Git的概念

【1】Git技术：公司必备，一定要会

【2】Git概念：

Git是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的项目。

【3】什么是版本控制?

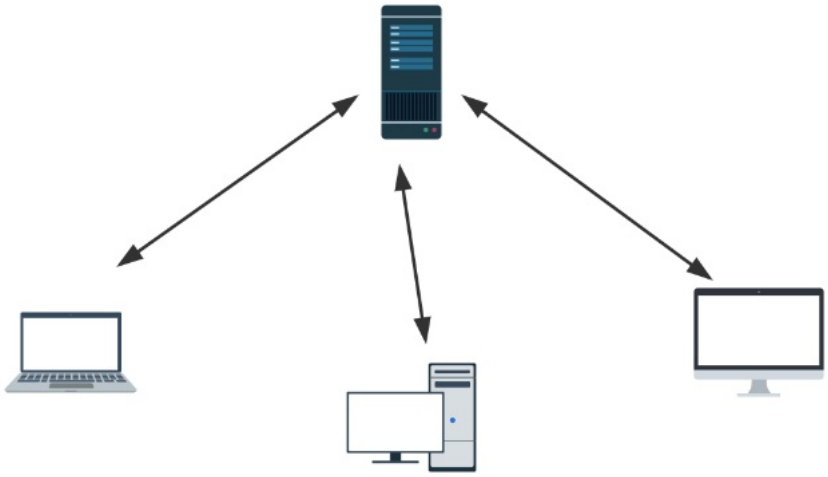
版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统 。

【4】为什么要使用版本控制?

软件开发中采用版本控制系统是个明智的选择。
有了它你就可以将某个文件回溯到之前的状态,甚至将整个项目都回退到过去某个时间点的状态。
就算你乱来一气把整个项目中的文件改的改删的删，你也照样可以轻松恢复到原先的样子。
但额外增加的工作量却微乎其微。你可以比较文件的变化细节,查出最后是谁修改了哪个地方,从而找出导致怪异问题出现的原因，又是谁在何时报告了某个功能缺陷等等。

【5】版本控制系统的分类：

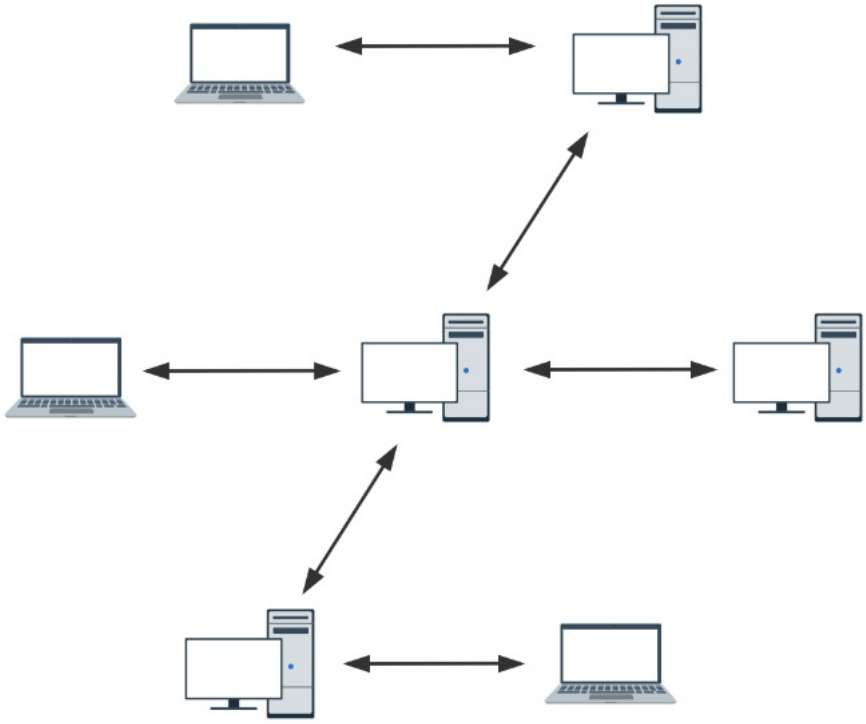
❉集中化的版本控制系统：



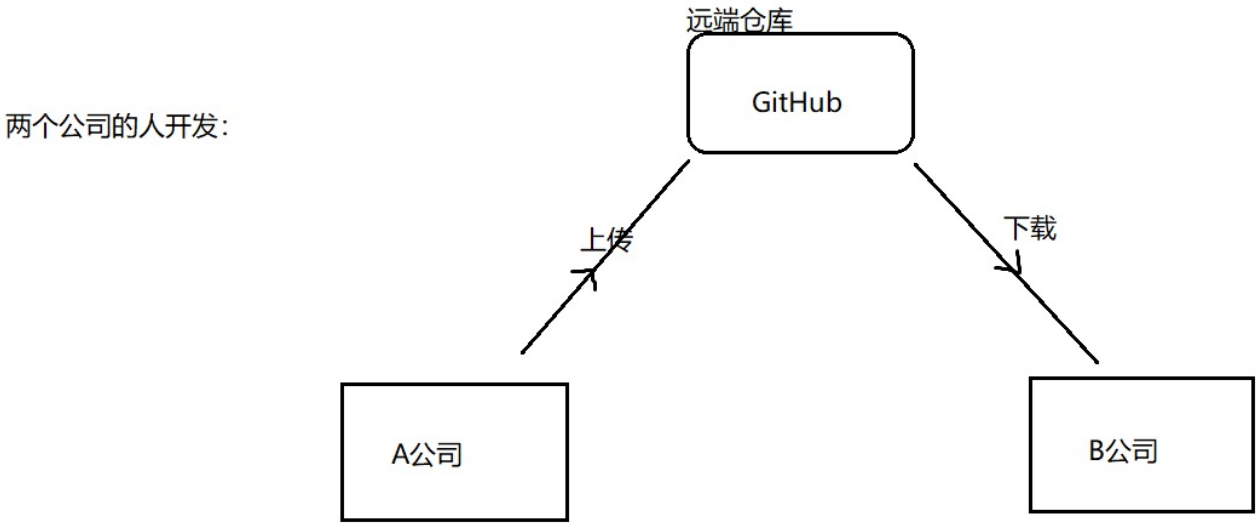
集中化的版本控制系统诸如CVS, SVN 以及Perforce 等，都有一个单一的集中管理的服务器,保存所有文件的修订版本，而协同工作的人们都通过客
户端连到这台服务器,取出最新的文件或者提交更新。多年以来,这已成为版本控制系统的标准做法,这种做法带来了许多好处,现在,每个人都可以在一定程度上看到项目中的其他人正在做些什么。而管理员也可以轻松掌控每个开发者的权限，并且管理一个集中化的版本控制系统;要远比在各个客户端上维护本地数据库来得轻松容易。
事分两面，有好有坏。这么做最显而易见的缺点是中央服务器的单点故障。如果服务器宕机一小时，那么在这一小时内， 谁都无法提交更新，也就无法协同工作。

❉分布式的版本控制系统

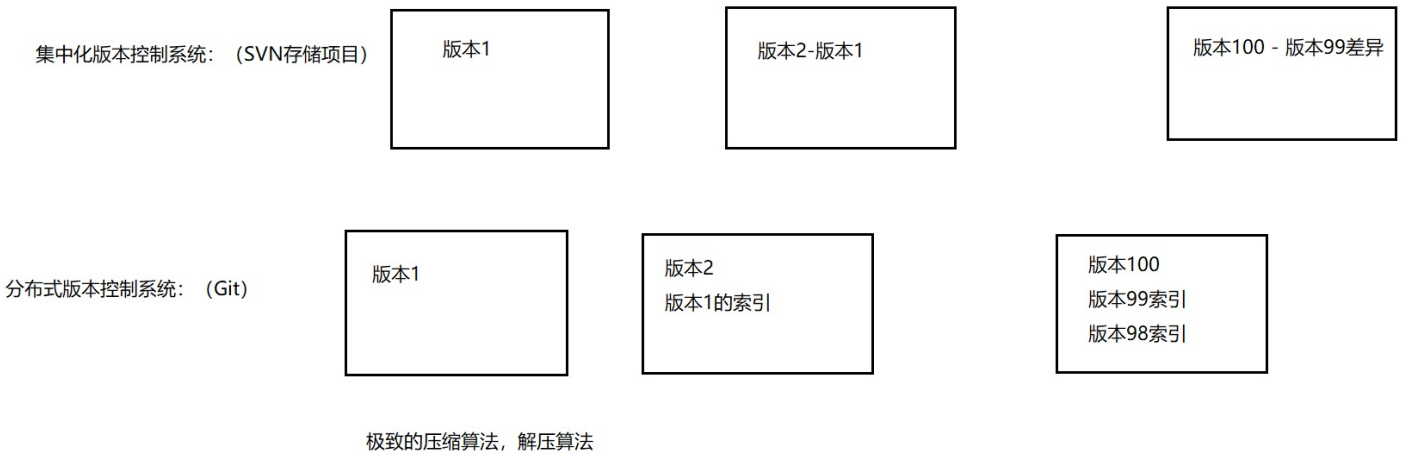
由于上面集中化版本控制系统的那些缺点，于是分布式版本控制系统面世了。
在这类系统中，像Git, BitKeeper 等,客户端并不只提取最新版本的文件快照，而是把代码仓库完整地镜像下来。



更进一步，许多这类系统都可以指定和若干不同的远端代码仓库进行交互。这样，你就可以在同一个项目中分别和不同工作小组的人相互协作。



分布式的版本控制系统在管理项目时存放的不是项目版本与版本之间的差异.它存的是索引(所需磁盘空间很少所以每个客户端都可以放下整个项目的历史记录)



Git简史

Linux ---> 人越来越多 ---->代码优化的越来越好 ---》项目管理工具
---》Bitkeeper ---》终止合作 ---》一周Git --》一个月内将Linux管理在Git上---》开源，免费 ---》用户群大

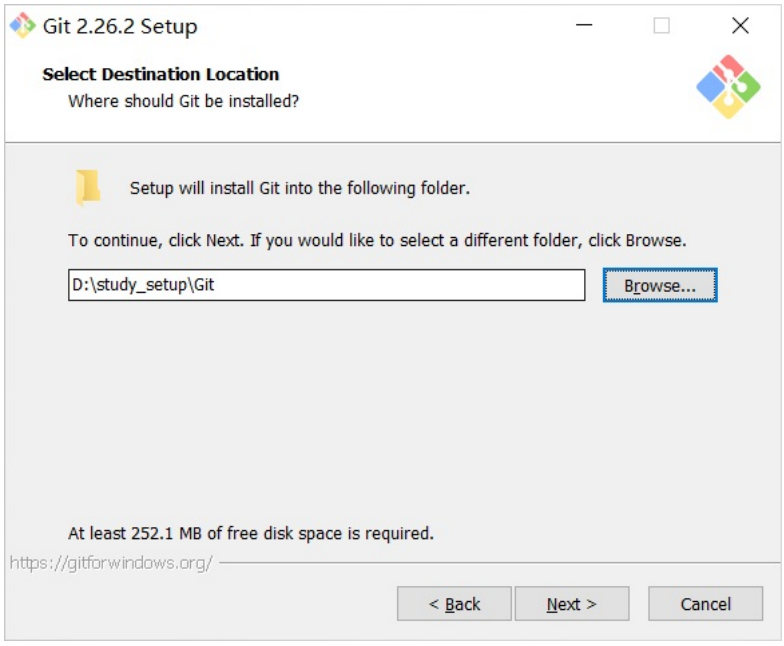
Git的安装过程

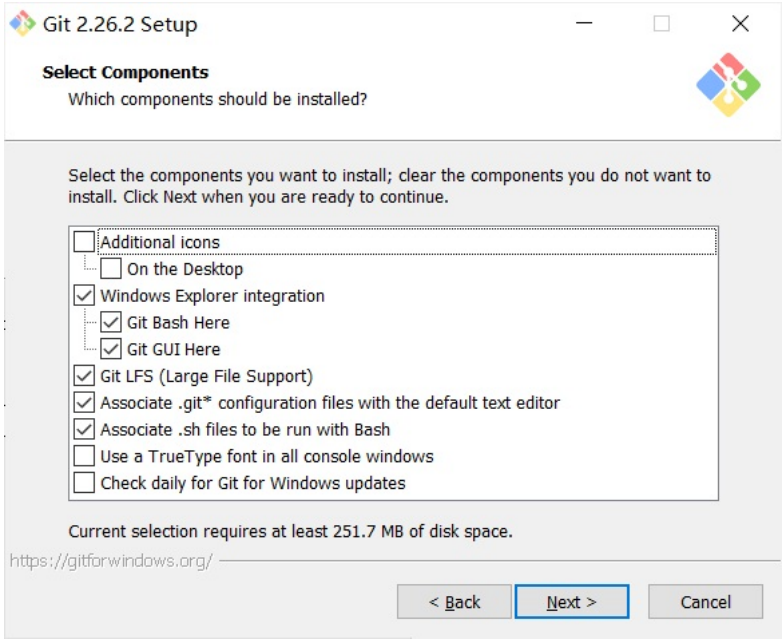
【1】Git官网：

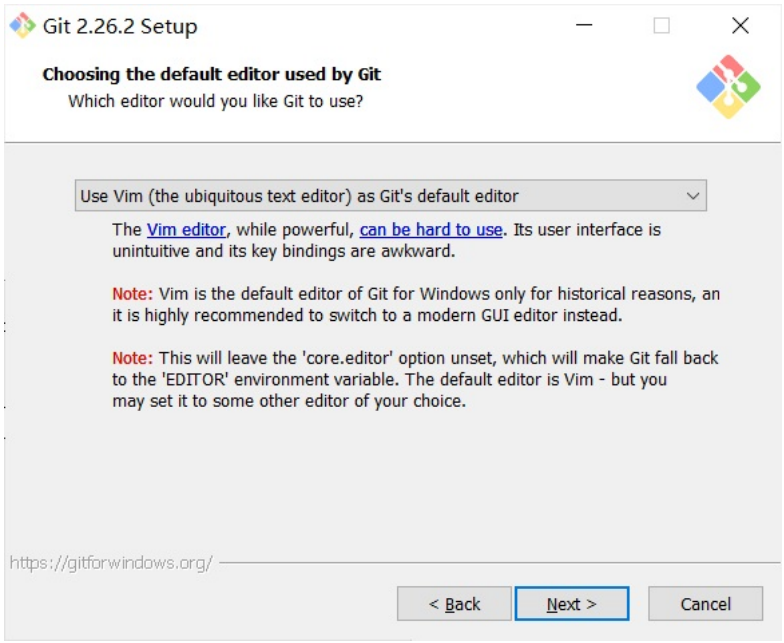
https://git-scm.com/

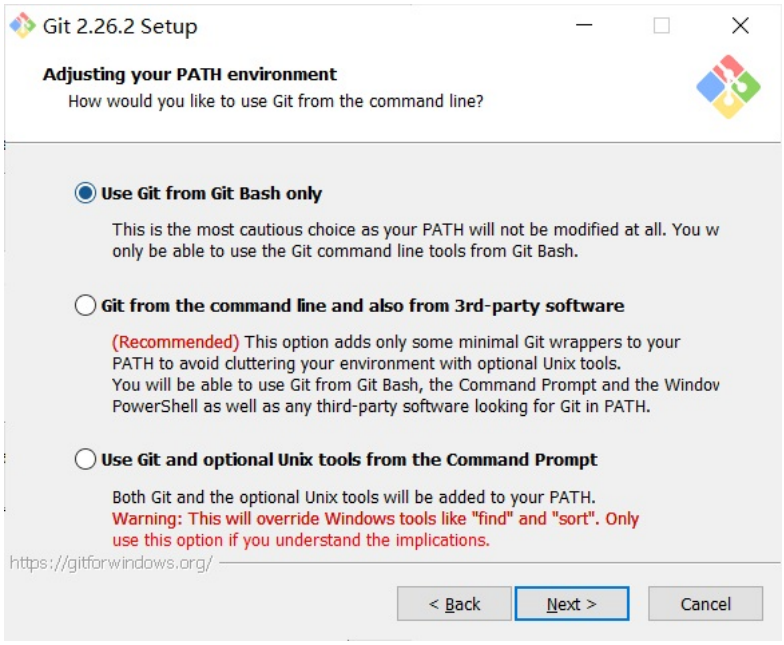
【2】安装过程：

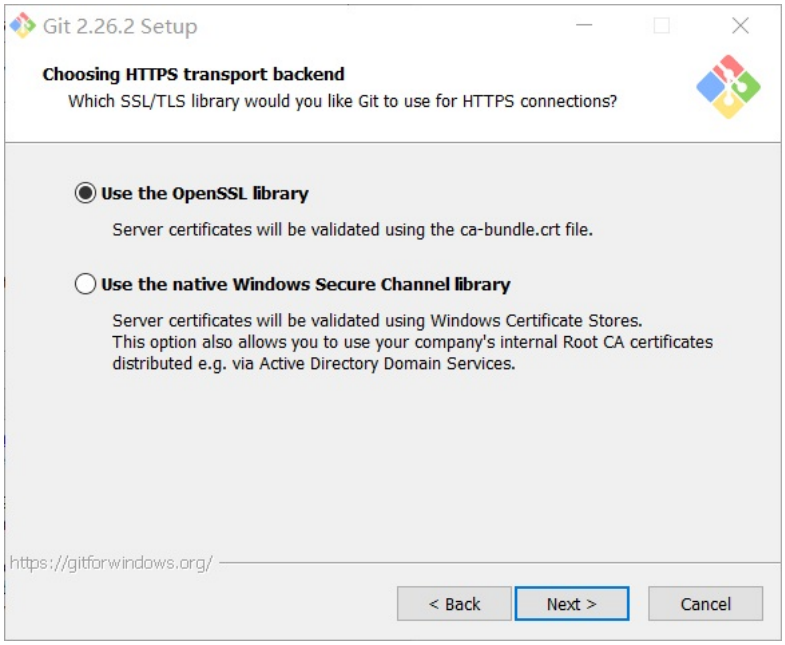
一直下一步

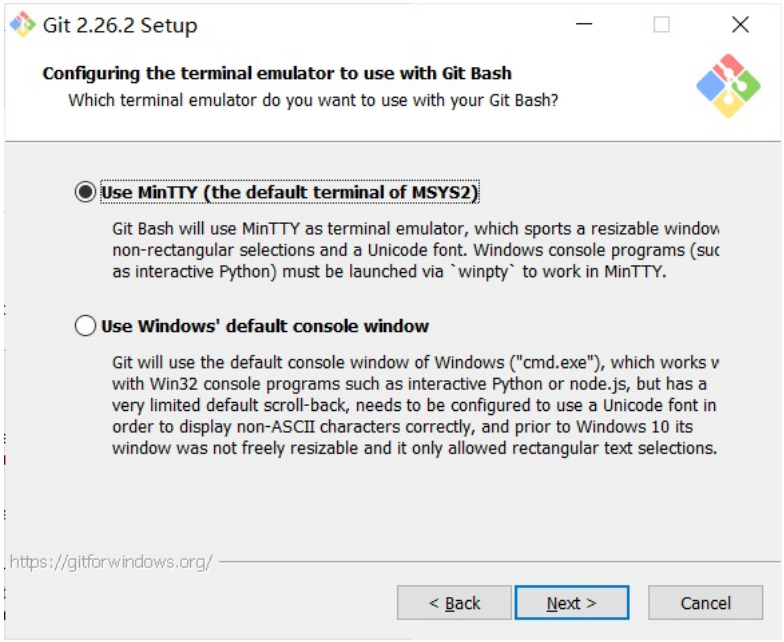



Git 2.26.2 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 251.7 MB of disk space.
https://gitforwindows.org/
< Back
Next >
Cancel

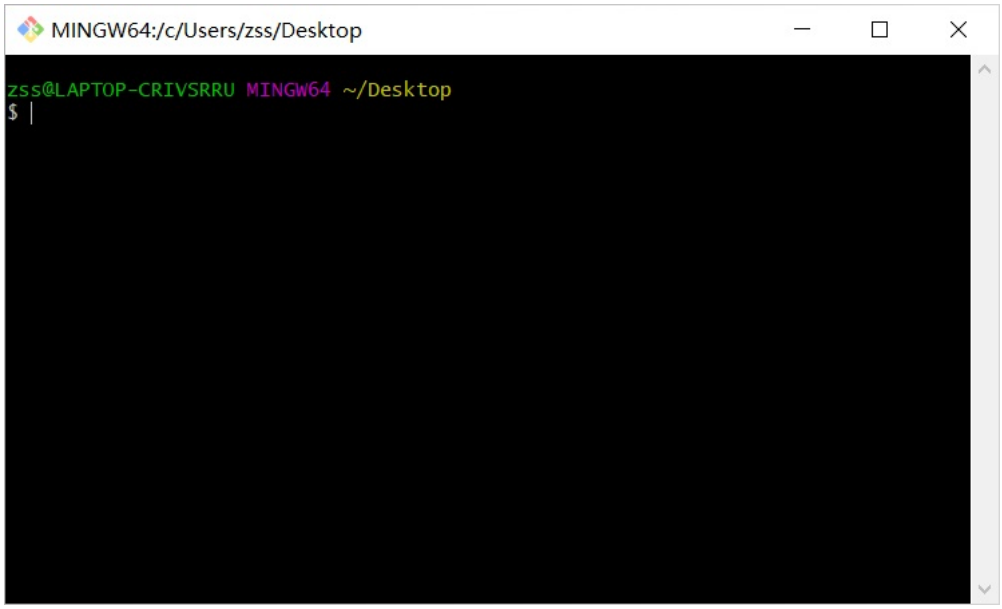

Git 2.26.2 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, an it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
< Back
Next >
Cancel


Git 2.26.2 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You w only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windov PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
< Back
Next >
Cancel

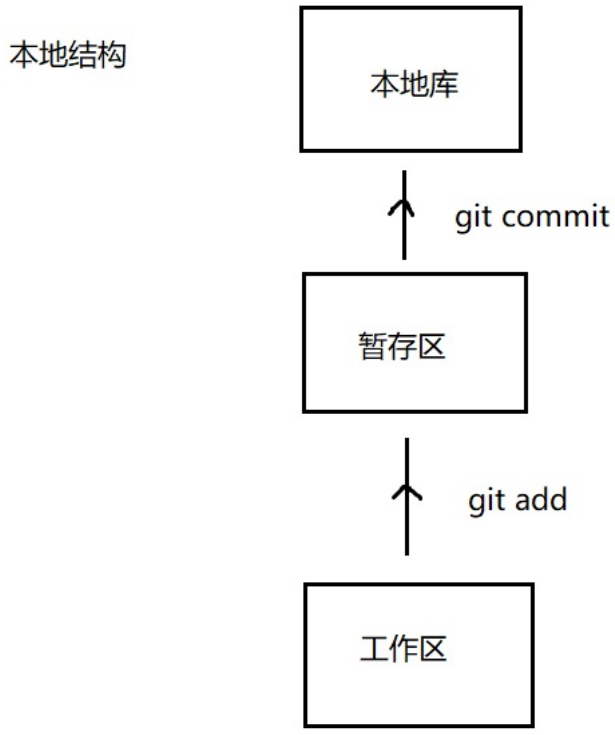




点击Git Bash Here打开Git终端：

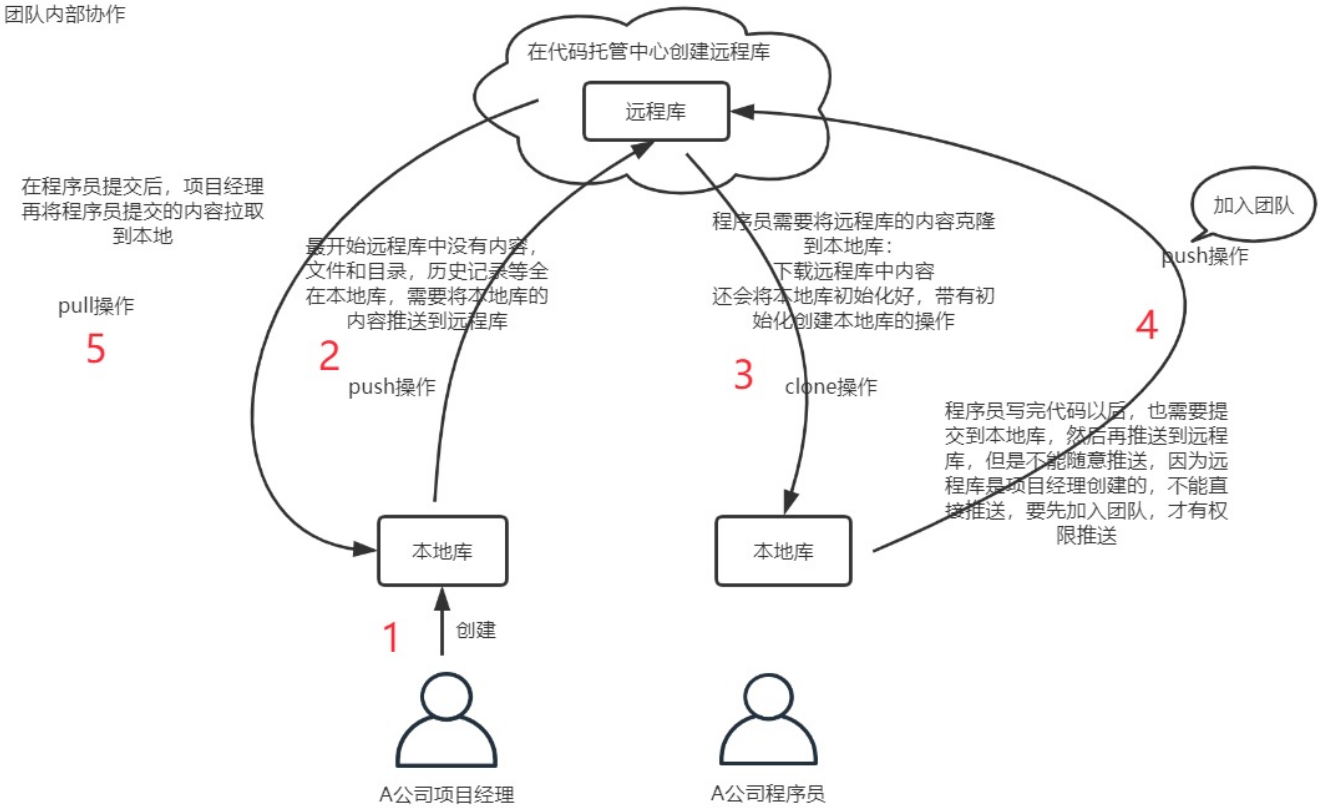


Git结构

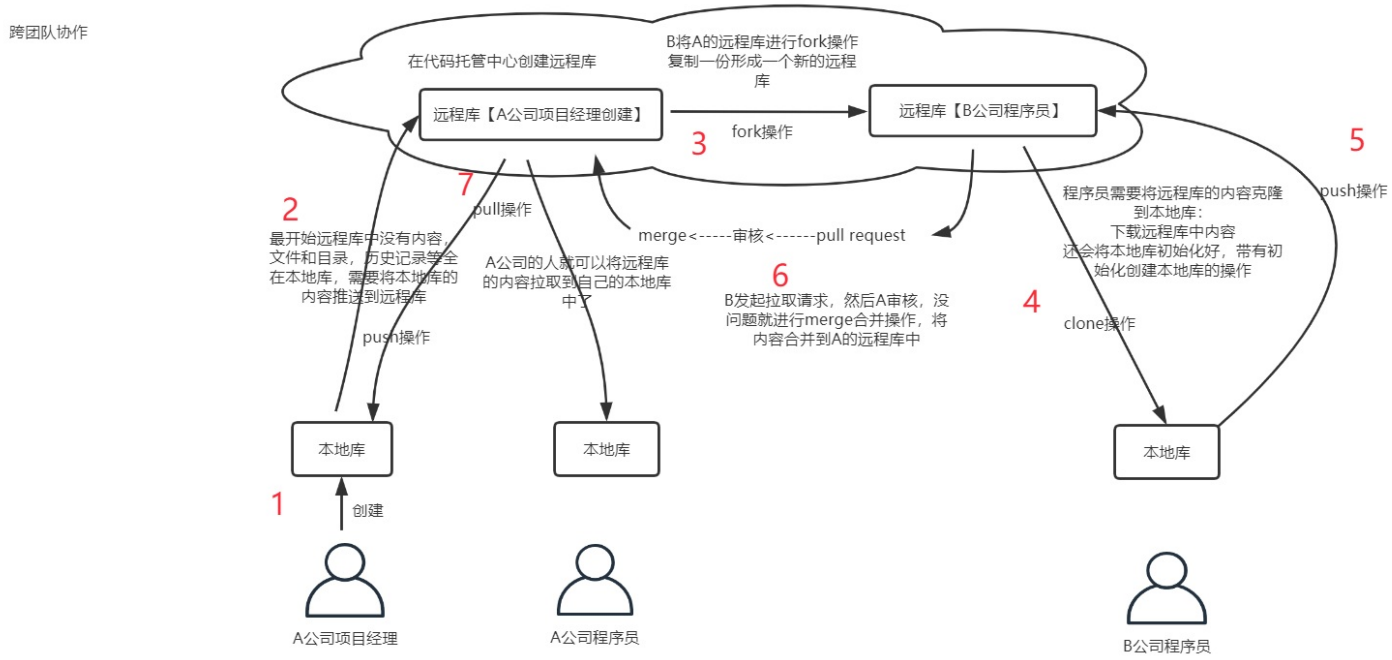


代码托管中心_本地库和远程库的交互方式

【1】代码托管中心是干嘛的呢?
我们已经有了本地库，本地库可以帮我们进行版本控制，为什么还需要代码托管中心呢?
它的任务是帮我们维护远程库，
下面说一下本地库和远程库的交互方式，也分为两种：
（1）团队内部协作



（2）跨团队协作



【2】托管中心种类：

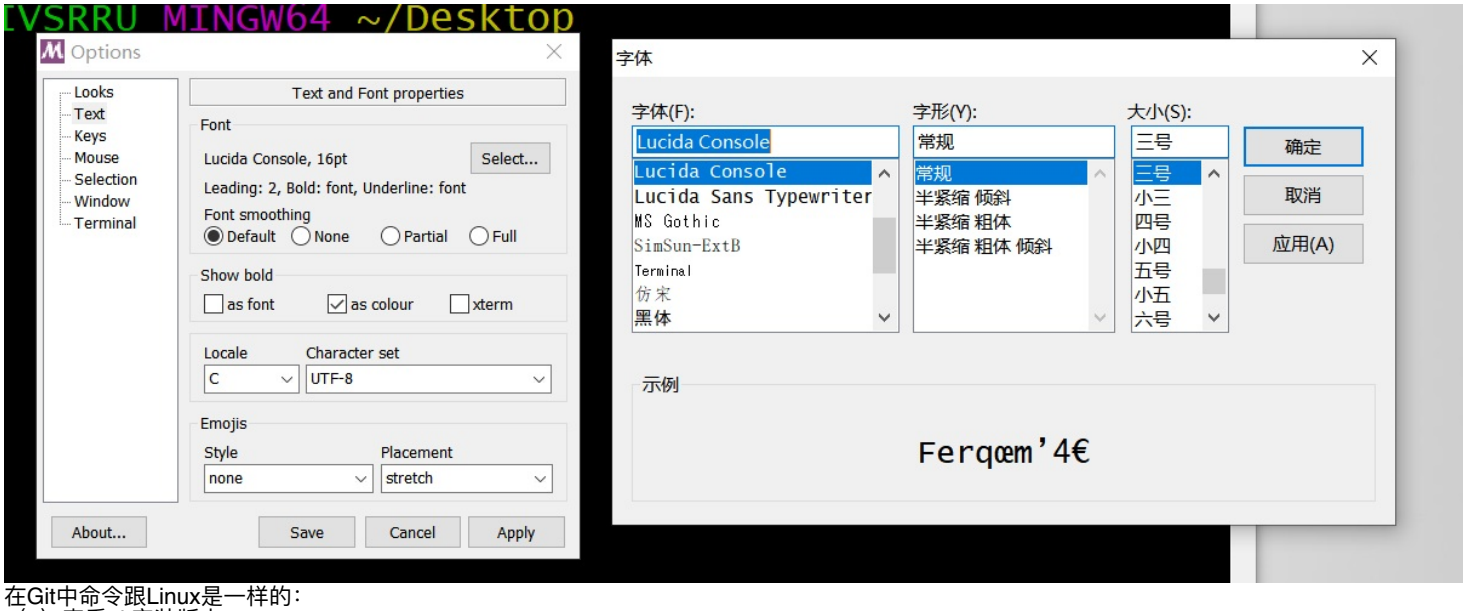
局域网环境下： 可以搭建 GitLab服务器作为代码托管中心，GitLab可以自己去搭建
外网环境下：可以由GitHub或者Gitee作为代码托管中心，GitHub或者Gitee是现成的托管中心，不用自己去搭建

初始化本地仓库

【1】创建一个文件夹：

GitResp

【2】打开Git终端：
Git Bash Here:
进入以后先对字体和编码进行设置：



在Git中命令跟Linux是一样的：

（1）查看git安装版本：

```
zss@LAPTOP-CRIVSRRU MINGW64 ~/Desktop
$ git --version
git version 2.26.2.windows.1
```

（2）清屏：

```
zss@LAPTOP-CRIVSRRU MINGW64 ~/Desktop
$ clear
```

（3）设置签名：

设置用户名和邮箱：

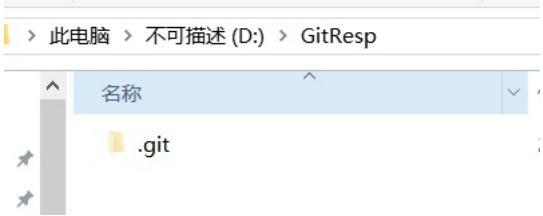
```
zss@LAPTOP-CRIVSRRU MINGW64 ~/Desktop
$ git config --global user.name "zhaoss"
```

```
zss@LAPTOP-CRIVSRRU MINGW64 ~/Desktop
$ git config --global user.email "chinazss@126.com"
```

（4）本地仓库的初始化操作：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp
$ git init
Initialized empty Git repository in D:/GitResp/.git/
```

.git目录是隐藏的：可以调出来查看：



查看.git下内容：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp/.git (GIT_DIR!)
$ ll
total 7
-rw-r--r-- 1 zss 197121  23 Apr 24 16:44 HEAD
-rw-r--r-- 1 zss 197121 130 Apr 24 16:44 config
-rw-r--r-- 1 zss 197121  73 Apr 24 16:44 description
drwxr-xr-x 1 zss 197121   0 Apr 24 16:44 hooks/
drwxr-xr-x 1 zss 197121   0 Apr 24 16:44 info/
drwxr-xr-x 1 zss 197121   0 Apr 24 16:44 objects/
drwxr-xr-x 1 zss 197121   0 Apr 24 16:44 refs/
```

注意事项： .git目录下的本地库相关的子目录和子文件不要删除，不要胡乱修改。

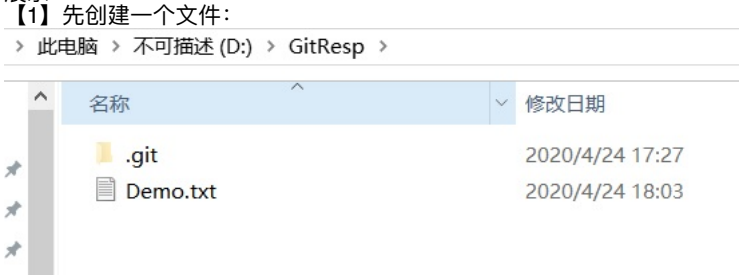
---

add和commit命令

添加文件： add 提交文件：commit

展示：

【1】先创建一个文件：



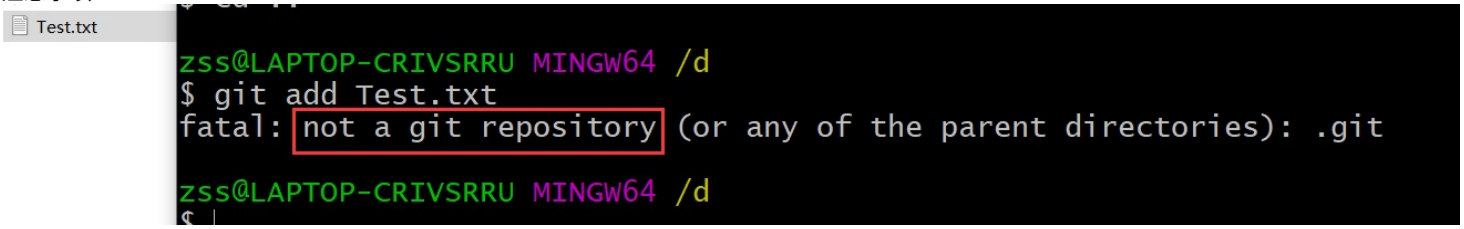
【2】将文件提交到暂存区：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Demo.txt
```

【3】将暂存区的内容提交到本地库：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "这是我提交的第一个文件 Demo.txt" Demo.txt
[master (root-commit) dbad5cb] 这是我提交的第一个文件 Demo.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Demo.txt
```

注意事项：



（1）不放在本地仓库中的文件，git是不进行管理
（2）即使放在本地仓库的文件，git也不管理，必须通过add,commit命令操作才可以将内容提交到本地库。

---

status命令

git status看的是工作区和暂存区的状态

创建一个文件，然后查看状态：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Demo2.txt

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

然后将Demo2.txt通过git add命令提交至：暂存区：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add "Demo2.txt"
```

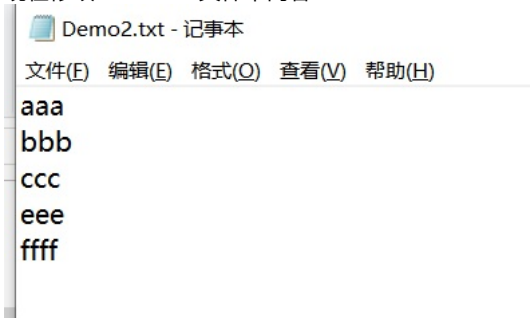
查看状态：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Demo2.txt
```

利用git commit 命令将文件提交至：本地库

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "这是我提交的第二个文件" "Demo2.txt"
[master 4cfe4fd] 这是我提交的第二个文件
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Demo2.txt
```

现在修改Demo2.txt文件中内容：



然后再查看状态：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (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:   Demo2.txt
```

重新添加至：暂存区：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add "Demo2.txt"
```

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Demo2.txt
```

然后将暂存区的文件提交到本地库：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "修改了文件Demo2.txt中内容" "Demo2.txt"
[master 4cd45d2] 修改了文件Demo2.txt中内容
 1 file changed, 5 insertions(+)
```

提交完再查看状态：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

---

log命令

git log 可以让我们查看提交的，显示从最近到最远的日志

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git log
commit 4cd45d2ab730fa46da82c23e3aaa8bec6774c7a5 (HEAD -> master)
Author: zhaoss <chinazss@126.com>
Date:   Sat Apr 25 13:33:33 2020 +0800

    修改了文件Demo2.txt中内容

commit 4cfe4fd6e2a2e21db27d95bf9441372a479066a9
Author: zhaoss <chinazss@126.com>
Date:   Sat Apr 25 13:30:17 2020 +0800

    这是我提交的第二个文件

commit dbad5cb0ac2424377733aab720caa4475aa12dbf
Author: zhaoss <chinazss@126.com>
Date:   Fri Apr 24 18:06:19 2020 +0800

    这是我提交的第一个文件 Demo.txt
```

当前历史记录对应的索引

key:索引

value:历史记录对应的具体的内容

---

log命令2

当历史记录过多的时候，查看日志的时候，有分页效果，分屏效果，一页展示不下：

```
commit 709c842ae9474b9aeb30ae9fc4d42887d9cabe14
Author: zhaoss <chinazss@126.com>
Date:   Sat Apr 25 15:23:01 2020 +0800

    提交了Demo7.txt

commit 10849ecd4ad3b2721bfdd58592c1ccb8e0866755
Author: zhaoss <chinazss@126.com>
Date:   Sat Apr 25 15:22:42 2020 +0800

    提交了Demo6.txt

commit 03b39b2f62d2bf0548a6f3e6dcd71b631b8184ad
Author: zhaoss <chinazss@126.com>
Date:   Sat Apr 25 15:13:17 2020 +0800

    提交了Demo4.txt

:
```

下一页：空格
上一页： b
到尾页了，显示END

退出：q

日志展示方式：
【1】方式1：git log ---》分页
【2】方式2：git log --pretty=onelint

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git log --pretty=oneline
d99f2acb54e3e2c9fcec3c0fa3dfa27ccf2c0916 (HEAD -> master) 提交了Demo10.txt
db78f2d9f81ea108c4c9b564c116d6153d8f395d 提交了Demo9.txt
8eb620b302252bc9b621149a212f537ad651005a 提交了Demo8.txt
709c842ae9474b9aeb30ae9fc4d42887d9cabe14 提交了Demo7.txt
10849ecd4ad3b2721bfdd58592c1ccb8e0866755 提交了Demo6.txt
03b39b2f62d2bf0548a6f3e6dcd71b631b8184ad 提交了Demo4.txt
f856aa14d485c2c3d359cccaeaa09c63ba204e68 提交了Demo3.txt
4cd45d2ab730fa46da82c23e3aaa8bec6774c7a5 修改了文件Demo2.txt中内容
4cfe4fd6e2a2e21db27d95bf9441372a479066a9 这是我提交的第二个文件
dbad5cb0ac2424377733aab720caa4475aa12dbf 这是我提交的第一个文件 Demo.txt
```

【3】方式3：git --oneline

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git log --oneline
d99f2ac (HEAD -> master) 提交了Demo10.txt
db78f2d 提交了Demo9.txt
8eb620b 提交了Demo8.txt
709c842 提交了Demo7.txt
10849ec 提交了Demo6.txt
03b39b2 提交了Demo4.txt
f856aa1 提交了Demo3.txt
4cd45d2 修改了文件Demo2.txt中内容
4cfe4fd 这是我提交的第二个文件
dbad5cb 这是我提交的第一个文件 Demo.txt
```

【4】方式4：git reflog
多了信息：HEAD@{数字}
这个数字的含义：指针回到当前这个历史版本需要走多少步

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git reflog
d99f2ac (HEAD -> master) HEAD@{0}: commit: 提交了Demo10.txt
db78f2d HEAD@{1}: commit: 提交了Demo9.txt
8eb620b HEAD@{2}: commit: 提交了Demo8.txt
709c842 HEAD@{3}: commit: 提交了Demo7.txt
10849ec HEAD@{4}: commit: 提交了Demo6.txt
03b39b2 HEAD@{5}: commit: 提交了Demo4.txt
f856aa1 HEAD@{6}: commit: 提交了Demo3.txt
4cd45d2 HEAD@{7}: commit: 修改了文件Demo2.txt中内容
4cfe4fd HEAD@{8}: commit: 这是我提交的第二个文件
dbad5cb HEAD@{9}: commit (initial): 这是我提交的第一个文件 Demo
```

---

reset命令

reset命令：前进或者后退历史版本

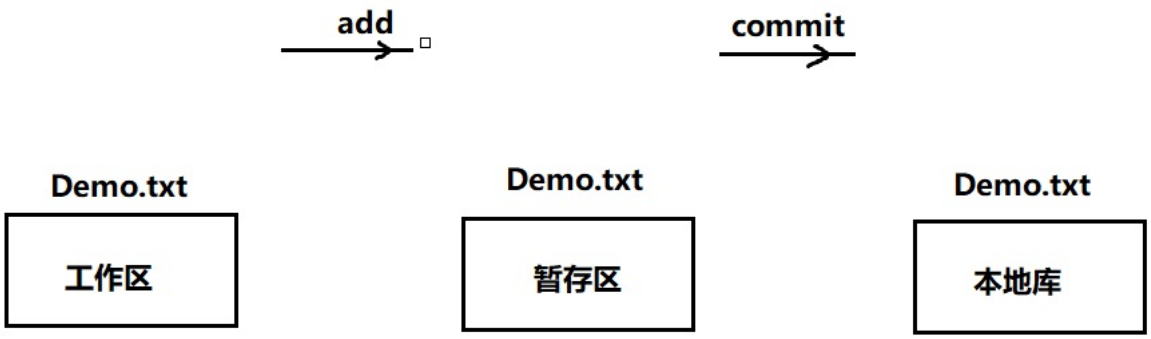
复制：在终端中选中就是复制了
粘贴： 右键：paste

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git reset --hard bc07b79
HEAD is now at bc07b79 insert ccc
```
z

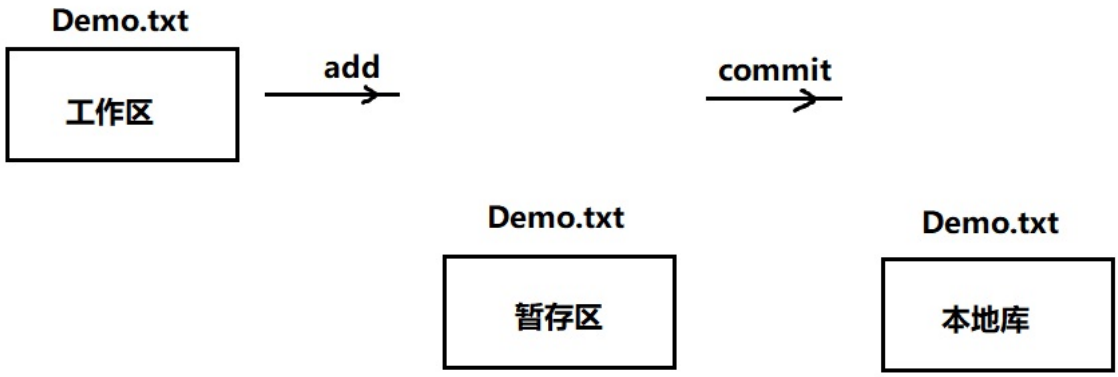
---

hard参数/mixed参数/soft参数

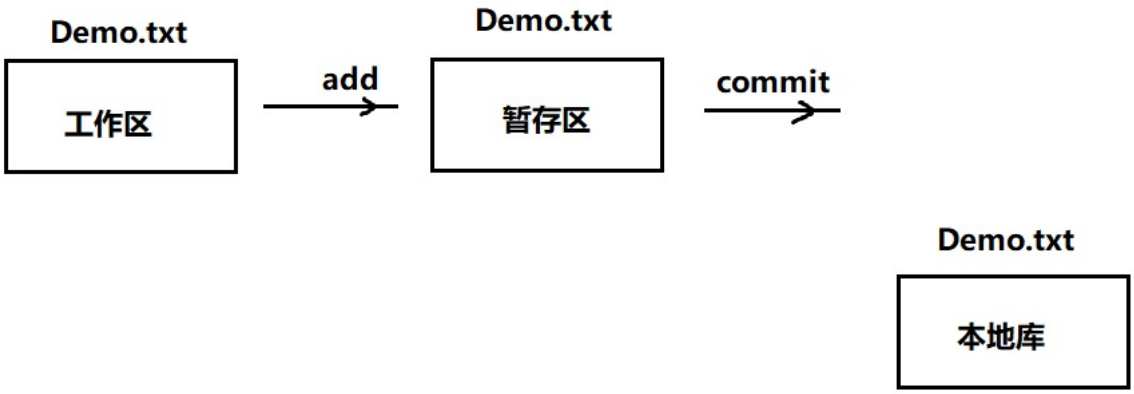
【1】hard参数：
git reset --hard [索引]
本地库的指针移动的同时，重置暂存区，重置工作区



【2】mixed参数：
本地库的指针移动的同时，重置暂存区，但是工作区不动



【3】soft参数：
本地库的指针移动的时候，暂存区，工作区都不动



总结：以后用的多的就是 第一种hard参数

删除文件_找回本地库删除的文件

【1】新建 一个Test2.txt文件
【2】将它add到暂存区中
【3】再通过commit提交到本地库

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "添加Test2.txt文件" Test2.txt
[master 99b26c8] 添加Test2.txt文件
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Test2.txt
```

【4】删除工作区中的Test2.txt

```
zss@LAPTOP-CRIVSRRU MINGW64
$ rm Test2.txt
```

【5】将删除操作同步到暂存区：
【6】将删除操作同步到本地库：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Test2.txt


zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "删除Test2.txt文件" Test2.txt
[master 22c85d6] 删除Test2.txt文件
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 Test2.txt
```

【7】查看日志：

```
$ git reflog
22c85d6 (HEAD -> master) HEAD@{0}: commit: 删除Test2.txt文件
99b26c8 HEAD@{1}: commit: 添加Test2.txt文件
a0f7d06 HEAD@{2}: commit: 提交Demo5.txt
bc07b79 HEAD@{3}: reset: moving to bc07b79
```

【8】找回本地库中删除的文件，实际上就是将历史版本切换到刚才添加文件的那个版本即可：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git reset --hard 99b26c8
HEAD is now at 99b26c8 添加Test2.txt文件
```

找回暂存区删除的文件

【1】删除工作区数据：

```
zss@LAPTOP-CRIVSRRU MING
$ rm Test2.txt
```

【2】同步到缓存区：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Test2.txt
```

【3】后悔了，恢复暂存区中数据：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (ma
$ git reset --hard 99b26c8
HEAD is now at 99b26c8 添加Test2.txt文件
```

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (ma
$ git reset --hard HEAD
HEAD is now at 99b26c8 添加Test2.txt文件
```

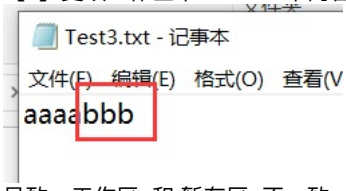
---

diff命令

【1】先创建一个文件，添加到暂存区，再提交到本地库：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test3.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "提交了Test3.txt" Test3.txt
[master ff3ab3b] 提交了Test3.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Test3.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

【2】更改工作区中Test3.txt中内容，增加内容：



导致：工作区 和 暂存区 不一致，比对：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git diff Test3.txt
diff --git a/Test3.txt b/Test3.txt
index 7284ab4..62ba7e8 100644
--- a/Test3.txt
+++ b/Test3.txt
@@ -1 +1 @@
-aaaa
\ No newline at end of file
+aaaabbb
\ No newline at end of file

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
```

Git 是按照行为单位管理数据
所以：删除一行，添加一行

总结： git diff [文件名] ---》 将工作区中的文件和暂存区中文件进行比较

多个文件的比对：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git diff
diff --git a/Test2.txt b/Test2.txt
index e69de29..e5a49f3 100644
--- a/Test2.txt
+++ b/Test2.txt
@@ -0,0 +1 @@
+qqq
\ No newline at end of file
diff --git a/Test3.txt b/Test3.txt
index 7284ab4..62ba7e8 100644
--- a/Test3.txt
+++ b/Test3.txt
@@ -1 +1 @@
-aaaa
\ No newline at end of file
+aaaabbb
\ No newline at end of file
```

总结：git diff --->比较工作区中和暂存区中 所有文件的差异

比较暂存区和本地库中差别：
git diff [历史版本][文件名] --->比较暂存区和本地库中内容

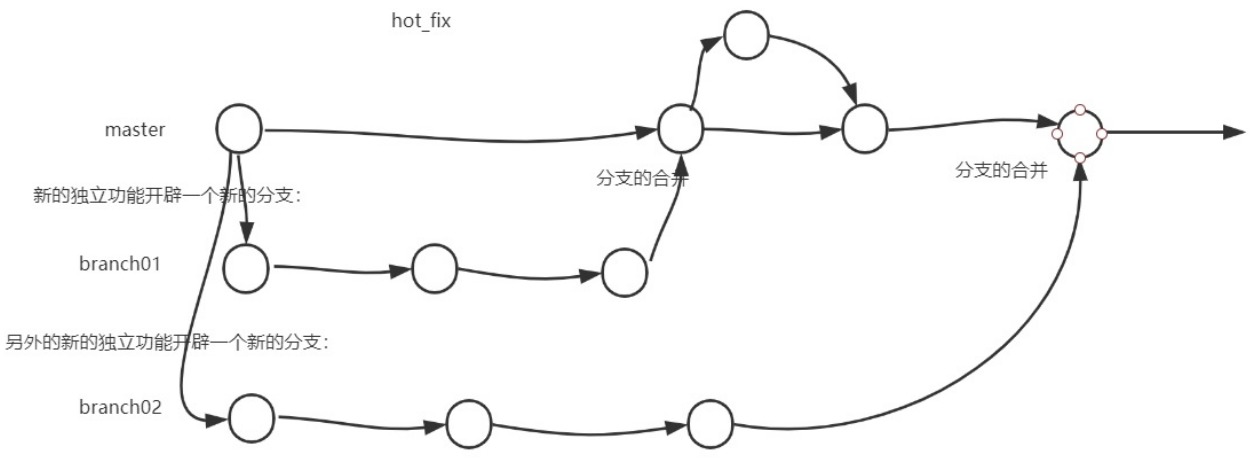
```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp
$ git diff ff3ab3b Test3.txt
diff --git a/Test3.txt b/Test3.txt
```

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (ma
$ git diff HEAD Test3.txt
diff --git a/Test3.txt b/Test3.txt
```

什么是分支

【1】什么是分支：
在版本控制过程中，使用多条线同时推进多个任务。这里面说的多条线，就是多个分支。

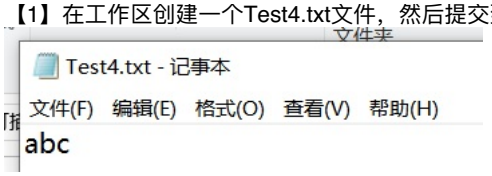
【2】通过一张图展示分支：



【3】分支的好处：
同时多个分支可以并行开发，互相不耽误，互相不影响，提高开发效率
如果有一个分支功能开发失败，直接删除这个分支就可以了，不会对其他分支产生任何影响。

查看，创建，切换分支

【1】在工作区创建一个Test4.txt文件，然后提交到暂存区，提交到本地库：




```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "添加了Test4.txt" Test4.txt
[master 8b9f916] 添加了Test4.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

【2】查看分支：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git branch -v
* master 8b9f916 添加了Test4.txt
```

【3】创建分支：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git branch branch01
```

再查看：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git branch -v
  branch01 8b9f916 添加了Test4.txt
* master   8b9f916 添加了Test4.txt
```

你在哪个分支上，是通过*来显示的

【4】切换分支：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git checkout branch01
Switched to branch 'branch01'
M       Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git branch -v
* branch01 8b9f916 添加了Test4.txt
  master   8b9f916 添加了Test4.txt
```

冲突问题，如何解决冲突题

【1】进入branch01分支，增加内容：


```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git status
On branch branch01
nothing to commit, working tree clean

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git add Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git commit -m "在分支01中增加内容"  Test4.txt
[branch01 a46723e] 在分支01中增加内容
 1 file changed, 2 insertions(+), 1 deletion(-)
```

【2】将分支切换到master:

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git checkout master
Switched to branch 'master'

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git branch -v
  branch01 a46723e 在分支01中增加内容
* master   8b9f916 添加了Test4.txt
```

然后在主分支下 加入内容:

```
  branch01 a46723e 在分支01中增加内容
* master   8b9f916 添加了Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git add Test4.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git commit -m "主分支中增加了内容" Test4.txt
[master 5c090c0] 主分支中增加了内容
 1 file changed, 2 insertions(+), 1 deletion(-)
```

【3】再次切换到branch01分支查看:

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git branch -v
* branch01 a46723e 在分支01中增加内容
  master   5c090c0 主分支中增加了内容

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ cat Test4.txt
abc
增加内容by branch01
```

这里的内容跟主分支中无关了

【4】将branch01分支 合并到 主分支：
（1）进入主分支：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (branch01)
$ git checkout master
Switched to branch 'master'

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$
```

（2）将branch01中的内容和主分支内容进行合并：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master)
$ git merge branch01
Auto-merging Test4.txt
CONFLICT (content): Merge conflict in Test4.txt
Automatic merge failed; fix conflicts and then commit the result.

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master|MERGING)
$
```

出现冲突

显示这个，代表处在合并状态中

查看文件：出现冲突：

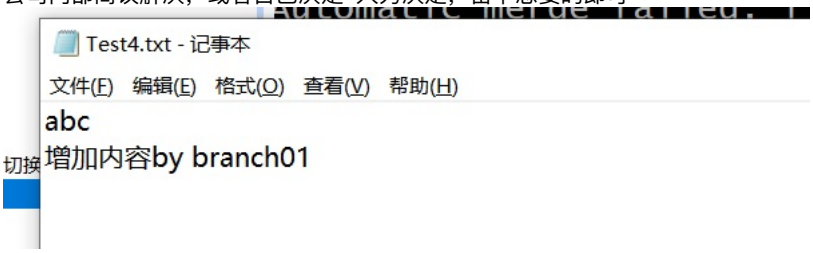
什么时候会出现冲突问题？在同一个文件的同一个位置修改

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp (master|MERGING)
$ cat Test4.txt
abc
<<<<<<< HEAD
增加内容by master
=======
增加内容by branch01
>>>>>>> branch01
```

就是当前分支修改的内容

就是你合并过来的那个分支中修改的内容

解决：
公司内部商议解决，或者自己决定 人为决定，留下想要的即可：

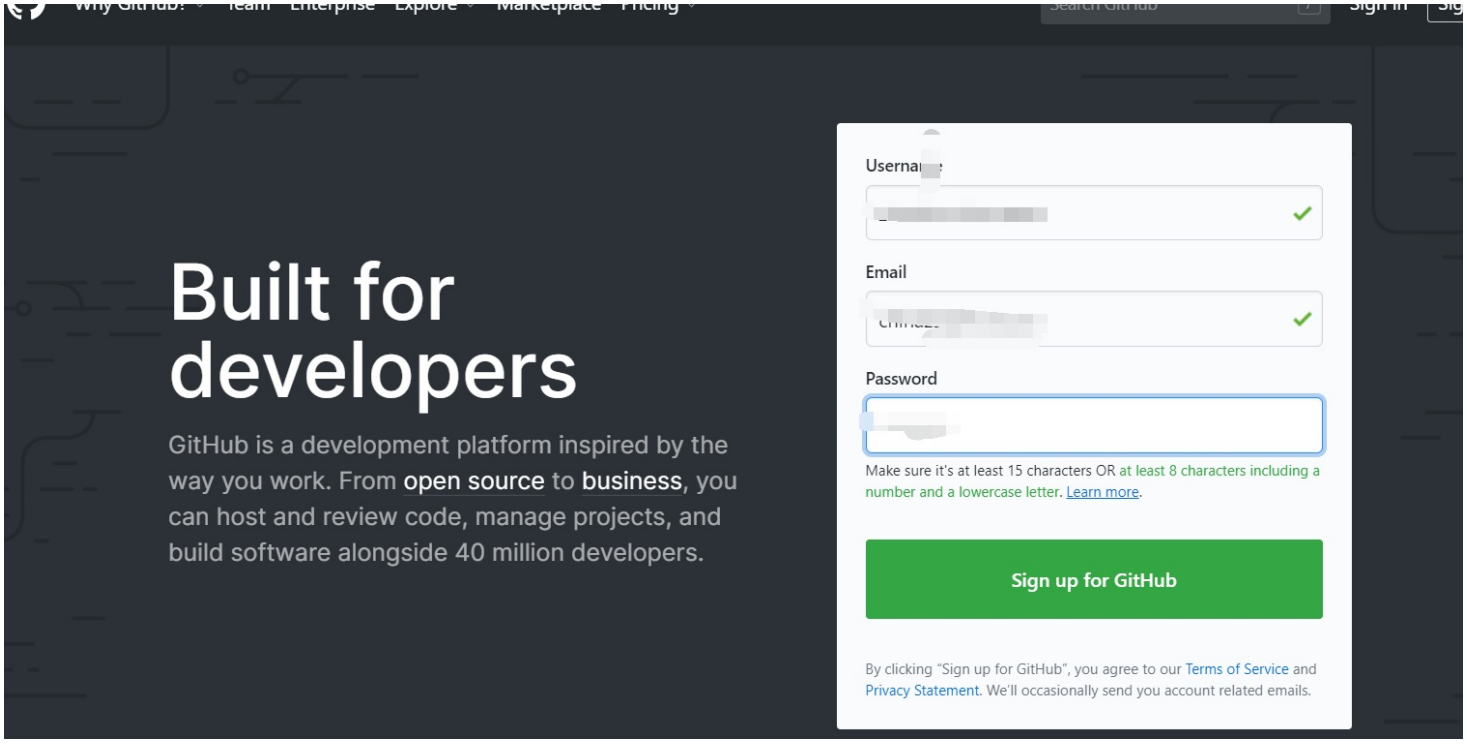


将工作区中内容添加到暂存区：

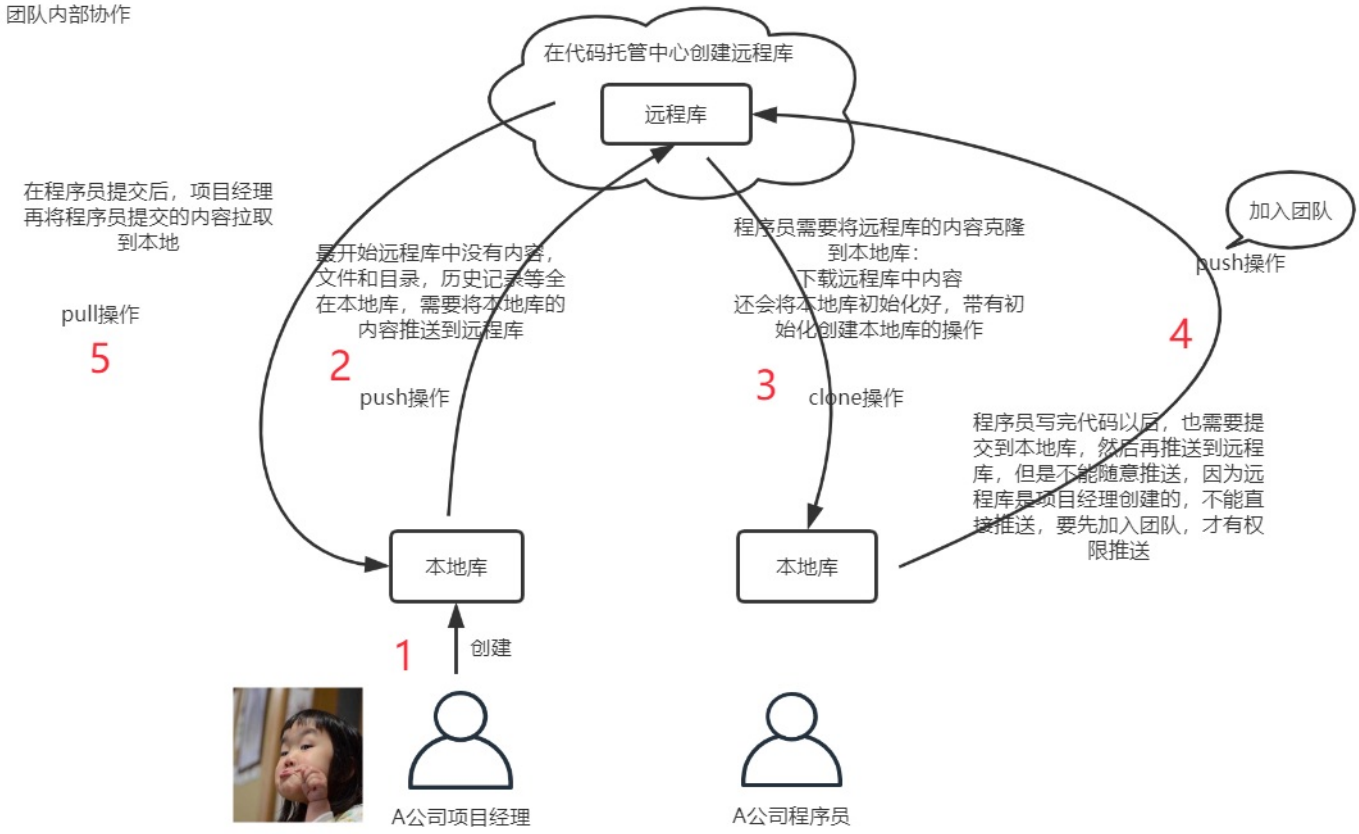
然后进行commit操作：

GitHub账号注册

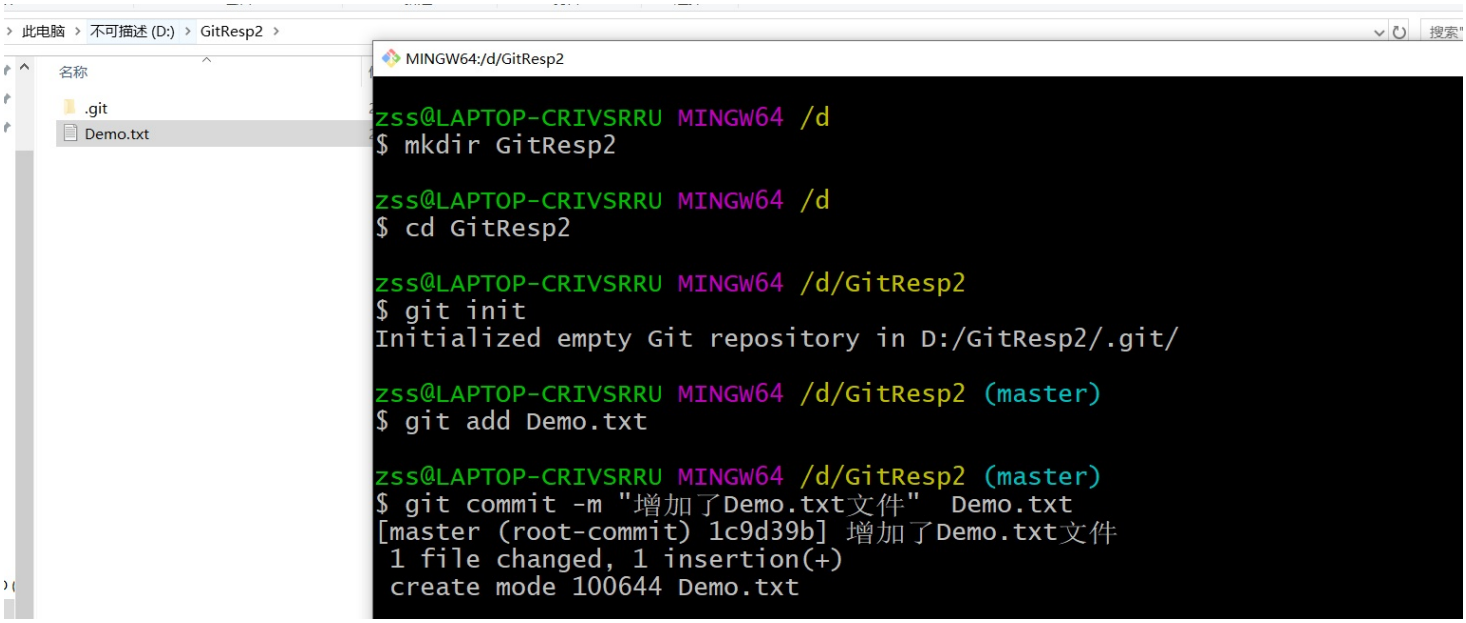
官网： https://github.com/



回顾本地库和远程库交互方式

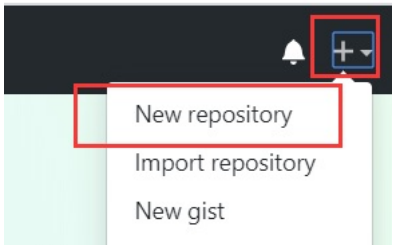


初始化本地库

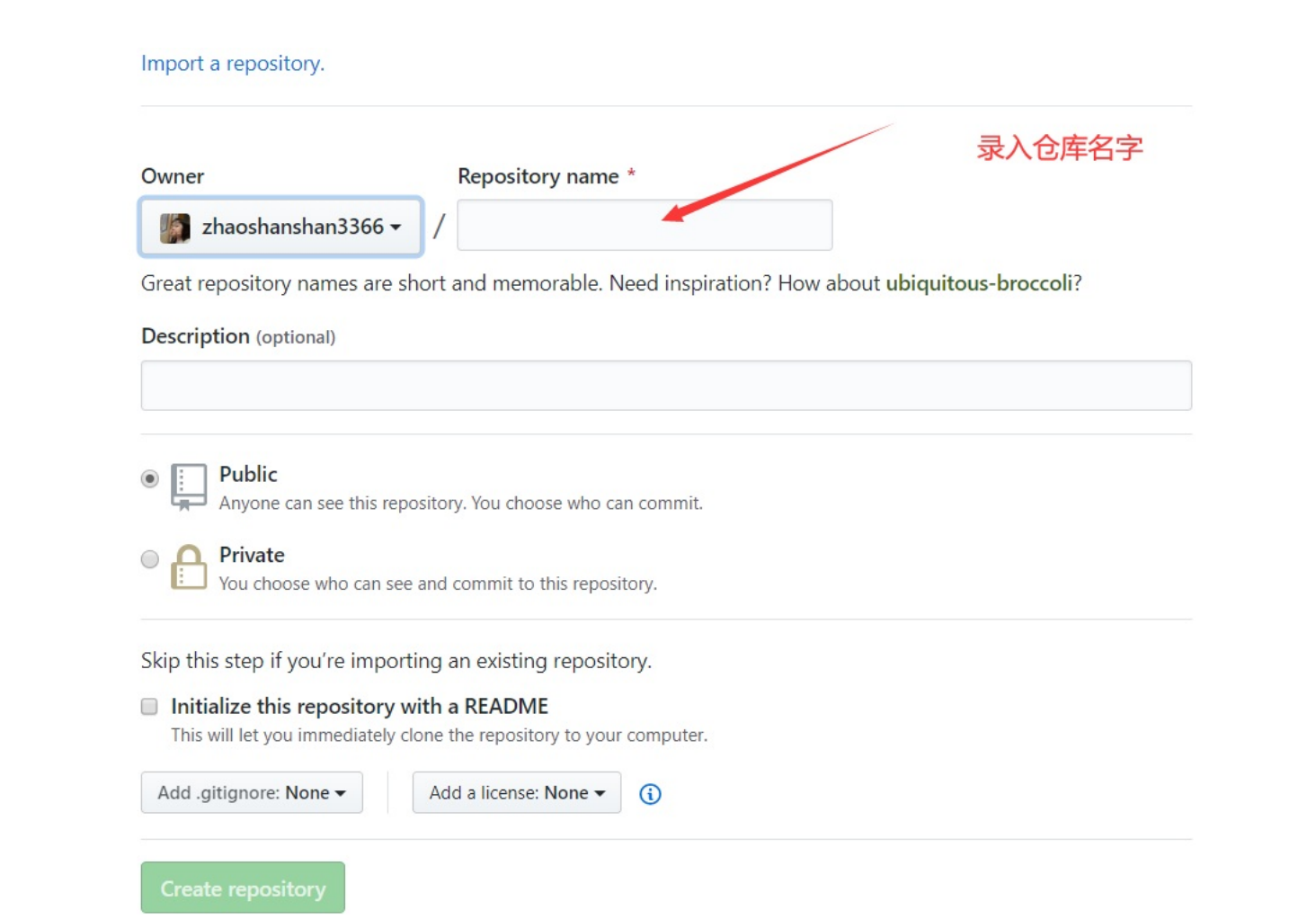


创建远程库

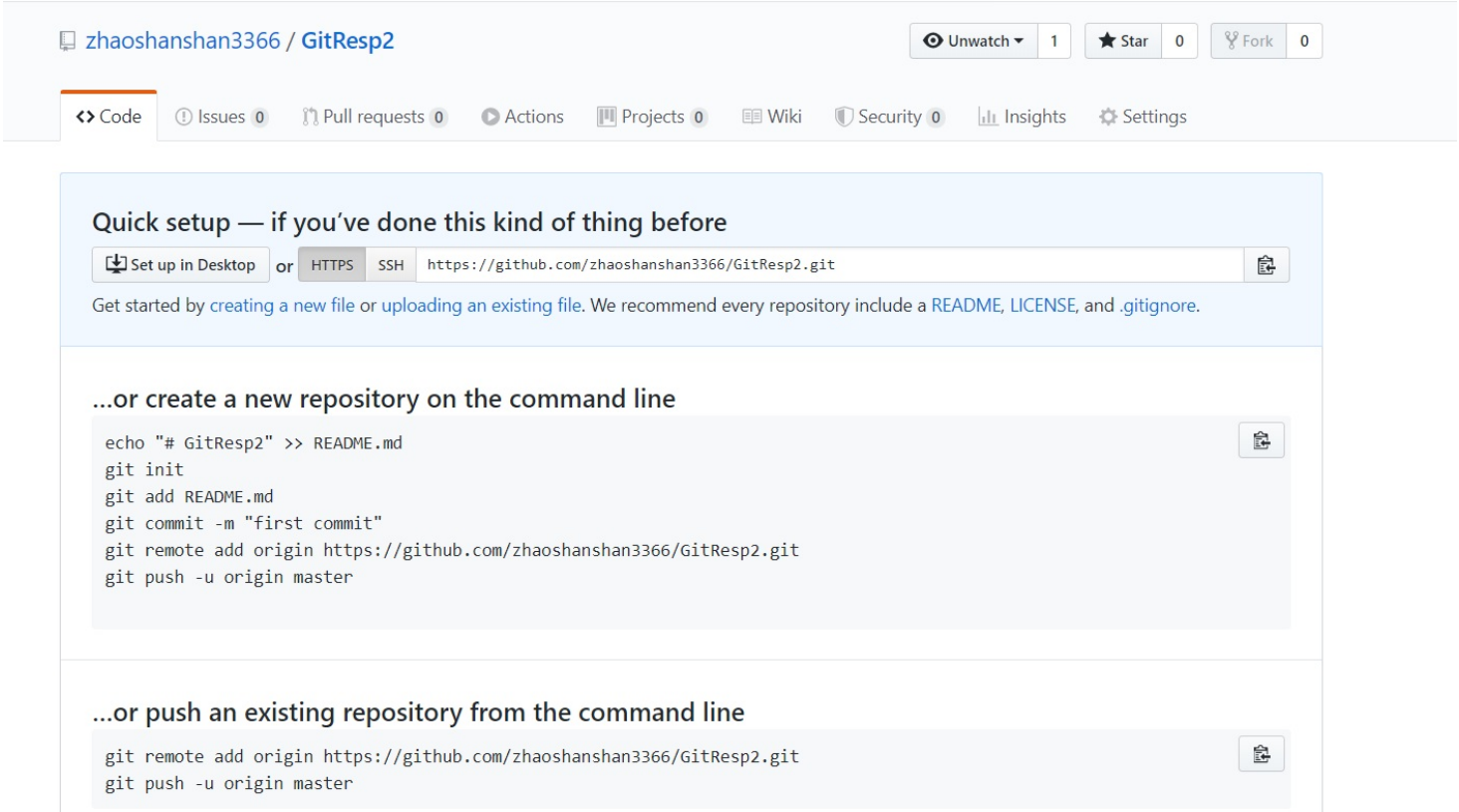
【1】创建远程库



【2】录入信息：

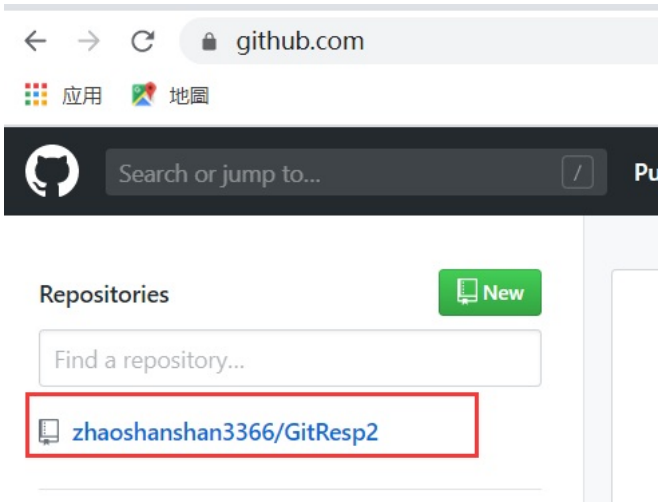


【3】完成状态：

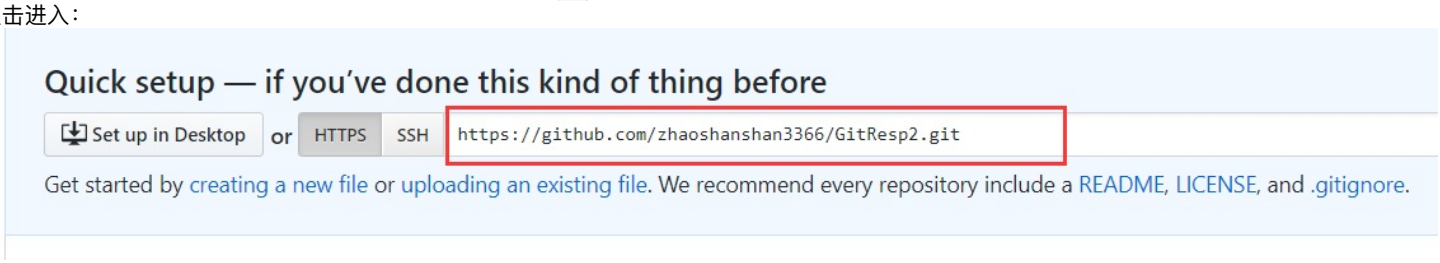


在本地创建远程库地址的别名

远程库的地址：



点击进入：



远程库地址比较长，每次复制比较麻烦
https://github.com/zhaoshanshan3366/GitResp2.git

在Git本地将地址保存，通过别名

查看别名：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitRes
$ git remote -v
```

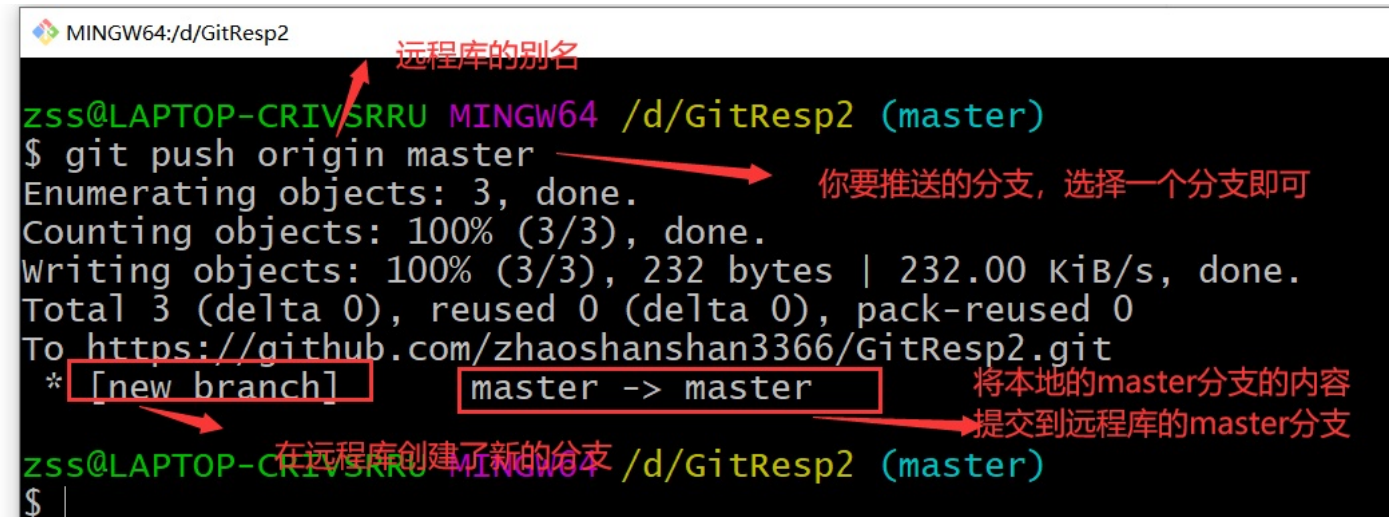
起别名：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git remote add origin https://github.com/zhaoshanshan3366/GitResp2.git
```

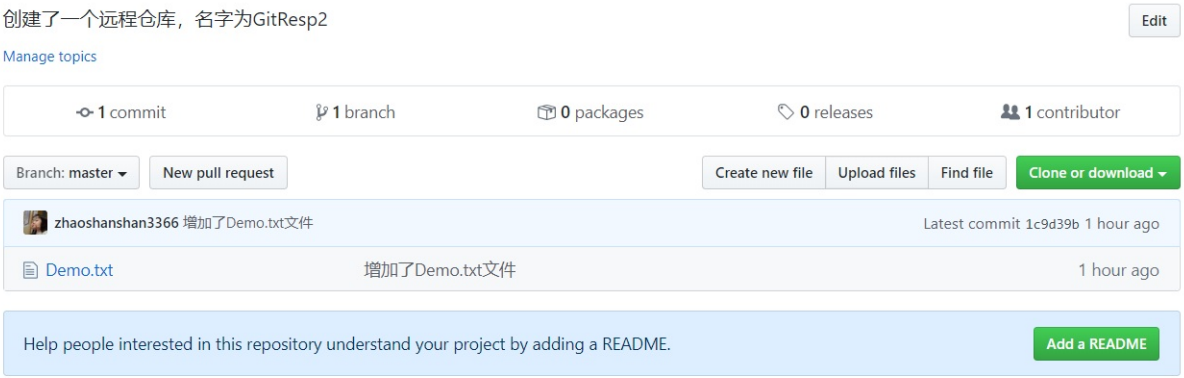
别名，可以随意按照你的需求去起名

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git remote -v
origin  https://github.com/zhaoshanshan3366/GitResp2.git (fetch)
origin  https://github.com/zhaoshanshan3366/GitResp2.git (push)
```

推送操作

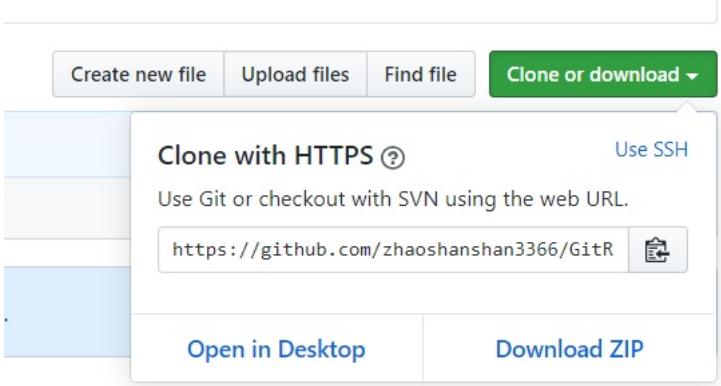


推送成功以后，查看自己的远程库：

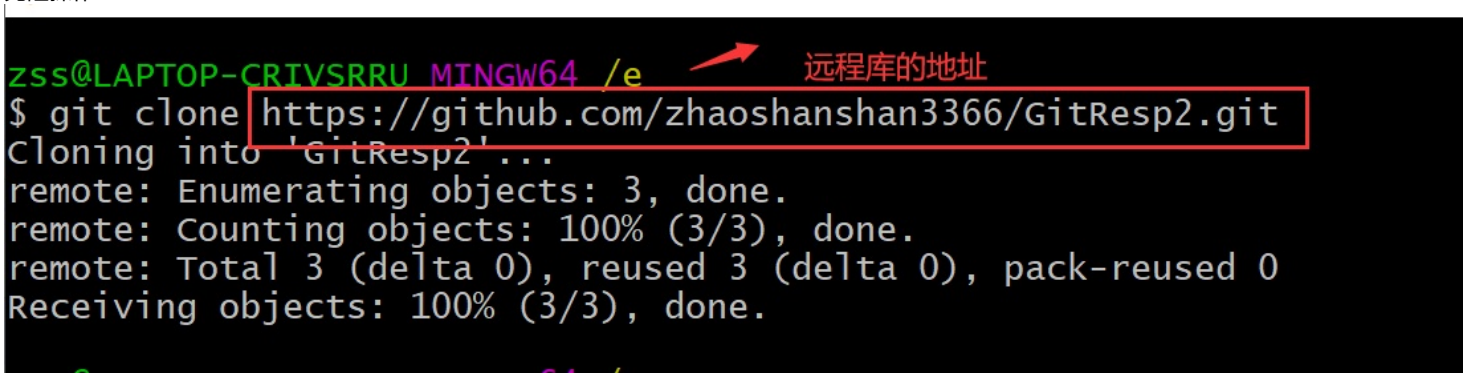


克隆操作

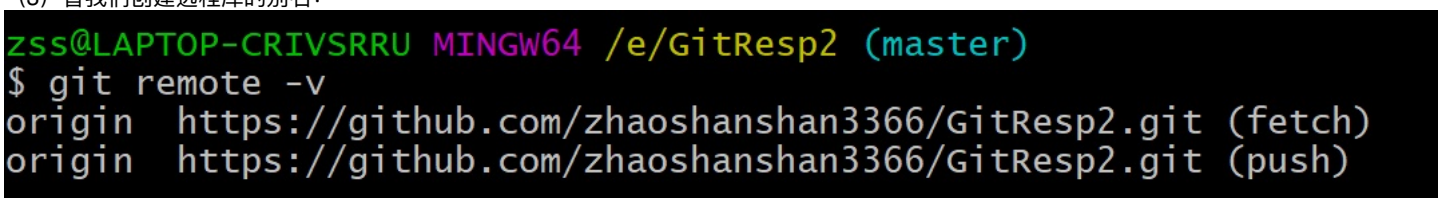
远程库地址复制：



克隆操作：

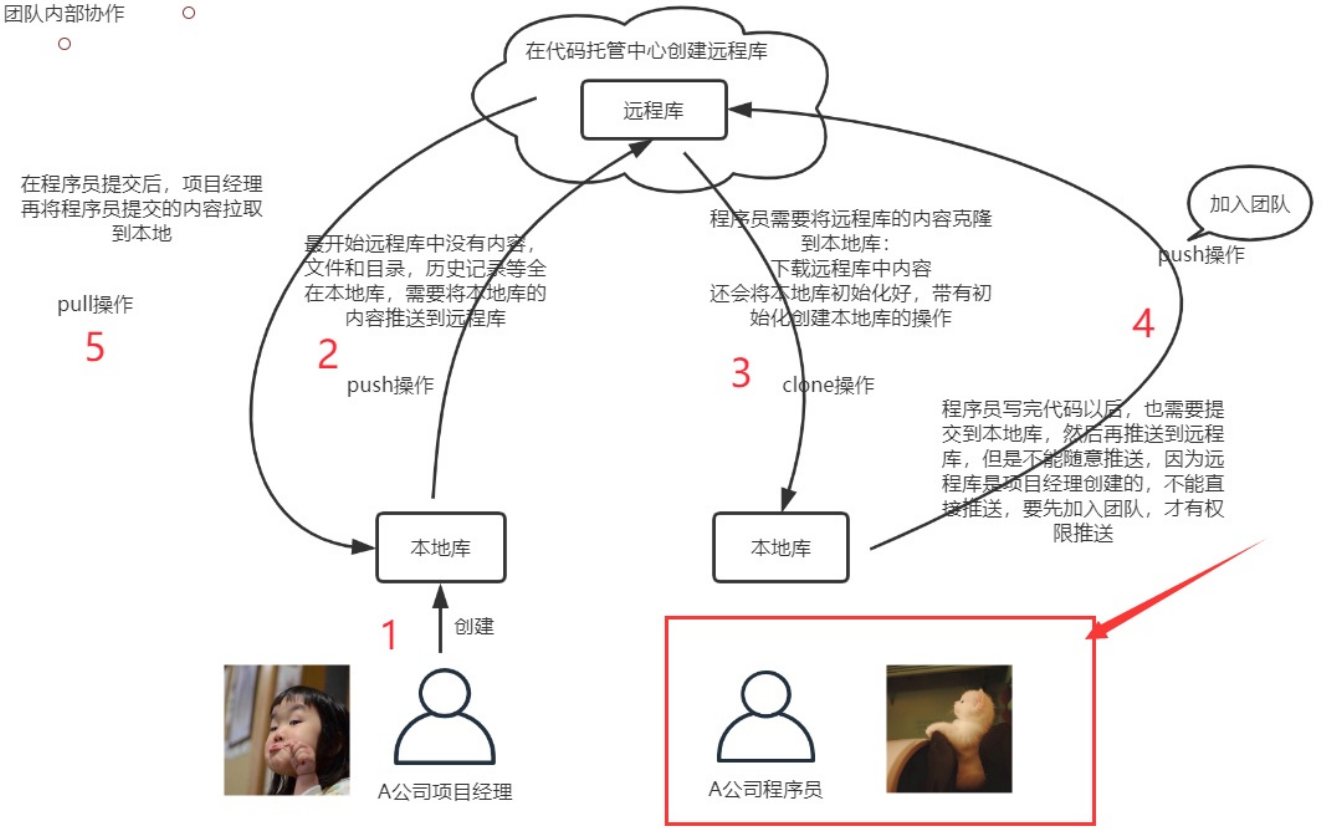


克隆操作可以帮我们完成：
（1）初始化本地库
（2）将远程库内容完整的克隆到本地
（3）替我们创建远程库的别名：



邀请加入团队，push操作

团队内部协作



【1】更新本地库信息：

```
zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git add Demo2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git commit -m "创建了Demo2.txt" Demo2.txt
[master db40be7] 创建了Demo2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Demo2.txt
```

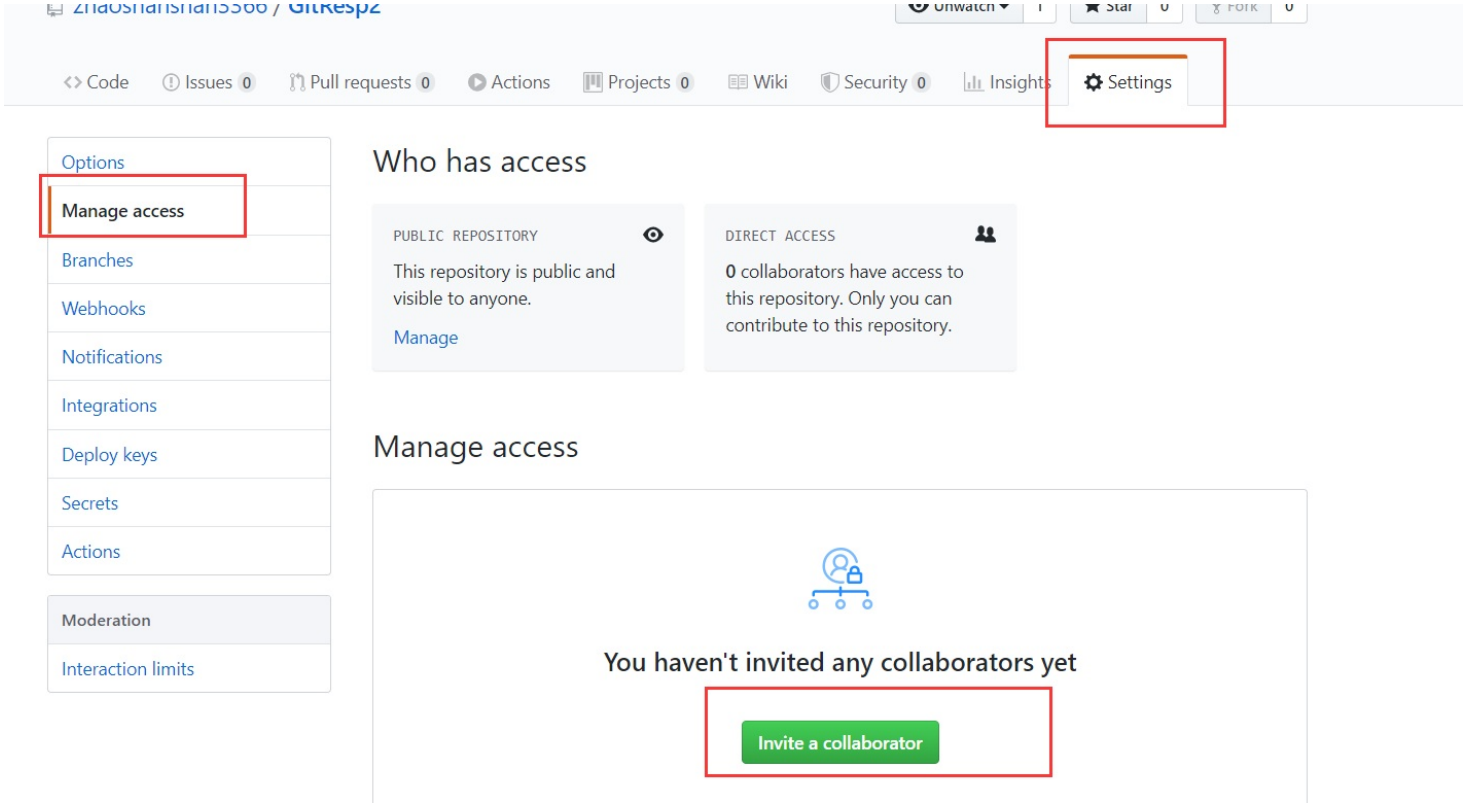
【2】push内容到远程库中去：

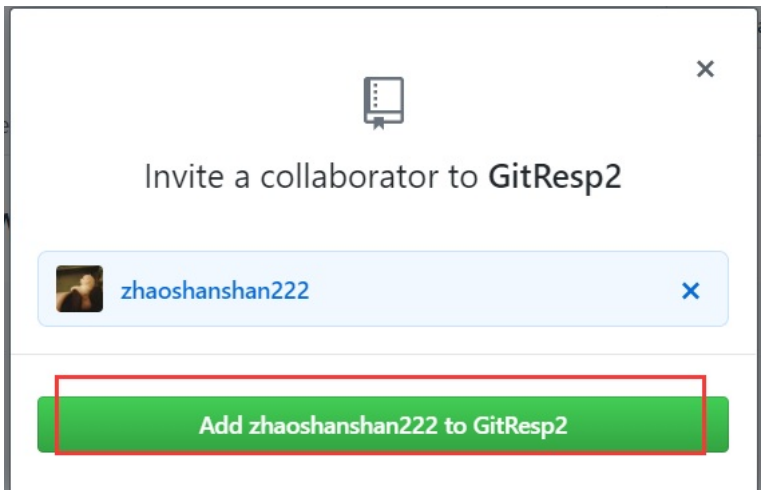
发现可以直接push进去，并没有让我录入账号密码，或者也没有提示错误 - --》结果 很诡异
原因：git使用的时候在本地有缓存：
将缓存删除：

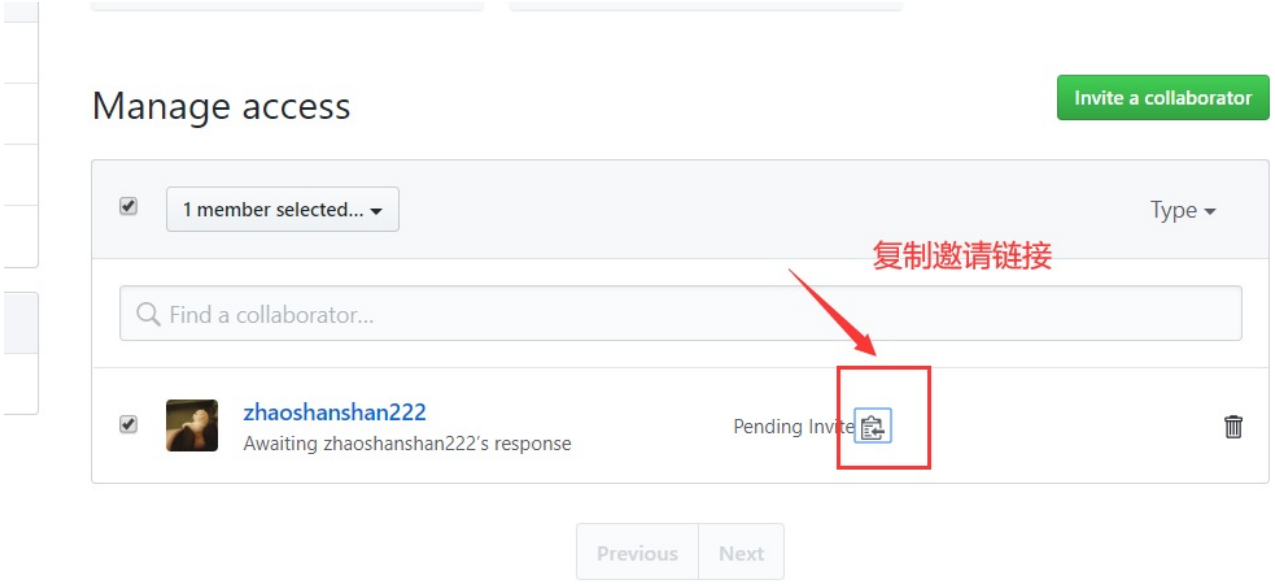
现在再次重新push，发现出错了：

```
zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git push origin master
remote: Permission to zhaoshanshan3366/GitResp2.git denied to zhaoshan
fatal: unable to access 'https://github.com/zhaoshanshan3366/GitResp2.
 requested URL returned error: 403
```

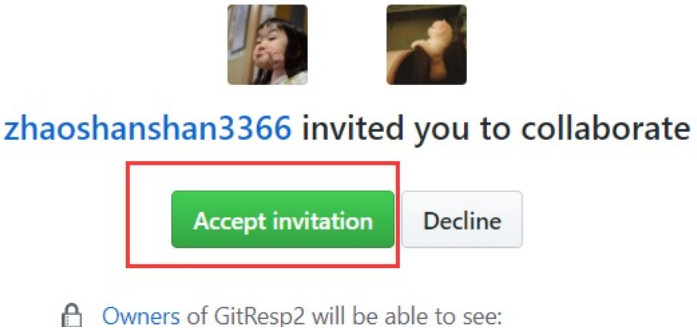
必须要加入团队：
登录项目经理的账号，邀请普通成员：





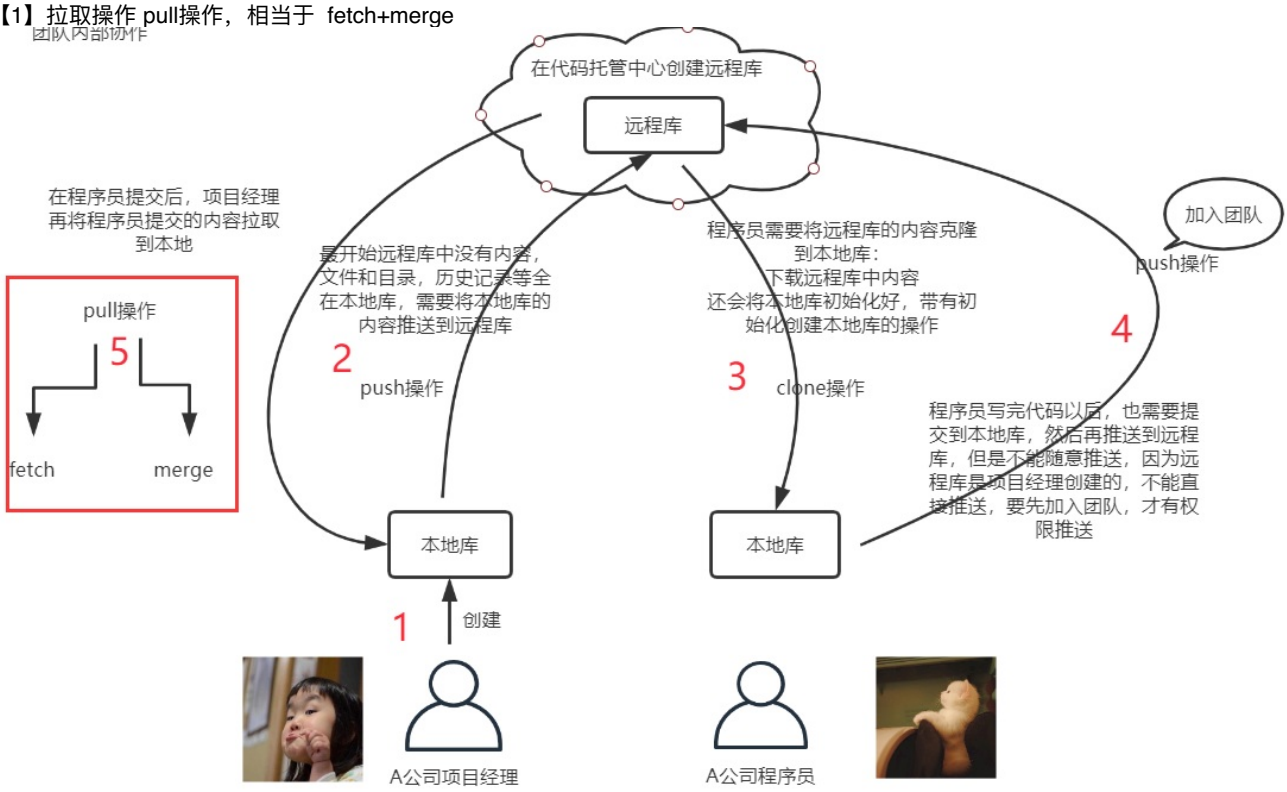


登录被邀请者的账号，接收邀请：（在地址栏录入邀请链接即可： https://github.com/zhaoshanshan3366/GitResp2/invitations ）



远程库修改的拉取操作

【1】拉取操作 pull操作，相当于 fetch+merge



【2】项目经理先先确认远程库内容是否更新了：

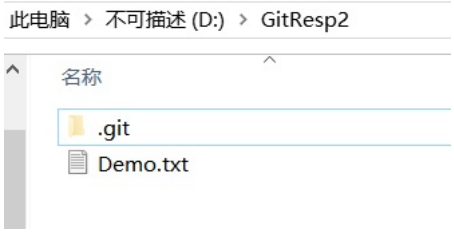
【3】项目经理进行拉取：
（1）先是抓取操作：fetch:

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git fetch origin master
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 17 (delta 6), reused 11 (delta 0), pack-reused 0
Unpacking objects: 100% (17/17), 1.42 KiB | 27.00 KiB/s, done.
From https://github.com/zhaoshanshan3366/GitResp2
 * branch            master     -> FETCH_HEAD
   1c9d39b..c091f46  master     -> origin/master
```

远程库别名

远程库上对应的分支

在抓取操作执行后，只是将远程库的内容下载到本地，但是工作区中的文件并没有更新。工作区中还是原先的内容：



抓取后可以去远程库看看内容是否正确：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git checkout origin/master
Note: switching to 'origin/master'.
```

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 ((c091f46...))
$ ll
total 3
-rw-r--r-- 1 zss 197121  8 Apr 27 18:19 Demo.txt
-rw-r--r-- 1 zss 197121 53 Apr 27 18:19 Demo2.txt
-rw-r--r-- 1 zss 197121  6 Apr 27 18:19 Demo3.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 ((c091f46...))
$ cat Demo3.txt
iiiiii
```

然后发现内容都正确，就可以进行合并操作了：
合并之前应该将分支切换回来：

```
zss@LAPTOP-CRIVSRRU MINGW64 /
$ git checkout master
Previous HEAD position was c0
```

(2) 进行合并：merge:

```
$ git merge origin/master
Updating 1c9d39b..c091f46
Fast-forward
 Demo.txt  | 2 +-
 Demo2.txt | 1 +
 Demo3.txt | 1 +
 3 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 Demo2.txt
 create mode 100644 Demo3.txt
```

远程库修改的拉取操作2

远程库的拉取可以直接利用pull命令来完成：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d
$ git pull origin master
```

fetch+merge操作：--->为了保险，慎重
pull --->代码简单，省事

协同开发合作时冲突的解决办法

【1】

向远程库推送数据：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git add Test.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git commit -m "创建了Test.txt" Test.txt
[master cefde78] 创建了Test.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Test.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object
To https://github.com/zhaoshanshan3366/GitResp2.git
   3b5e332..cefde78  master -> master
```

【2】

做了一个拉取操作：

```
zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reuse
Unpacking objects: 100% (3/3), 261 bytes | 23.00 KiB/s, d
From https://github.com/zhaoshanshan3366/GitResp2
 * branch            master     -> FETCH_HEAD
   3b5e332..cefde78  master     -> origin/master
Updating 3b5e332..cefde78
Fast-forward
 Test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Test.txt
```

到这里为止，现在远程合作没有任何问题。
现在操作同一个文件的同一个位置的时候，就会引起冲突：

【3】

再次做了推送操作：

```
zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git add Test.txt

zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git commit -m "更新了Test.txt" Test.txt
[master 685ea39] 更新了Test.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

zss@LAPTOP-CRIVSRRU MINGW64 /e/GitResp2 (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s,
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), complete
To https://github.com/zhaoshanshan3366/GitResp
   cefde78..685ea39  master -> master
```

改动位置：

【4】

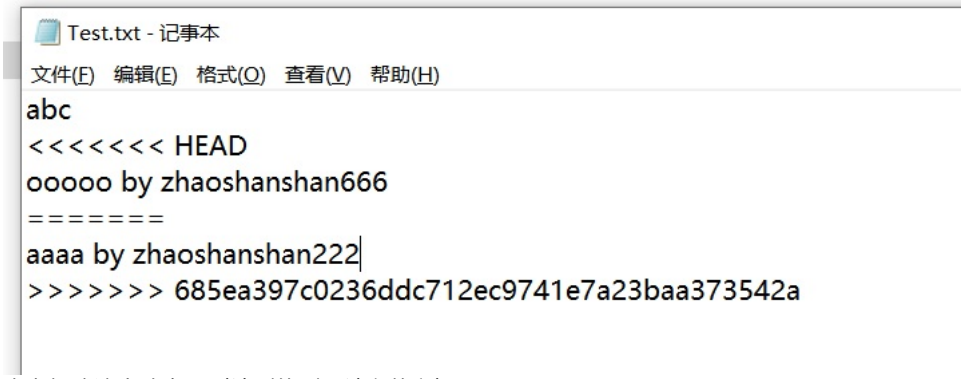
改动Test.txt中内容，然后进行推送：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git add Test.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git commit -m "更新了Test.txt" Test.txt
[master a3a64ab] 更新了Test.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git push origin master
fatal: HttpRequestException encountered.
   [redacted]
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/zhao
it/'
```

发现 推送失败！
在冲突的情况下，先应该拉取下来，然后修改冲突，然后再推送到远程服务器：
先拉取：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), 269 bytes | 24.00 KiB/s, done.
From https://github.com/zhaoshanshan3366/GitResp2
 * branch            master     -> FETCH_HEAD
   cefde78..685ea39  master     -> origin/master
Auto-merging Test.txt
CONFLICT (content): Merge conflict in Test.txt
Automatic merge failed; fix conflicts and then commit the resul
```

查看冲突：



人为解决这个冲突：（该删的删，该留的留）

解决完冲突以后，向服务器推送：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master|MERGING)
$ git add Test.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master|MERGING)
$ git commit -m "解决了冲突问题" Test.txt
fatal: cannot do a partial commit during a merge.

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master|MERGING)
$ git commit -m "解决了冲突问题"
[master c5e4f3a] 解决了冲突问题
```

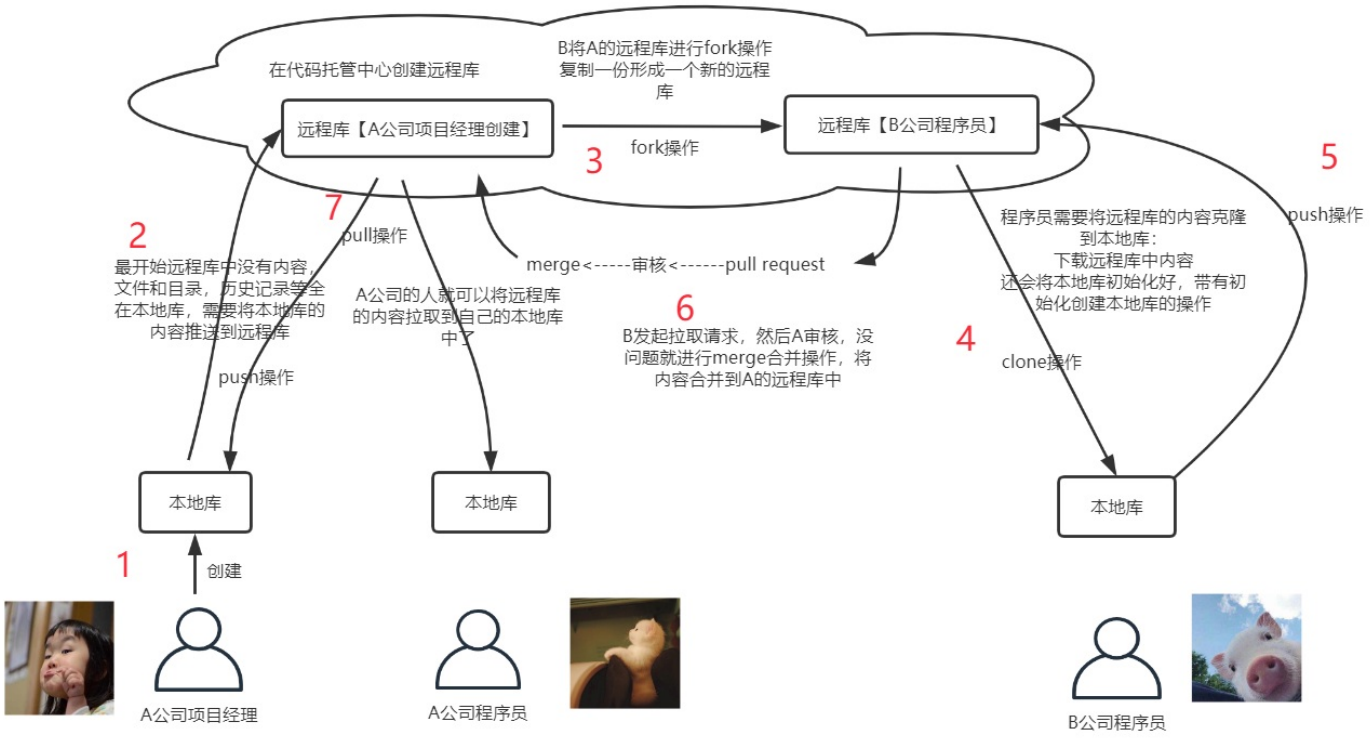
注意在解决冲突问题的时候，提交不可以带文件名，否则提交失败

推送：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git push origin master
Enumerating objects: 10, done.
```

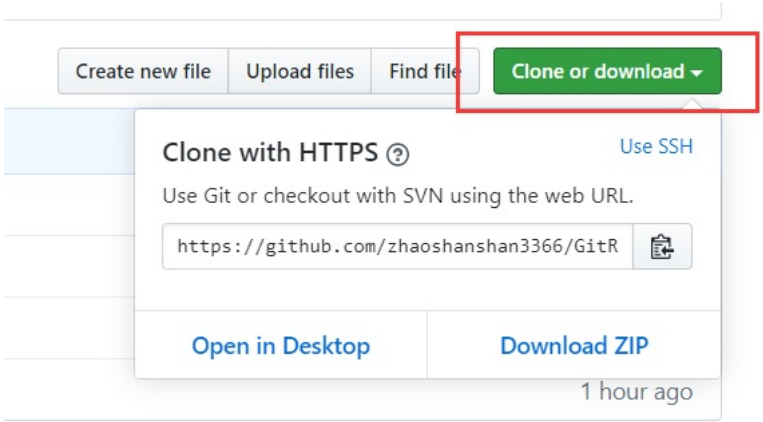
解决了冲突问题：

回顾跨团队合作交互方式



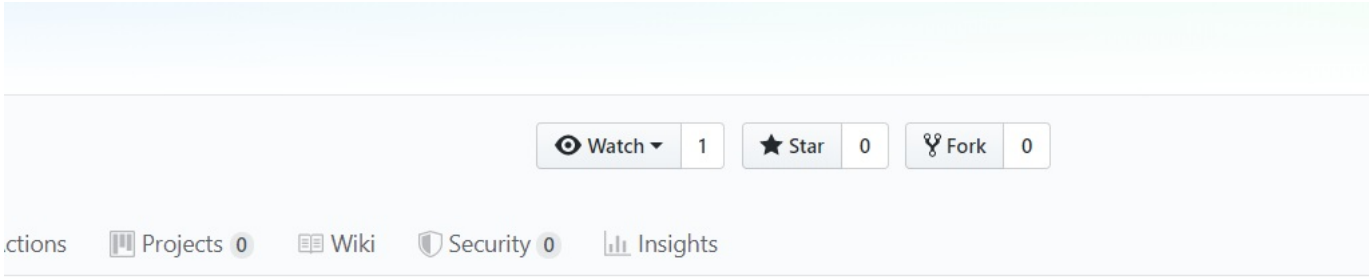
跨团队合作

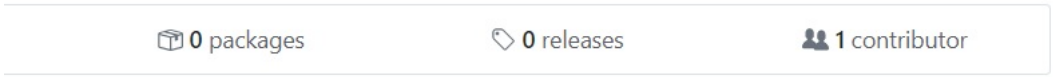
【1】得到远程库的地址： 



地址：
https://github.com/zhaoshanshan3366/GitResp2.git

【2】进行fork操作： 
进入到账号后：复制地址：https://github.com/zhaoshanshan3366/GitResp2.git
然后点击下面的fork操作：





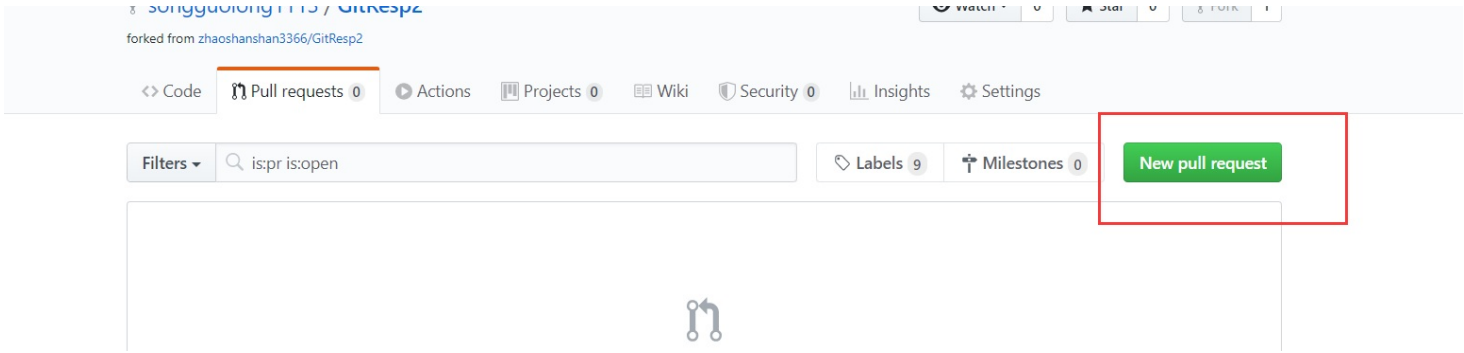
【3】然后就可以克隆到本地，并且进行修改： 

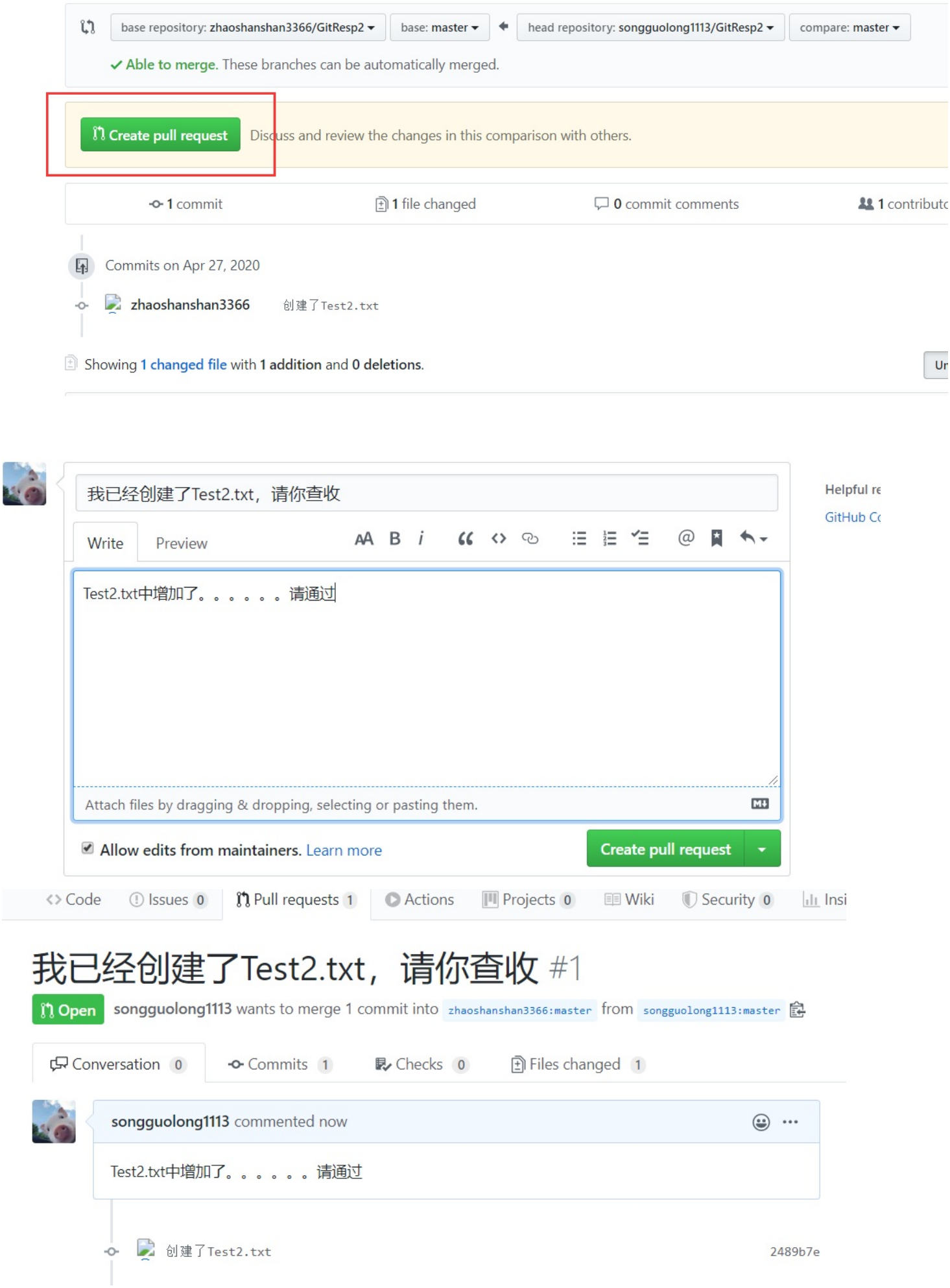
```
zss@LAPTOP-CRIVSRRU MINGW64 /c
$ git clone https://github.com/songguolong1113/GitResp2.git
Cloning into 'GitResp2'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 40 (delta 13), reused 29 (delta 2), pack-reused 0
Receiving objects: 100% (40/40), done.
Resolving deltas: 100% (13/13), done.
```

然后更改数据：添加到暂存区，然后提交到本地库，然后push到远程库：

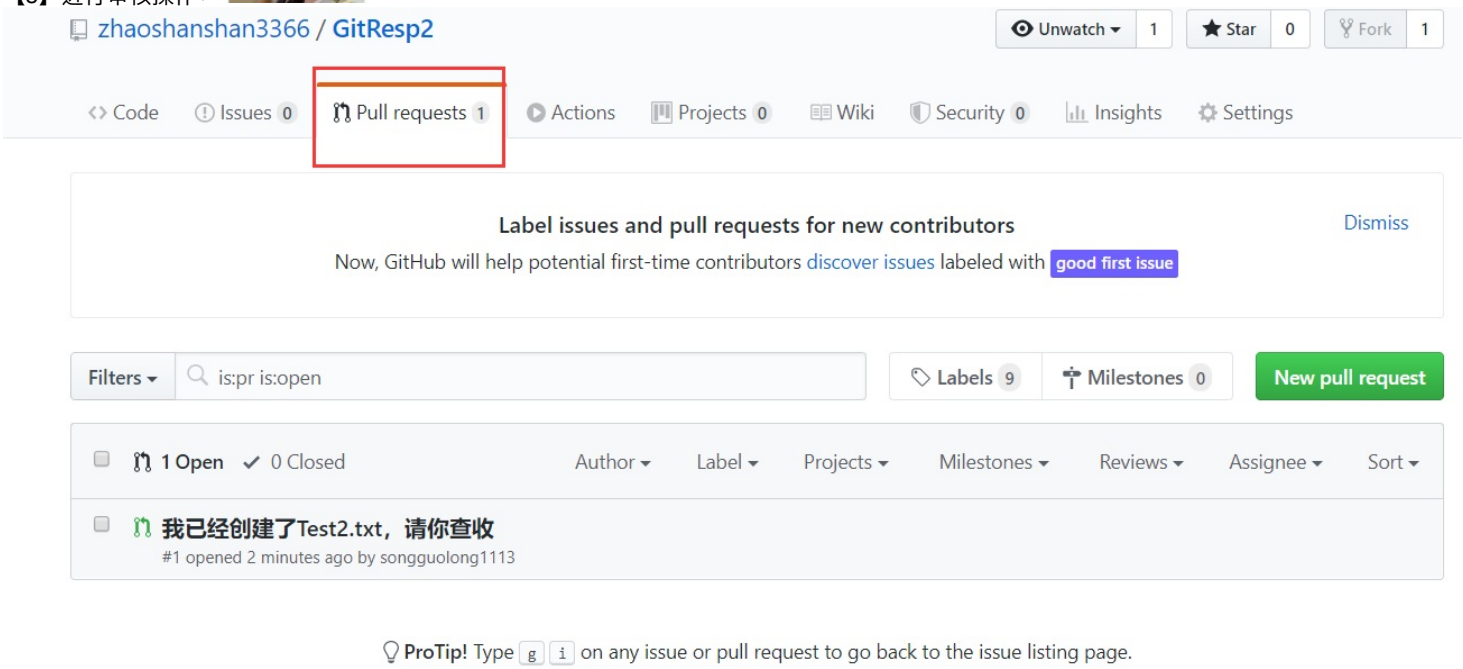
```
zss@LAPTOP-CRIVSRRU MINGW64 /c/GitResp2 (master)
$ git add Test2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /c/GitResp2 (master)
$ git commit -m "创建了Test2.txt" Test2.txt
[master 2489b7e] 创建了Test2.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Test2.txt

zss@LAPTOP-CRIVSRRU MINGW64 /c/GitResp2 (master)
$ git push https://github.com/songguolong1113/GitResp2.git master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/songguolong1113/GitResp2.git
   c5e4f3a..2489b7e  master -> master
```

【4】进行pull request操作： 



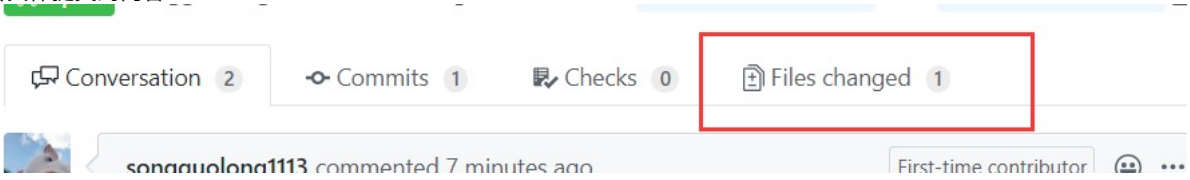

base repository: zhaoshanshan3366/GitResp2
base: master
head repository: songguolong1113/GitResp2
compare: master
Able to merge. These branches can be automatically merged.
Create pull request
Discuss and review the changes in this comparison with others.
1 commit
1 file changed
0 commit comments
1 contributo
Commits on Apr 27, 2020
zhaoshanshan3366
创建了Test2.txt
Showing 1 changed file with 1 addition and 0 deletions.
我已经创建了Test2.txt，请你查收
Write
Preview
Test2.txt中增加了。。。。。。请通过
Attach files by dragging & dropping, selecting or pasting them.
Allow edits from maintainers. Learn more
Create pull request
Helpful re
GitHub Co
Code
Issues 0
Pull requests 1
Actions
Projects 0
Wiki
Security 0
Insi
我已经创建了Test2.txt，请你查收 #1
Open
songguolong1113 wants to merge 1 commit into zhaoshanshan3366:master from songguolong1113:master
Conversation 0
Commits 1
Checks 0
Files changed 1
songguolong1113 commented now
Test2.txt中增加了。。。。。。请通过
创建了Test2.txt
2489b7e

【5】进行审核操作：

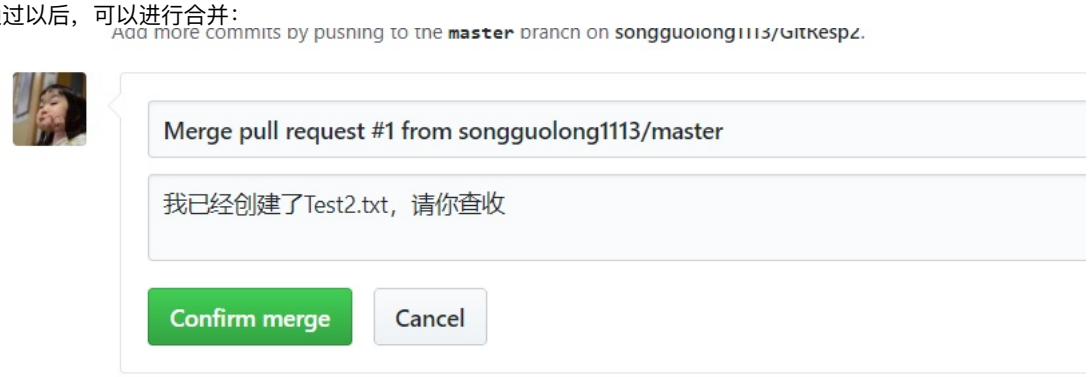


可以互相留言：

查看具体提交的内容：



确定通过以后，可以进行合并：



SSH免密登录

免密操作：

【1】进入用户的主目录中：

```
$ cd ~
```

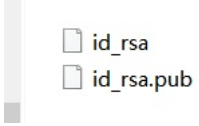
【2】执行命令，生成一个.ssh的目录：

```
zss@LAPTOP-CRIVSRRU MINGW64 ~
$ ssh-keygen -t rsa -C chinazss@126.com
Generating public/private rsa key pair.
```

keygen --- > key generation
注意：C要大写
后面的邮箱，是你的github注册的账号的时候对应的邮箱
三次回车确认默认值即可

```
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/zss/.ssh/id_rsa):
Created directory '/c/Users/zss/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/zss/.ssh/id_rsa
Your public key has been saved in /c/Users/zss/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:O1DVmEAiiNpHp9kEfOWyR34HLdRZdP5ubgy7Qid3cIO chinazss@126.com
The key's randomart image is:
+---[RSA 3072]----+
| . oo..o+.o= +o .|
|. . o.+o oo.+  o |
|.. . B. + o .  .o|
|. . + .*   o  E +|
|   .  o S . .  o.|
|       o o .o +..|
|        o  . + =o|
|         .  . .oo|
|            ..o.|
+----[SHA256]-----+
```

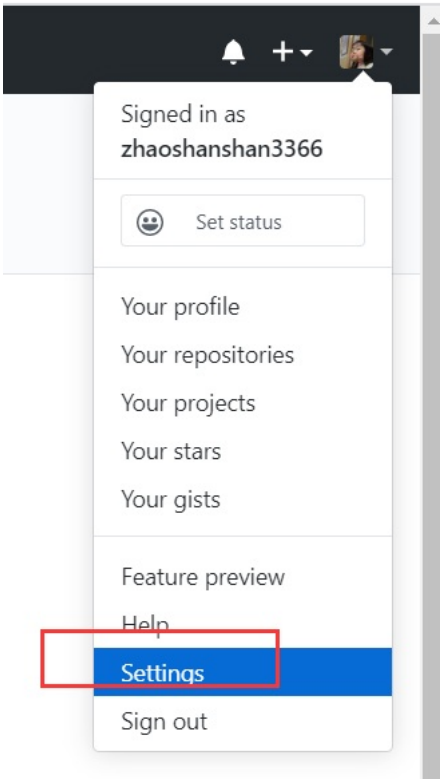
发现在.ssh目录下有两个文件：

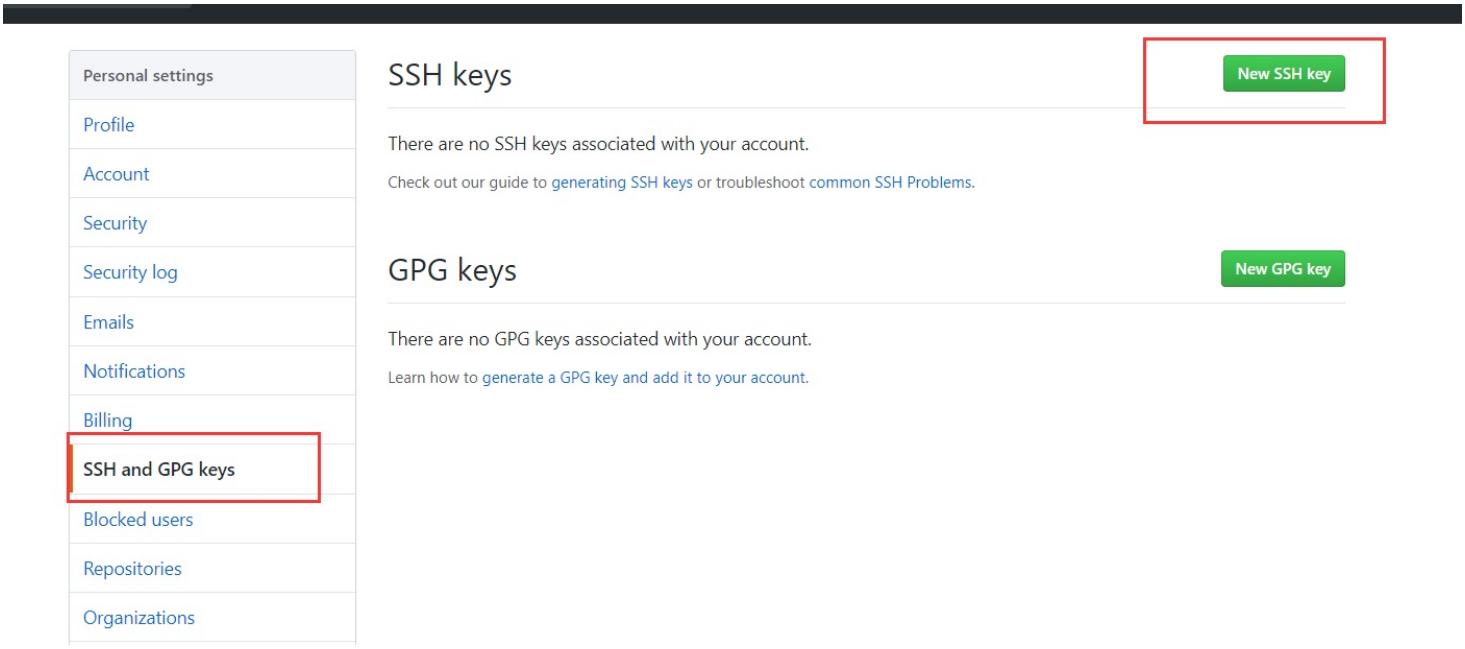


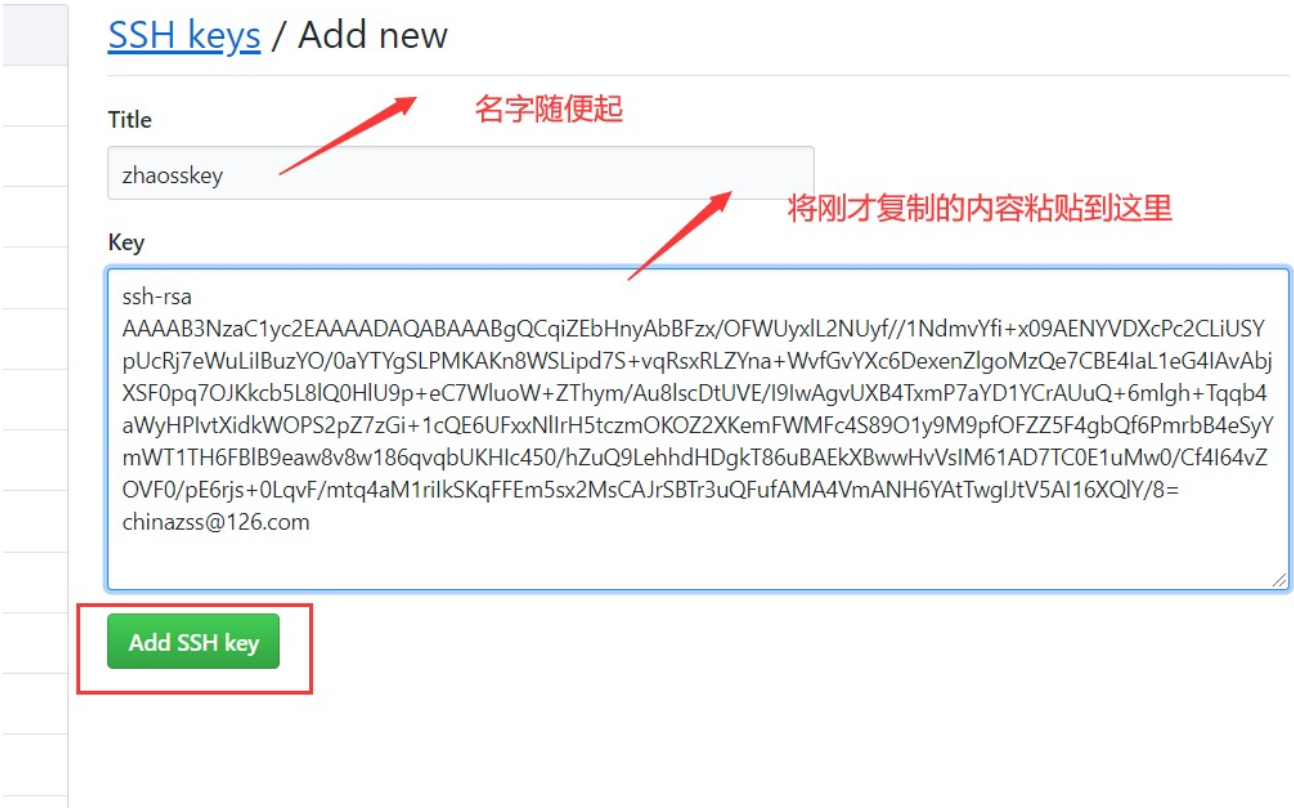
【3】打开id_rad.pub文件，将里面的内容进行复制操作：

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCqiZEbHnyAbBFzx/OFWUyxlL2NUyf//1NdmvYfi+x09AENYVDXcPc2CLiUSYpUcRj7eWuLiIBuzYO/0aY
chinazss@126.com
```

【4】打开github账号：


Signed in as
zhaoshanshan3366
Set status
Your profile
Your repositories
Your projects
Your stars
Your gists
Feature preview
Help
Settings
Sign out


Personal settings
Profile
Account
Security
Security log
Emails
Notifications
Billing
SSH and GPG keys
Blocked users
Repositories
Organizations
SSH keys
New SSH key
There are no SSH keys associated with your account.
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.



【5】生成密钥以后，就可以正常进行push操作了：
对ssh远程地址起别名：

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git remote add origin_ssh git@github.com:zhaoshanshan3366/GitResp2.git
```

展示别名：

```
$ git remote -v
origin  https://github.com/zhaoshanshan3366/GitResp2.git (fetch)
origin  https://github.com/zhaoshanshan3366/GitResp2.git (push)
origin_ssh      git@github.com:zhaoshanshan3366/GitResp2.git (fetch)
origin_ssh      git@github.com:zhaoshanshan3366/GitResp2.git (push)
```

创建一个文件：

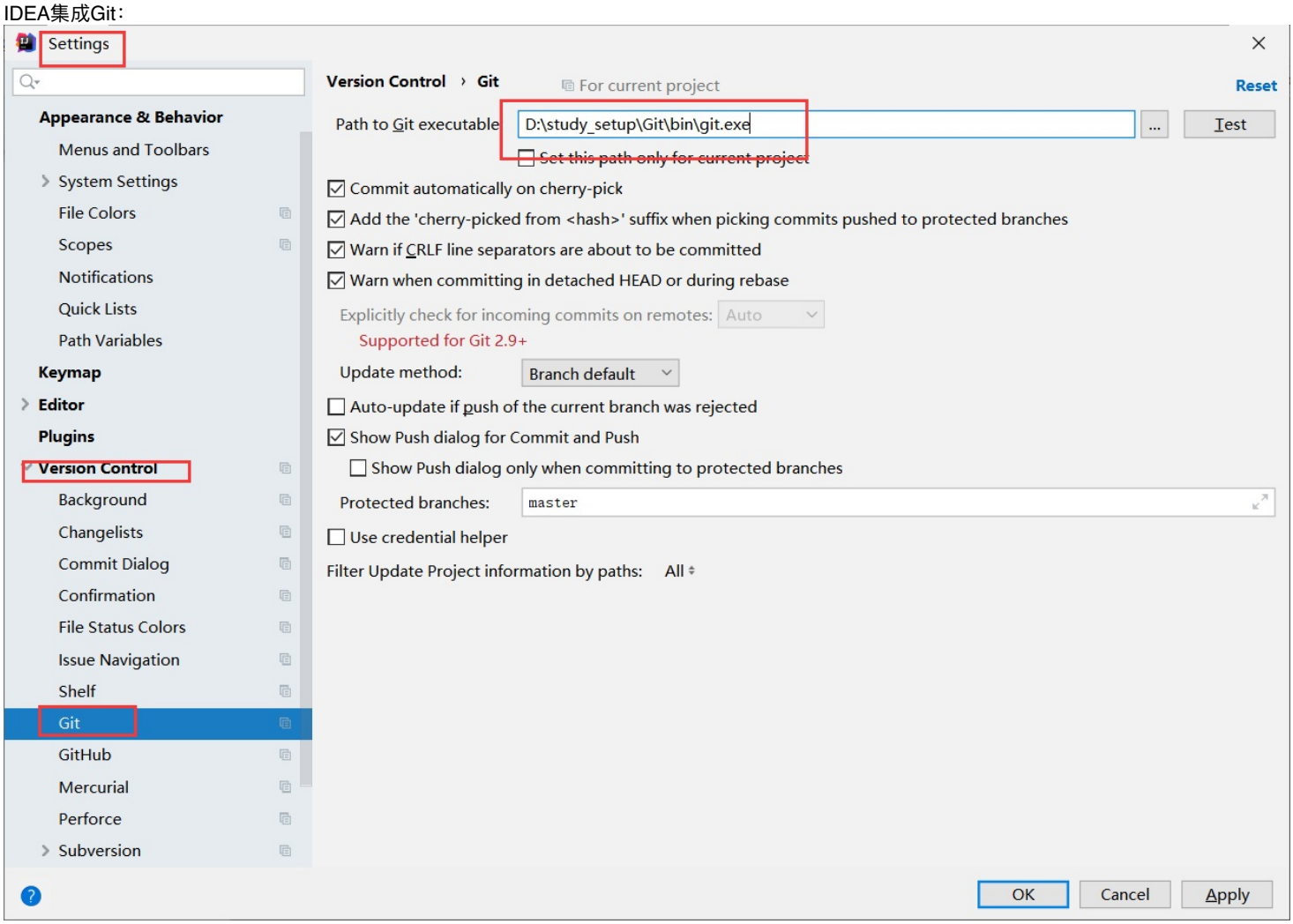
添加到暂存区，提交到本地库，然后push到远程库（地址用的是ssh方式的地址）

```
zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git add Test100.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git commit -m "创建了Test100.txt" Test100.txt
[master 6b25fb0] 创建了Test100.txt
 1 file changed, 1 insertion(+)
 create mode 100644 Test100.txt

zss@LAPTOP-CRIVSRRU MINGW64 /d/GitResp2 (master)
$ git push origin_ssh master
Warning: Permanently added the RSA host key for IP address
e list of known hosts.
```

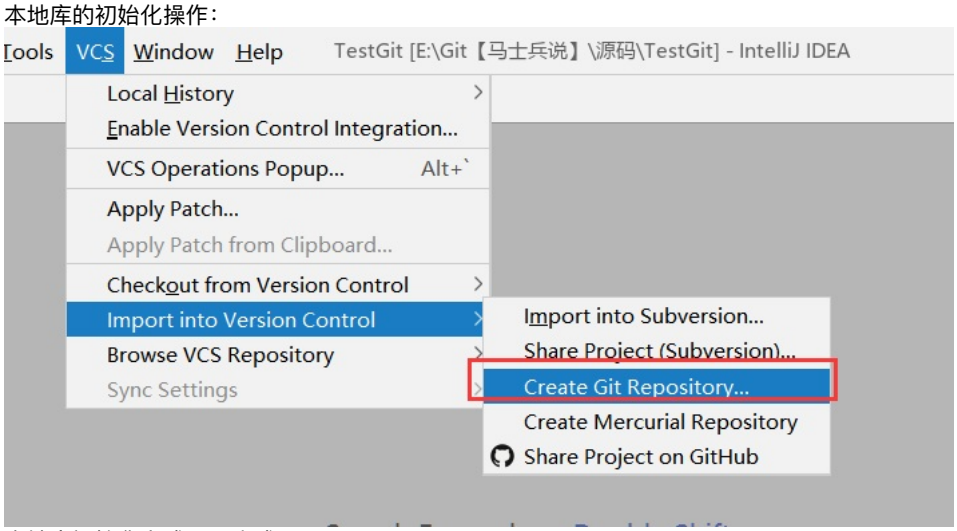
ssh方式好处：　不用每次都进行身份验证
缺陷：只能针对一个账号

IDEA集成Git

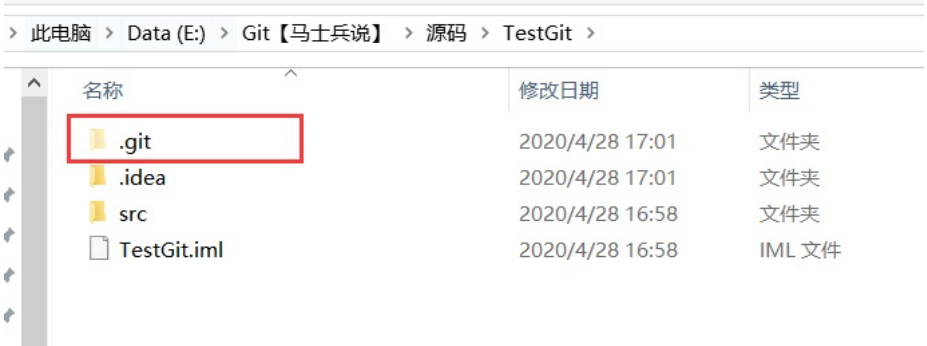
IDEA集成Git：



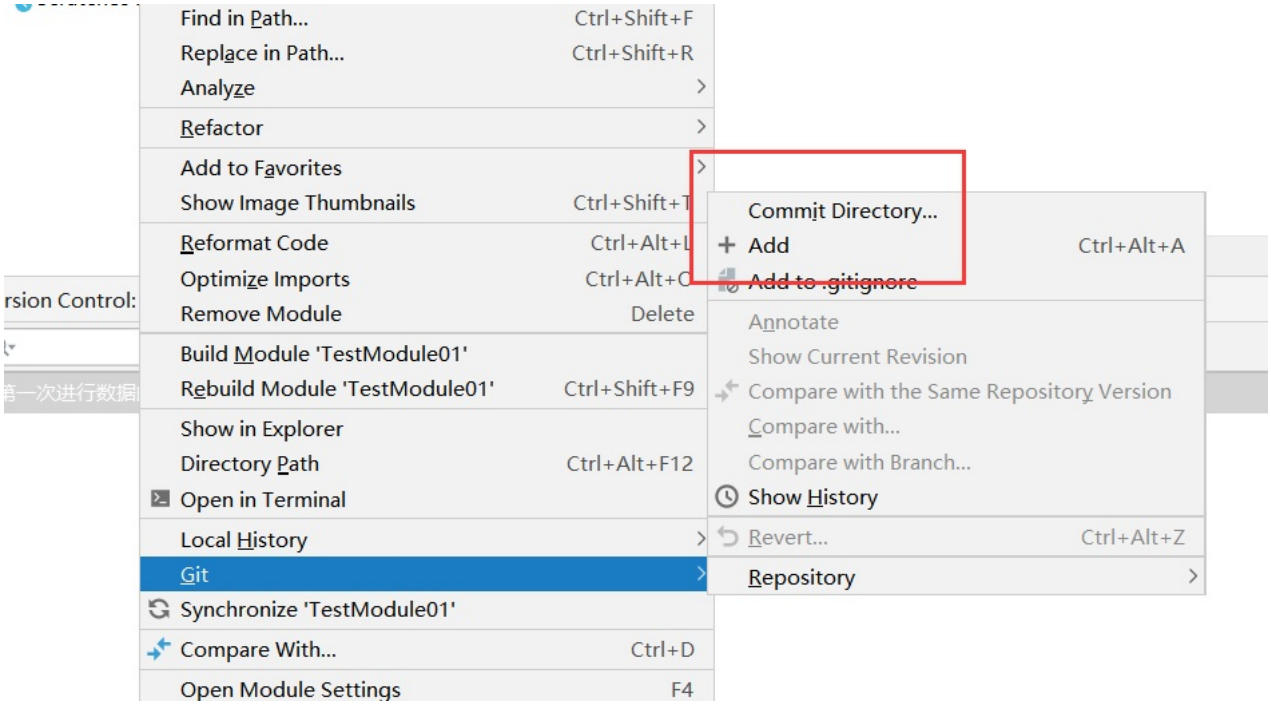
本地库的初始化操作：

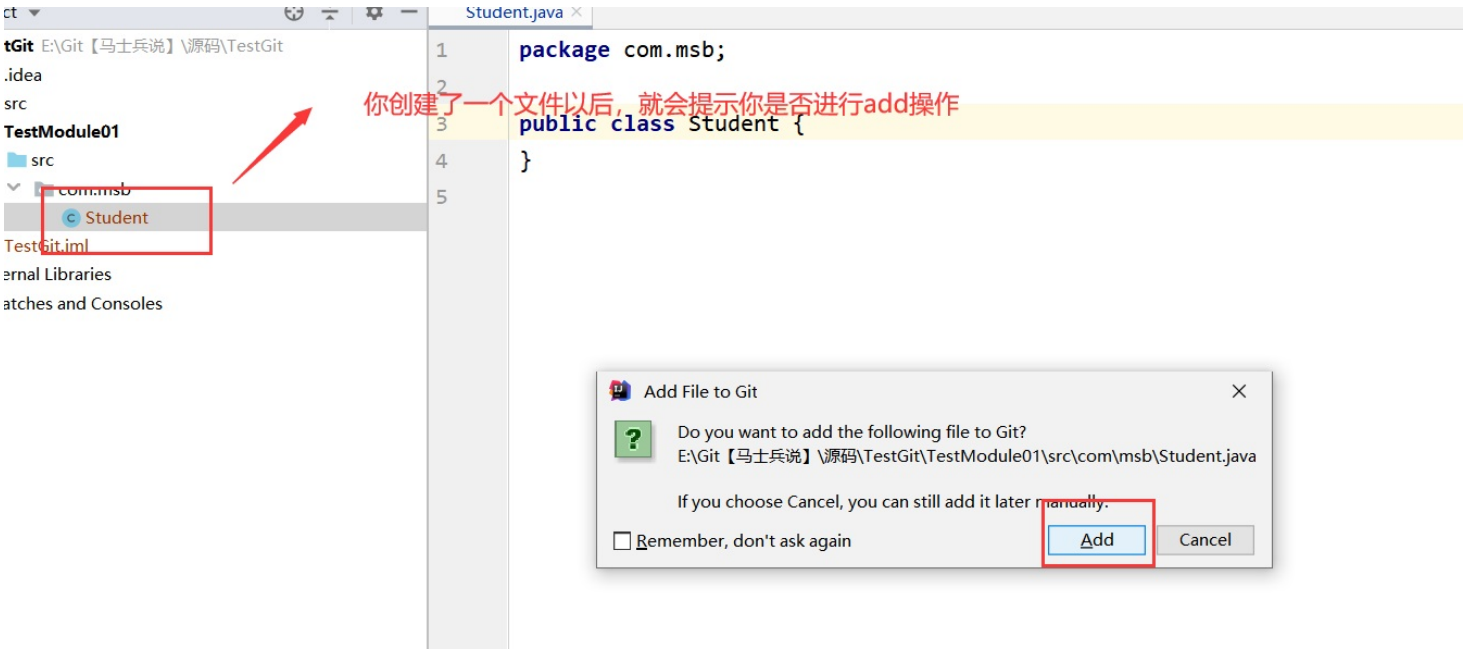


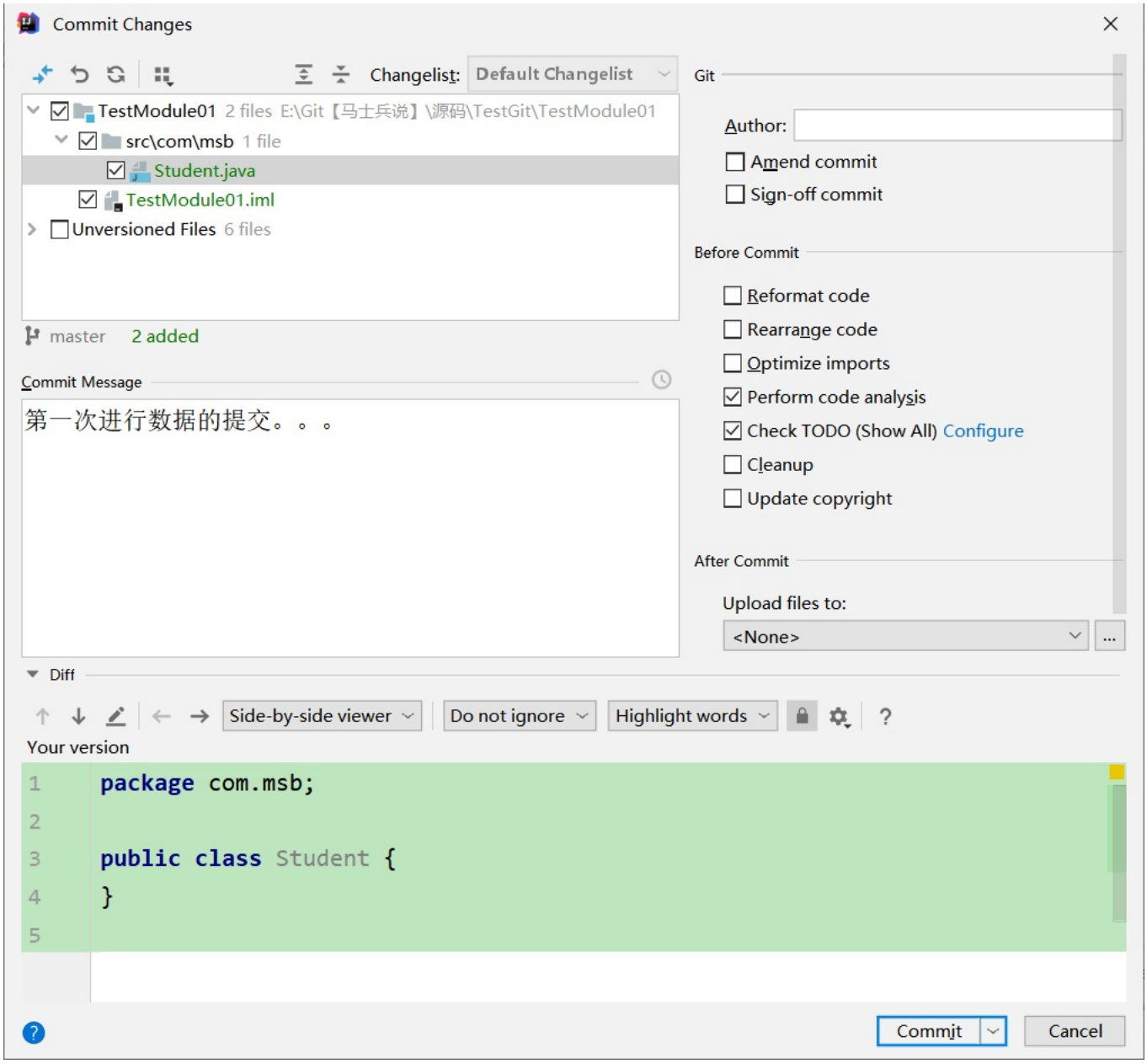
本地库初始化完成了，生成了.git目录：



添加到暂存区，再提交到本地库操作； add +commit:


Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Remove Module Delete
Build Module 'TestModule01'
Rebuild Module 'TestModule01' Ctrl+Shift+F9
Show in Explorer
Directory Path Ctrl+Alt+F12
Open in Terminal
Local History
Git
Synchronize 'TestModule01'
Compare With... Ctrl+D
Open Module Settings F4
Commit Directory...
Add Ctrl+Alt+A
Add to .gitignore
Annotate
Show Current Revision
Compare with the Same Repository Version
Compare with...
Compare with Branch...
Show History
Revert... Ctrl+Alt+Z
Repository


tGit E:\Git【马士兵说】\源码\TestGit
.idea
src
TestModule01
src
com.msb
Student
TestGit.iml
ernal Libraries
atches and Consoles
Student.java
1 package com.msb;
2
3 public class Student {
4 }
5
你创建了一个文件以后，就会提示你是否进行add操作
Add File to Git
Do you want to add the following file to Git?
E:\Git【马士兵说】\源码\TestGit\TestModule01\src\com\msb\Student.java
If you choose Cancel, you can still add it later manually.
Remember, don't ask again
Add
Cancel



当你更改内容以后，前面跟本地库内容不一致的地方会显示绿色：

```
2
3   public class Student {
4       private int age;
5       private String name;
6       private String sex;
7   }
8
```

本地库和远程库的交互

因为他们是两个不同的项目，要把两个不同的项目合并，git需要添加一句代码，在 `git pull` 之后，这句代码是在git 2.9.2版本发生的，最新的版本需要添加 `--allow-unrelated-histories` 告诉 git 允许不相关历史合并。

假如我们的源是origin，分支是master，那么我们需要这样写 `git pull origin master --allow-unrelated-histories`

这个方法只解决因为两个仓库有不同的开始点，也就是两个仓库没有共同的 commit 出现的无法提交。如果使用本文的方法还无法提交，需要看一下是不是发生了冲突，解决冲突再提交

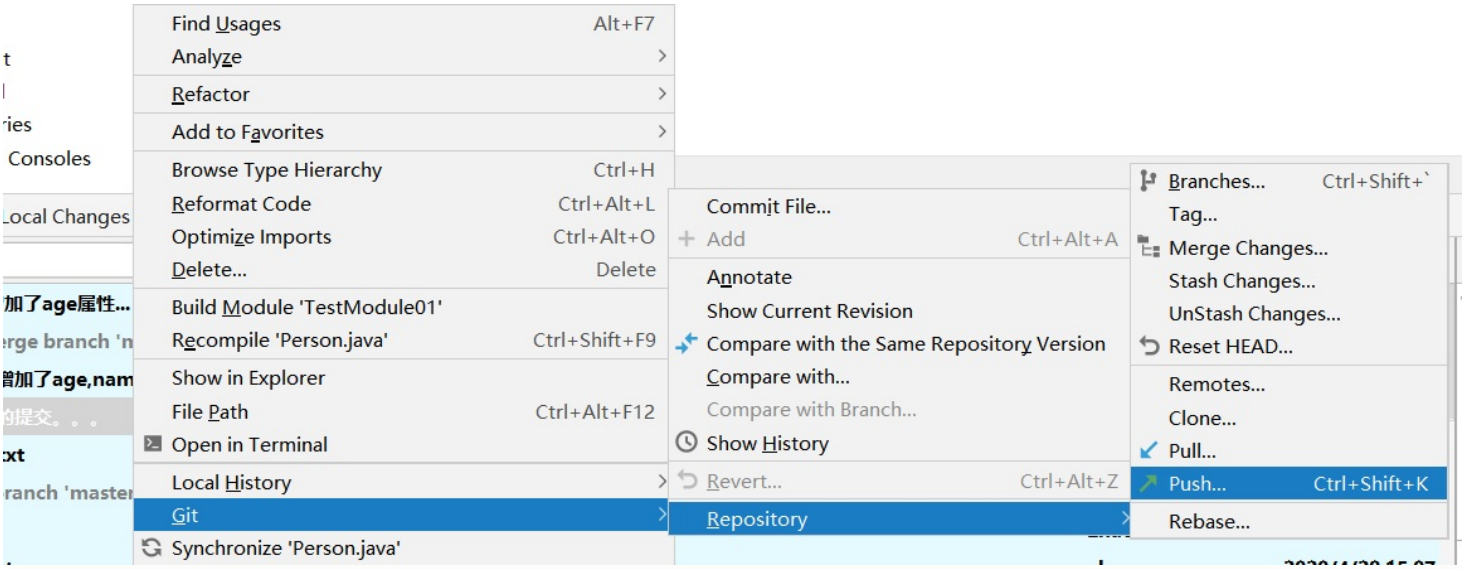
push推送： `git push -u origin master -f`

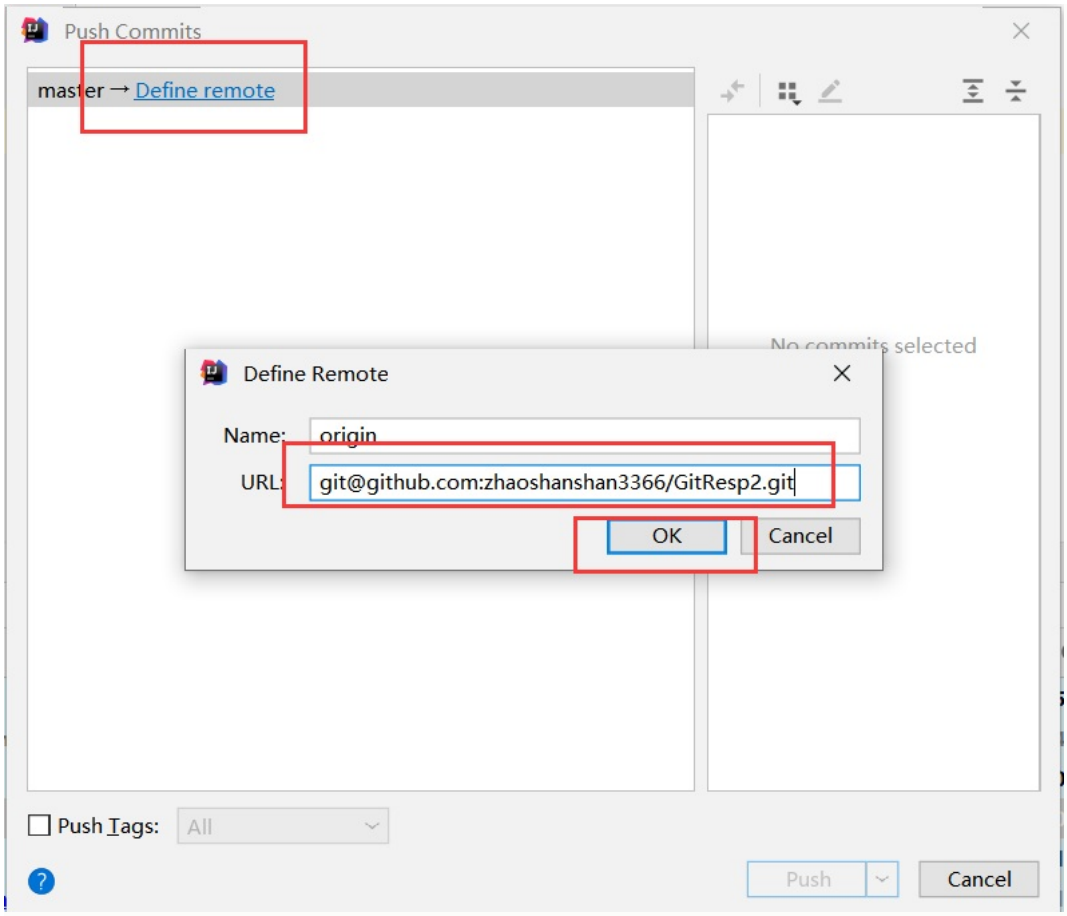
```
zss@LAPTOP-CRIVSRRU MINGW64 /e/Git【马士兵说】/源码/TestGit (master)
$ git pull git@github.com:zhaoshanshan3366/GitResp2.git master --allow-unrelated
-histories
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
```

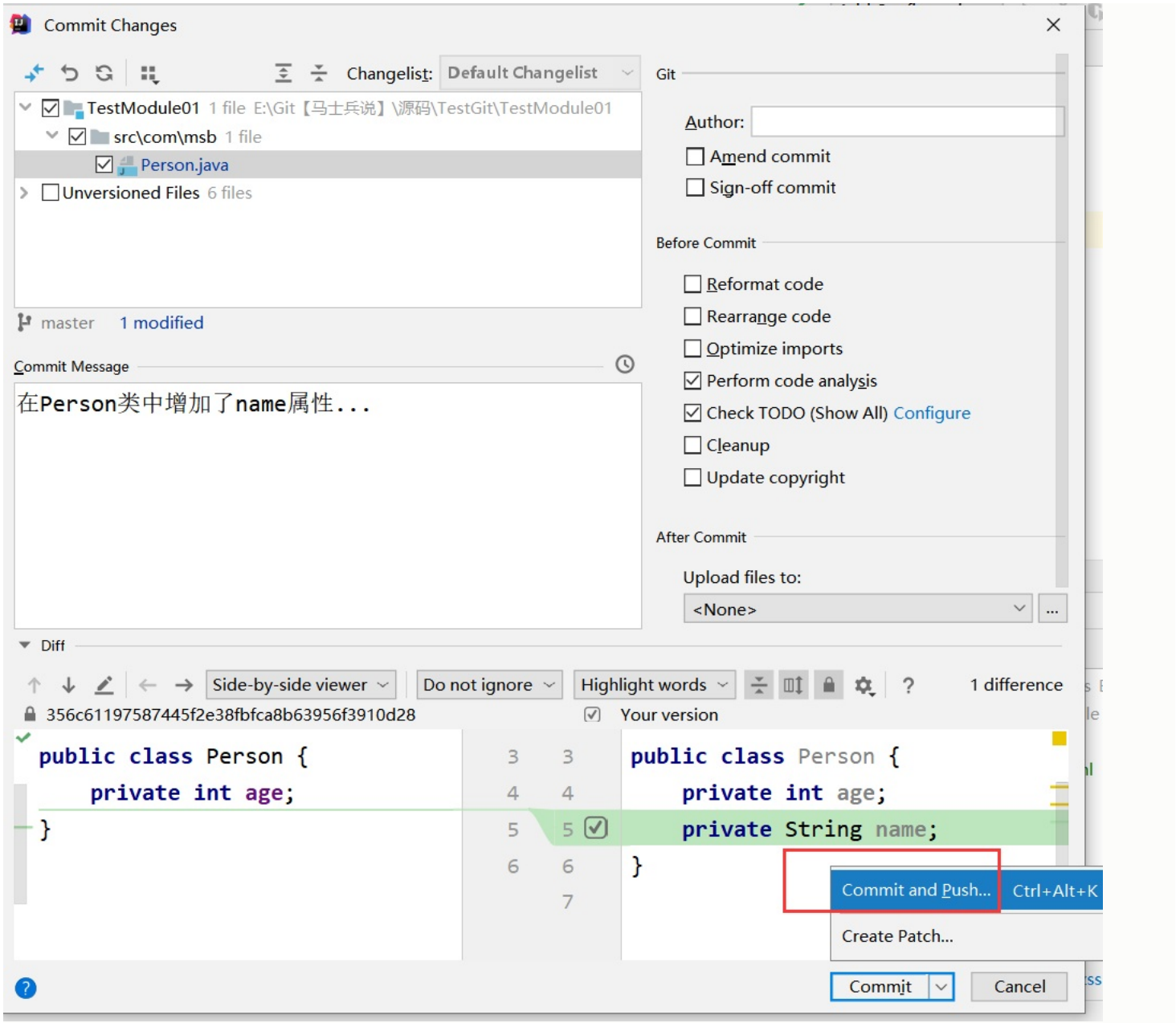
```
zss@LAPTOP-CRIVSRRU MINGW64 /e/Git【马士兵说】/源码/TestGit (master)
$ git push -u git@github.com:zhaoshanshan3366/GitResp2.git master -f
Warning: Permanently added the RSA host key for IP address '13.229.188.59' to t
```

到这里 远程库和本地库就可以进行交互了。

在IDEA中进行推送:



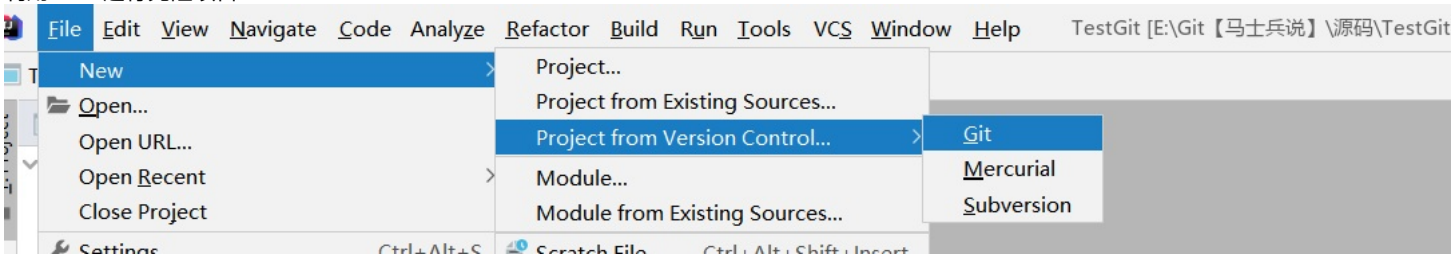

Push Commits
master → Define remote
No commits selected
Define Remote
Name: origin
URL: git@github.com:zhaoshanshan3366/GitResp2.git
OK
Cancel
Push Tags:
All
Push
Cancel

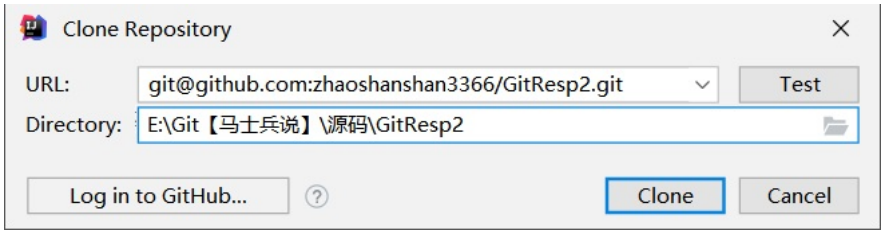


一般在开发中先pull操作，再push操作，不会直接进行push操作！！

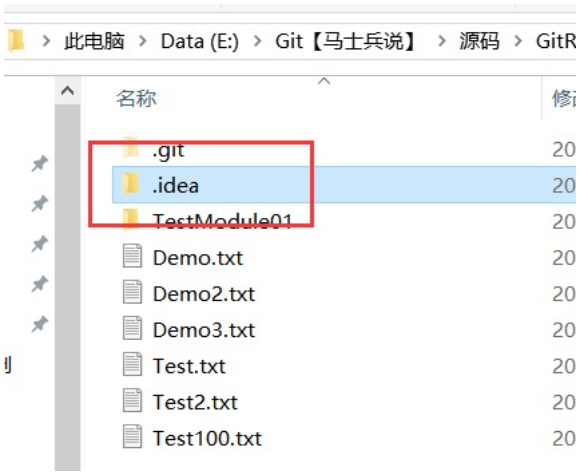
使用IDEA克隆远程库到本地

利用IDEA进行克隆项目：





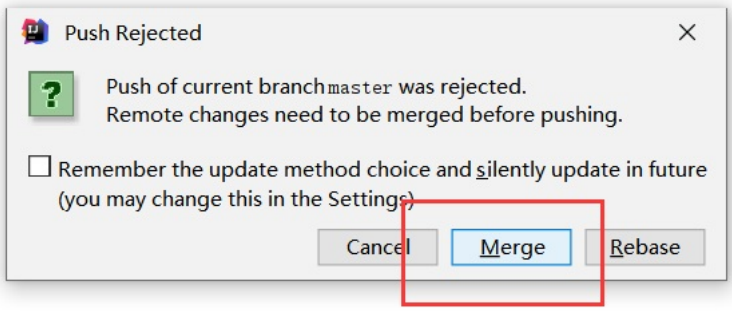
克隆到本地后：
这个目录既变成了一个本地仓库，又变成了工作空间。



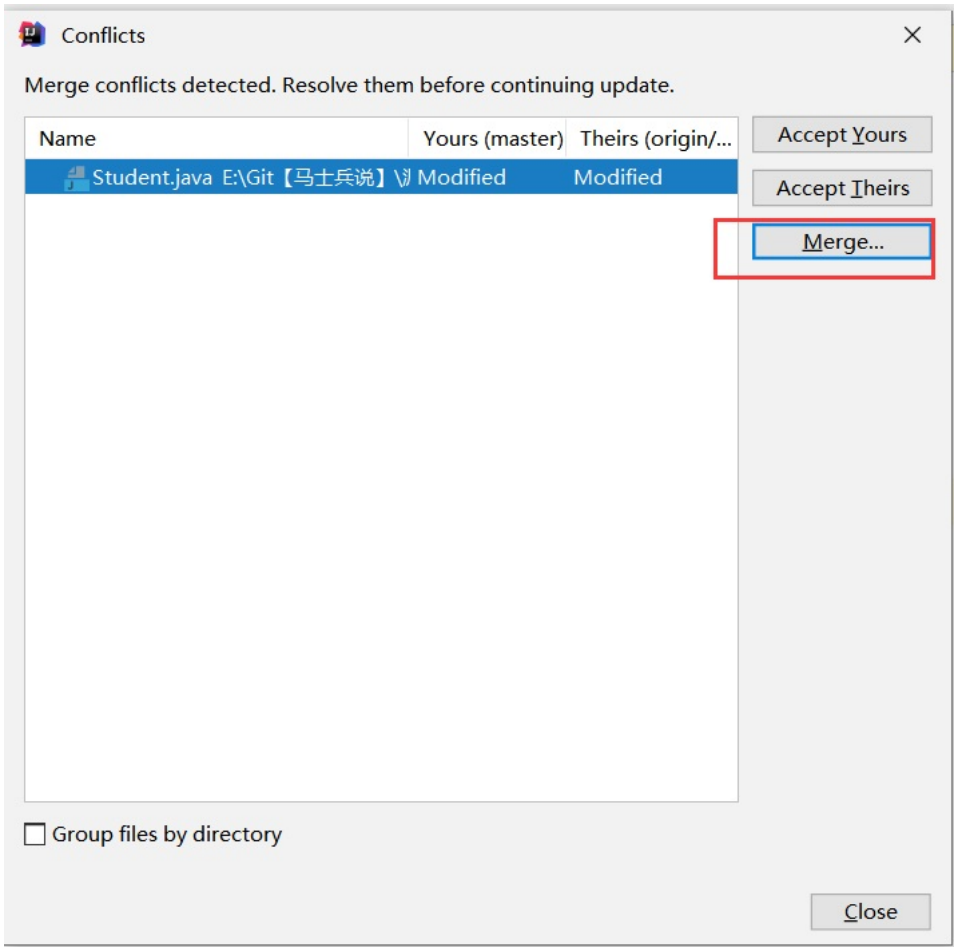
解决冲突

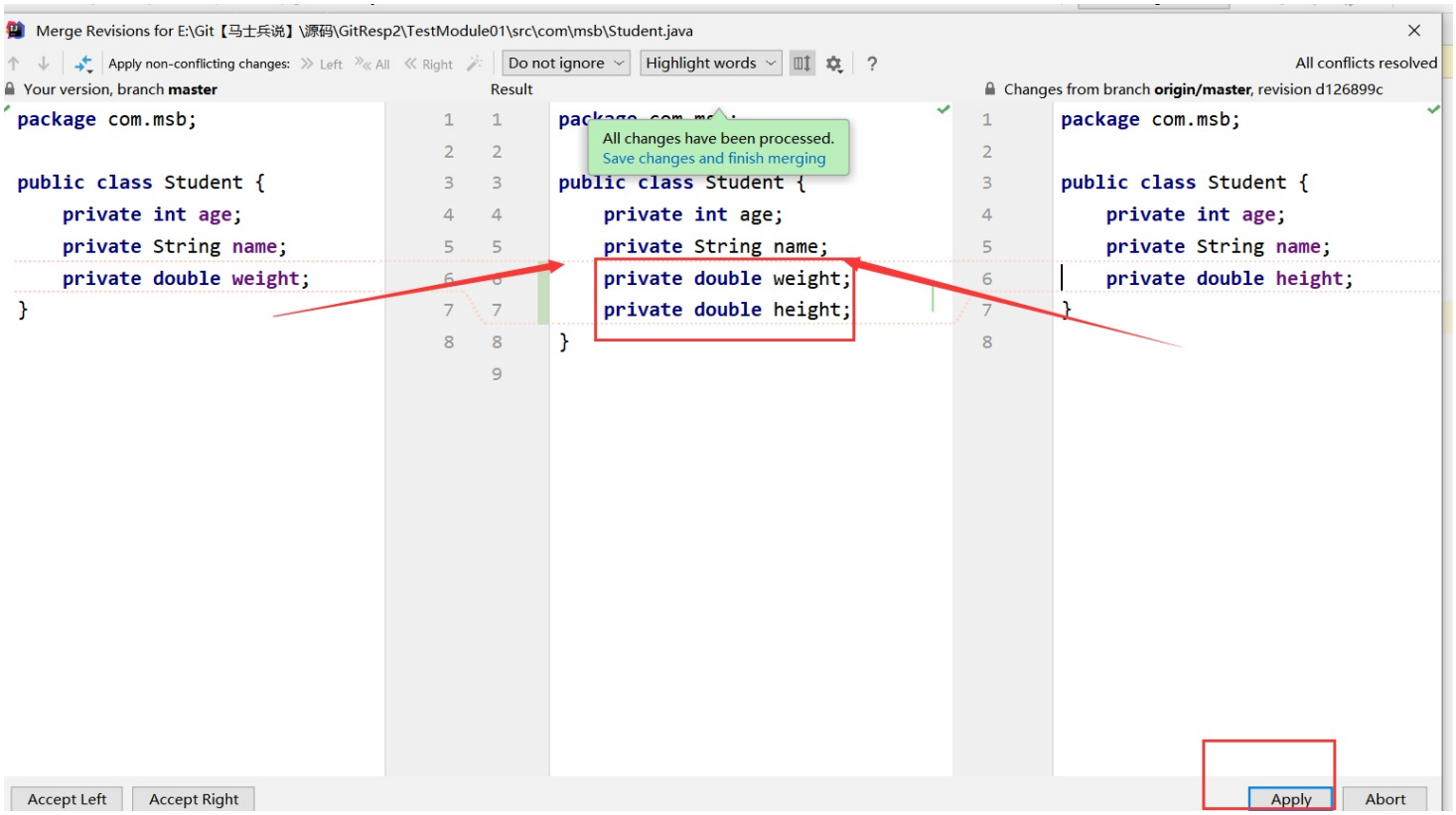
【1】在你push以后，有冲突的时候提示合并操作：

```
public class Student {
    private int age;
    private String name;
    private double weight;
}
```



合并：





如何避免冲突

【1】团队开发的时候避免在一个文件中改代码
【2】在修改一个文件前，在push之前，先pull操作