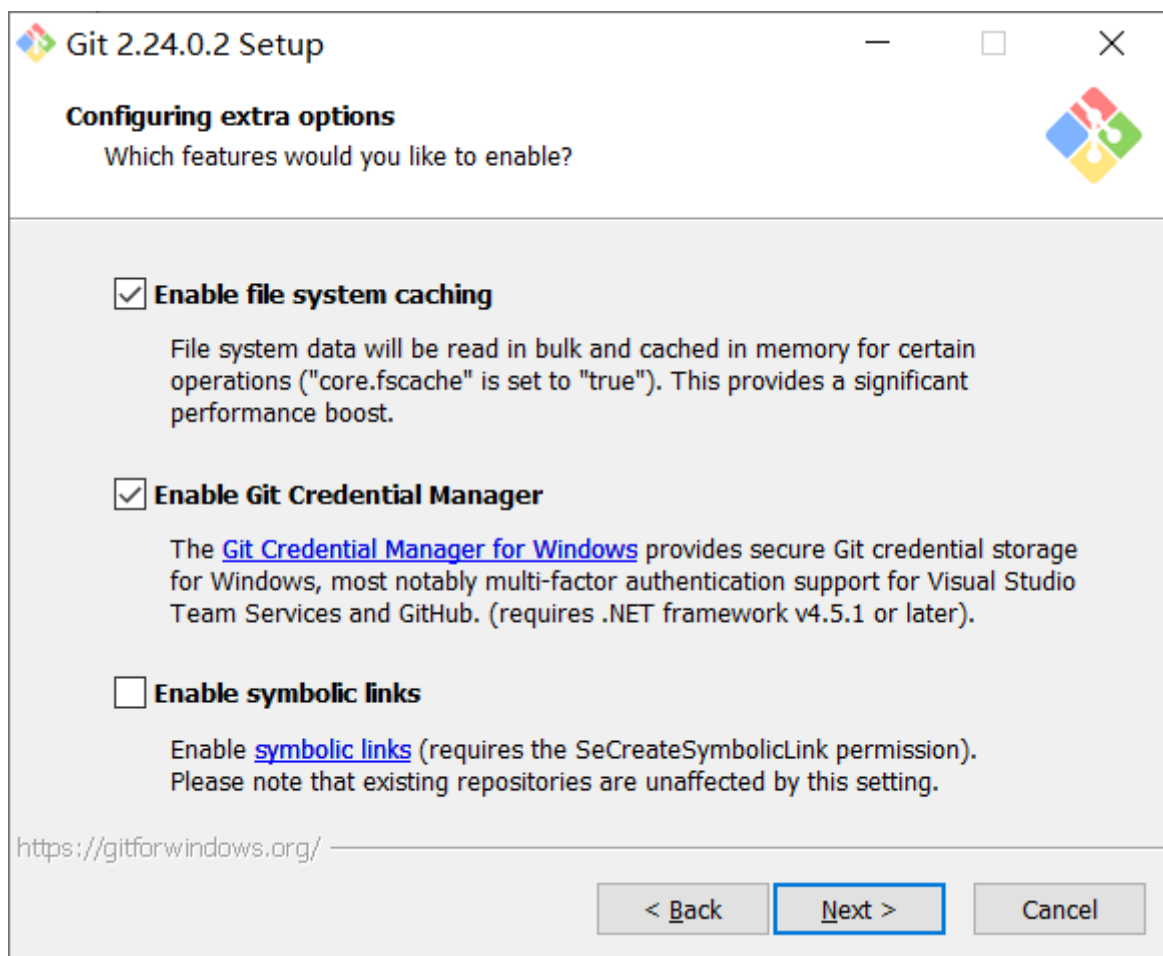
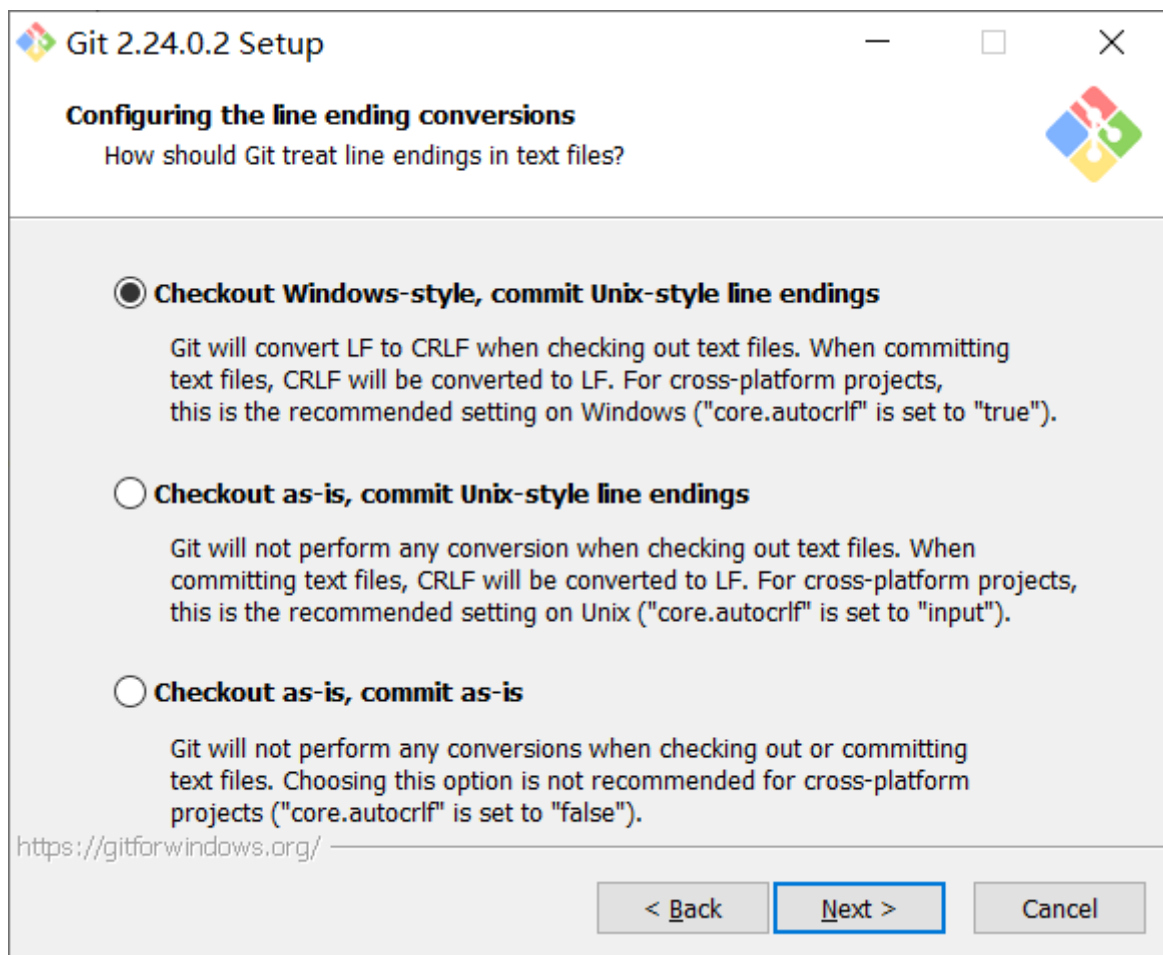
1. Git-2.24.0.2-64-bit.exe
2. 双击下一步，选择全英文路径

3.



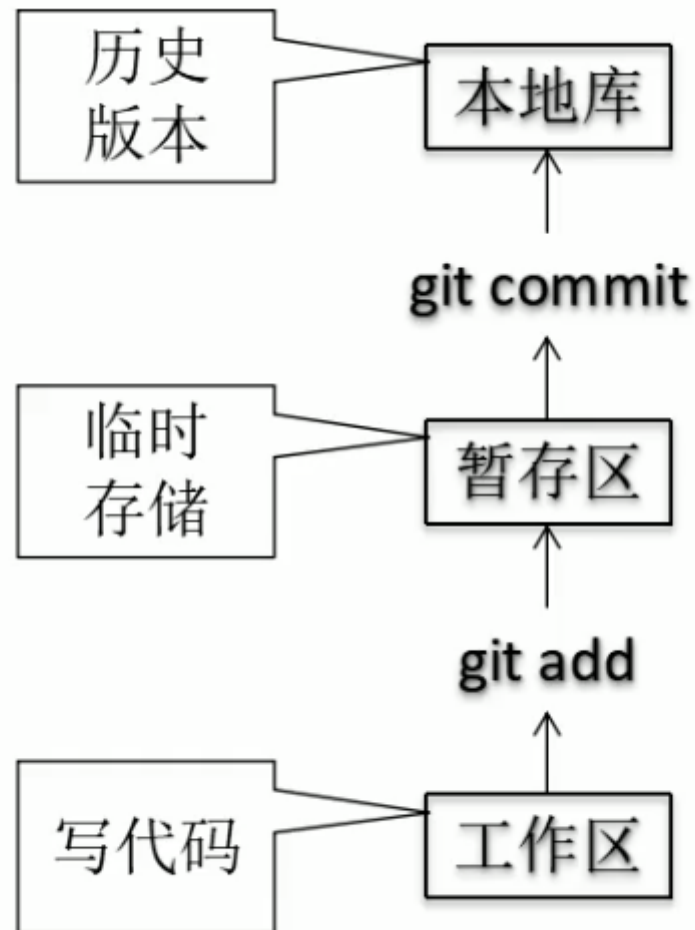






点击install, 在任何目录下Git Bash Here可以打开Git的终端

3.Git结构



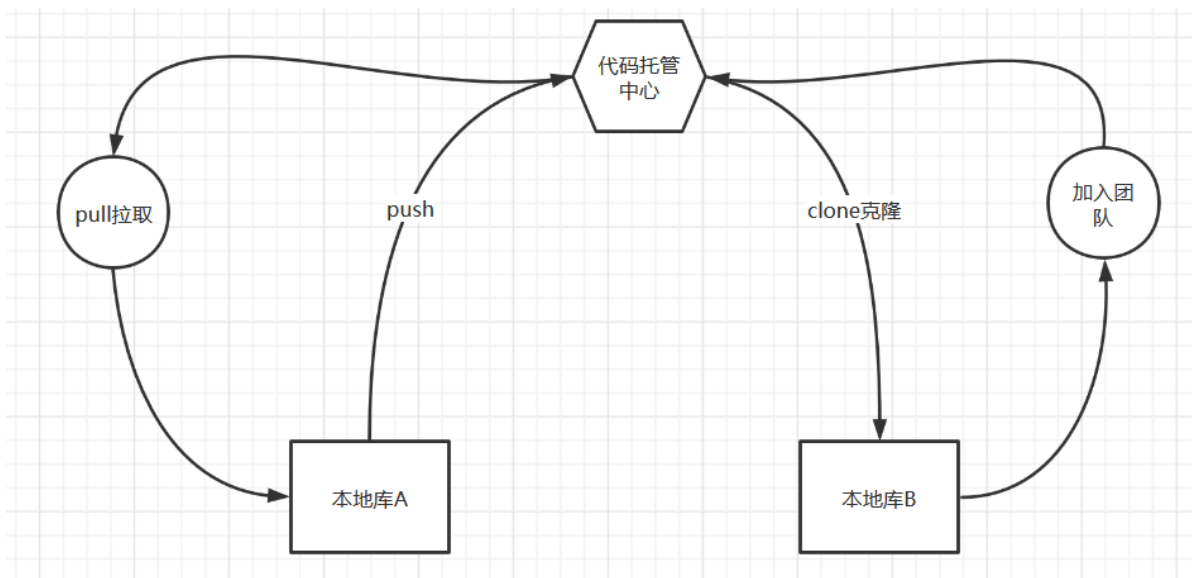
4.Git和代码托管中心

代码托管中心的任务：维护远程库

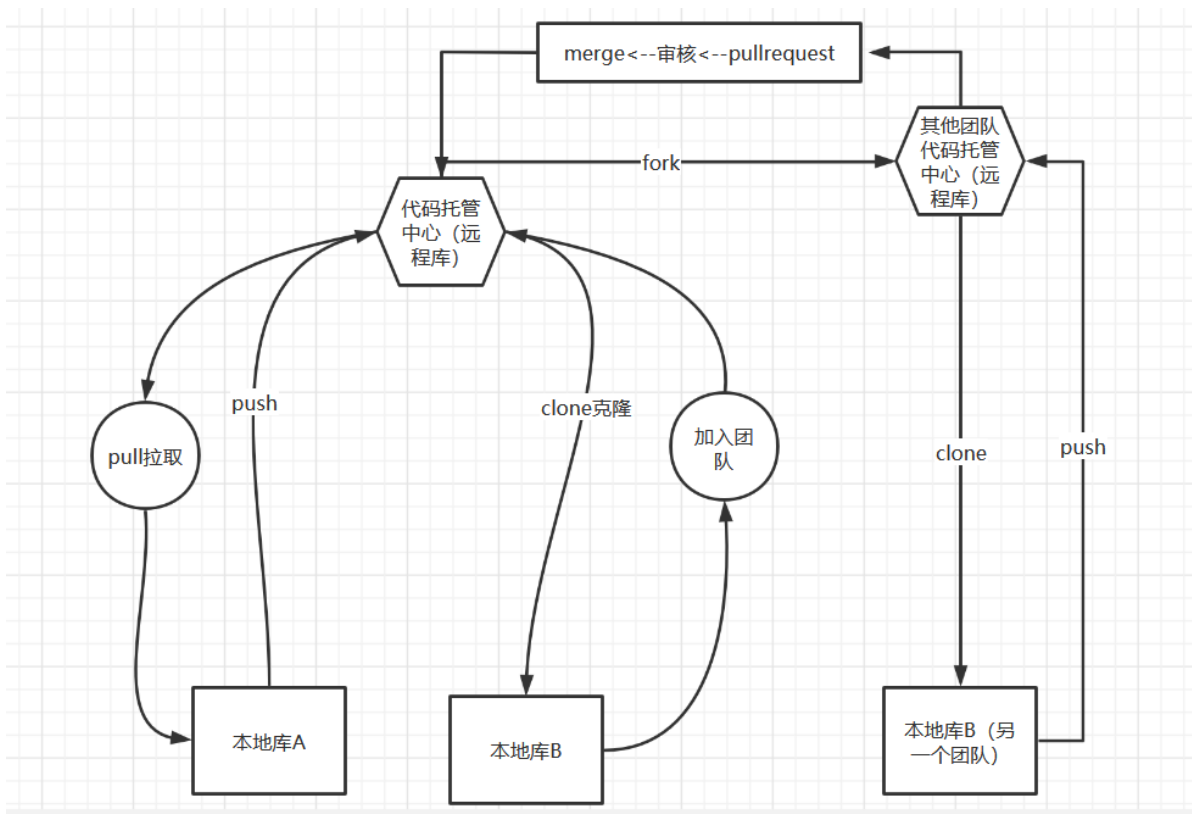
- 局域网环境
 - GitLab服务器。可以自己搭建一个GitLab服务器**
- 外网环境
 - GitHub
 - 码云

5.本地库和远程库的关系

5.1团队内协作的关系



5.2跨团队合作



6.Git命令行操作

6.1本地库初始化

```
git init
```

注意：生成的 `.git` 目录中存放的是本地库相关文件，不要删除

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat
$ git init
Initialized empty Git repository in D:/Git/workspace/WeChat/.git/
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ ls -la
total 4
drwxr-xr-x 1 HX 197121 0 12月 7 11:32 ./
drwxr-xr-x 1 HX 197121 0 12月 7 11:31 ../
drwxr-xr-x 1 HX 197121 0 12月 7 11:32 .git/
```

6.2 设置签名

- 用户名: lusutao
- Email地址: 805634005@qq.com (可以为空)
 - 作用: 区分不同开发人员的身份
 - 辨析: 这里设置的签名和登录远程库 (代码托管中心) 的账号 密码没有任何关系
 - 命令
 - 系统级别/仓库级别 (仅在当前本地库范围内有效)

```
git config user.name lusutao
git config user.email 805634005@qq.com
信息保存的位置 ./git/config 文件
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.name lusutao

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.email 8056340052

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.email 805634005@qq.com

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
[user]
    name = lusutao
    email = 805634005@qq.com
```

- 系统用户级别: 登录当前操作系统的用户范围
 - git config --global user.name lusutao
 - git config --global user.email 805634005@qq.com
 - 信息保存的位置 是用户家目录下面的。 ~ /.gitconfig 文件

```
HX@LAPTOP-4KP112IL MINGW64 ~
$ cat .gitconfig
[user]
    name = lusutao1
    email = 805634005@briup.com
```

- 系统优先级

- 就近原则：项目级别优先于系统用户级别，二者都有时采用项目级别的签名
- 如果只有系统用户级别的签名，就以系统用户级别的签名为准
- 二者都没有不允许

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.name lusutao

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.email 8056340052

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git config user.email 805634005@qq.com

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
[user]
    name = lusutao
    email = 805634005@qq.com

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$
```

6.3git基本操作

6.3.1 添加提交以及查看状态操作

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

输入 `git status`会显示该信息，此时 使用`vim hello.txt`命令创建文件，（按`i`进入编辑，再`ESC+ZZ`保存并关闭`vim`）

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ vim hello.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt

nothing added to commit but untracked files present (use "git add" to track)
```

`git add hello.txt`

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git add hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
```

输入 `git add` 命令的以后再次输入 `git status`

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt
```

文件显示绿色，此时文件被放入暂存区（使用 `git rm --cached hello.txt` 将文件从暂存区中取消）

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git rm --cached hello.txt
rm 'hello.txt'

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.txt

nothing added to commit but untracked files present (use "git add" to track)
```

此时文件名再次为红色

然后输入 `git commit`

```

1
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # on branch master
6 #
7 # Initial commit
8 #
9 # changes to be committed:
10 #       new file:   hello.txt
11 #

```

进入vim模式，可以进行编辑，对此次commit进行注释

```

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git commit hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master (root-commit) a43c01b] My first commit.new file hello.txt
1 file changed, 3 insertions(+)
create mode 100644 hello.txt

```

保存退出

```

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git status
On branch master
nothing to commit, working tree clean

```

此时暂存区为空，工作区也没有新建

再次vim 修改文件以后 查询

```

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ vim hello.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

此时可以add也可以直接commit

```

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (master)
$ git commit
[master 95d1931] 这是第一次修改后的文件
1 file changed, 1 insertion(+)

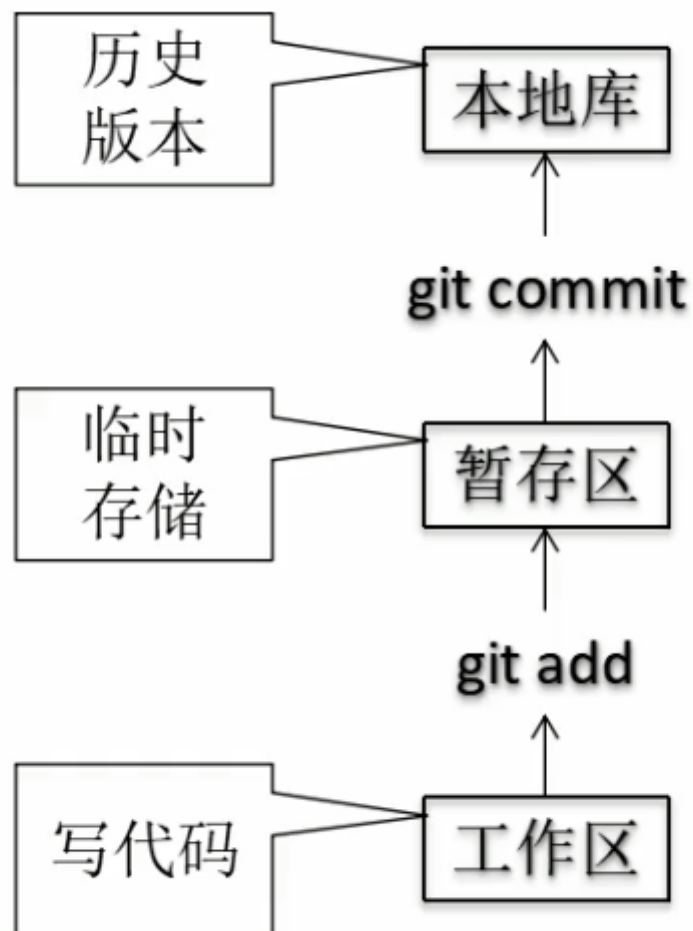
```

如果使用 `git commit -m "第二次修改后的文件" good.txt` 可以不通过vim直接commit

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git commit -m "第二次修改后的文件" hello.txt
warning: LF will be replaced by CRLF in hello.txt.
The file will have its original line endings in your working directory
[master 4458107] 第二次修改后的文件
1 file changed, 1 insertion(+)
```

命令小结:

- 状态查看
 - `git status`
- 添加操作
 - `git add [file name]`
- 提交操作
 - `git commit -m "第二次修改后的文件" good.txt` 或者 `git commit [file name]`



6.3.2 查看版本

```
git log
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/wechat (master)
$ git log
commit 44581072909ba412aca1a1c3abb77b4319647b12 (HEAD -> master)
Author: lusutao <805634005@qq.com>
Date: Mon Dec 7 14:29:14 2020 +0800

    第二次修改后的文件

commit 95d1931a38ca4fbcfce756c844efa7976aade2bd
Author: lusutao <805634005@qq.com>
Date: Mon Dec 7 14:24:54 2020 +0800

    这是第一次修改后的文件

commit a43c01bc526c6e6e0bde7da8c065ba8cdcc0a0c1
Author: lusutao <805634005@qq.com>
Date: Mon Dec 7 14:13:50 2020 +0800

    My first commit.new file hello.txt
```

如果版本过多，可以空格往下翻页。b向上翻页。q推出

```
$ git log --pretty=oneline 简洁显示
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/wechat (master)
$ git log --pretty=oneline
44581072909ba412aca1a1c3abb77b4319647b12 (HEAD -> master) 第二次修改后的文件
95d1931a38ca4fbcfce756c844efa7976aade2bd 这是第一次修改后的文件
a43c01bc526c6e6e0bde7da8c065ba8cdcc0a0c1 My first commit.new file hello.txt
```

```
$ git log --oneline 更为简洁，hash值也只显示一部分
```

```
$ git log --oneline
4458107 (HEAD -> master) 第二次修改后的文件
95d1931 这是第一次修改后的文件
a43c01b My first commit.new file hello.txt
```

```
git reflog Head@{移动到当前版本需要多少步}
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/wechat (master)
$ git reflog
4458107 (HEAD -> master) HEAD@{0}: commit: 第二次修改后的文件
95d1931 HEAD@{1}: commit: 这是第一次修改后的文件
a43c01b HEAD@{2}: commit (initial): My first commit.new file hello.txt
```

6.3.3 版本前进后退

- 基于索引值操作（推荐使用）

```
git reset --hard [局部哈希值]
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reflog
4458107 (HEAD -> master) HEAD@{0}: commit: 第二次修改后的文件
95d1931 HEAD@{1}: commit: 这是第一次修改后的文件
a43c01b HEAD@{2}: commit (initial): My first commit.new file hello.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reset --hard 95d1931
HEAD is now at 95d1931 这是第一次修改后的文件
```

再次查询

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reflog
95d1931 (HEAD -> master) HEAD@{0}: reset: moving to 95d1931
4458107 HEAD@{1}: commit: 第二次修改后的文件
95d1931 (HEAD -> master) HEAD@{2}: commit: 这是第一次修改后的文件
a43c01b HEAD@{3}: commit (initial): My first commit.new file hello.txt
```

此时指针移动完成，此时文件变为该版本的内容

```
UUUUU

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reset --hard a43c01b
HEAD is now at a43c01b My first commit.new file hello.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ cat hello.txt
No commits yet
No commits yet
hello
```

- 使用^符号（了解）
 - 只能往后退，不能前进 `git reset --hard HEAD ^^^` 几个^符号回退几个版本
- 使用~符号（了解）
 - 再很多版本时，不适合使用^，这时使用~符号
 - `git reset --hard HEAD~ 2` 则是回退的版本数

6.3.4 hard 和soft以及mixed参数对比

`git help reset` 可以打开本地的帮助文档

- --soft
 - 仅仅是再本地库移动head指针

Does not touch the index file or the working tree at all (but resets the head to `<commit>` , just like all modes do). This leaves all your changed files "Changes to be committed", as `git status` would put it.

- --mixed
 - 再本地库移动head指针
 - 重置暂存区

Resets the index but not the working tree (i.e., the changed files are preserved but not marked for commit) and reports what has not been updated. This is the default action.

If `-N` is specified, removed paths are marked as intent-to-add (see `git-add(1)`).

- --hard

- 再本地库移动head指针
- 重置暂存区
- 重置工作区

Resets the index and working tree. Any changes to tracked files in the working tree since `<commit>` are discarded.

6.3.4 文件的删除并找回

先创建文件a.txt,将其提交到暂存区中,使用 `rm a.txt` 命令,再删除后,只是文件会删除,而删除文件的操作还在。

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reflog
d923bd3 (HEAD -> master) HEAD@{0}: commit: delete a.txt
a2a469c HEAD@{1}: commit: 新文件
dfe3f0a HEAD@{2}: commit: 第三次修改后的文件
95d1931 HEAD@{3}: reset: moving to 95d1931
a43c01b HEAD@{4}: reset: moving to a43c01b
95d1931 HEAD@{5}: reset: moving to 95d1931
4458107 HEAD@{6}: commit: 第二次修改后的文件
95d1931 HEAD@{7}: commit: 这是第一次修改后的文件
a43c01b HEAD@{8}: commit (initial): My first commit.new file hello.txt
```

此时只需要更换版本就可以找回文件

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git reset --hard a2a469c
HEAD is now at a2a469c 新文件

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ ll
total 2
-rw-r--r-- 1 HX 197121  7 12月  7 15:35 a.txt
-rw-r--r-- 1 HX 197121 53 12月  7 15:15 hello.txt
```

如何将删除的文件找回:

- 前提: 删除前, 文件存在时候的版本提交到了本地库
- 操作: `git reset --hard[指针位置]`
 - 删除操作已经提交到本地库: 指针位置指向历史记录
 - 删除操作尚未提交到本地库 (还在暂存区): 指针位置使用Head命令

6.3.5 比较文件

`git diff [file name]` 再修改本地文件后, 文件还没有进行add或者commit时使用

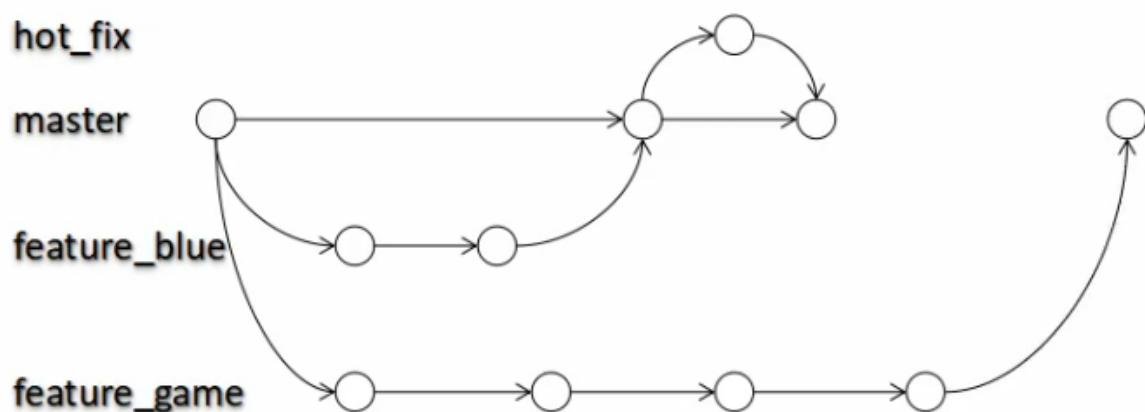
[illegible]

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git add a.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/weChat (master)
$ git diff a.txt
```

小结

- ## 7.分支



7.1分支的好处

- 同时并行推进多个功能开发，提高开发效率
- 各个分支再开发过程中，如果一个分支开发失败，不会对其他分支有任何影响，失败的分支删除重新开始即可

7.2分支操作

- 创建分支

```
git branch 分支名
```

- 查看分支

```
git branch  
git branch -v
```

- 切换分支

```
git checkout 分支名  
git checkout -b 分支名    #创建分支并直接切换到该分支
```

- 合并分支 相当于把修改了的文件拉过来
 - 第一步：切换到接受修改的分支（被合并，增加到新内容上）
 - 第二部：执行merge【分支名】将需要合并的分支连接到所在的分支上

```
git merge xxx
```

注意：合并分支的时候要明确谁谁合并
我在a分支里面修改了。要合并到master，就先切换到master，然后合并b

- 删除分支

```
git branch -d 分支名
```

7.3.冲突

再多个分支都修改了同一文件后进行合并时，便会产生冲突

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (hot_fix)
$ git merge master
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.
```

此时合并失败，需要去手动去解决

分支的表现：

```

1 <<<<<<< HEAD
2 i bug修复zzz
3 冲突测试
4 分支冲突测aa试
5 =====
6 aaa
7 sdsadsaZZ
8 >>>>>>> master
~

```

如何解决?

```

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (hot_fix|MERGING)
$ git status
On branch hot_fix
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   a.txt

no changes added to commit (use "git add" and/or "git commit -a")

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (hot_fix|MERGING)
$ git add a.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (hot_fix|MERGING)
$ git commit -m "冲突解决"
[hot_fix 61b92ed] 冲突解决

HX@LAPTOP-4KP112IL MINGW64 /workspace/WeChat (hot_fix)
$ cat a.txt
冲突解决

```

在冲突后，需要自己去编辑文件，将其中的====》》》》《《《《等特殊符号删除，在经过协商后将文件修改到满意的程度，保存退出。然后进行 `git add【文件名】`，再 `git commit -m"注释"` 命令，注意：在此时commit是不能带文件名的，否则会报错

7.4分支管理

git分支管理的本质就是创建和移动指针

8.创建远程库

8.1 首先创建本地库

```
HX@LAPTOP-4KP112IL MINGW64 /workspace
$ mkdir huashan

HX@LAPTOP-4KP112IL MINGW64 /workspace
$ git init
Initialized empty Git repository in D:/Git/workspace/.git/

HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ vim hsjf.txt

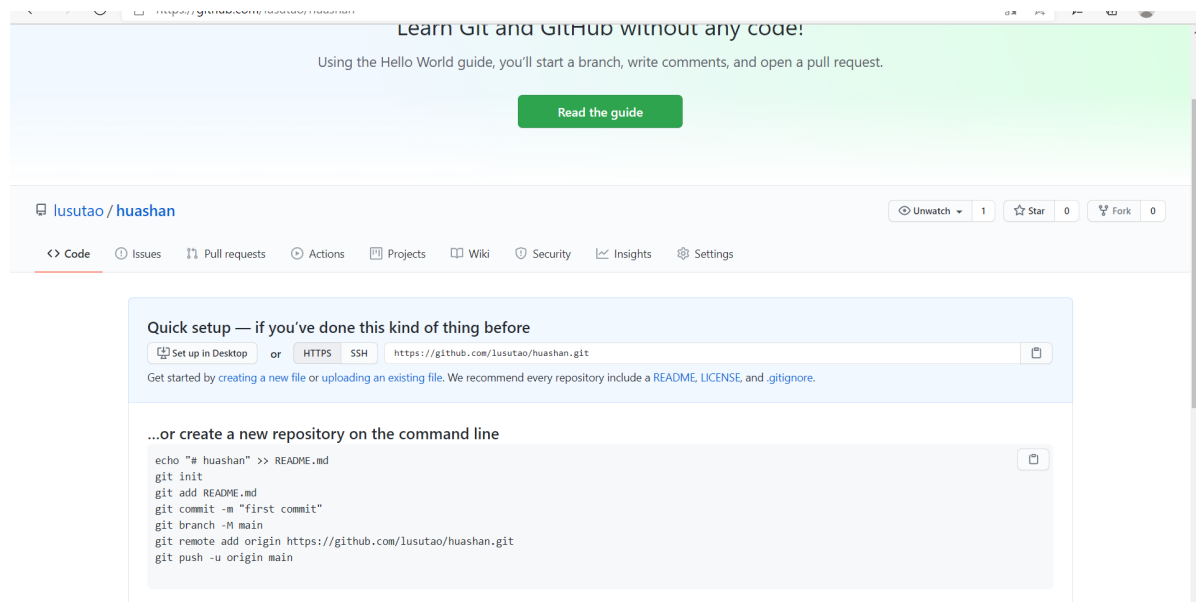
HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ git add hsjf.txt
warning: LF will be replaced by CRLF in hsjf.txt.
The file will have its original line endings in your working directory

HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ git commit -m "华山1" hsjf.txt
warning: LF will be replaced by CRLF in hsjf.txt.
The file will have its original line endings in your working directory
[master (root-commit) 11961ab] 华山1
1 file changed, 1 insertion(+)
create mode 100644 hsjf.txt
```

8.2 首先创建本地库

8.3 创建远程库

登陆GitHub



8.4 通过push推送本地库文件

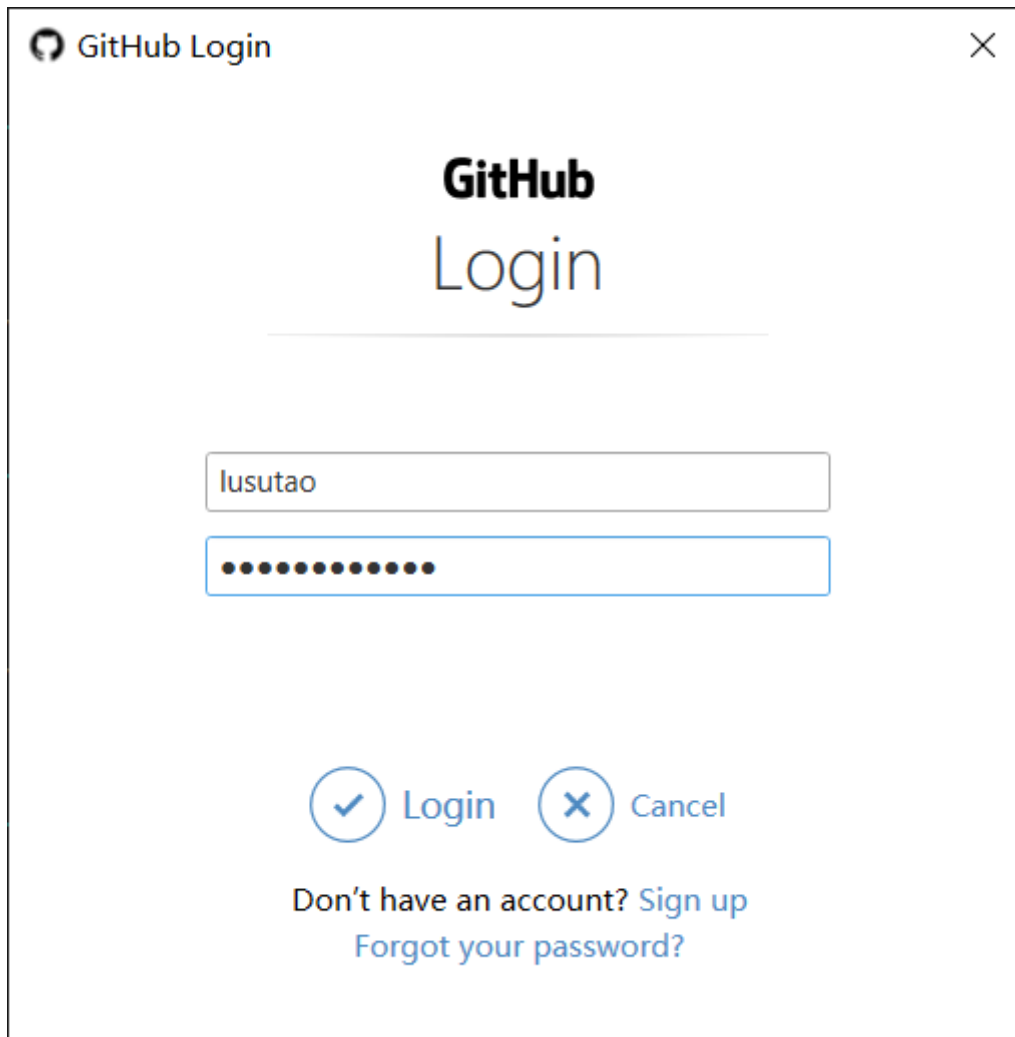
可以使用 `git remote add lst https://github.com/lusutao/huashan.git` 来给地址取别名为lst

```
HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ git remote add lst https://github.com/lusutao/huashan.git
fatal: remote lst already exists.

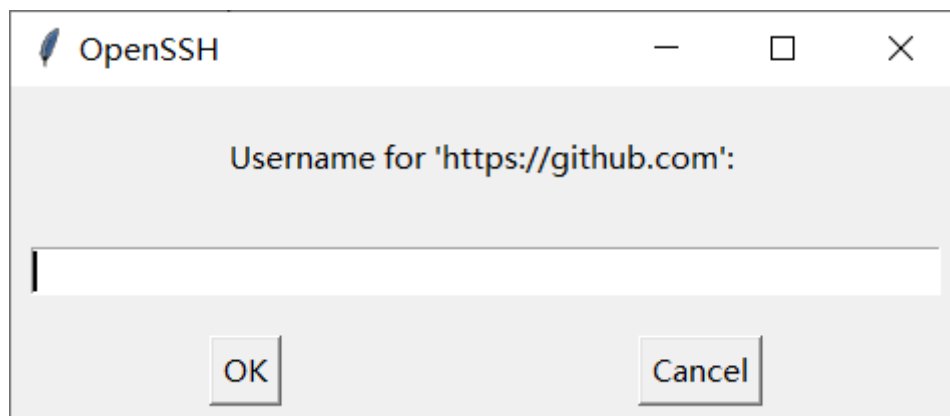
HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ git remote
lst
```

接着进行 `git push lst master` 操作，此时需要验证，第一次填写GitHub的邮箱，第二次填写GitHub的密码

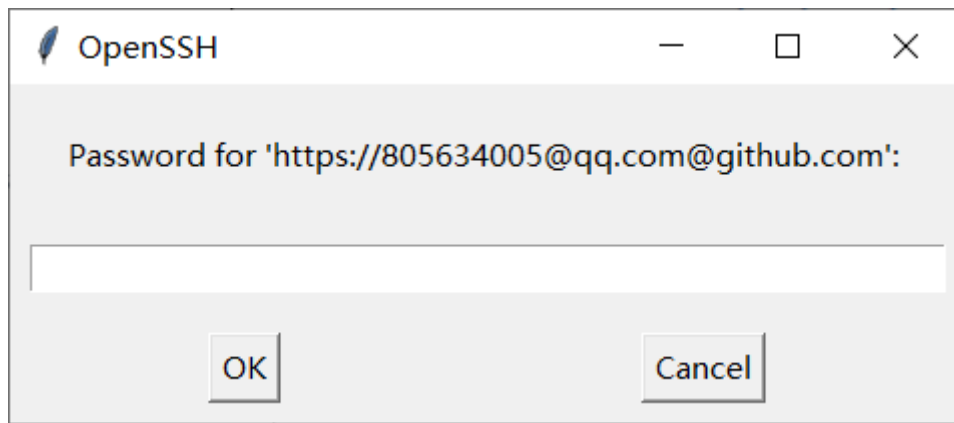
push以后的弹窗



A screenshot of the GitHub Login dialog box. The title bar says "GitHub Login" with a close button. The main text "GitHub Login" is centered. Below it is a horizontal line. There are two input fields: the first contains "lusutao" and the second is a password field with dots. At the bottom, there are two buttons: "Login" (with a checkmark icon) and "Cancel" (with an X icon). Below the buttons, there is a link "Don't have an account? Sign up" and another link "Forgot your password?".



A screenshot of the OpenSSH Username dialog box. The title bar says "OpenSSH" with standard window controls. The main text is "Username for 'https://github.com':". Below it is a text input field. At the bottom, there are two buttons: "OK" and "Cancel".



```
HX@LAPTOP-4KP112IL MINGW64 /workspace (master)
$ git push 1st master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 225 bytes | 225.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/lusutao/huashan.git
* [new branch]      master -> master
```

此时则push完成

8.5 克隆

```
git clone[远程地址]
```

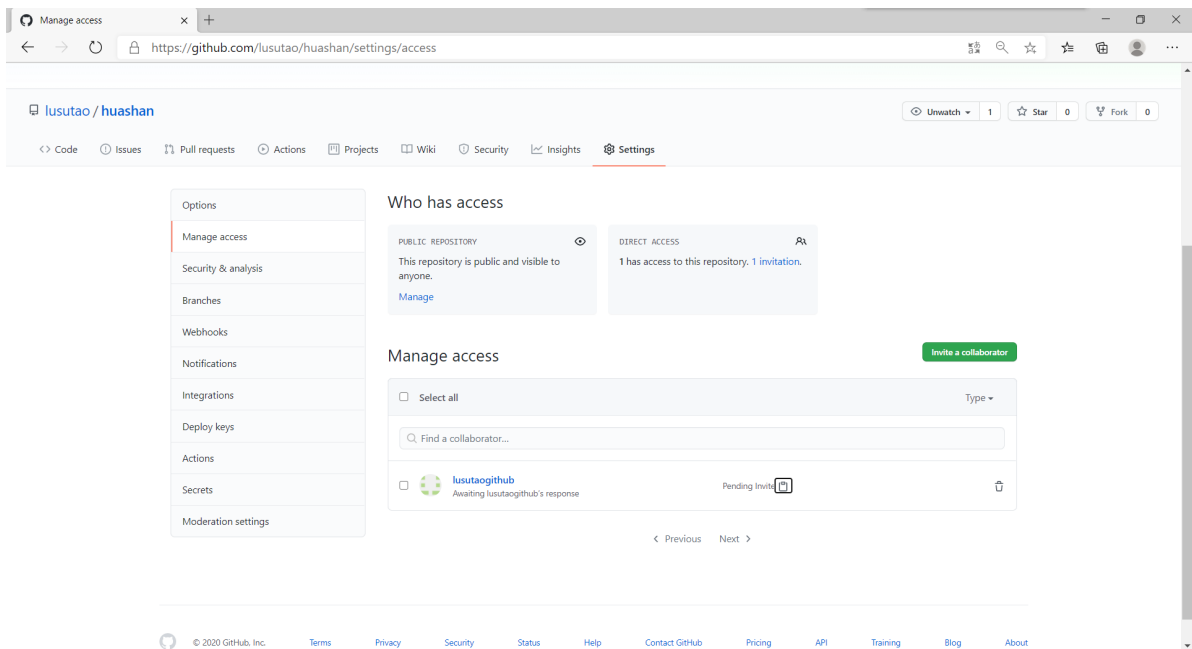
```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan_1hc (master)
$ git clone https://github.com/lusutao/huashan.git
Cloning into 'huashan'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
```

克隆的三个效果

- 完整的吧远程库下载到本地
- 替我们创建roigin远程地址别名
- 初始化本地库

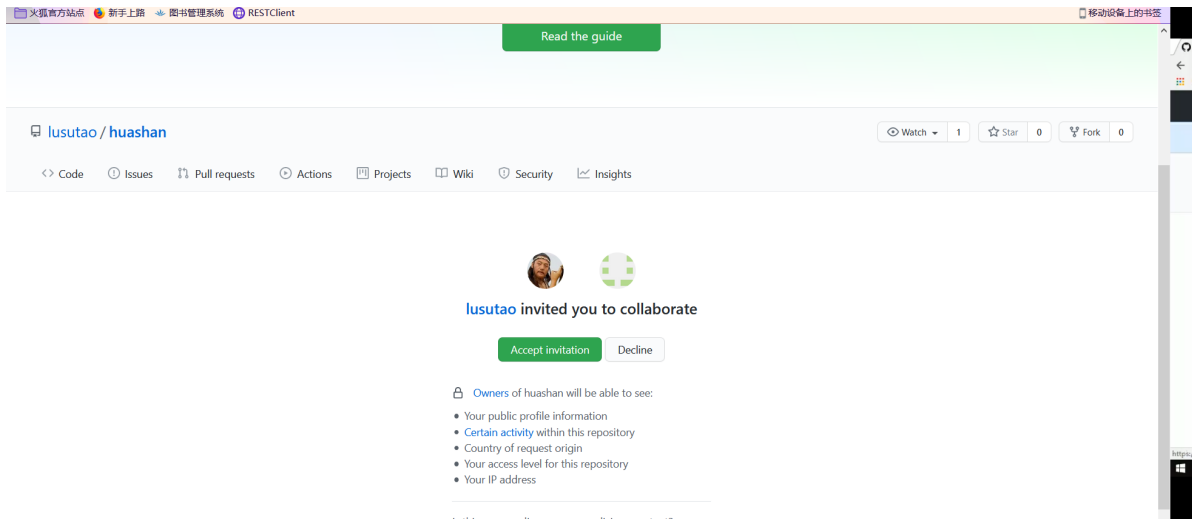
8.6 邀请另一个用户加入团队

在创建远程库的账号中查找新用户账号，进行邀请，并复制链接



然后登陆新用户的GitHub账号

直接在该用户下访问复制的链接



点击同意

此时便和可以push

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan_lhc/huashan/huashan (master)
$ git push lsgithub master
Logon failed, use ctrl+c to cancel basic credential prompt.
remote: You must verify your email address.
remote: See https://github.com/settings/emails.
fatal: unable to access 'https://github.com/lusutao/huashan.git/': The request
returned error: 403

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan_lhc/huashan/huashan (master)
$ git push lsgithub master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 366 bytes | 15.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/lusutao/huashan.git
    ddddf91..e9e4e8d  master -> master

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan_lhc/huashan/huashan (master)
$ |
```

此时新用户便有了push的权限

8.7 远程数据库的拉取

在一个用户提交修改后的文件后，另一个用户需要拉取的操作。

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git fetch lst master
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/lusutao/huashan
* branch                master      -> FETCH_HEAD
  ddddf91..e9e4e8d      master      -> lst/master

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ cat hsjf.txt
华山剑法-ybq

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git checkout lst/master
Note: switching to 'lst/master'.

You are in 'detached HEAD' state. You can look around, make e
xperimental
changes and commit them, and you can discard any commits you
make in this
state without impacting any branches by switching back to a b
ranch.

If you want to create a new branch to retain commits you crea
te, you may
do so (now or later) by using -c with the switch command. Exa
mple:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detach
```

```
git switch -

Turn off this advice by setting config variable advice.detach
edHead to false

HEAD is now at e9e4e8d 令狐冲

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan ((e9e4e8d...))
$ cat hsjf.txt
华山剑法-ybq
独孤九剑-lhc

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan ((e9e4e8d...))
$
```

此时另一个用户的文件也修改为新的文件，即拉取完成

这是将本地文件也进行了修改，此时本地huashan文件夹和huashan_lhc文件夹的hsjf.txt内容一致

MINGW64:/workspace/huashan

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan ((e9e4e8d..
$ git checkout master
Previous HEAD position was e9e4e8d 令狐冲
Switched to branch 'master'

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ cat hsjf.txt
华山剑法-ybq

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git merge lst/master
FETCH_HEAD      HEAD            lst/master      master

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git merge lst/master
Updating ddddf91..e9e4e8d
Fast-forward
 huashan/hsjf.txt | 1 +
 1 file changed, 1 insertion(+)

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ cat hsjf.txt
华山剑法-ybq
独孤九剑-lhc
```

拉取小结：

pull=fetch+merge

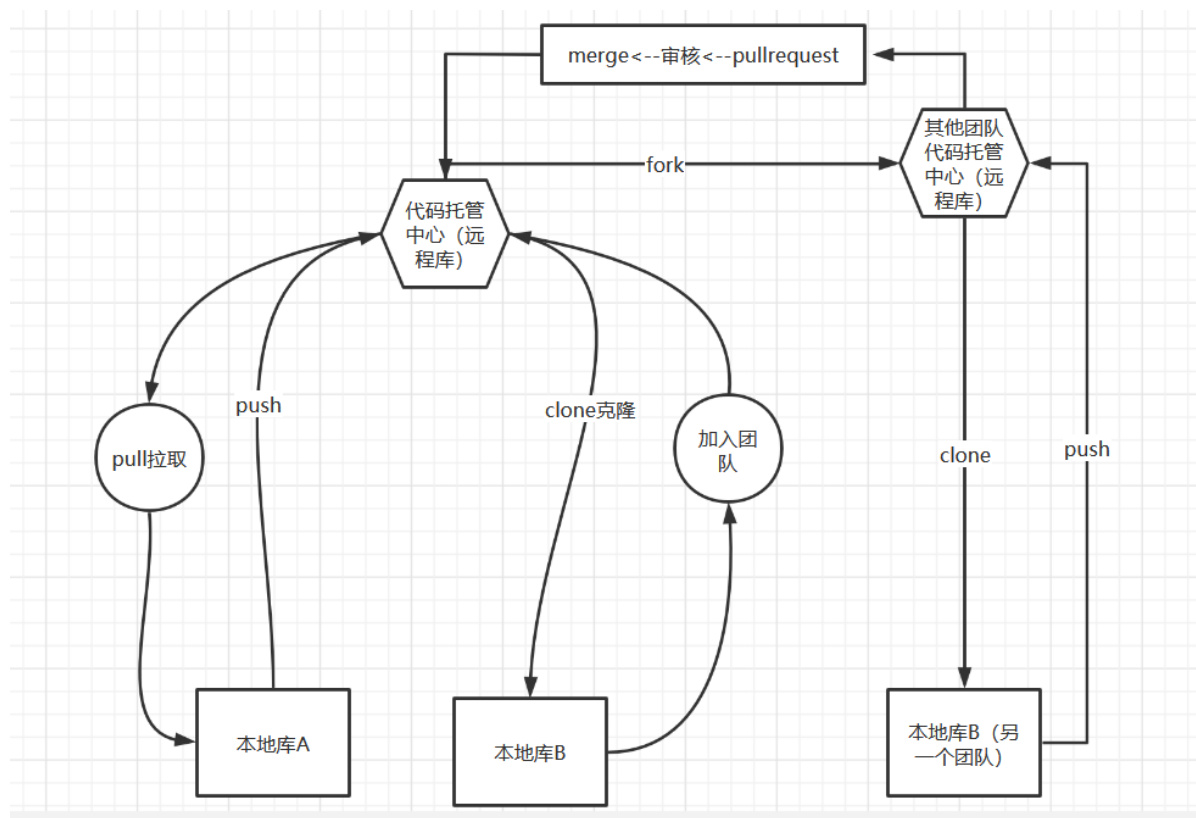
- git fetch 【远程地址别名】 【远程分支名】
- git merge 【远程地址别名/远程分支名】

8.8 解决冲突

要点

- 如果不是基于github远程库的最新版所做的修改，而是和最新版有冲突的话，不能推送，需要先拉取
- 拉取下来后会进入冲突状态，按照“分支冲突解决”操作解决即可

9.跨团队



9.ssh免密登陆

9.1设置

首先回到家目录 `cd ~`

```

HX@LAPTOP-4KP112IL MINGW64 ~
$ ssh-keygen -t rsa -C 805634005@qq.com
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HX/.ssh/id_rsa):
Created directory '/c/Users/HX/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/HX/.ssh/id_rsa.
Your public key has been saved in /c/Users/HX/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ZqTOXS1EMBMuEriiY90TccQe2v29QIHB3vOfOCGDAJY 805634005@qq.com
The key's randomart image is:
+---[RSA 3072]-----+
|  +oo.o =o. |
| E.+ + o + |
| . =.* + + . |
| ...+ * B . . |
| ... o * S o . |
| o. . * o = . |
| .. + = = . |
|          + o |
|          . |
+---[SHA256]-----+

```

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



mykey

SHA256:ZqTOXS1EMBMuEriiY90TCcQe2v29QIH83v0FOCGDAJY

Added on 8 Dec 2020

Never used — Read/write

[Delete](#)

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GPG keys

[New GPG key](#)

9.2测试

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ vim a.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git add a.txt
warning: LF will be replaced by CRLF in huashan/a.txt.
The file will have its original line endings in your working directory

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git commit -m"免密登陆" a.txt
warning: LF will be replaced by CRLF in huashan/a.txt.
The file will have its original line endings in your working directory
[master 024a283] "免密登陆"
 1 file changed, 1 insertion(+)
 create mode 100644 huashan/a.txt

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git remote -v
lst      https://github.com/lusutao/huashan.git (fetch)
lst      https://github.com/lusutao/huashan.git (push)
```

```
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git remote add lst_ssh git@github.com:lusutao/huashan.git

HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git remote -v
lst      https://github.com/lusutao/huashan.git (fetch)
lst      https://github.com/lusutao/huashan.git (push)
lst_ssh  git@github.com:lusutao/huashan.git (fetch)
lst_ssh  git@github.com:lusutao/huashan.git (push)

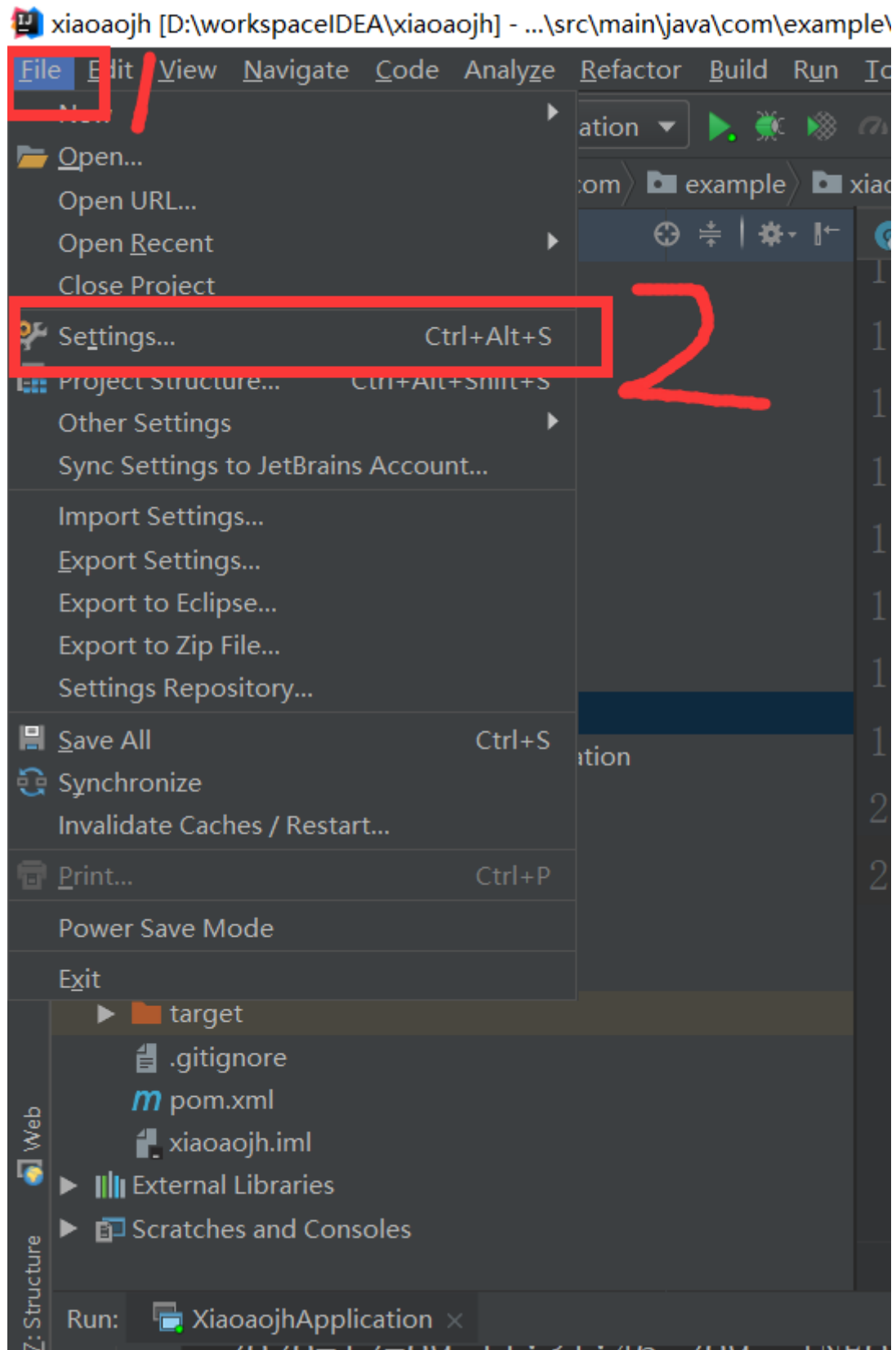
HX@LAPTOP-4KP112IL MINGW64 /workspace/huashan (master)
$ git push lst_ssh master
The authenticity of host 'github.com (192.30.255.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXupJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,192.30.255.113' (RSA) to the list of known hosts.
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 424 bytes | 20.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:lusutao/huashan.git
   d7d9ce2..024a283  master -> master
```

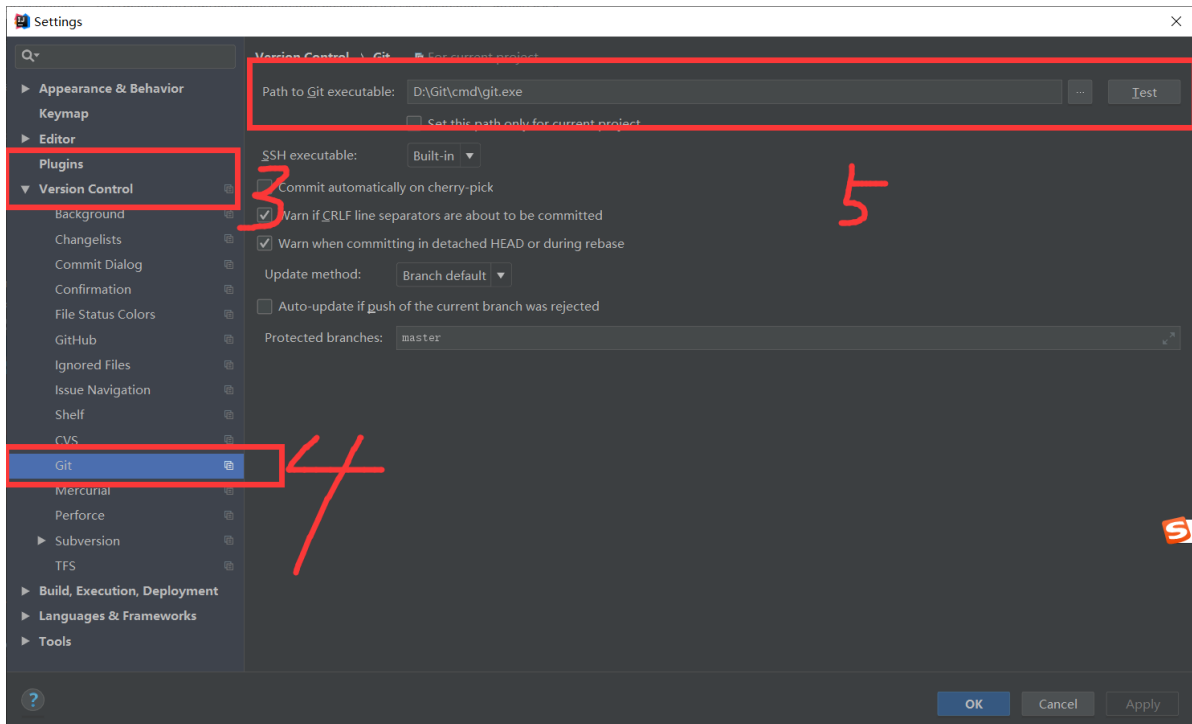
这是第一次免密push，在之后就不需要密码了

10.idea整合Git

10.1整合

设置流程



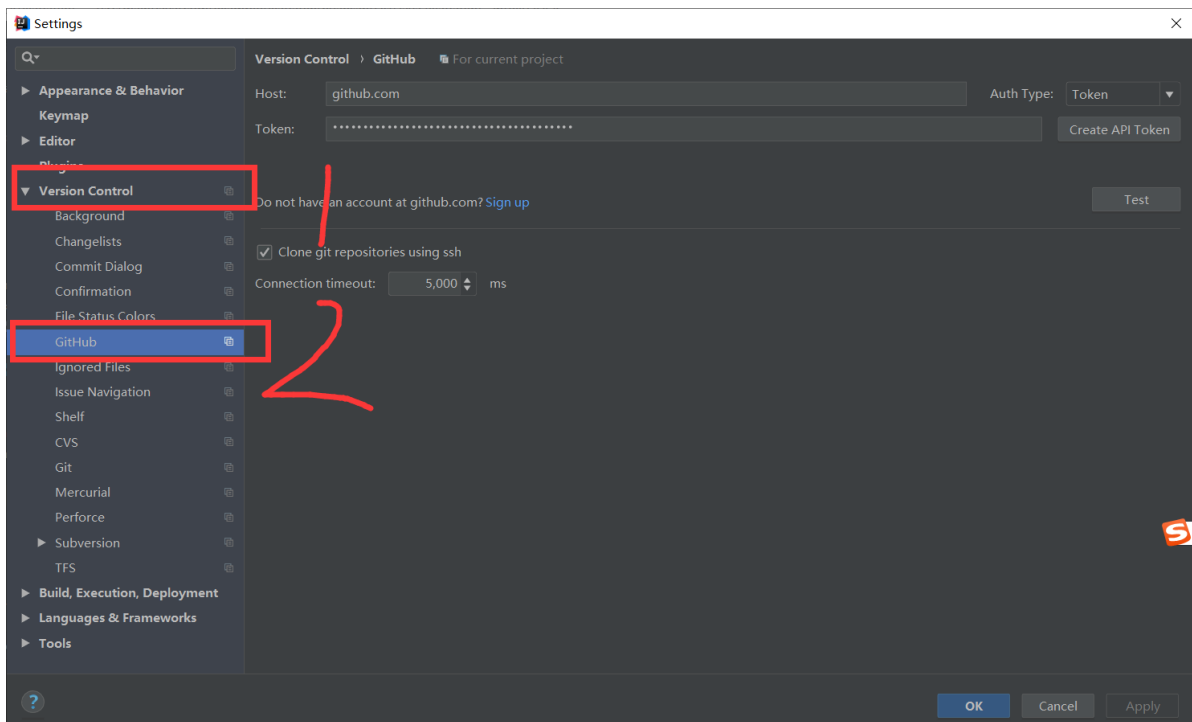


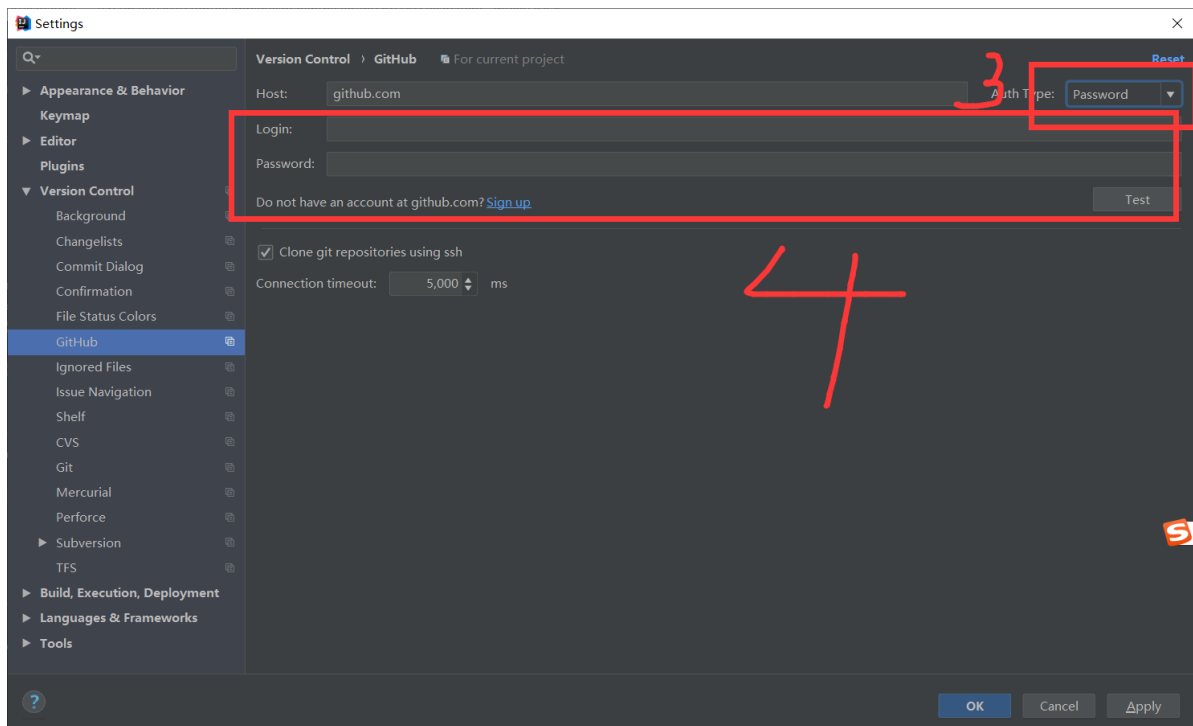
注意：



这是在安装目录的cmd下的git.exe

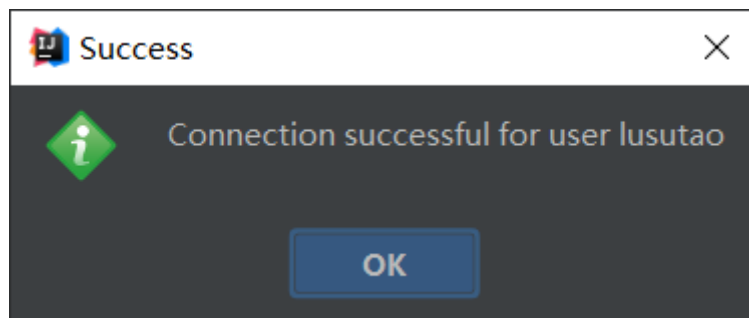
配置完Git，接下来配置github



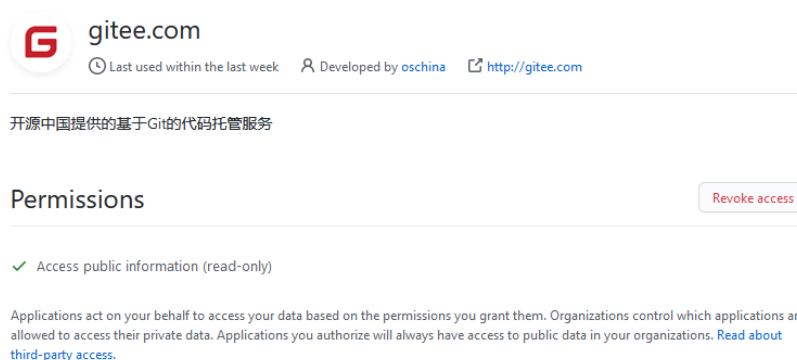
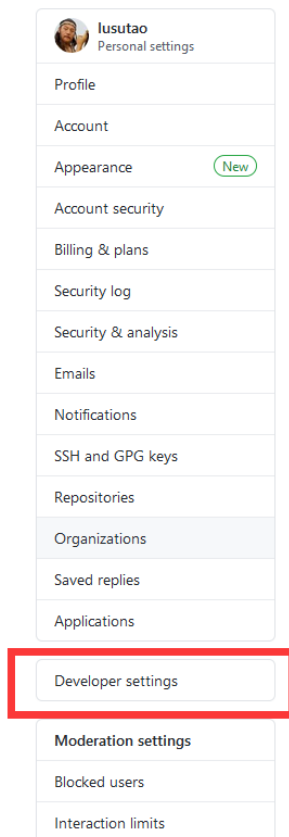
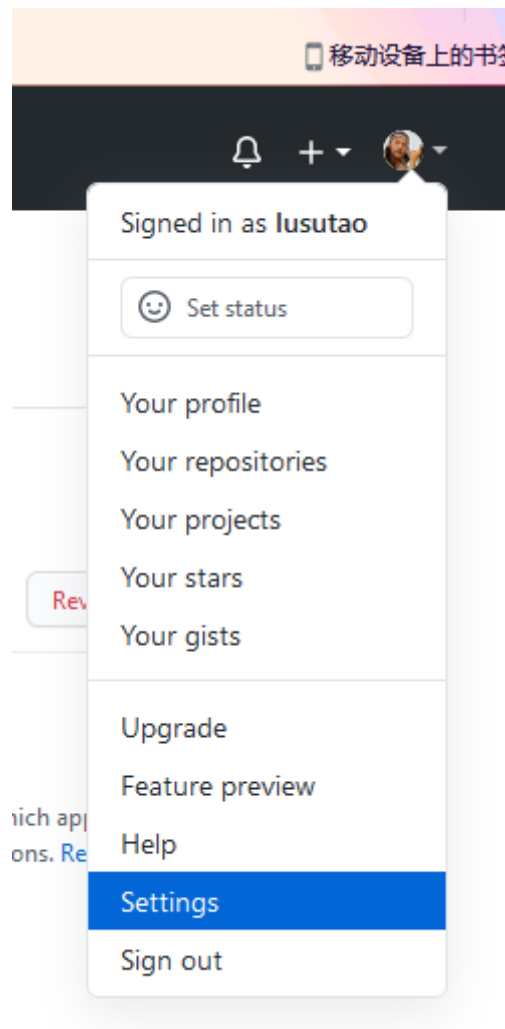


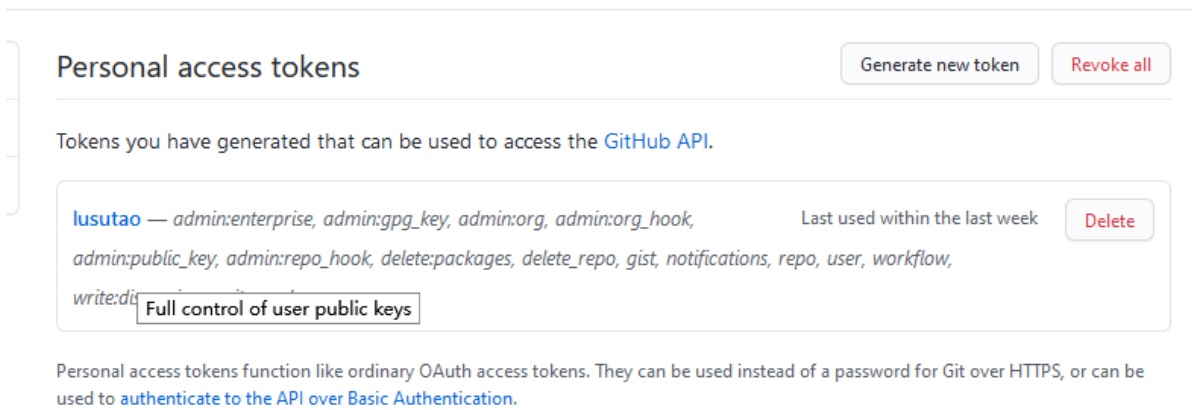
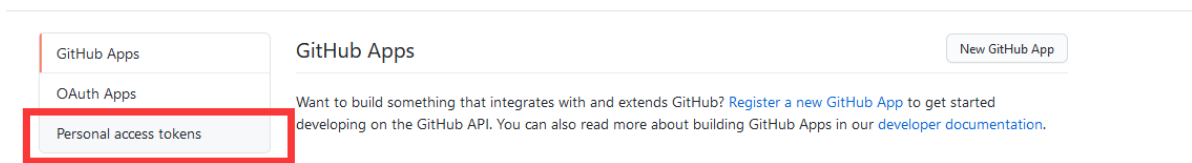
login填写github账号

password填写github密码，最后点击 **Test** 按钮，如果出现：

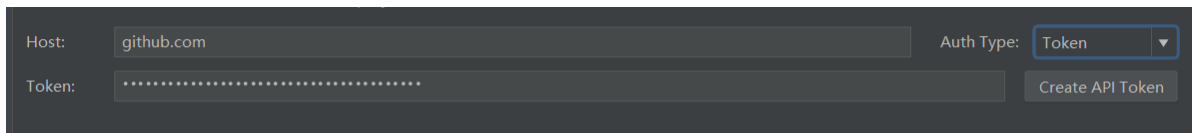


则说明配置成功，如果失败，可以使用第二种方式：在Auth Type的按钮中将password改为token，
接下来去登陆github，选择setting

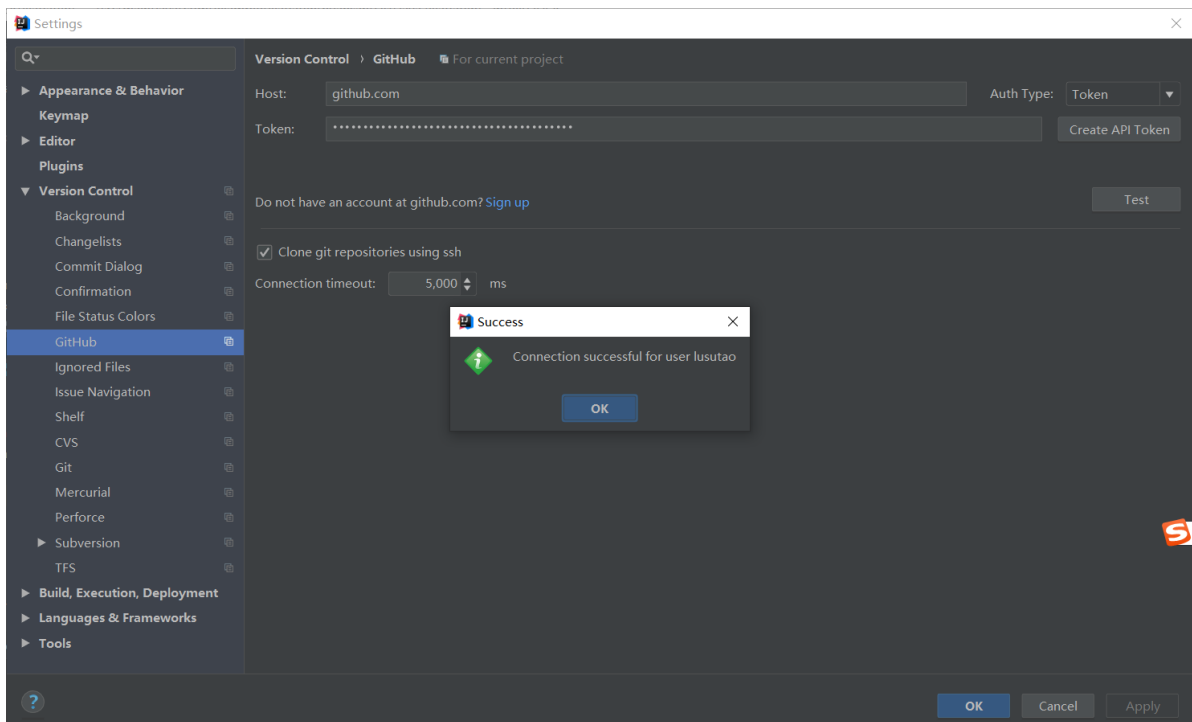




在这里便可以找到自己账号独有的token，将其复制到

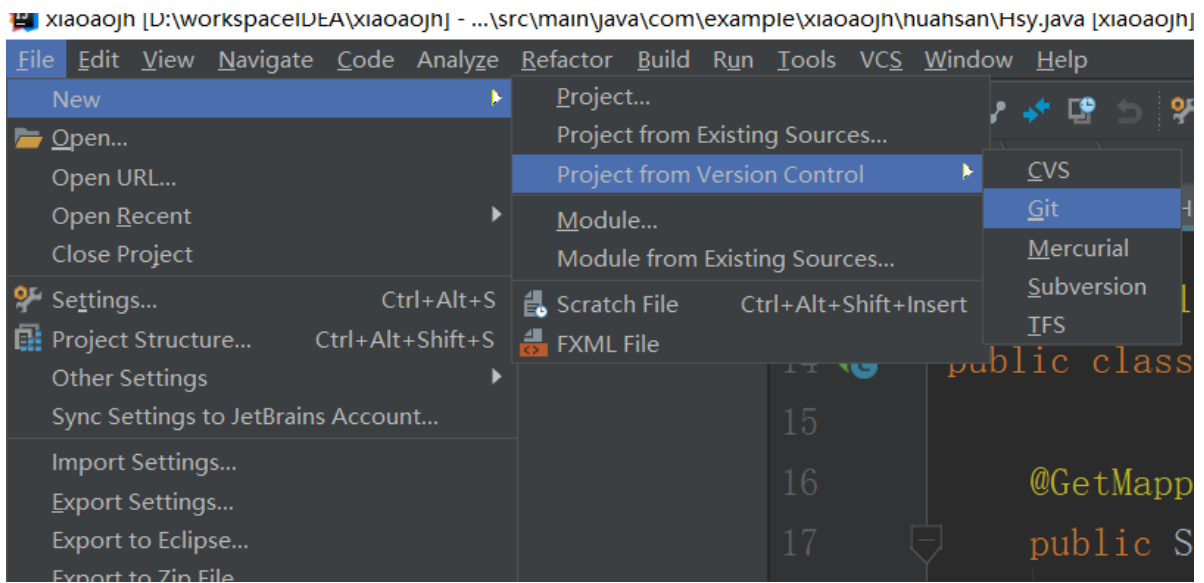


点击Test，出现

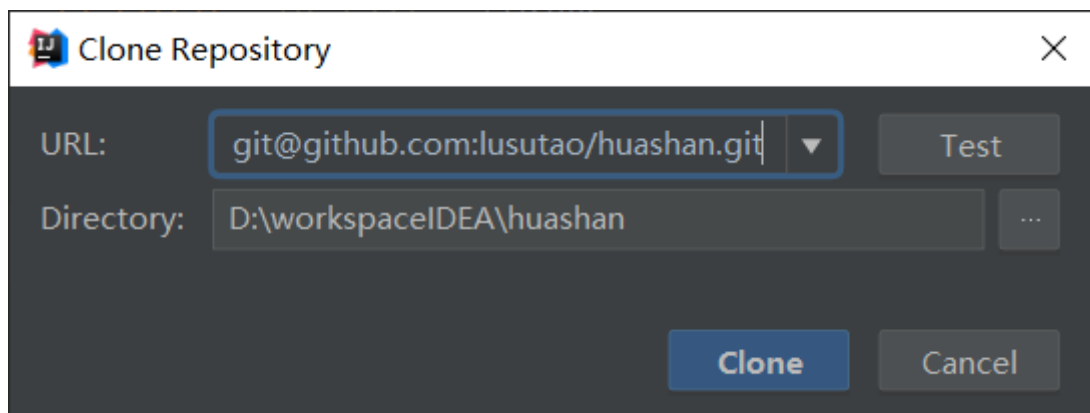


便说明配置成功

10.2 拉取



点击出现

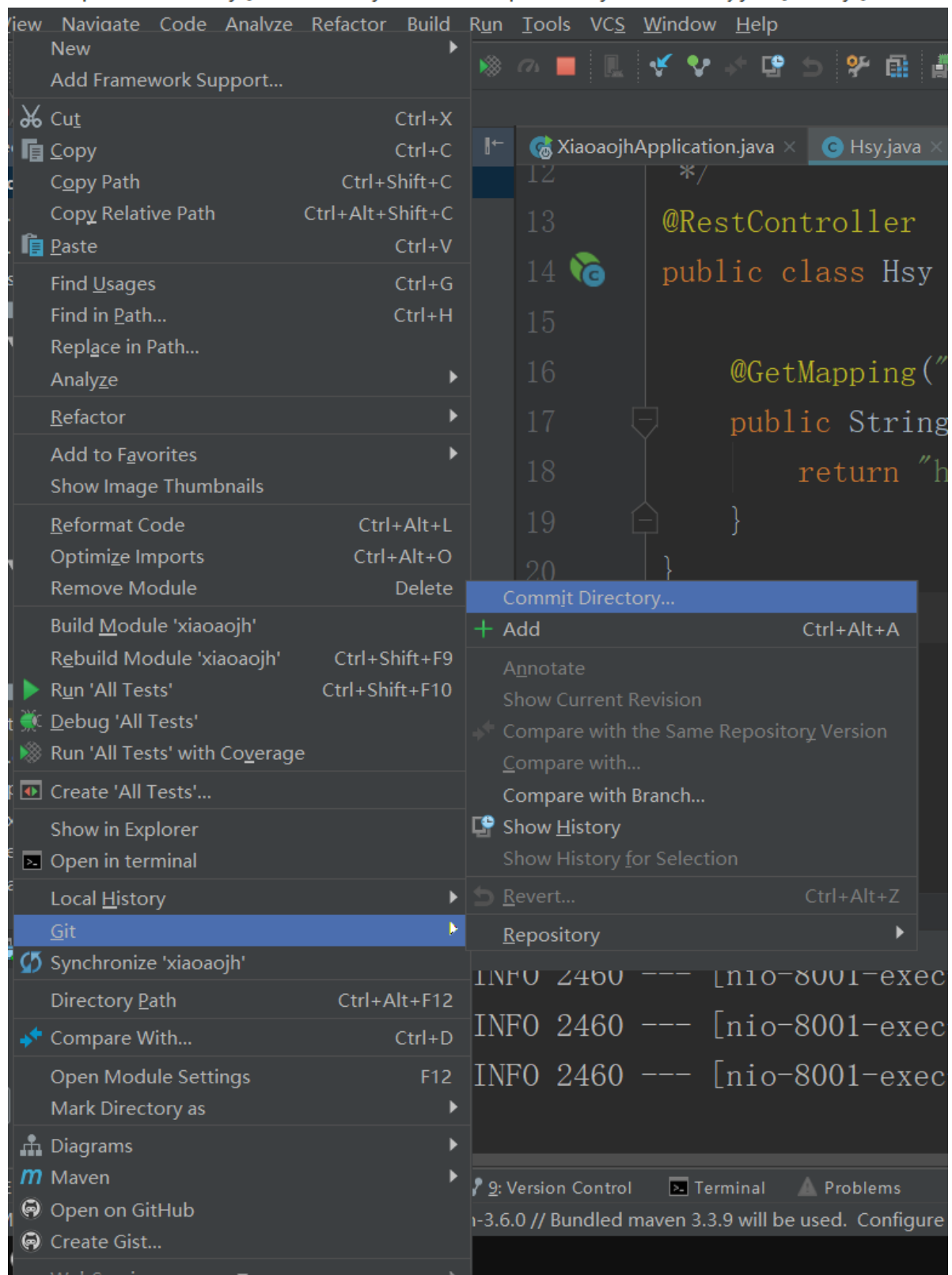


url填写git项目的地址

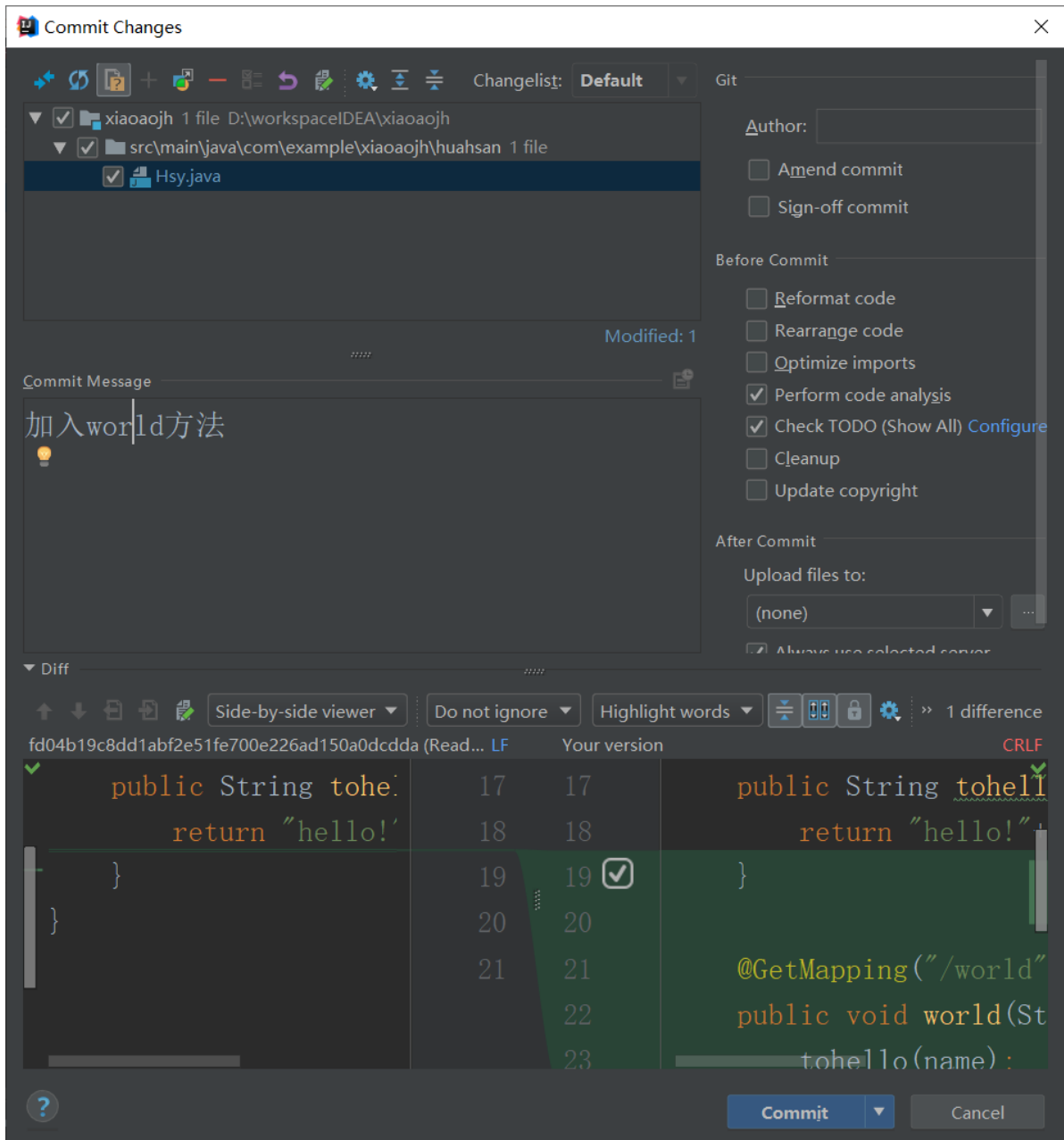
directory为放到本地的地址

填完之后直接clone就可以了，第一次使用会弹出输入线上git网站的用户名和密码。这样就完成了将项目拉下本地的操作了。

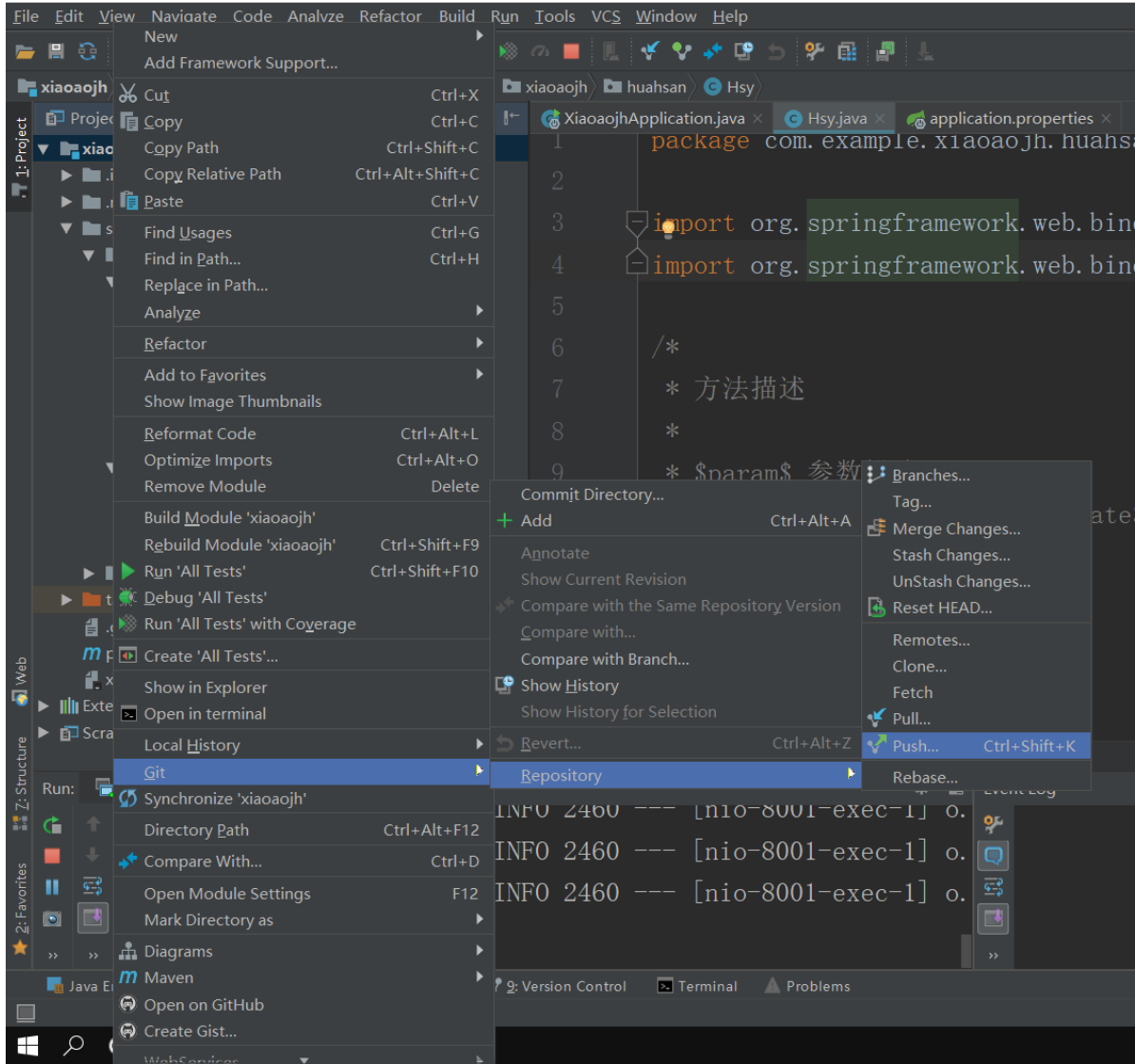
10.2 提交和更新

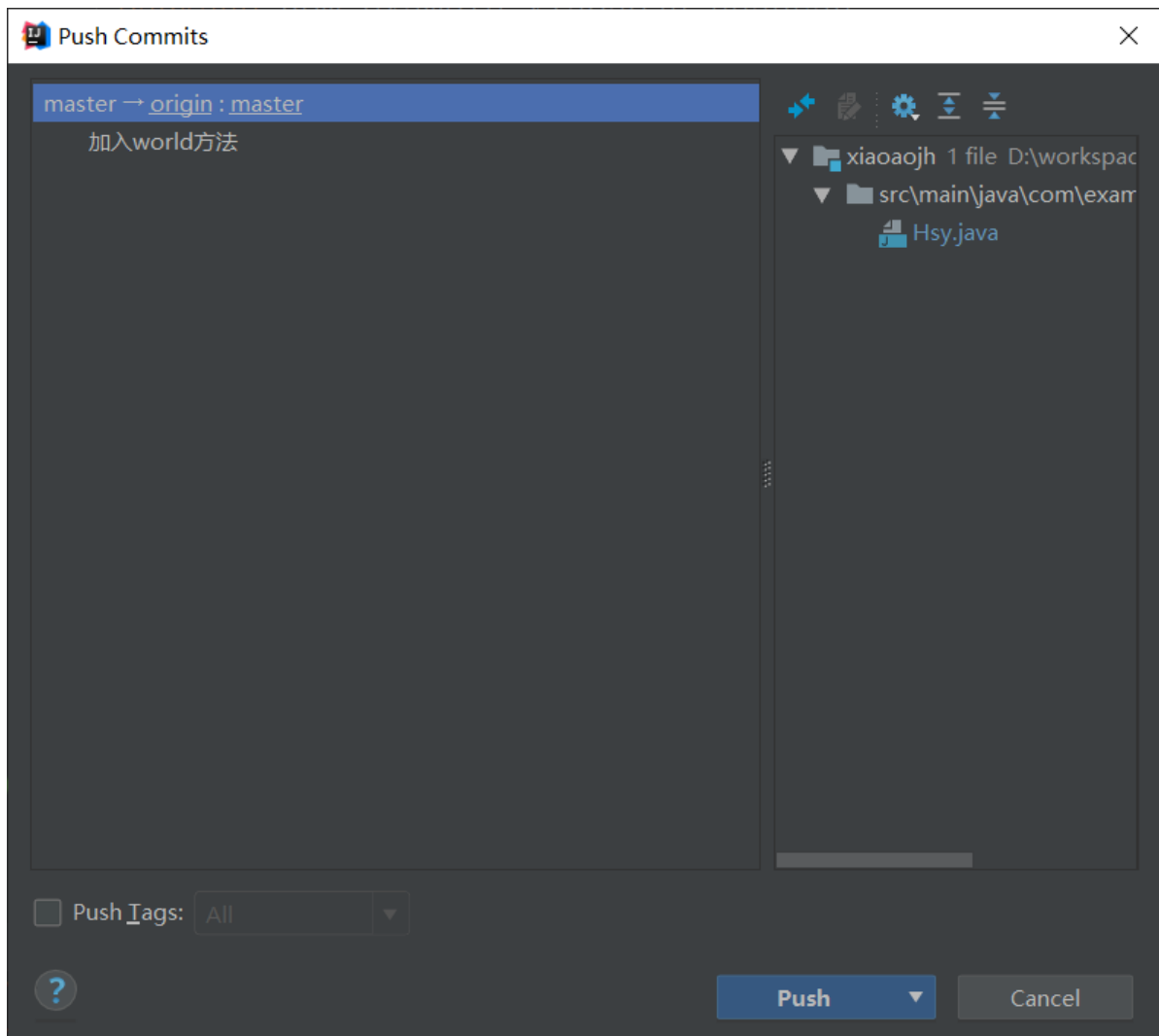


点击以后，如果第一次提交，需要提交所有需要的文件，在之后提交，便只需要提交修改过的文件

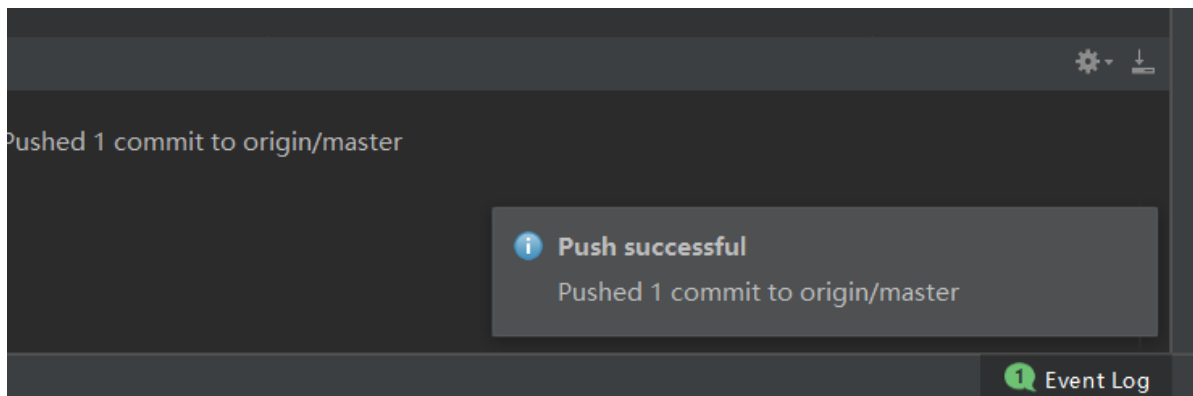


此时将文件提交到本地库中，





点击push将修改更新到远程库（github）中

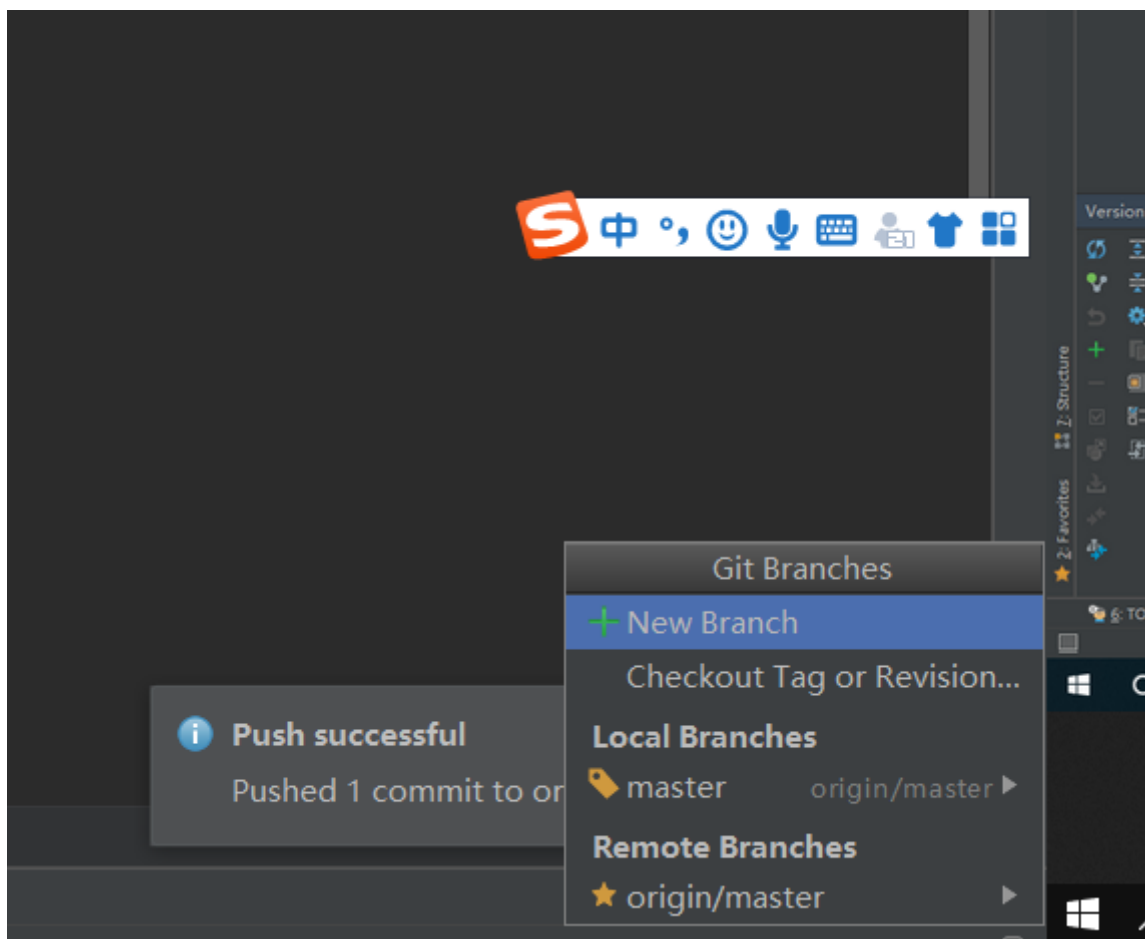


出现这个就说明操作成功。

idea整合git总结

IDEA使用git其实很方便的基本上git所有操作都可以完成。

在IDEA的最右下角有切换分支的功能，合并功能，有兴趣可以自己再深入摸索：



11.gitee



选择信息：

请选择一个模板

☐ 使用仓库模板创建并包含所有分支和历史提交

仓库名称

huashan

归属

lusutao

/

huashan

仓库地址: <https://gitee.com/lusutao/huashan>

仓库介绍 非必填

华山

是否开源

☒ 私有 ☐ 公开

私有仓库的非仓库成员无法访问该仓库的代码和其他任何形式的资源

私有仓库最多支持 5 人协作（如拥有多个私有仓库，所有协作人数总计不得超过 5 人）

企业仓库，更适合使用 Gitee 企业版，[了解更多 >>](#)

选择语言

Java

添加 .gitignore

Actionscript

简易的命令入门教程:

Git 全局设置:

```
git config --global user.name "lusutao"
git config --global user.email "8426802+lusutao@user.noreply.gitee.com"
```

创建 git 仓库:

```
mkdir huashan
cd huashan
git init
touch README.md
git add README.md
git commit -m "first commit"
git remote add origin https://gitee.com/lusutao/huashan.git
git push -u origin master
```

已有仓库?

```
cd existing_git_repo
git remote add origin https://gitee.com/lusutao/huashan.git
git push -u origin master
```

具体使用和github类似，可以参考github的使用。可以参考

<https://www.cnblogs.com/yiven/p/8465054.html>

12.tortoisegit的安装和使用

TortoiseGit 简称 tgit，中文名海龟Git。TortoiseGit是一个开放的GIT版本控制系统的源客户端。

TortoiseGit 支持Winxp/vista/win7/Win10，提供有中文版支持。

TortoiseGit 可以恢复您的文件的旧版本，并研究如何以及合适改变了历史数据，谁改变了它。下面教程针对使用TortoiseGit 的用户。

12.1 安装

12.1.1 第一步

下载Git

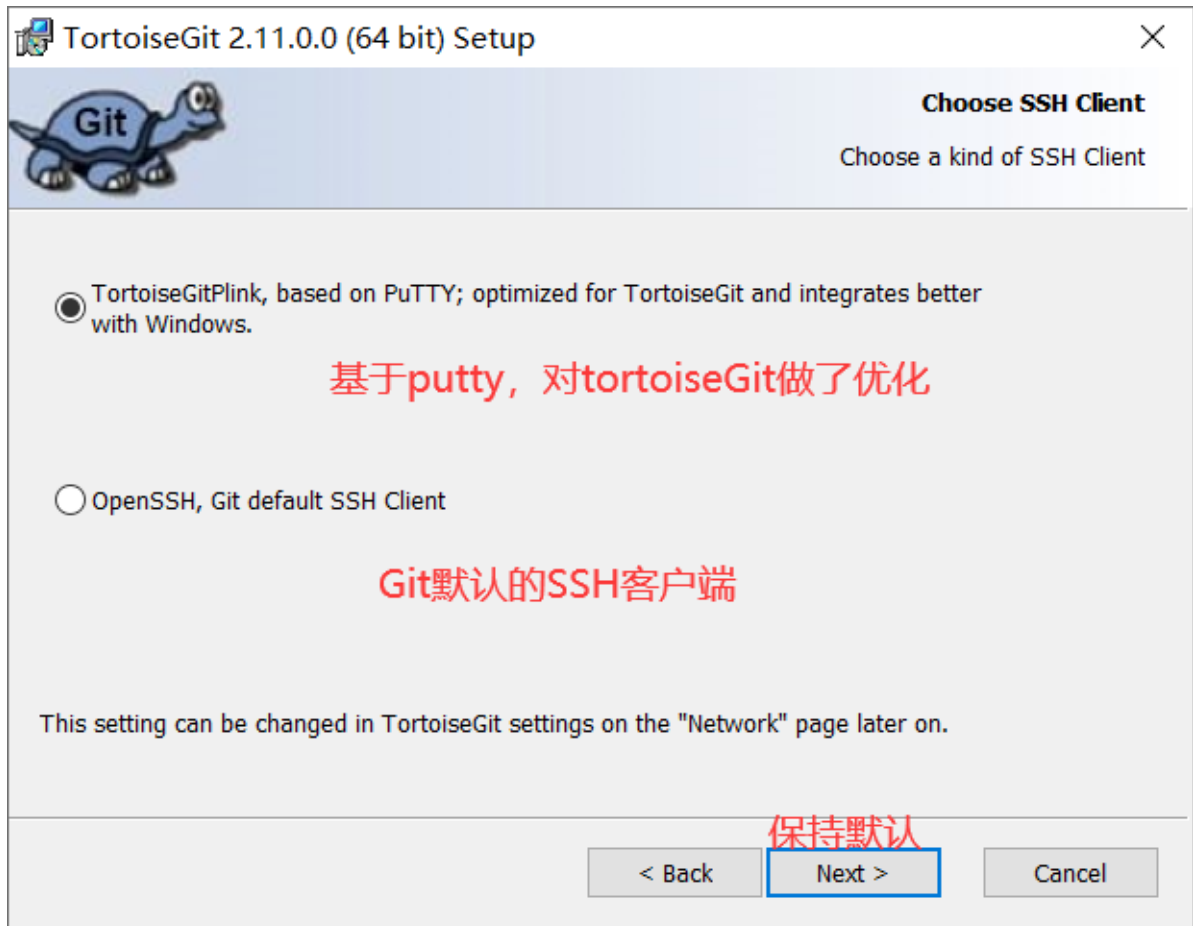
12.1.2 第二步

下载Tortoisegit及中文语言包，并安装

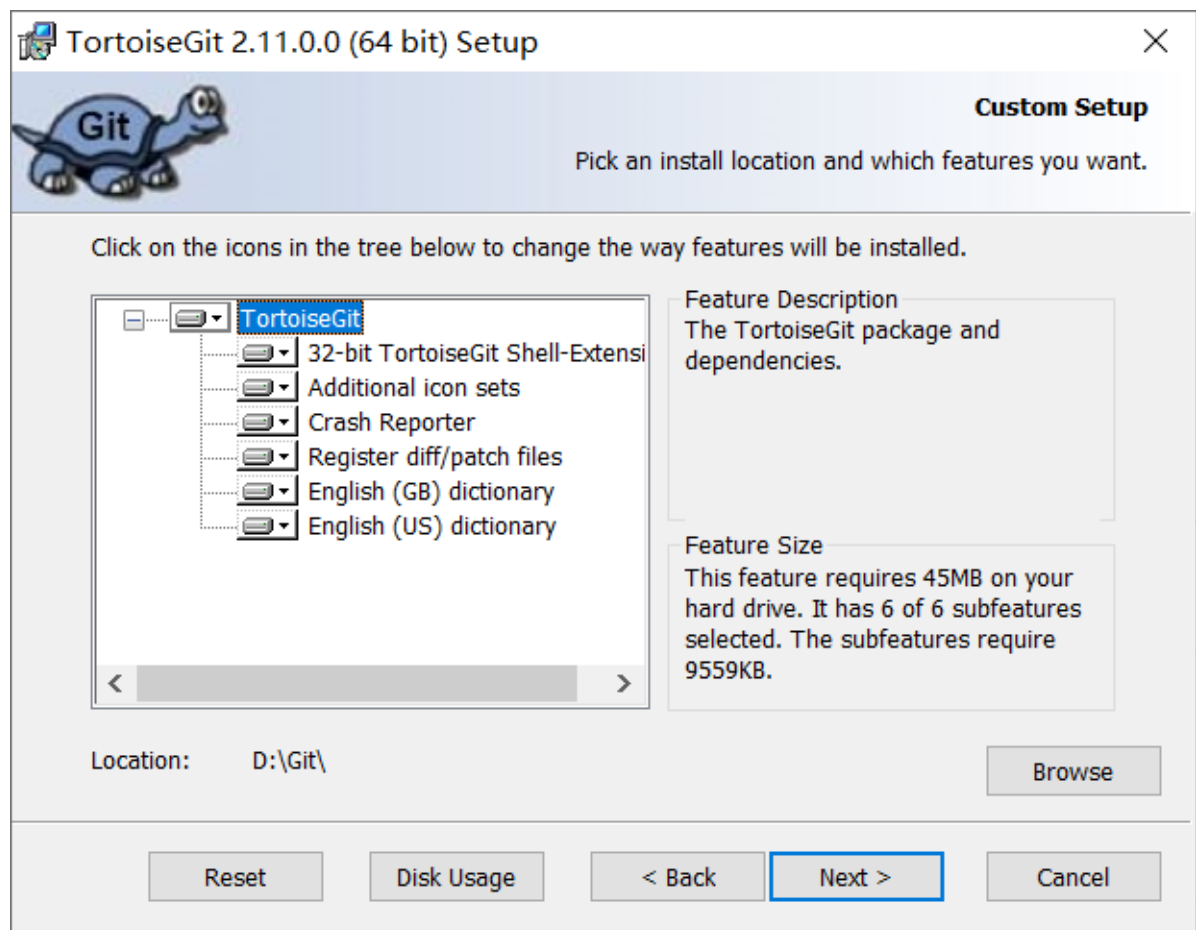
注意 下载安装的顺序

双击安装程序,一直点击下一步

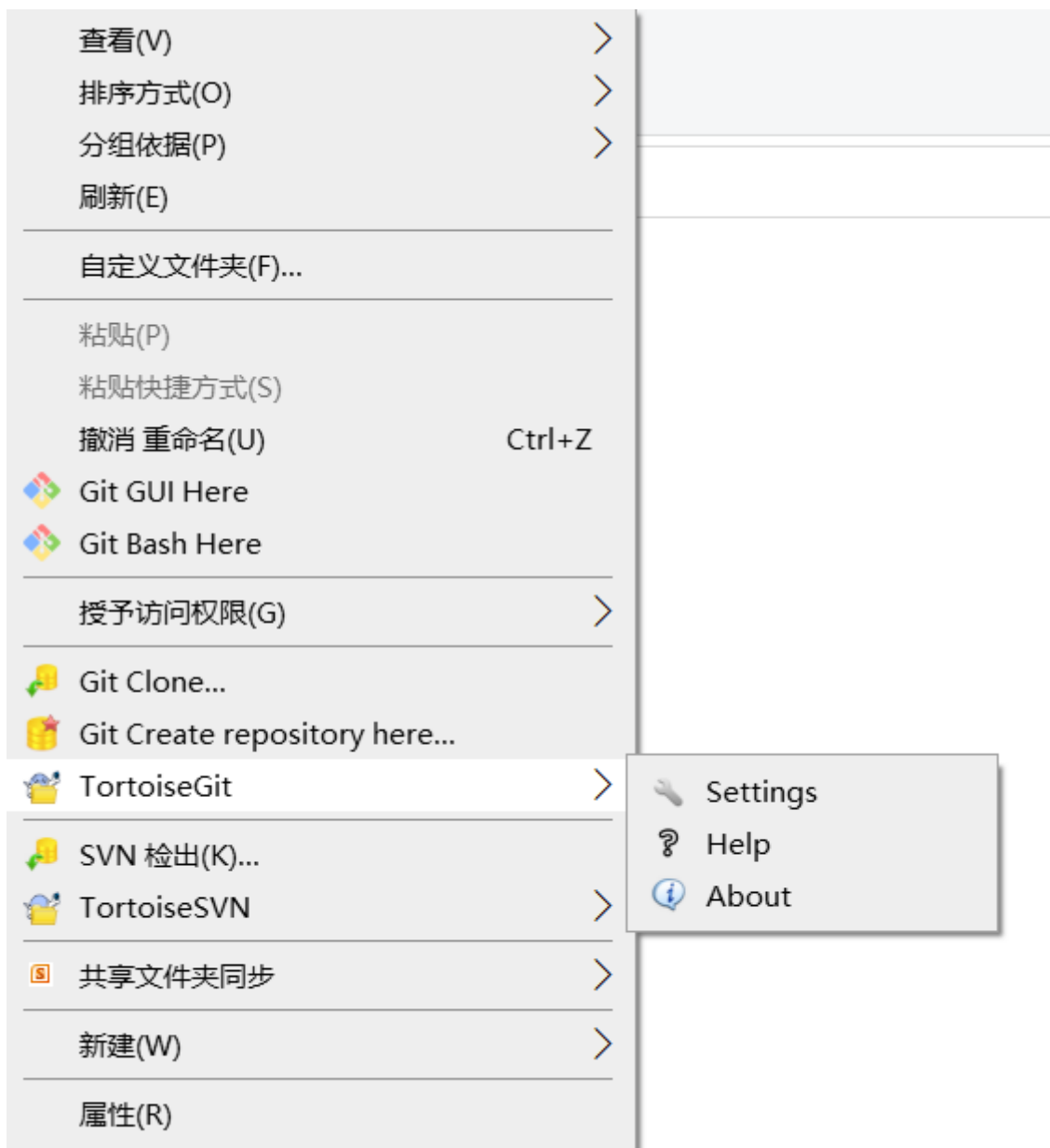
选择SSH客户端. 可以选择 TortoiseGitPlink(位于TortoiseGit安装目录/bin 下), 也可以选择 Git 默认的SSH客户端,位于 Git安装目录/bin/ssh.exe(如果配置了 Path,那直接是 ssh.exe)



下一步:选择安装目录



安装完毕后鼠标右键，会发现多了关于TortoiseGit的选项



12.1.3 第三步

安装中文语言包，双击语言包

12.2配置

首次启动，需要进行“首次启动向导”（后续也可以重新进行首次启动向导）

一直点击下一步，知道出现OpenSSH，选择该选项，再点击生成密钥对。此时需要在Generate中不停移动鼠标才可以生成密钥，否则将不会生成

PutTY Key Generator ? X

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAkSB9u9C2tEDbB8ta35EqXtF3Ot0kdUE
+o31R/ZMUUVh5QLWiKT9F1vBKxwQLK+LfgksHvHGVrX5BuR
+N3FfBgWWskbrR6xNO/bru2VwFKvejewipWRL0mT1wNTMs8bP98pimbMqhj1EnL3uXZH7
Uxv5BAwK1CBgOMN8MmAMI7dGUAcuGI/K7T6zvZcbWfCOErB40EHZc4Q90gQjWFArNAY
wTqBZSp9a7hgQmaChHllqosGFC/SH87mKF
```

Key fingerprint: ssh-rsa 2048 33:a2:ec:69:0a:be:f3:14:fb:5f:70:a1:3e:c1:25:af

Key comment: rsa-key-20201210

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

Parameters

Type of key to generate:

☒ RSA ☐ DSA ☐ ECDSA ☐ Ed25519 ☐ SSH-1 (RSA)


Number of bits in a generated key: 2048

将选择的内容复制到git服务器上，并且将密钥保存在一个不常用的目录上，以保证不会被删除。

接下来的密钥配置和上文的ssh免密登陆操作一致。

SSH keys New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **小乌龟**
SHA256:ZqTOXS1EMBMuEn1Y98TCcQe2v29Q1HB3v0FOCGDAJY
Added on 10 Dec 2020
Never used — Read/write Delete

[Check out our guide to generating SSH keys or troubleshoot common SSH Problems.](#)

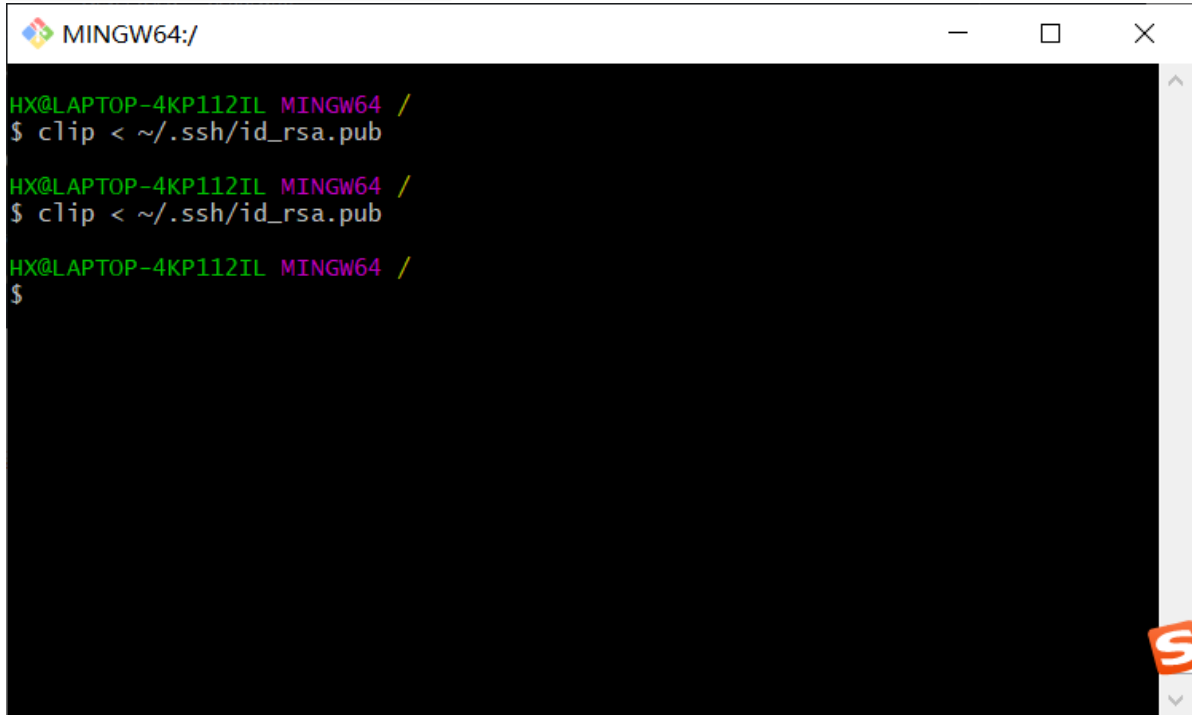
GPG keys New GPG key

There are no GPG keys associated with your account.

[Learn how to generate a GPG key and add it to your account.](#)

注意·

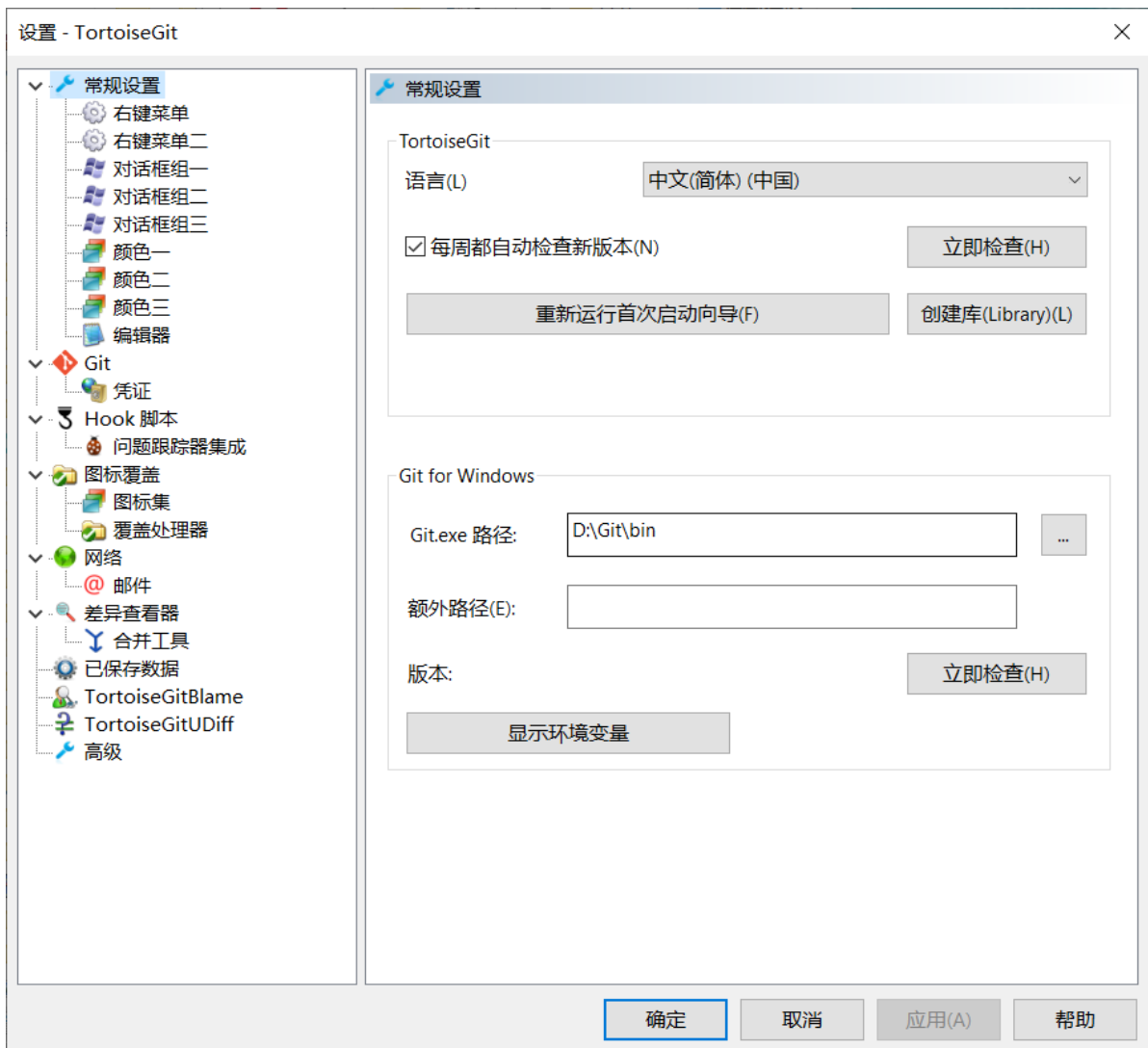
此时复制的密钥格式有误，需要去转格式



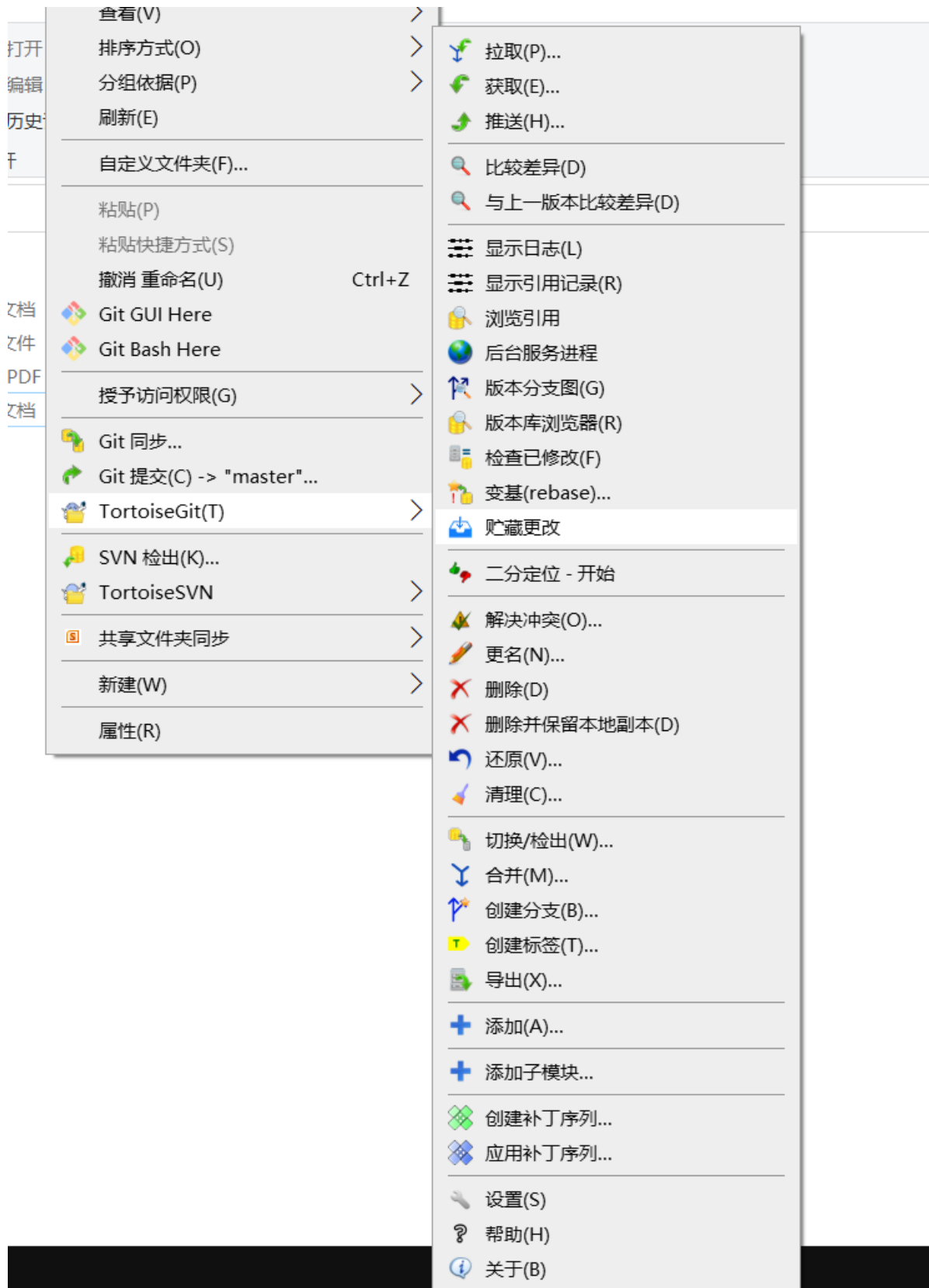
```
MINGW64:/  
HX@LAPTOP-4KP112IL MINGW64 /  
$ clip < ~/.ssh/id_rsa.pub  
HX@LAPTOP-4KP112IL MINGW64 /  
$ clip < ~/.ssh/id_rsa.pub  
HX@LAPTOP-4KP112IL MINGW64 /  
$
```

在git bash中输入 `clip < ~/.ssh/id_rsa.pub` 输入该命令后，则可以添加，并且此时需要将github中以前设置的ssh key删除

安装并设置完语言包后的界面

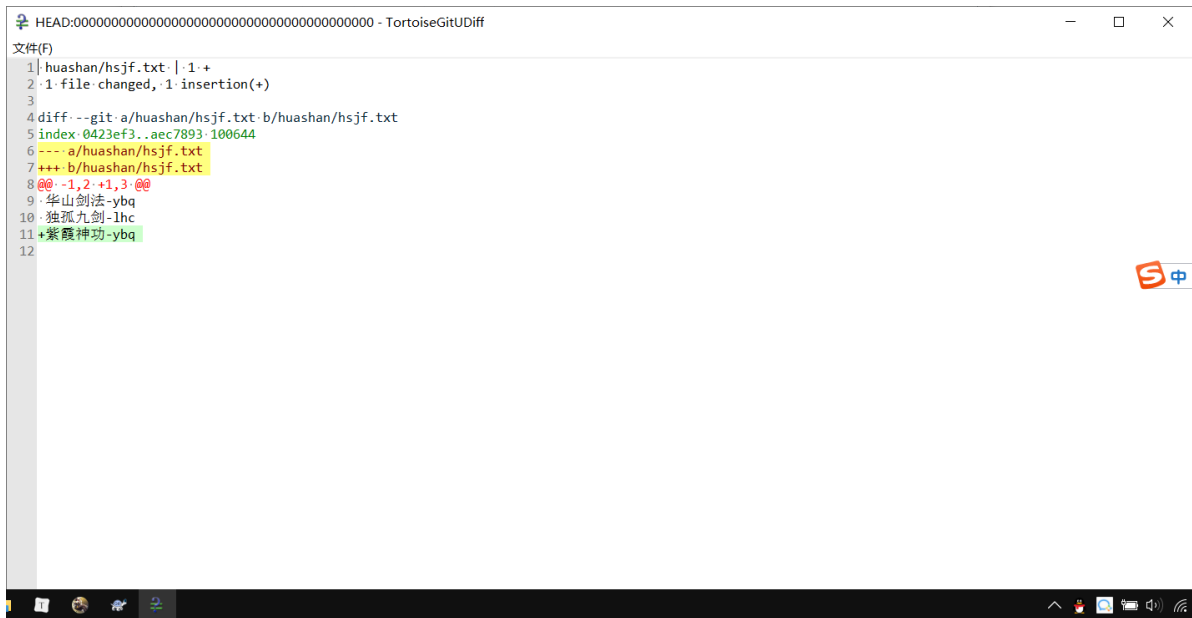


12.3使用

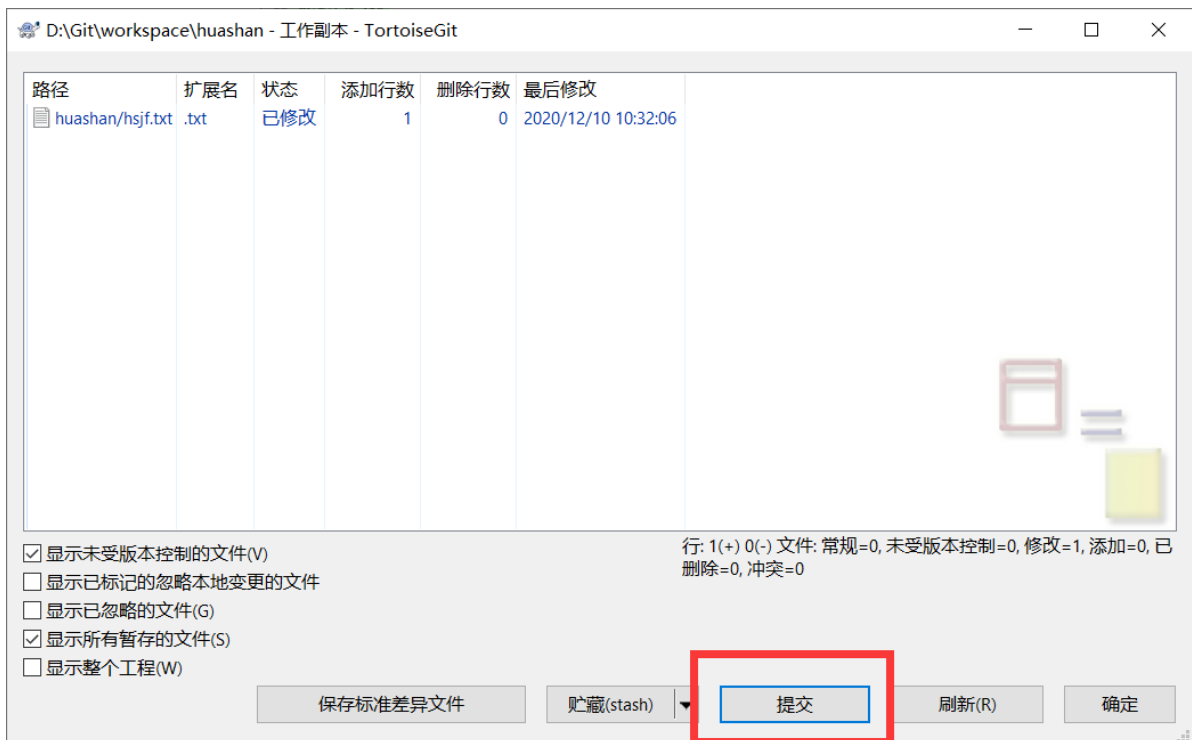


可以直接进行拉取推送对比不同等操作

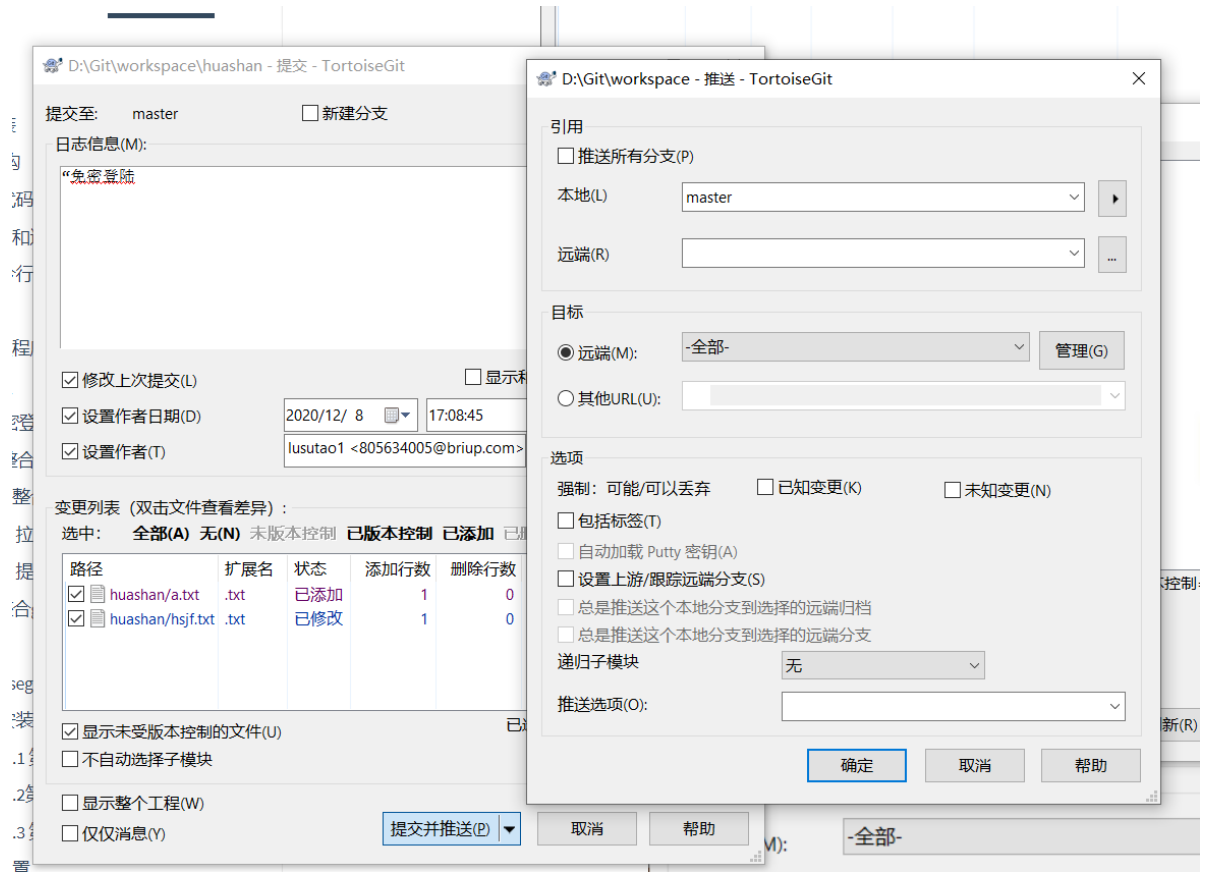
例如对比不同：

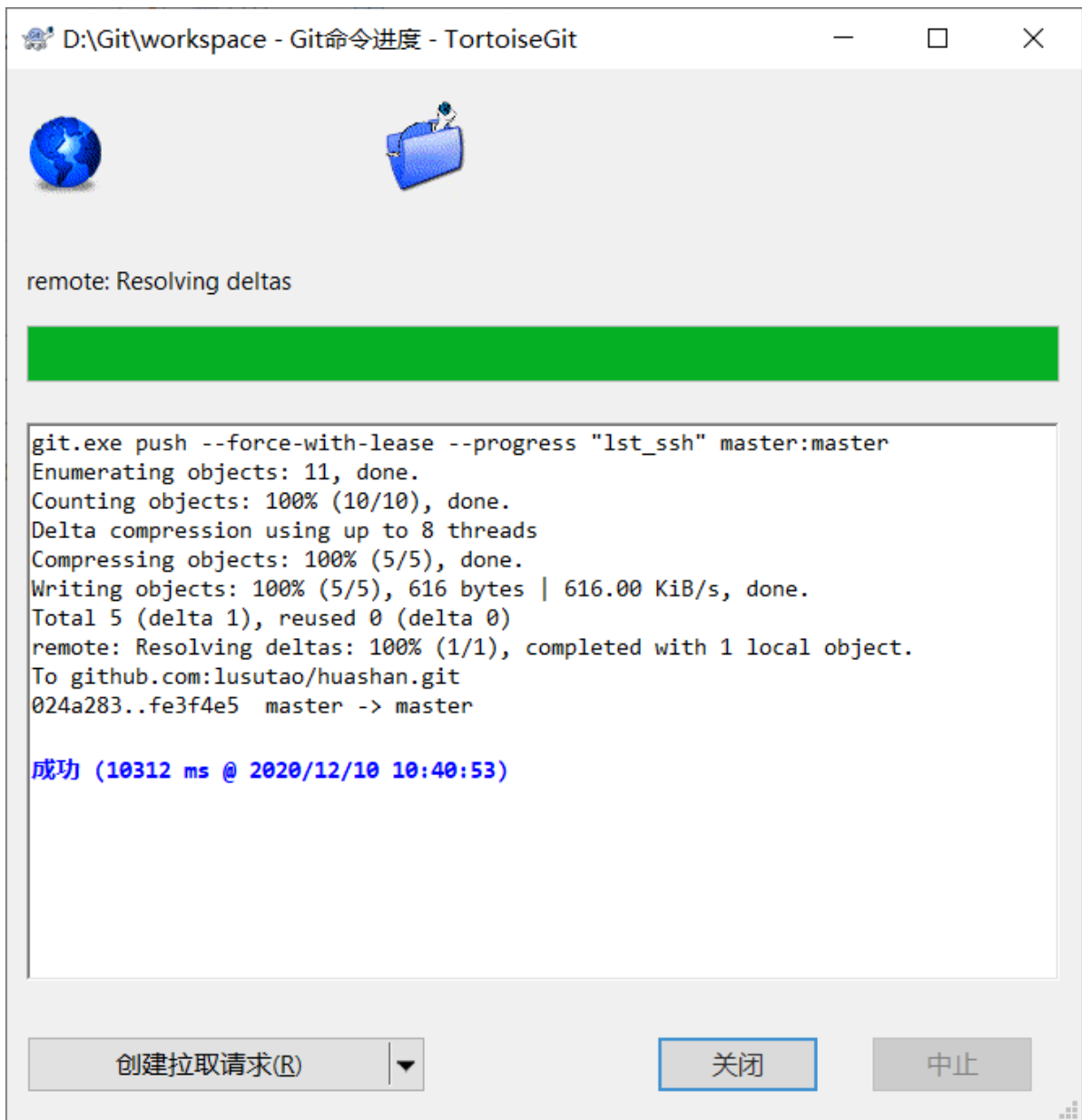


并将其提交



并且可以选择提交方式





提交并推送成功。

注意

如果想要push，需要先pull后commit之后才可以push，否则会报错