1. Git概述和环境搭建

> Git是一个开源的分布式版本控制系统，区别于集中式的系统，Git系统中每一个终端都有一套完整的版本库，脱离了网络，也能管理版本，能提高效率。
>
> 可以有效、高速地处理从很小到非常大的项目版本管理。
>
> 在团队开发中代码同步是非常重要的，大致开发的流程如下：
>
> 1. 在服务器上创建git仓库
> 2. 开发者A和B从服务器上克隆仓库
> 3. A和B在本地创建分支，负责开发自己的模块
> 4. A同步最新代码到本地，推送代码到服务器仓库
> 5. B同步最新代码到本地，推送代码到服务器仓库
> 6. 如果A和B在更新同步代码的时候有冲突，则解决冲突。若无冲突，则代码自动合并

基本概念：

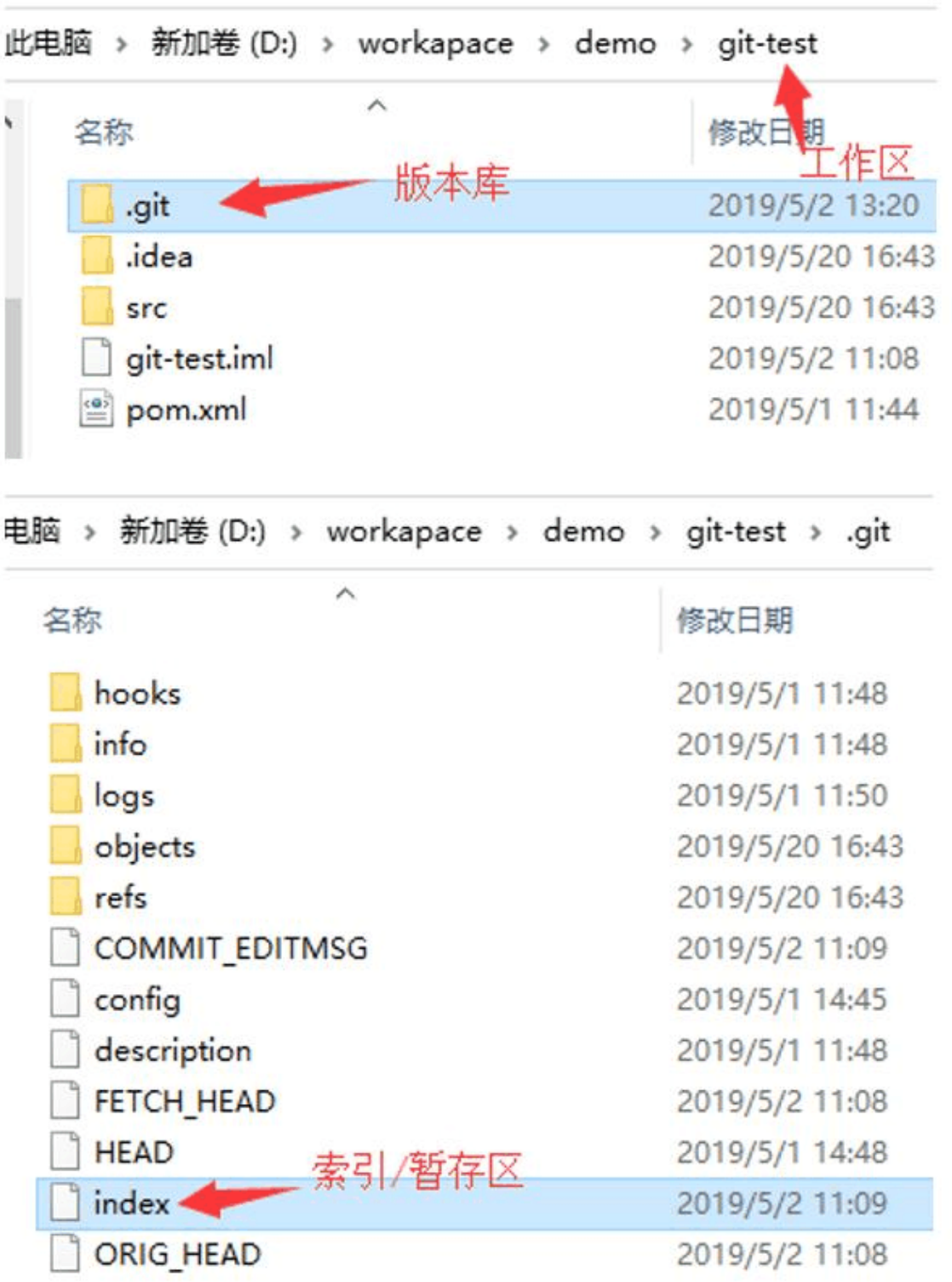
工作区：就是你在电脑里能看到的目录。

暂存区：一般存放在 ".git目录下" 下的index文件（.git/index）中，所以也叫索引

版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库

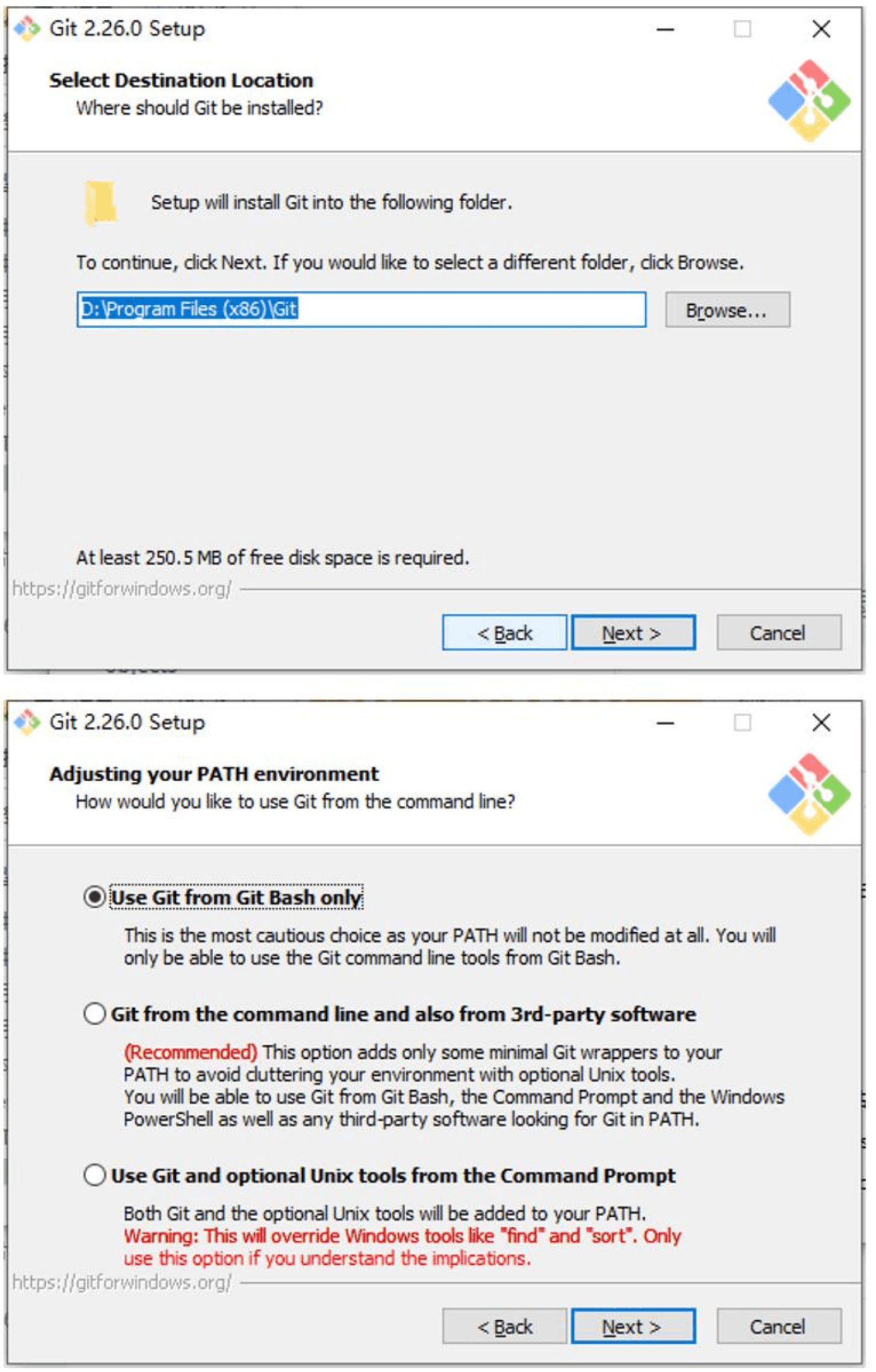
使用git add添加一个文件的时候，会先加入到暂存区；

通过git commit之后，暂存区的内容同步到正式的版本库。

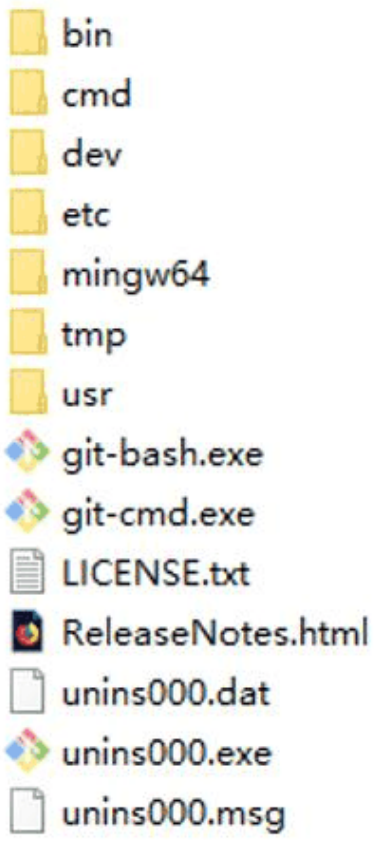


1.1 安装git客户端

下载git安装包https://git-scm.com/downloads，双击安装，选择安装位置，点下一步就行



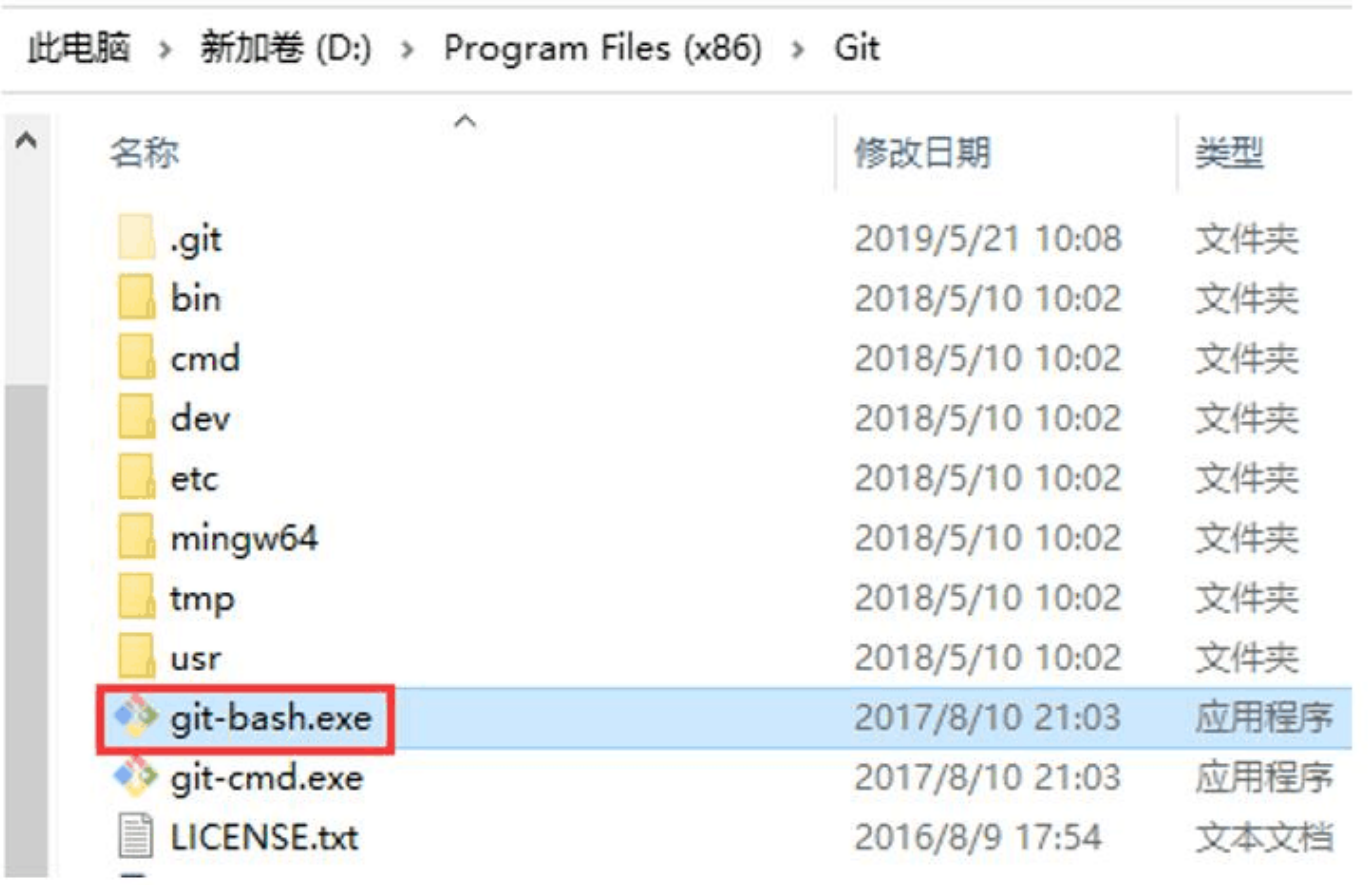
安装后的目录内容如下：



1.2 注册账号

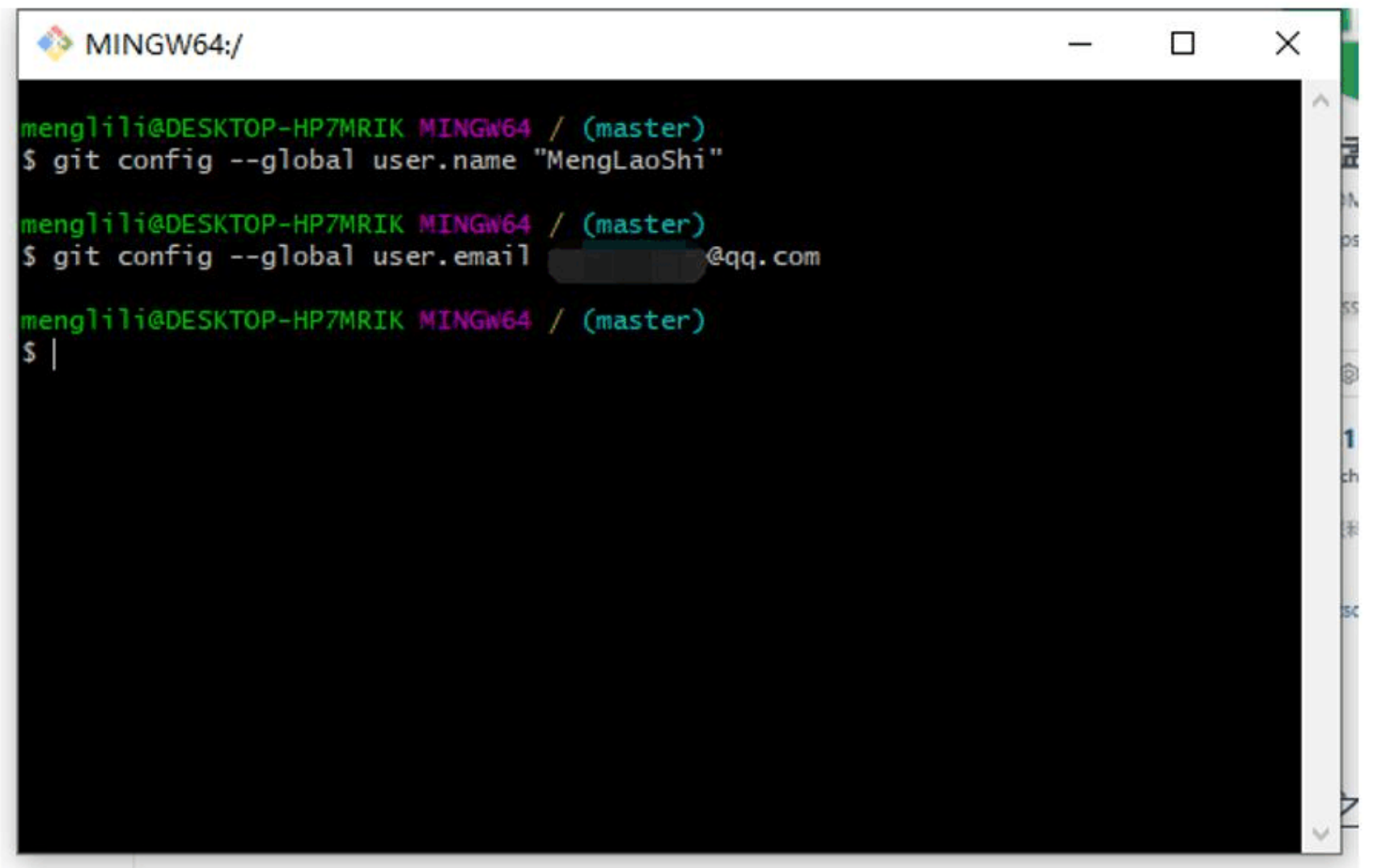
> 在我们团队开发，或者把代码共享给其他人的时候，需要有一个远程仓库，常用的远程git服务有：
>
> GitHub(**https://github.com/**)：公有库免费，私有付费。
>
> 码云(**https://gitee.com**)：国内的服务商，公有、私有都免费。
>
> GitLab(**https://gitlab.com**)：与前两者不同，GitLab是可以部署到自己服务器上的web服务。
>
> 在开始学习git之前，需要注册一个账号，码云或者GitHub的都可以(两个平台是独立的，二者不能通用)。

注册成功之后，打开git安装路径下的git-nash.exe

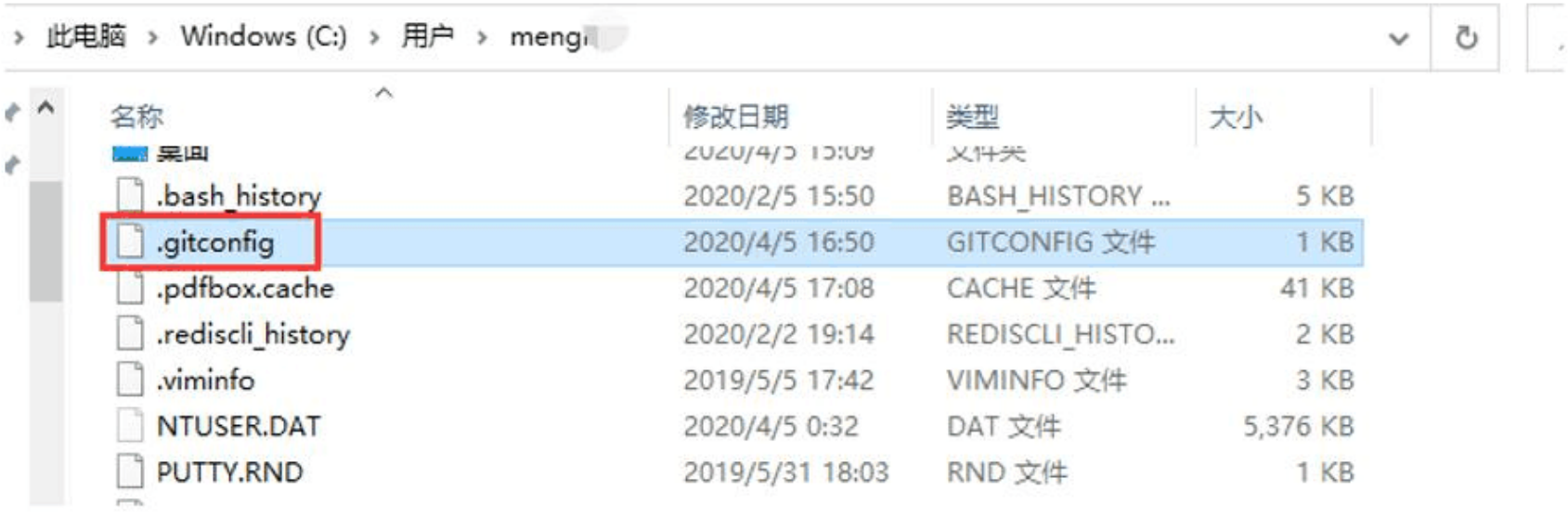


> 输入用下面的命令,做一个全局配置，后序提交代码的时候都会携带这个信息，最好用前面注册git的邮箱

```
git config --global user.name "你的名字"
git config --global user.email "你的邮箱"
```

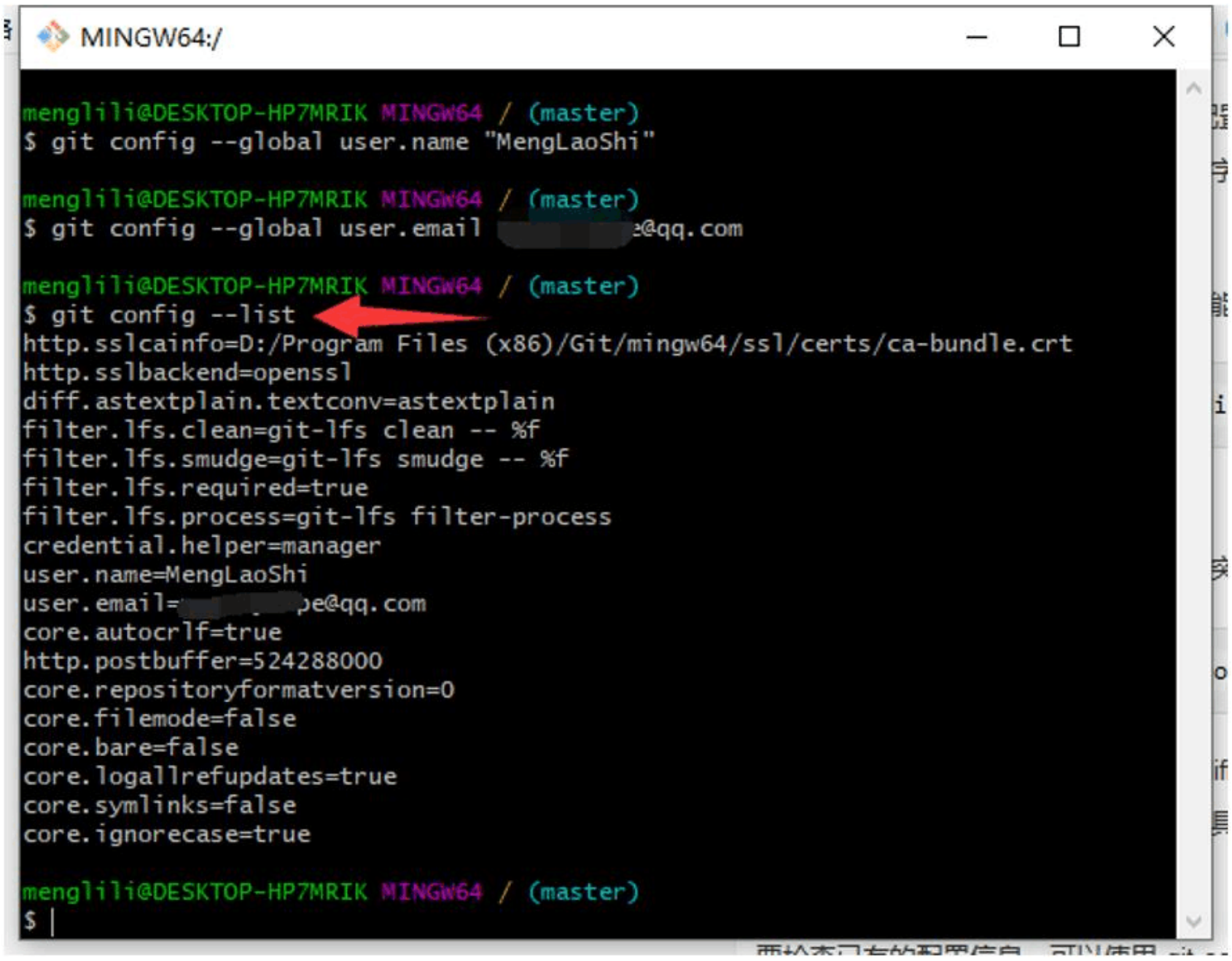


> 之后在你的电脑用户目录下会出现一个.gitconfig文件，全局配置就设置好了



使用命令：`git config --list`

可以查看配置信息



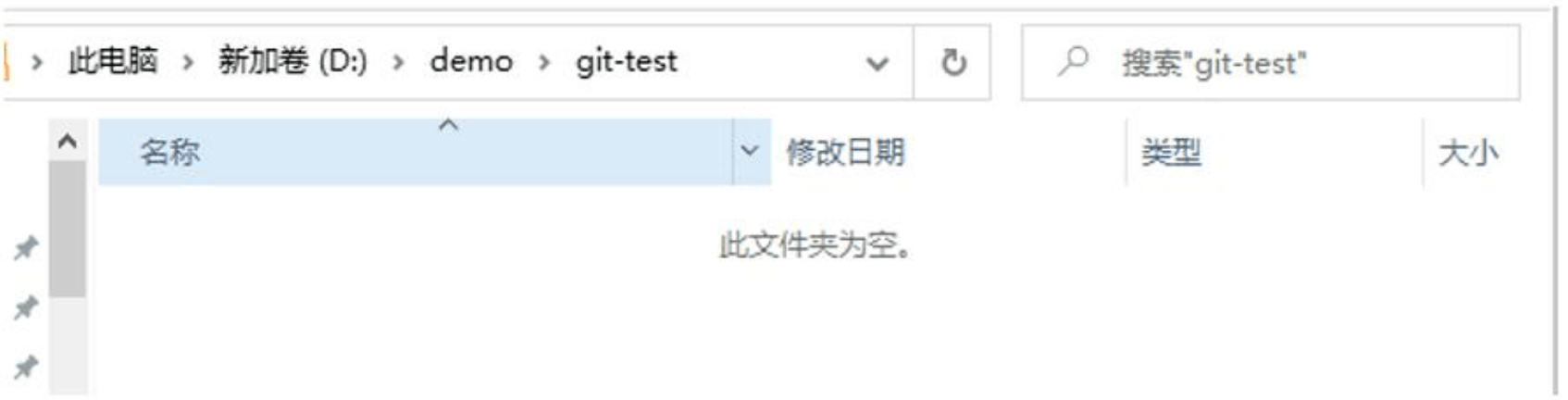
--global参数代表的是全局配置，对当前用户的所有仓库有效，常用

--local只对某个仓库有效，不写的话默认就是local

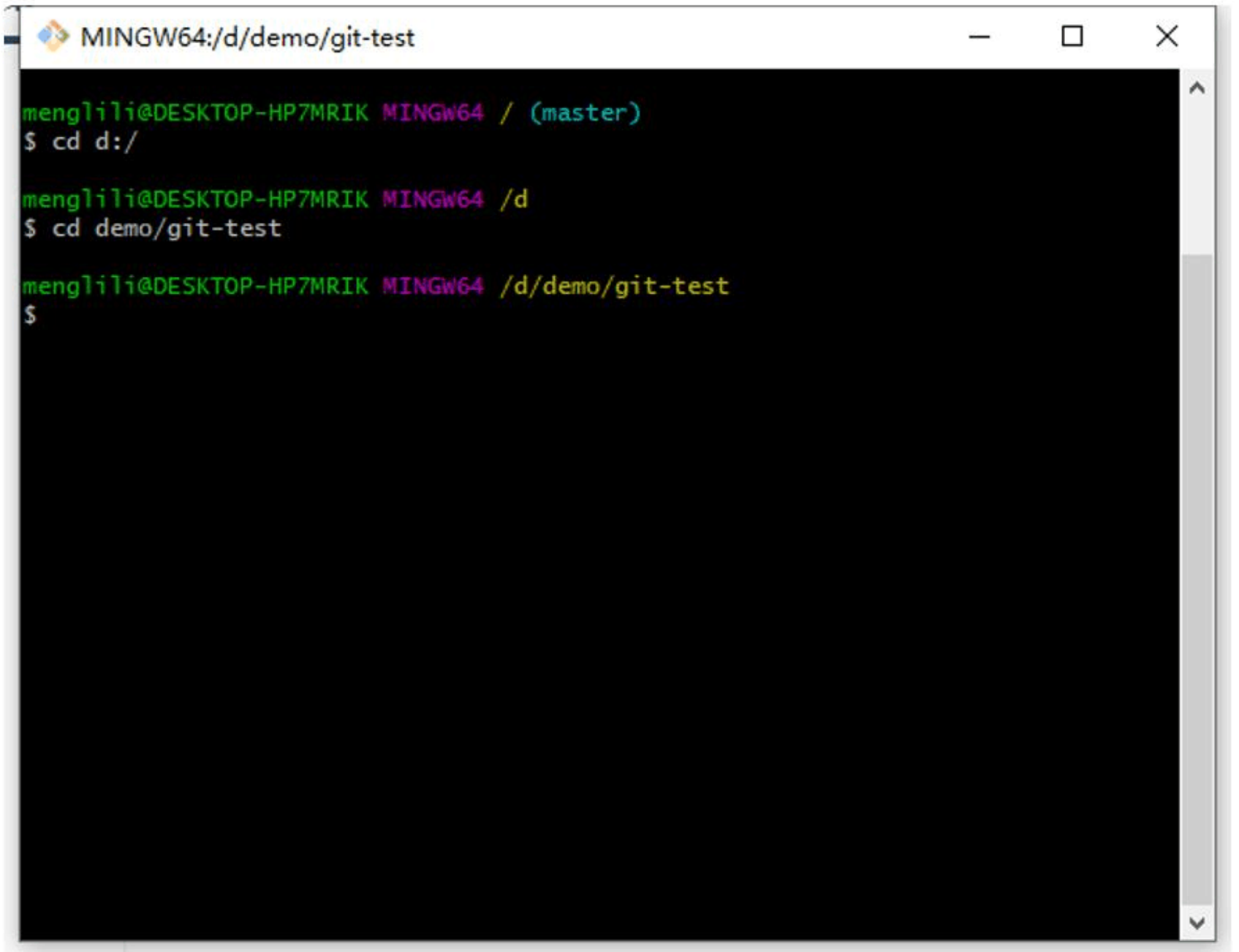
--system对系统所有登录的用户有效，很少用

2. 基础命令

创建一个空文件夹用来当本地仓库：



在git-bash中切换目录



git中支持linux中的命令，比如

`cd` 打开目录

`vi` 编辑文本 `:wq` 保存并退出

`pwd` 是显示当前目录

`ls` 显示文件夹中的内容

`cat` 查看文件内容

> 如果没有linux基础也没关系，记住这几个命令，或者直接操作文件也可以。

MINGW64:/d/demo/git-test

```
menglili@DESKTOP-HP7MRIK MINGW64 / (master)
$ cd d:/demo/git-test

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ pwd
/d/demo/git-test

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```

2.1 git init

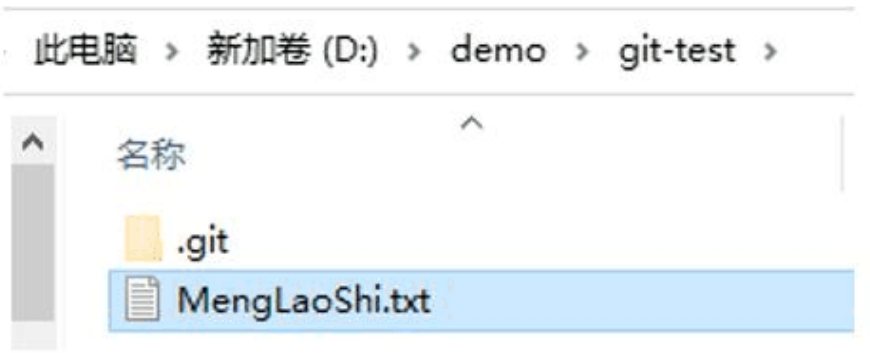
> 执行`git init`

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test
$ git init
Initialized empty Git repository in D:/demo/git-test/.git/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

> 之后在目录里会看到一个`.git`文件夹

2.2 git add

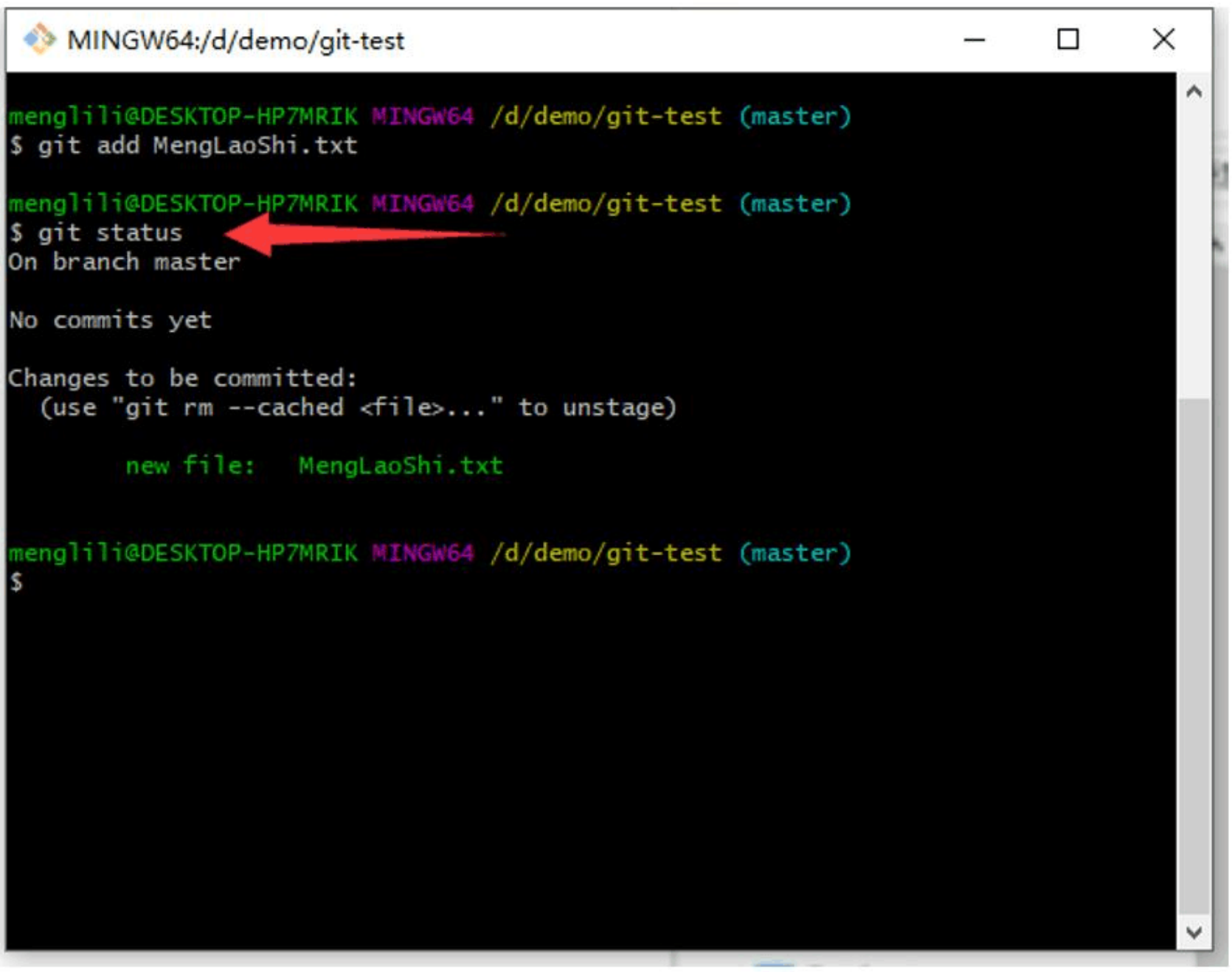
> 在目录下手动创建一个文件(随便什么文件都行，就是我们平时用的文件)：



> 将文件添加到git中的命令是：`git add 文件名`
>
> `git add -u` 可以将当前托管的文件一起提交，不需要指定文件名
>
> 此时的文件进入了git的`暂存区`

2.3 git status

> 通过命令`git status`可以看到当前暂存区的状态：



2.4 git commit

> 通过 `git commit` 命令正式提交到版本库（只提交暂存区中的数据），其中 `-m` 参数表示的是提交时的注释信息，我们要写好当前版本都修改了什么，方便以后版本回退的时候，退到指定的版本。

> 当commit提交之后，再次查看暂存区，发现已经没有东西了：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git commit -m '第一次提交'
[master (root-commit) 212543a] 第一次提交
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git status
On branch master
nothing to commit, working tree clean

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

> commit只提交暂存区中的数据，所以需要先add，如果是文件已经add过了，可以使用-am参数直接提交

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git commit -am '第二次提交'
[master c9b98d1] 第二次提交
 1 file changed, 2 insertions(+)

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```

> 如果想修改最近一次的提交信息message,使用 `git commit --amend`

2.5 git diff

> 接下来我们修改文件内容，然后保存：

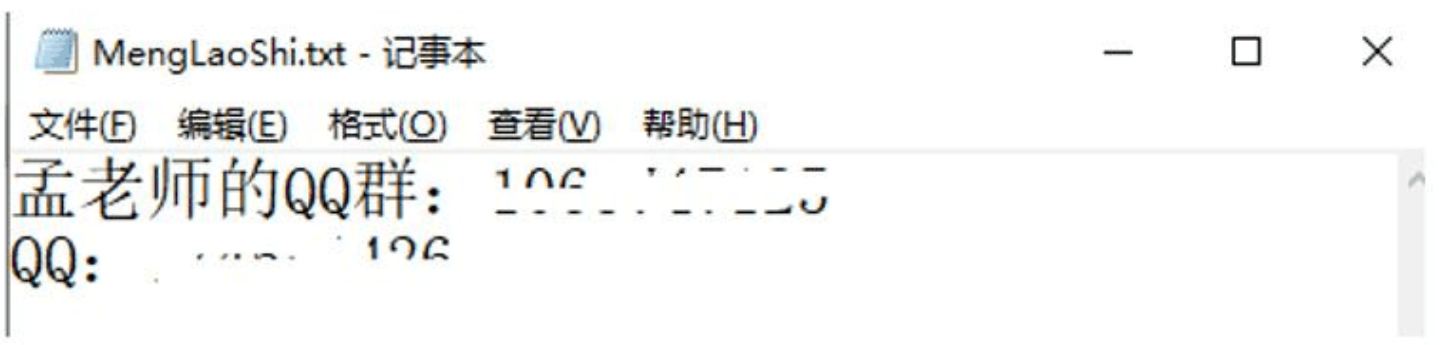
再执行 `git add` 和 `git status`,可以看见文件是一个 `modified` 状态:

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git add MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   MengLaoShi.txt
```

执行 `git diff --cached` 可以查看暂存区中的变动(和HEAD的差别):

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git diff --cached
diff --git a/MengLaoShi.txt b/MengLaoShi.txt
index e69de29..925272c 100644
--- a/MengLaoShi.txt
+++ b/MengLaoShi.txt
@@ -0,0 +1 @@
+孟老师的QQ群: 1063747125
\ No newline at end of file

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```

如果不加 `--cached` 参数，则是查看工作区和暂存区的差别:



```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git diff
diff --git a/MengLaoShi.txt b/MengLaoShi.txt
index 925272c..5291500 100644
--- a/MengLaoShi.txt
+++ b/MengLaoShi.txt
@@ -1 +1,2 @@
-孟老师的QQ群：1063747125
\ No newline at end of file
+孟老师的QQ群：1063747125
+QQ：574549426
\ No newline at end of file

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```

`git diff -- 文件名` 是只查看某一个文件的区别

`git diff 分支1 分支2 -- 文件名` 查看两个分支之间的区别，分支可以写成commit号

2.6 git reset

`git reset` 用于回滚暂存区中的操作。对文件进行修改：

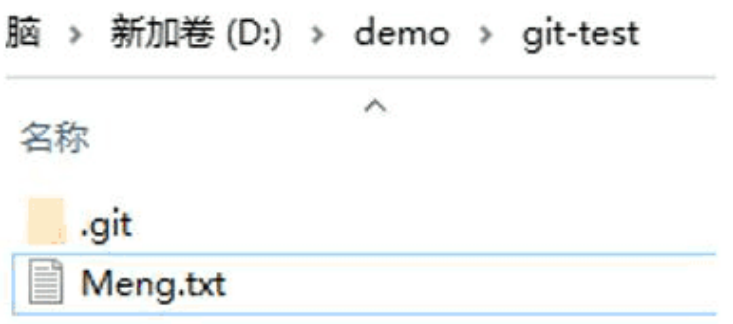
通过add添加到暂存区，然后reset，可以看到刚才暂存区的操作被取消了

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git add MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git diff --cached
diff --git a/MengLaoShi.txt b/MengLaoShi.txt
index 5291500..0d9118b 100644
--- a/MengLaoShi.txt
+++ b/MengLaoShi.txt
@@ -1,2 +1,3 @@
 孟老师的QQ群：1063747125
-QQ：574549426
\ No newline at end of file
+QQ：574549426
+============
\ No newline at end of file

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git reset
Unstaged changes after reset:
M       MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git diff --cached

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

2.7 git mv

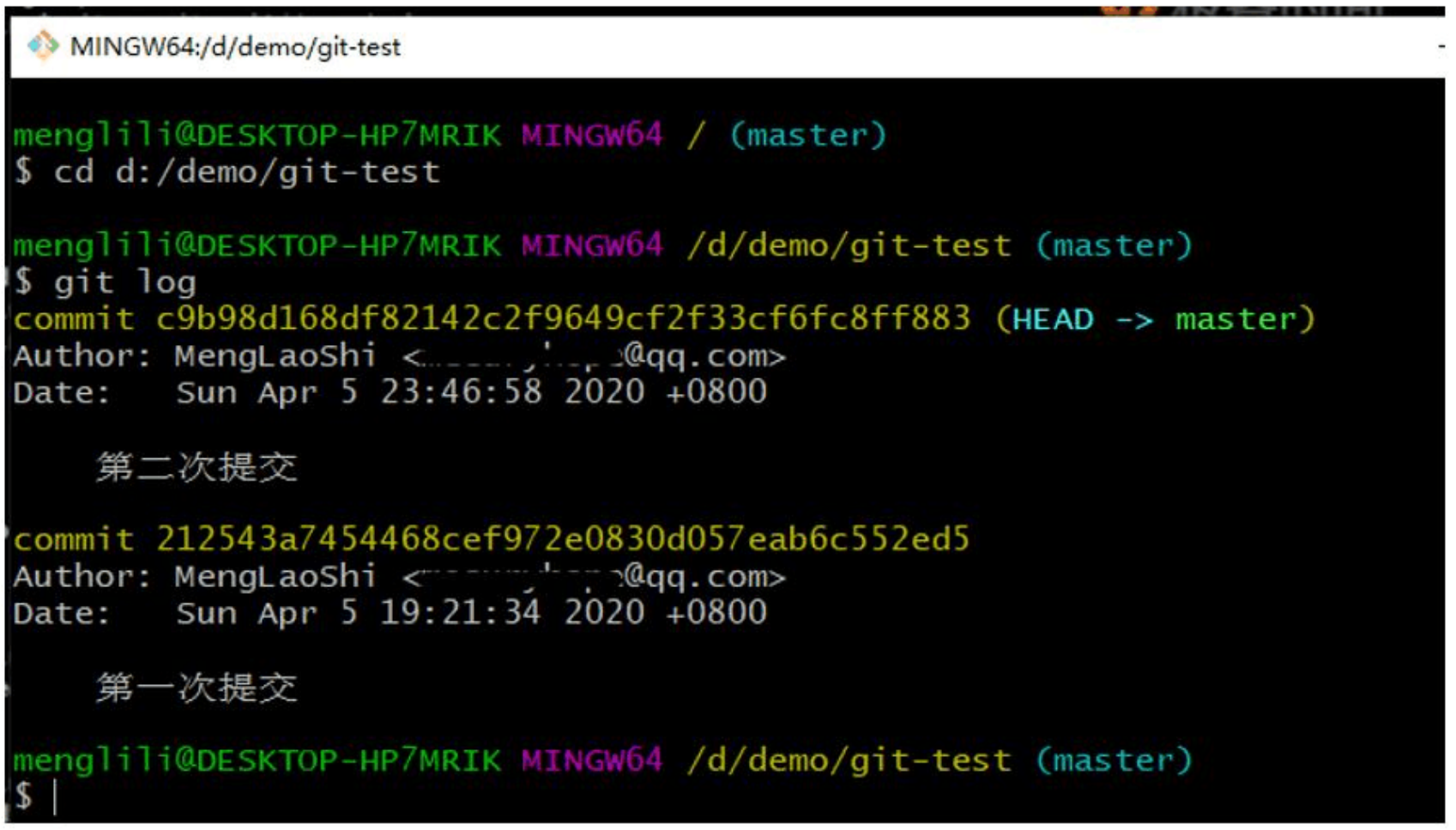
> 文件重命名`git mv 原文件 目标文件`
>
> 这个命令会同时修改git库和你本地的文件名

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git mv MengLaoShi.txt Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git commit -m '重命名'
[master b29a662] 重命名
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename MengLaoShi.txt => Meng.txt (100%)

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```



2.8 git log

> 查看日志 `git log`



> 参数:
>
> `--oneline` 简洁的查看历史
>
> `-n数字` 最近几次的记录
>
> `--all` 查看所有分支所有记录
>
> `--graph` 图形化方式

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git log --graph
* commit b29a662e2098bbf4e8a0c3c1a629d0e396efa293 (HEAD -> master)
| Author: MengLaoShi <[redacted]@qq.com>
| Date:   Mon Apr 6 02:35:04 2020 +0800
|
|     重命名
|
* commit c9b98d168df82142c2f9649cf2f33cf6fc8ff883
| Author: MengLaoShi <[redacted]@qq.com>
| Date:   Sun Apr 5 23:46:58 2020 +0800
|
|     第二次提交
|
* commit 212543a7454468cef972e0830d057eab6c552ed5
  Author: MengLaoShi <[redacted]@qq.com>
  Date:   Sun Apr 5 19:21:34 2020 +0800

      第一次提交
```

2.9 git branch

> 分支是git中非常重要的一个概念,在开发时我们可以创建多个分支，每个分支间是相互独立的，互不影响。当分支开发完毕后，可以选择合并到主干。
>
> `git branch`命令用于操作git分支:
>
> `git branch -v` 看本地有多少分支
>
> `git branch -d 分支名` 删除分支

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git branch -v
* master b29a662 重命名
```

2.10 git checkout

> 可以从主干或其它分支快速创建一个新分支或切换分支，使用`git checkout`命令，在checkout的时候我们可以指定一个版本号，这个版本号就是在`git log`中查到的hash值。版本号在使用的时候可以只使用前几位，只要当前的几位能够确定唯一标识就可以。
>
> `git checkout -b 新分支 原分支/conmmit` 创建分支
>
> `git checkout 分支` 切换分支

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git log -n1
commit b29a662e2098bbf4e8a0c3c1a629d0e396efa293 (HEAD -> master, temp)
Author: MengL  Shi <          e@qq.com>
Date:   Mon Apr  6 02:35:04 2020 +0800

    重命名
```

hash版本号

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git checkout -b temp b29a662
Switched to a new branch 'temp'
M       Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git branch -v
  master b29a662 重命名
* temp   b29a662 重命名
```

新分支

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git branch -d nb
error: The branch 'nb' is not fully merged.
If you are sure you want to delete it, run 'git branch -D nb'.

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git branch -D nb
Deleted branch nb (was 3046b53).

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

> 接下来修改文件，然后提交，使`git log`可以查看到当前分支的历史，使用`git log --all`可以看所有分支：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ vi Meng.txt
```

```
MINGW64:/d/demo/git-test
孟老师的QQ群：1063747125
QQ：574549426
============
temp branch
```

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git commit -am 'add new line to temp'
[temp f7f4cb6] add new line to temp
 1 file changed, 3 insertions(+), 1 deletion(-)

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log
commit f7f4cb629ba513cfcdf487d77812fa96d34ac77a (HEAD -> temp)
Author: MengLaoShi <[redacted]@qq.com>
Date:   Mon Apr 6 11:31:49 2020 +0800

    add new line to temp

commit b29a662e2098bbf4e8a0c3c1a629d0e396efa293 (master)
Author: MengLaoShi <[redacted]@qq.com>
Date:   Mon Apr 6 02:35:04 2020 +0800

    重命名

commit c9b98d168df82142c2f9649cf2f33cf6fc8ff883
Author: MengLaoShi <[redacted]@qq.com>
Date:   Sun Apr 5 23:46:58 2020 +0800

    第二次提交

commit 212543a7454468cef972e0830d057eab6c552ed5
Author: MengLaoShi <[redacted]@qq.com>
Date:   Sun Apr 5 19:21:34 2020 +0800

    第一次提交
```

分支在temp

2.11 分离头指针

> 当checkout一个commit的hash时，会看到如下情况：

```
$ git log -n1
commit 86ee20937a4a2d534f37610[redacted]9d09d2c44eea64fb6 (HEAD -> temp)
Author: 扣了 <574549426@qq.com>
Date:   Mon Apr 6 15:56:11 2020 +0800

    添加文件夹

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git checkout 86ee20937a4a2d534f37610
Note: checking out '86ee20937a4a2d534f37610'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 86ee209... 添加文件夹

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test ((86ee209...))
$
```

> 这个提示的意思是，当前状态是分离头指针，即没有分支与当前的编辑版本关联；我们可以选择提交或废弃当前的编辑。但如果不与分支关联，当切换分支的时候，当前编辑的版本将全部丢失。

> 分离头指针，可以用在一些尝试性的修改上，比如我想改一些功能，但是不确定最终是否可行，那在分离头指针的状态，如果当前编辑不可行，就直接切换分支，丢弃当前的编辑。
>
> 我们尝试修改并提交：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test ((86ee209...))
$ vi Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test ((86ee209...))
$ git commit -am '分离头指针修改'
[detached HEAD 3046b53] 分离头指针修改
 1 file changed, 3 insertions(+), 1 deletion(-)

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test ((3046b53...))
$
```

> 此时如果我们切换分支，会出现如下情况：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test ((3046b53...))
$ git checkout temp
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  3046b53 分离头指针修改

If you want to keep it by creating a new branch, this may be a good time
to do so with:

 git branch <new-branch-name> 3046b53

Switched to branch 'temp'

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

切换分支

> 通过git log去查询，看不到任何与这个记录相关的东西：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log
commit 86ee20937a4a2d534f376109d09d2c44eea64fb6 (HEAD -> temp)
Author: 扣丁 <574549426@qq.com>
Date:   Mon Apr 6 15:56:11 2020 +0800

    添加文件夹

commit f7f4cb629ba513cfcdf487d77812fa96d34ac77a
Author: MengLaoShi <574549426@qq.com>
Date:   Mon Apr 6 11:31:49 2020 +0800

    add new line to temp

commit b29a662e2098bbf4e8a0c3c1a629d0e396efa293 (master)
Author: MengLaoShi <mecuryhope@qq.com>
Date:   Mon Apr 6 02:35:04 2020 +0800

    重命名
```

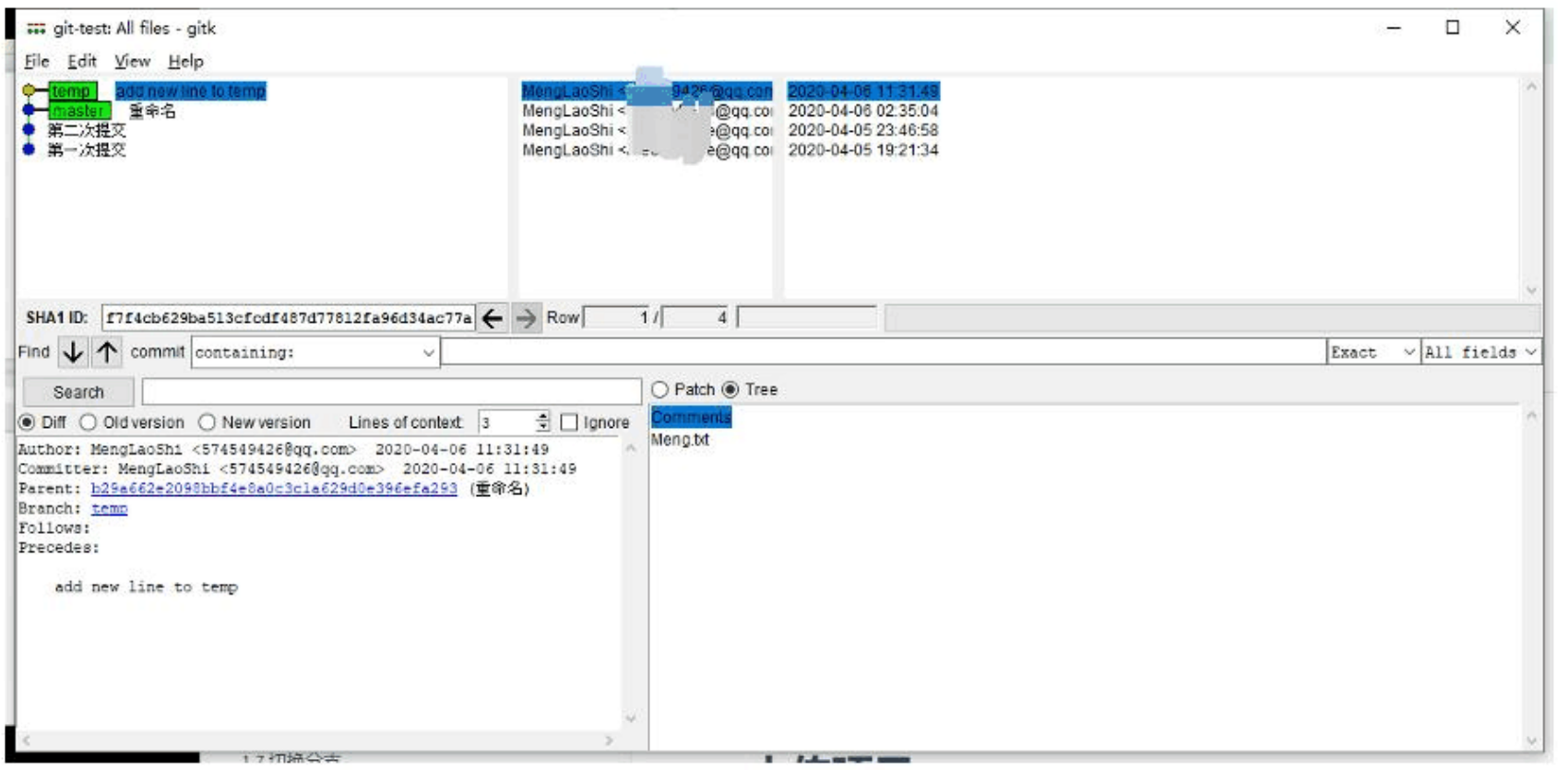
如果想保留刚才的修改，就去执行创建分支的命令：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git branch nb 3046b53
```

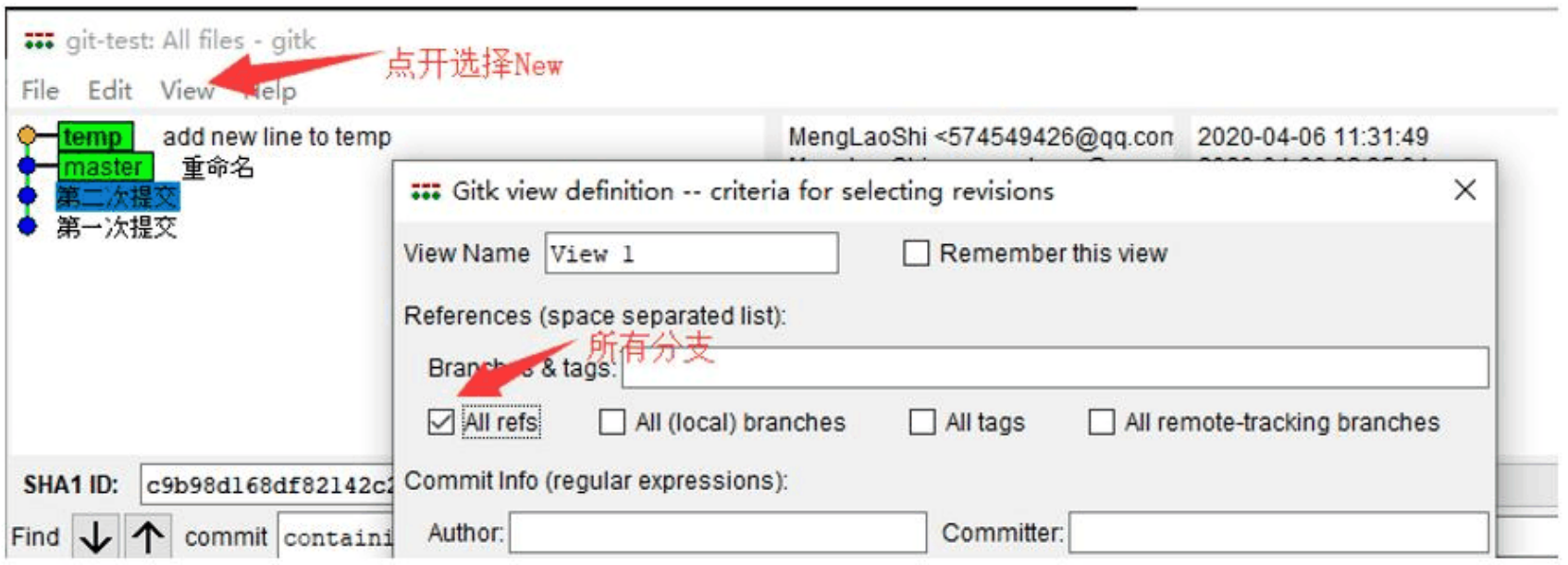
```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log --all
commit 3046b536a28c84c1fcc5531f9ef337301d1b6f0e (nb)
Author: 扣丁 <574549426@qq.com>
Date:   Mon Apr 6 16:52:14 2020 +0800

    分离头指针修改

commit 86ee20937a4a2d534f376109d09d2c44eea64fb6 (HEAD -> temp)
Author: 扣丁 <574549426@qq.com>
Date:   Mon Apr 6 15:56:11 2020 +0800

    添加文件夹
```

2.12 gitk

> `gitk`能够打开一个图形界面



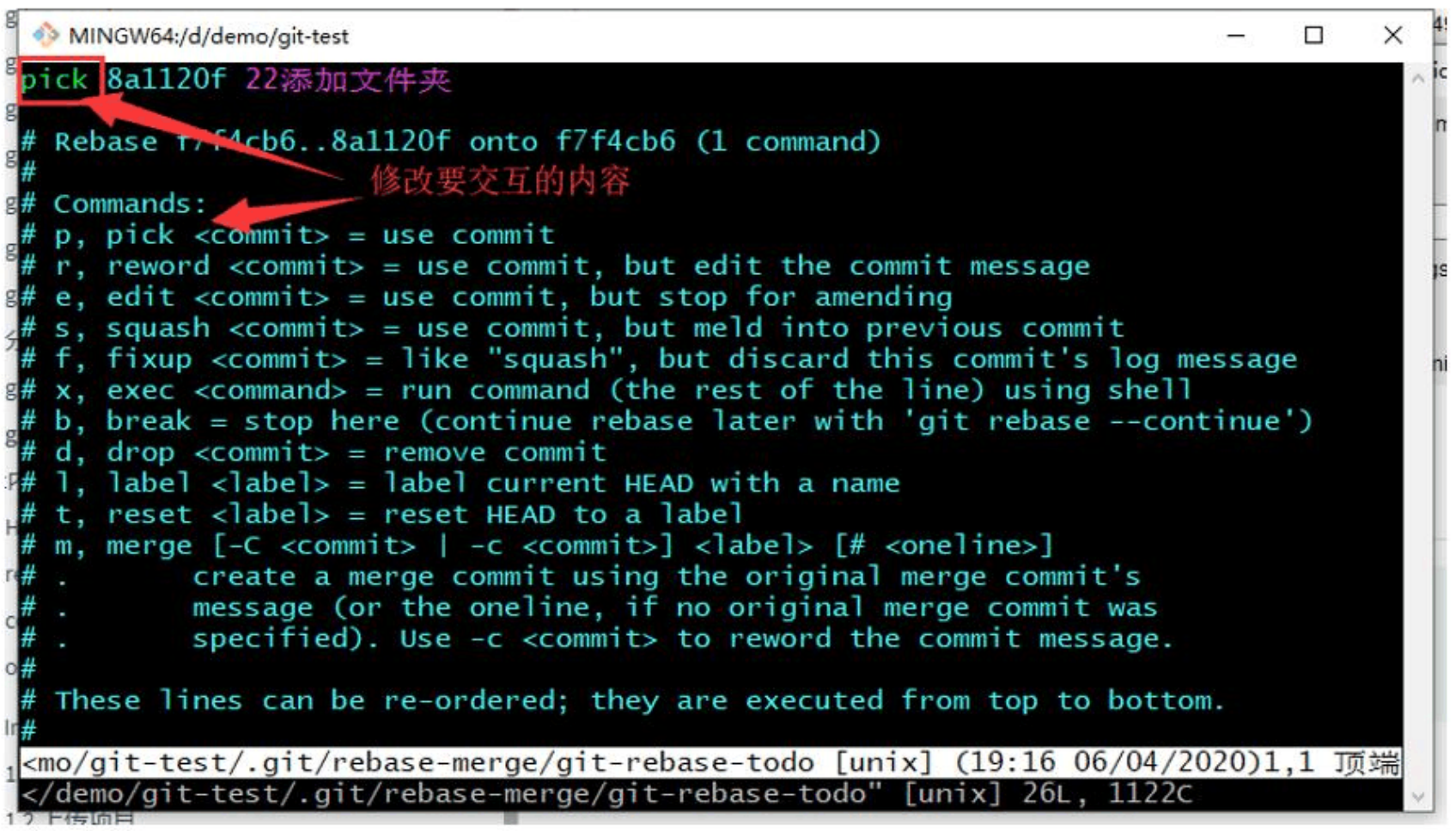
> 我们可以看到各个版本的信息，包括父子关系、提交时间。tree可以显示目录结构。在View中可以定义显示的视图



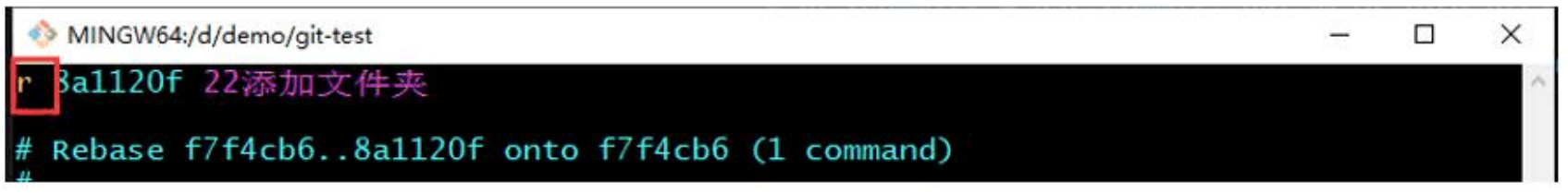
2.13 git rebase

这个操作交变基操作，可以指定修改历史版本中的提交信息

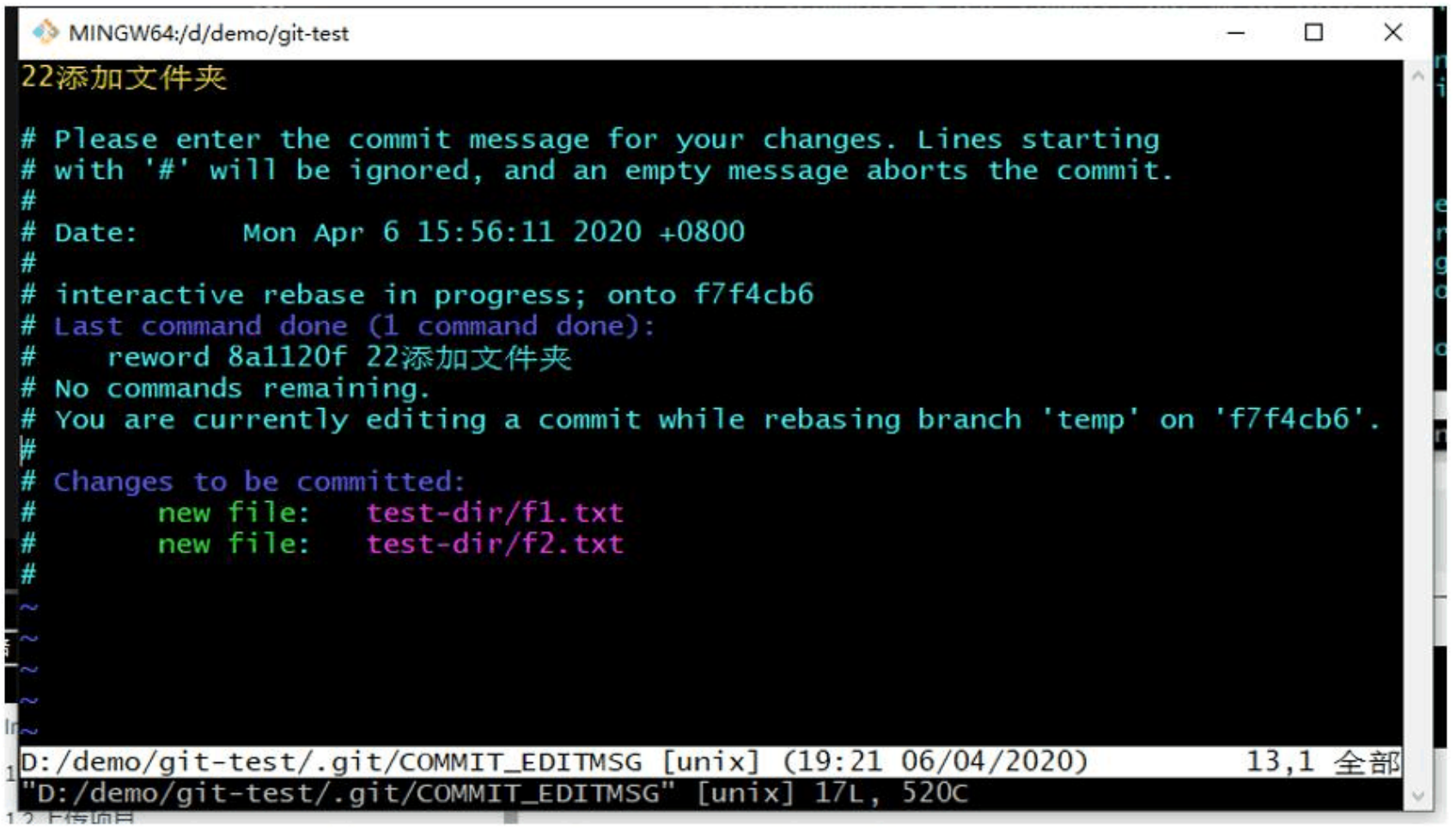
`git rebase -i hash值` (hash选的是父亲)可以打开一个交互界面：



我们只修改提交信息，所以选择 `r` 命令



保存退出后，会自动弹出一个新的交互界面，修改，并保存退出




```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git rebase -i  b29a662e2098bbf4e8a0c3c1a629d0e396efa293
[detached HEAD 2a62884] add a new line to temp
 Author: MengLaoShi <574549426@qq.com>
 Date: Mon Apr 6 11:31:49 2020 +0800
 1 file changed, 3 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/temp.
```

2.14 git rm

> 删除一个文件`git rm 文件名`删除一个文件，会删除工作区中的文件，并把变动放在暂存区
>
> 加`--cached`参数则删除暂存区中的东西，不删除工作区

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git rm MengLaoShi.txt
rm 'MengLaoShi.txt'
```

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    MengLaoShi.txt


menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git commit -m '删除文件'
[temp 0b7c41c] 删除文件
 1 file changed, 2 deletions(-)
 delete mode 100644 MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
nothing to commit, working tree clean

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

2.15 git stash

> 如果我的文件，有一部分新功能A在暂存区，但临时增加了别的任务B，需要在之前的版本上修改，而我的新功能A还没开发完，这时我们可以使用`git stash`命令来保存暂存区，此时文件会恢复到暂存区之前的样子。

MINGW64:/d/demo/git-test

```
$ git diff --cached
diff --git a/a.txt b/a.txt
index ccc3e7b..7d565d5 100644
--- a/a.txt
+++ b/a.txt
@@ -1 +1,2 @@
 aaaaa
+正在进行修改，还没提交

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git stash
Saved working directory and index state WIP on temp: ec660f7 创建两个新文件

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git stash list
stash@{0}: WIP on temp: ec660f7 创建两个新文件

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
nothing to commit, working tree clean

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

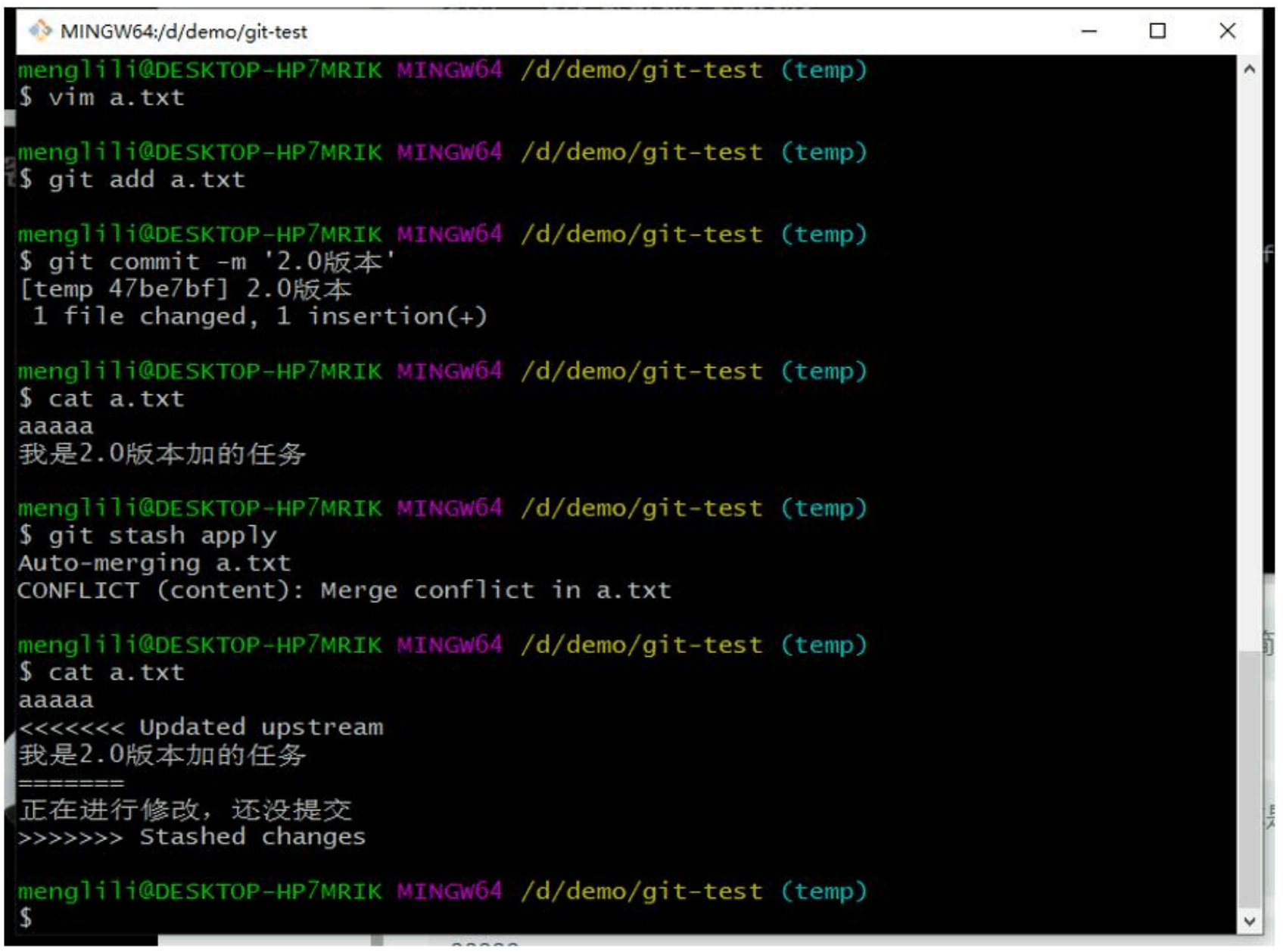
> 上述文件中，原来（版本库、工作区同步）我的a.txt中内容如下，简称版本1

```
aaaaa
```

> 我在后面追加了一行文字`正在进行修改，还没提交`，此时这个版本是在暂存区的，简称版本2，如下：

```
aaaaa
正在进行修改，还没提交
```

> 这时我接到了一个任务，要修改版本1中的bug，简称版本1.1，可是我的版本2已经开发一半了。回退到版本1就要删除掉版本2已经写的代码，这是不科学的。使用`git stash`命令，它会保留当前的暂存区，并将本地文件恢复到原来版本1的样子。做完版本1.1的任务后，使用`git stash apply`会将文件内容再恢复到版本2，继续上次未完成的编辑。



`git stash apply`和`git stash pop`命令都能将stash缓存的东西恢复，区别是apply会将缓存的信息保留在`git stash list`中，而pop则直接移出list。

3. git内部构成

> 进入到当前工作区的 `.git` 目录中：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ pwd
/d/demo/git-test

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ cd .git

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ ll
total 15
-rw-r--r-- 1 menglili 197121  21 4月    6 11:31 COMMIT_EDITMSG
-rw-r--r-- 1 menglili 197121 130 4月    5 18:16 config
-rw-r--r-- 1 menglili 197121  73 4月    5 18:16 description
-rw-r--r-- 1 menglili 197121 299 4月    6 12:39 gitk.cache
-rw-r--r-- 1 menglili 197121  23 4月    6 12:57 HEAD
drwxr-xr-x 1 menglili 197121   0 4月    5 18:16 hooks/
-rw-r--r-- 1 menglili 197121 137 4月    6 12:57 index
drwxr-xr-x 1 menglili 197121   0 4月    5 18:16 info/
drwxr-xr-x 1 menglili 197121   0 4月    5 19:21 logs/
drwxr-xr-x 1 menglili 197121   0 4月    6 11:31 objects/
-rw-r--r-- 1 menglili 197121  41 4月    6 00:15 ORIG_HEAD
drwxr-xr-x 1 menglili 197121   0 4月    5 18:16 refs/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
```

3.1 HEAD

> HEAD文件中存放的是当前目录工作的分支

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ cat HEAD
ref: refs/heads/master
```

> 当切换分支后，HEAD的内容会发生改变：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ cat HEAD
ref: refs/heads/master

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ cd .. 记得先切换目录

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git checkout temp
Switched to branch 'temp'

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ cd .git/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ cat HEAD
ref: refs/heads/temp

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
```

> 查看HEAD和父之间的差异，用`git diff HEAD HEAD^`
>
> `^`代表parent，`^^`就是父亲的父亲，等同于`~2`

3.2 refs

> refs是引用，里面包含了分支和标签等一些信息

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ ll .git/refs
total 0
drwxr-xr-x 1 menglili 197121 0 4月    6 11:31 heads/
drwxr-xr-x 1 menglili 197121 0 4月    5 18:16 tags/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ ll .git/refs/heads/
total 2                                    分支信息
-rw-r--r-- 1 menglili 197121 41 4月    6 02:35 master
-rw-r--r-- 1 menglili 197121 41 4月    6 11:31 temp

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

> 继续查看master文件，可以看见文件中保存的内容是和hash版本号相关的：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ ll .git/refs/heads/
total 2
-rw-r--r-- 1 menglili 197121 41 4月    6 02:35 master
-rw-r--r-- 1 menglili 197121 41 4月    6 11:31 temp

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ cat .git/refs/heads/master                  显示文件内容
b29a662e2098bbf4e8a0c3c1a629d0e396efa293

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git cat-file -t b29a662e2098bbf4
commit                                  查看文件类型

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git branch -v
  master b29a662 重命名
* temp   f7f4cb6 add new line to temp

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

3.3 config

> config中记录的是当前目录的配置，通过`git config --local`设置的就是这个文件

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git (GIT_DIR!)
$ cd ..

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git config --local user.name "扣丁"

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        symlinks = false
        ignorecase = true
[user]
        name = 扣丁

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

3.4 objects

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ cd .git/objects/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git/objects (GIT_DIR!)
$ ll
total 0
drwxr-xr-x 1 menglili 197121 0 4月    6 00:13 0d/
drwxr-xr-x 1 menglili 197121 0 4月    5 19:21 21/
drwxr-xr-x 1 menglili 197121 0 4月    6 11:31 43/
drwxr-xr-x 1 menglili 197121 0 4月    5 23:46 52/
drwxr-xr-x 1 menglili 197121 0 4月    5 19:37 92/
drwxr-xr-x 1 menglili 197121 0 4月    5 23:46 a5/
drwxr-xr-x 1 menglili 197121 0 4月    6 11:31 b2/
drwxr-xr-x 1 menglili 197121 0 4月    6 02:35 b6/
drwxr-xr-x 1 menglili 197121 0 4月    5 23:46 c9/
drwxr-xr-x 1 menglili 197121 0 4月    5 19:08 e6/
drwxr-xr-x 1 menglili 197121 0 4月    5 19:21 eb/
drwxr-xr-x 1 menglili 197121 0 4月    6 11:31 f7/
drwxr-xr-x 1 menglili 197121 0 4月    5 18:16 info/
drwxr-xr-x 1 menglili 197121 0 4月    5 18:16 pack/

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git/objects (GIT_DIR!)
```

文件目录

> 可以随便打开一个目录，查看文件类型和文件内容(不同的人测试结果可能不同，多切换几个文件夹就能找到文件了)：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git/objects (GIT_DIR!)
$ ll 0d
total 1
-r--r--r-- 1 menglili 197121 65 4月    6 00:13 9118b36d94fcec68577333a1f637b7923b6b6c

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git/objects (GIT_DIR!)
$ git cat-file -t 0d9118b36d94fcec68577
blob

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test/.git/objects (GIT_DIR!)
$ git cat-file -p 0d9118b36d94fcec68577
孟老师的QQ群：1063747125
QQ：574549426
============
```

-t查看类型

-p查看内容

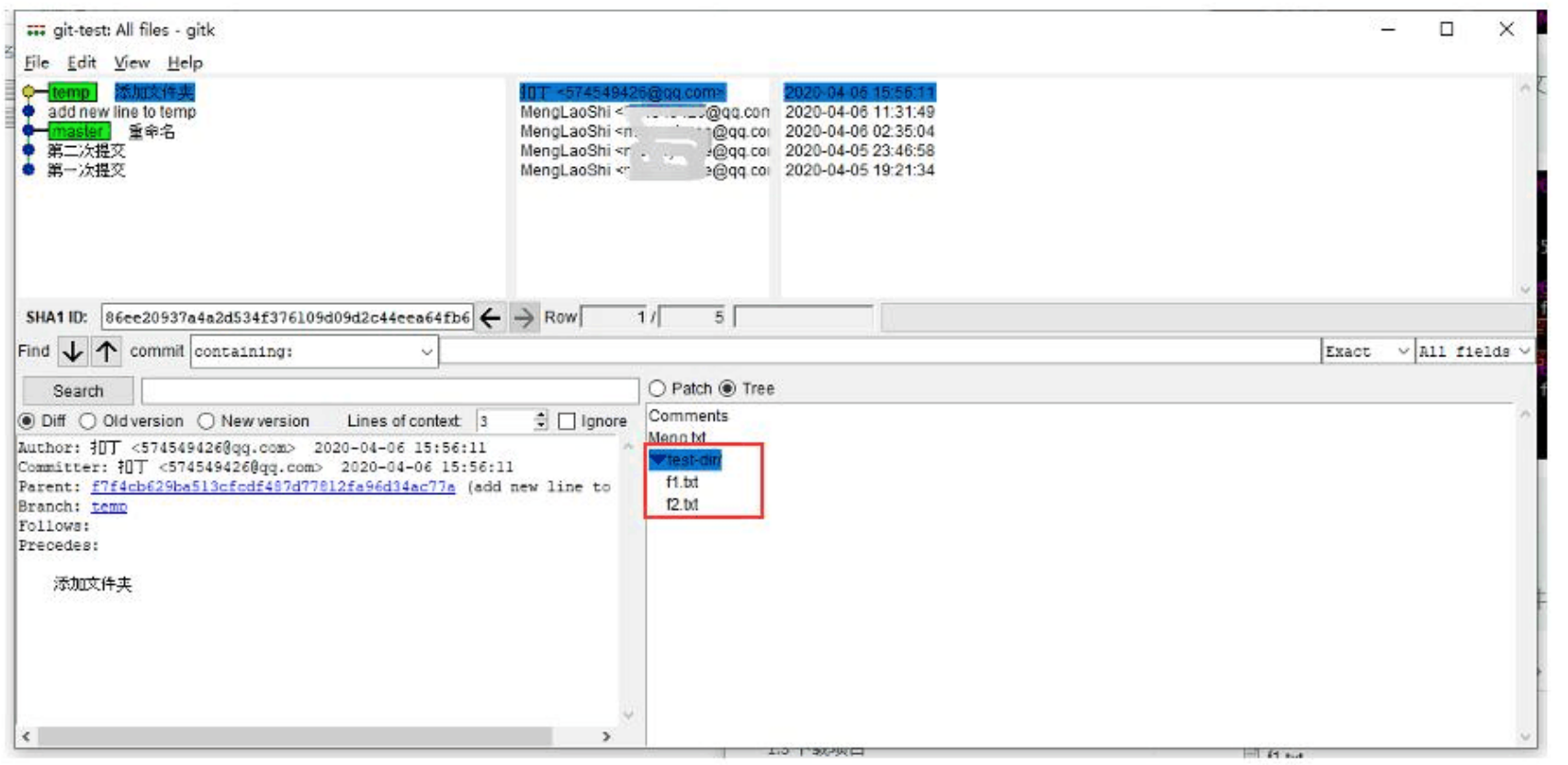
文件名规则是 文件夹名+hash

> git中核心的对象主要有 `commit`， `tree` 和 `blob` 三种类型。git中认为两个文件中如果内容相同，就是唯一的一个blob（和内容相关，和文件名无关，节约存储空间）。
>
> 在工作区中添加一个test-dir文件夹，并创建两个文件，提交到git中：

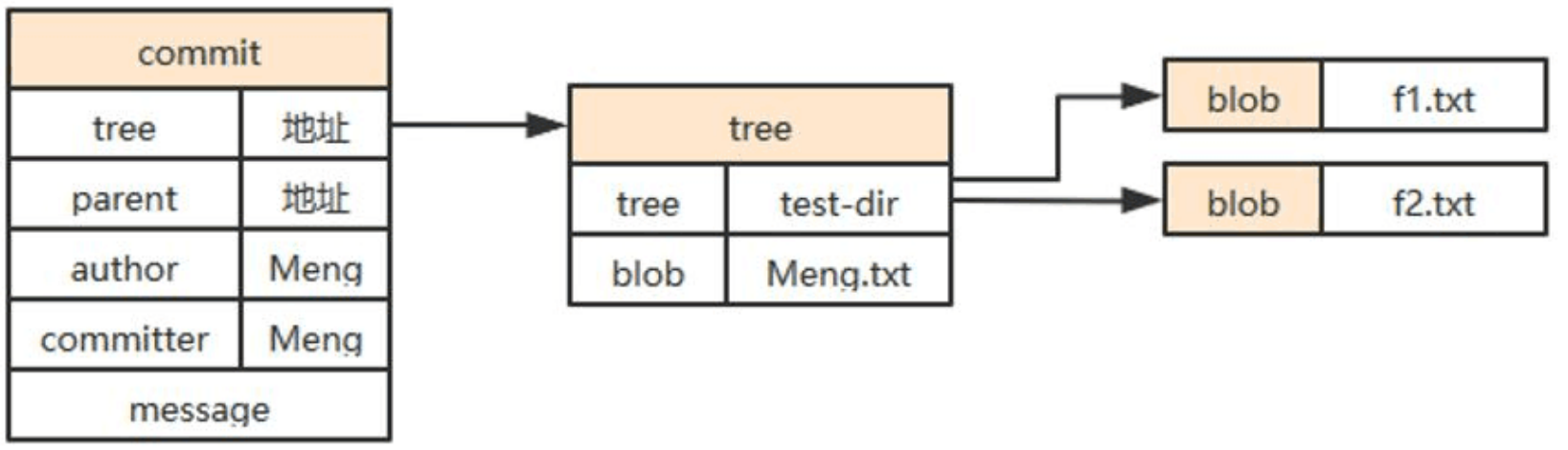
此电脑 › 新加卷 (D:) › demo › git-test › test-dir

名称	修改日期	类型	大小
f1.txt	2020/4/6 15:55	文本文档	0 KB
f2.txt	2020/4/6 15:55	文本文档	0 KB

> 在gitk中查看，可以看见目录结构



> 在git中每一个commit对应一个tree，它就是工作区那个文件夹
>
> 工作区tree下面会有其它的tree和blob，如下图(为方便理解，写了文件名，实际上只跟地址相关，和文件名无关)：




```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log -n1
commit 86ee20937a4a2d534f376109d09d2c44eea64fb6 (HEAD -> temp)
Author: 扣丁 <574549426@qq.com>
Date:   Mon Apr 6 15:56:11 2020 +0800

    添加文件夹

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git cat-file -t 86ee20937a4a2d534f376109d09d2c44eea64fb6
commit

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git cat-file -p 86ee20937a4a2d534f376109d09d2c44eea64fb6
tree 9c7ca9313e09433bb8035688ef7d0a08fd204ee7
parent f7f4cb629ba513cfcdf487d77812fa96d34ac77a
author 扣丁 <574549426@qq.com> 1586159771 +0800
committer 扣丁 <574549426@qq.com> 1586159771 +0800

添加文件夹

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git cat-file -p  9c7ca9313e09433bb8035688ef7d0a08fd204ee7
100644 blob 43598215ffdfdfc9e6e940b4307d5496ae9c1aea    Meng.txt
040000 tree 932a61f7815a7b06ef9a1ec9d79feb35eb5cc9eb    test-dir

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git cat-file -p  932a61f7815a7b06ef9a1ec9d79feb35eb5cc9eb
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    f1.txt
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    f2.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

4. 版本回退

4.1 暂存区恢复和HEAD一样

> 修改文件，并提交到暂存区：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ vi Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git add Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Meng.txt
```

> 使用`git reset HEAD -- 文件名`撤销暂存区中的变动,其中文件名是可选参数，不写就是撤销所有，(它不会撤销你文件中的修改,即不改变工作区，撤销的是add)

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git reset HEAD
Unstaged changes after reset:
M       Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
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:   Meng.txt

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

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git diff --cached

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

没有变动

4.2 工作区恢复成暂存区

> 修改本地文件，然后想恢复到暂存区中的版本，使用··

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ vim Meng.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git status
On branch temp
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:   Meng.txt

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

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git checkout -- Meng.txt
```

会覆盖本地文件

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)

4.3 回退到历史的commit(慎用)

> 使用 `git reset --hard 版本` 可以直接回退到历史版本的节点

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log
commit cb7cea9f8ef03564115f27ea5d220c389f2f297a (HEAD -> temp)
Author: 扣丁 <574549426@qq.com>
Date:   Mon Apr 6 23:34:12 2020 +0800

    提交

commit b8412cdda233c05bef1f0a659a7390e057e8d48a
Author: MengLaoShi <574549426@qq.com>
Date:   Mon Apr 6 11:31:49 2020 +0800

    我进行了合并
    add a new line to temp

    添加文件夹==

commit b29a662e2098bbf4e8a0c3c1a629d0e396efa293 (master)
Author: MengLaoShi <[illegible]@qq.com>
Date:   Mon Apr 6 02:35:04 2020 +0800

    重命名

commit c9b98d168df82142c2f9649cf2f33cf6fc8ff883
Author: MengLaoShi <[illegible]e@qq.com>
Date:   Sun Apr 5 23:46:58 2020 +0800

    第二次提交

commit 212543a7454468cef972e0830d057eab6c552ed5
Author: MengLaoShi <[illegible]e@qq.com>

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git reset --hard c9b98d168df82142c2f9649cf2f33cf6fc8ff883
HEAD is now at c9b98d1 第二次提交

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ ls
MengLaoShi.txt
```

这是重命名之前的版本

重命名被撤销了

并且回退的这个版本，之后的提交都会被永久删除了：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git reset --hard c9b98d168df82142c2f9649cf2f33cf6fc8ff883
HEAD is now at c9b98d1 第二次提交

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ ls
MengLaoShi.txt

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git log
commit c9b98d168df82142c2f9649cf2f33cf6fc8ff883 (HEAD -> temp)
Author: MengLaoShi <[redacted]e@qq.com>
Date:   Sun Apr 5 23:46:58 2020 +0800

    第二次提交

commit 212543a7454468cef972e0830d057eab6c552ed5
Author: MengLaoShi <[redacted]@qq.com>
Date:   Sun Apr 5 19:21:34 2020 +0800

    第一次提交

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$
```

5. 忽略文件

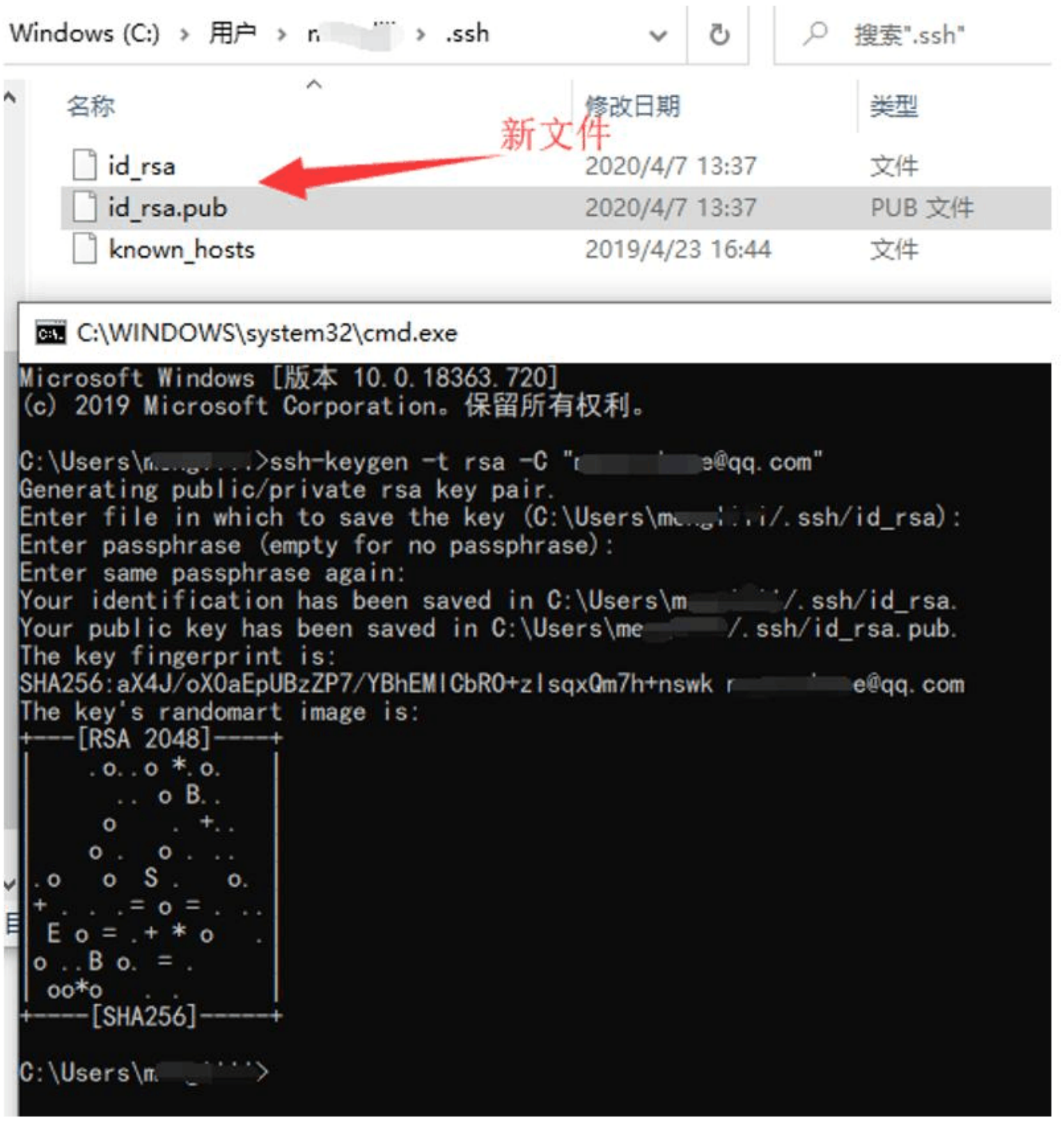
> 当工作区下有一些文件不需要git管理的时候，在工作区下添加一个 `.gitignore` 文件，必须叫这个名字。文件中的语法如下：
>
> 匹配文件夹，用 dir_name `/` 比如忽略docs文件夹，就写 `docs/`
>
> 匹配文件名，用 fileName，比如忽略f.txt，就写 `f.txt`
>
> 支持通配符，比如所有的jar包都忽略，就写 `*.jar`
>
> 需要注意的是，已经添加到版本库的文件，后加入到.gitignore，此时是无效的。必须删除版本库中该文件的信息才可以。

6. 同步到远程库

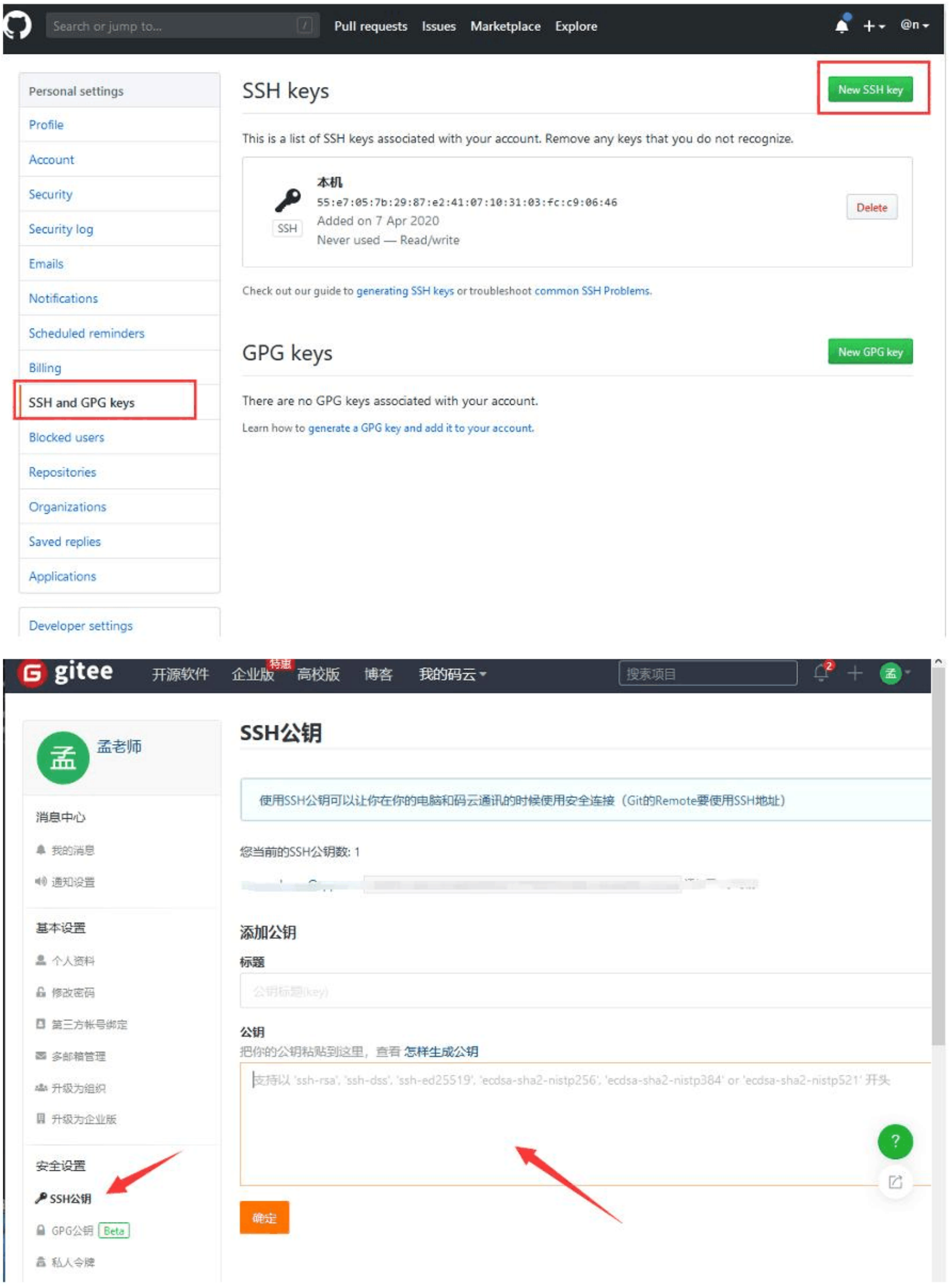
6.1 公私钥配置

> 打开cmd窗口，执行如下命令(邮箱换成自己注册的邮箱)，其它不用填，连续三次回车，之后在系统的用户目录下会有一个.ssh的文件夹：

```
ssh-keygen -t rsa -C "xxxxx@xxxxx.com"
```

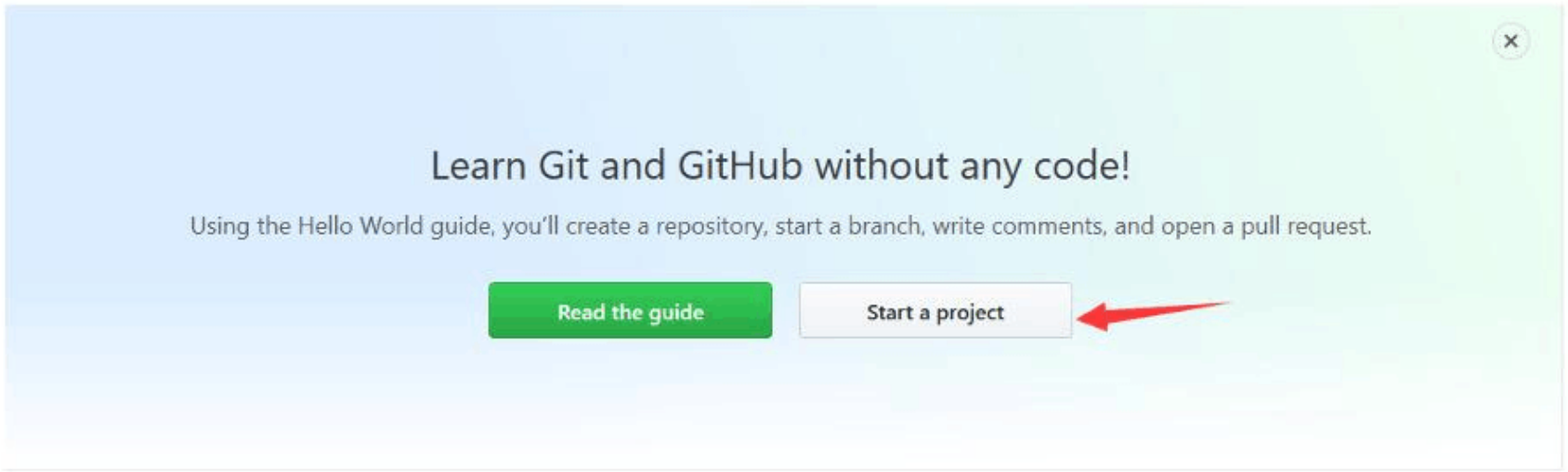


> 把id_rsa.pub中的内容复制到GitHub或者码云里(选一个你常用的就行):



6.2 创建远程仓库

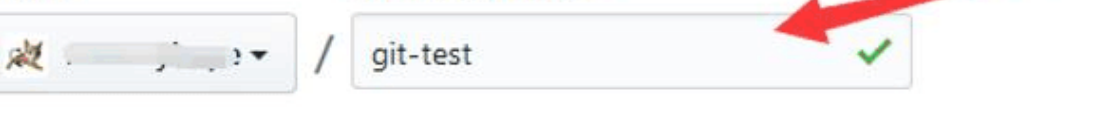
> 因为码云是中文版的，比较容易操作，后面以GitHub为例讲解。



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Owner / **Repository name** *



Great repository names are short and memorable. Need inspiration? How about **crispy-octo-robot**?

Description (optional)

我的第一个git项目

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

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

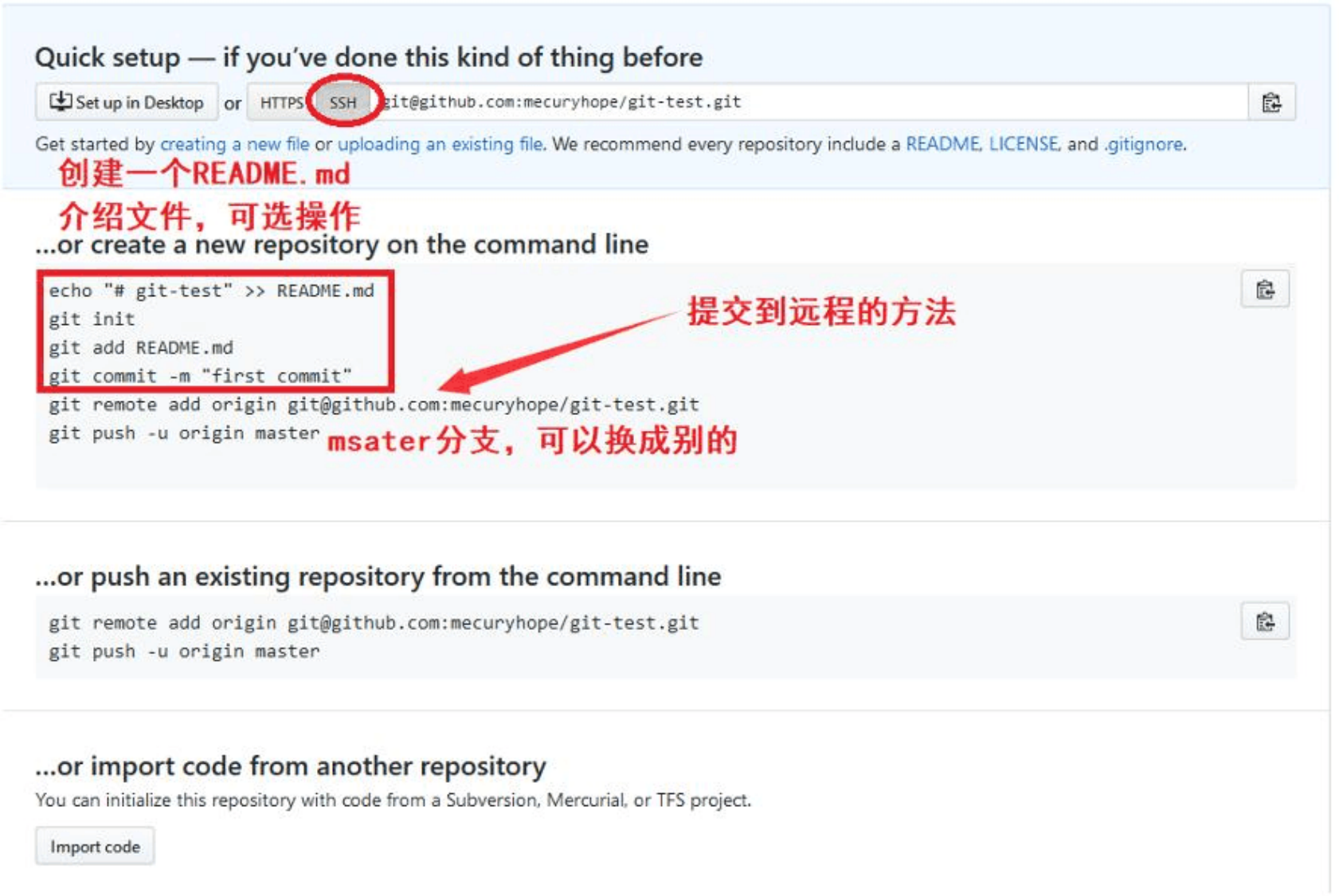
Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None**

Create repository

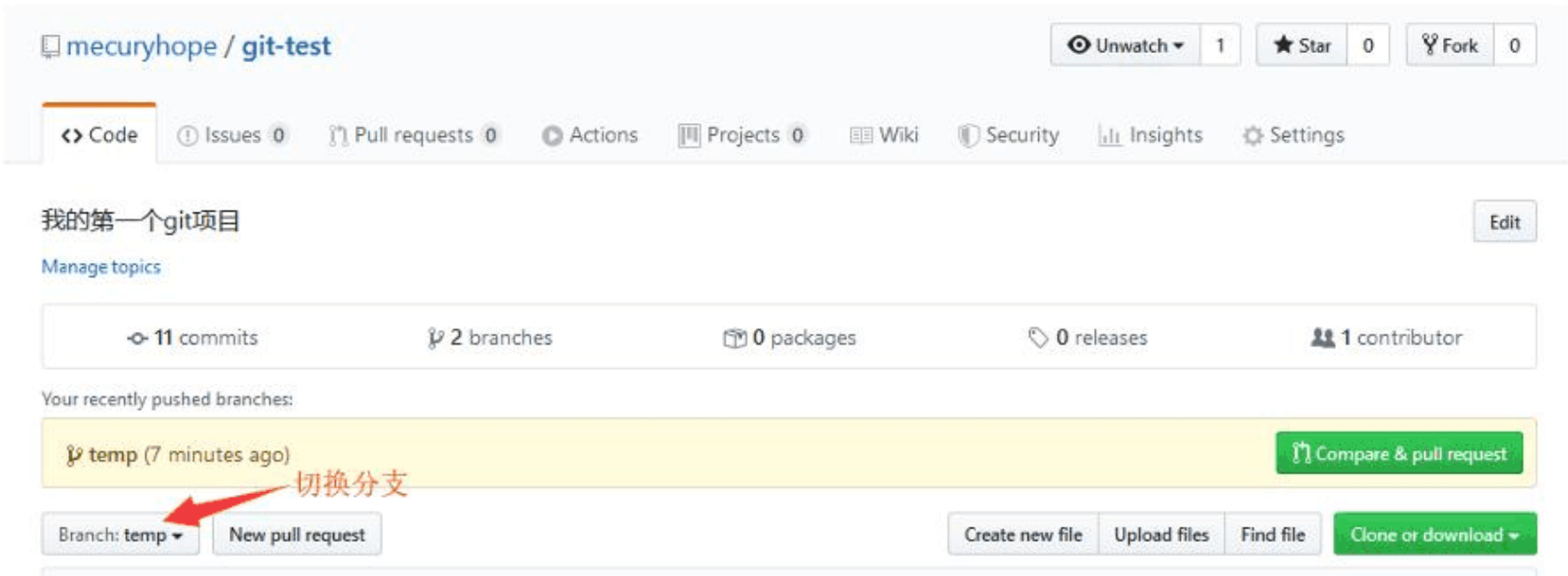
创建项目成功之后会出现如下界面，按照命令提示在本地git中执行，可以将本地库同步到GitHb中



> 其中 `git remote add 名字 地址` 是连接远程库的命令
>
> `git push -u 名字 分支` 是推送本地代码到远程仓库

6.3 push同步到远程库

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git remote add origin git@github.com:mecuryhope/git-test.git

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git push -u origin master
The authenticity of host 'github.com (13.229.188.59)' 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,13.229.188.59' (RSA) to the list of known
 hosts.
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (8/8), 724 bytes | 362.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mecuryhope/git-test.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
```



通过`git remote -v`可以查看远程信息：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (temp)
$ git remote -v
origin  git@github.com:mecuryhope/git-test.git (fetch)
origin  git@github.com:mecuryhope/git-test.git (push)
```

6.4 删除本地库和远程库的关联

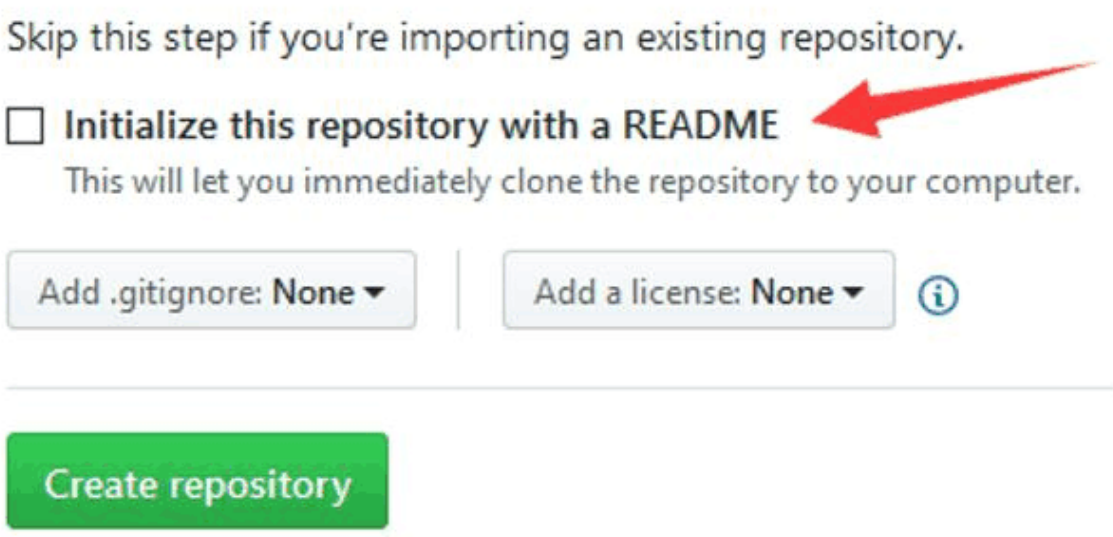
> 使用`git remote remove 名字`

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git remote remove origin

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git remote -v

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

7. 解决冲突

7.1 远程和本地冲突

> fetch命令会把远端库同步到本地，但是不会和本地分支关联。
>
> 如果远程库不是空的，或者远程库中有本地没有的东西，比如很多初学者在在创建工程的时候选了README



> 此时同步会出现问题，本地不能像远端提交数据：

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git remote add origin git@github.com:mecuryhope/git-test.git

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git push -u origin master
Warning: Permanently added the RSA host key for IP address '13.250.177.223' to t
he list of known hosts.
To github.com:mecuryhope/git-test.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:mecuryhope/git-test.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
```

> 通过fetch同步下来，会产生新分支

```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git fetch origin master
Warning: Permanently added the RSA host key for IP address '52.74.223.119' to th
e list of known hosts.
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 675 bytes | 61.00 KiB/s, done.
From github.com:mecuryhope/git-test
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$
```

> 接着用merge命令，把分支合并进来，merge的时候要注意解决冲突

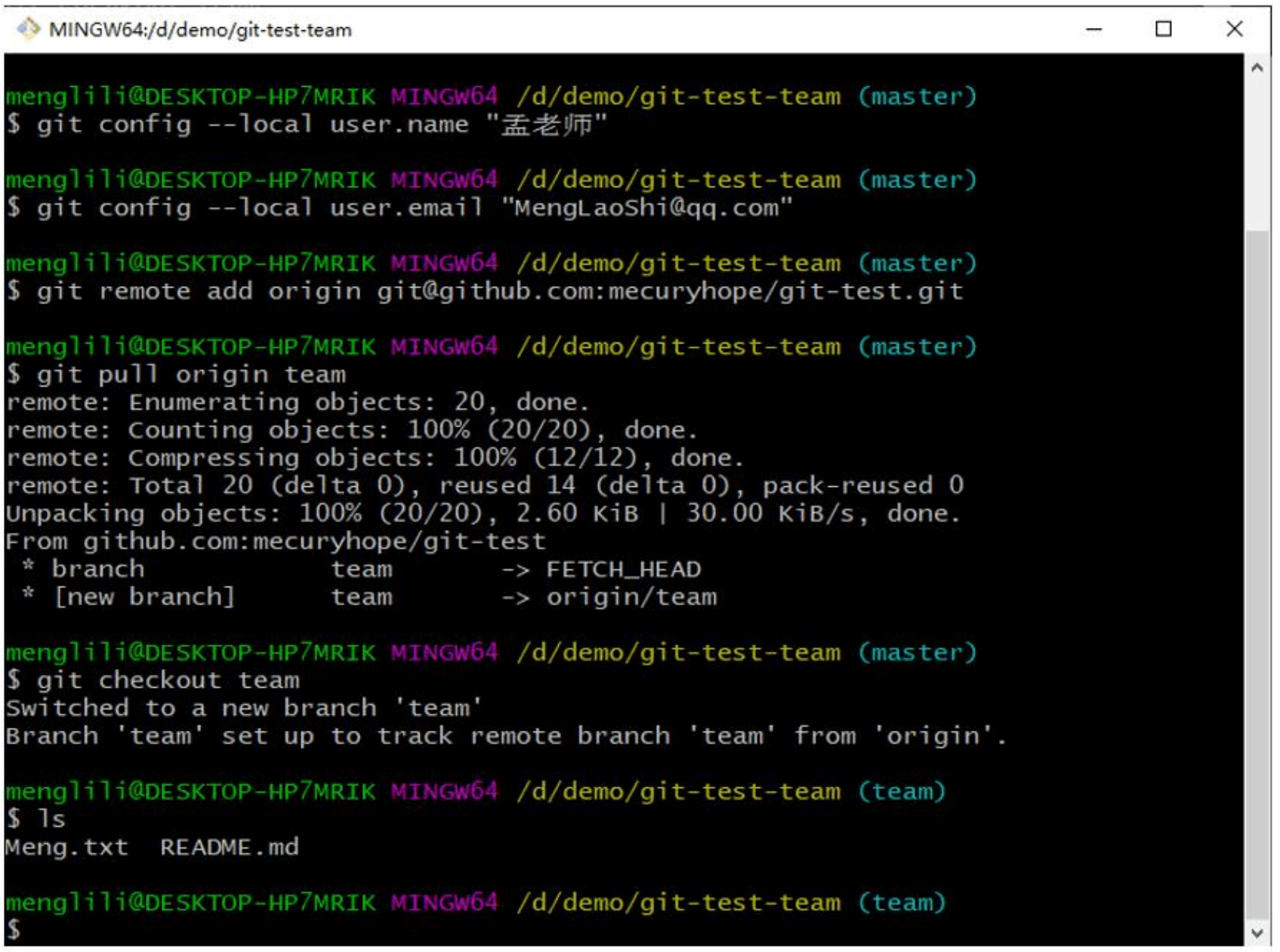
```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git merge origin/master
Updating b29a662..44fb8a5
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

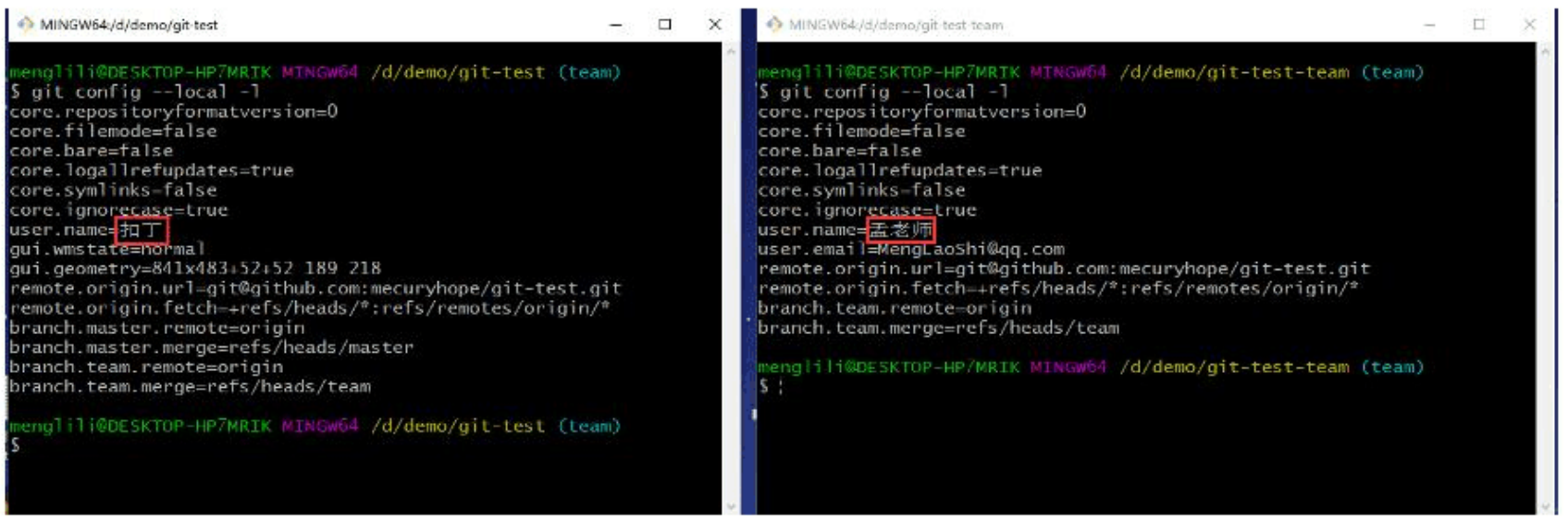
```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (master)
$ git merge origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

> 还有一个命令是`git pull`，相当于fetch+merge的命令。pull和fetch的区别是，fetch不会直接进行分支合并，需要merge。而pull直接合并，可能会产生冲突，需要手动解决

7.2 多人开发解决冲突

> 正常情况需要至少两个人才能测试，为了方便理解，我们在本地复制两份工作区，通过设置local的user.name区分不同开发者。我们重新创建一个分支team来演示这个功能：

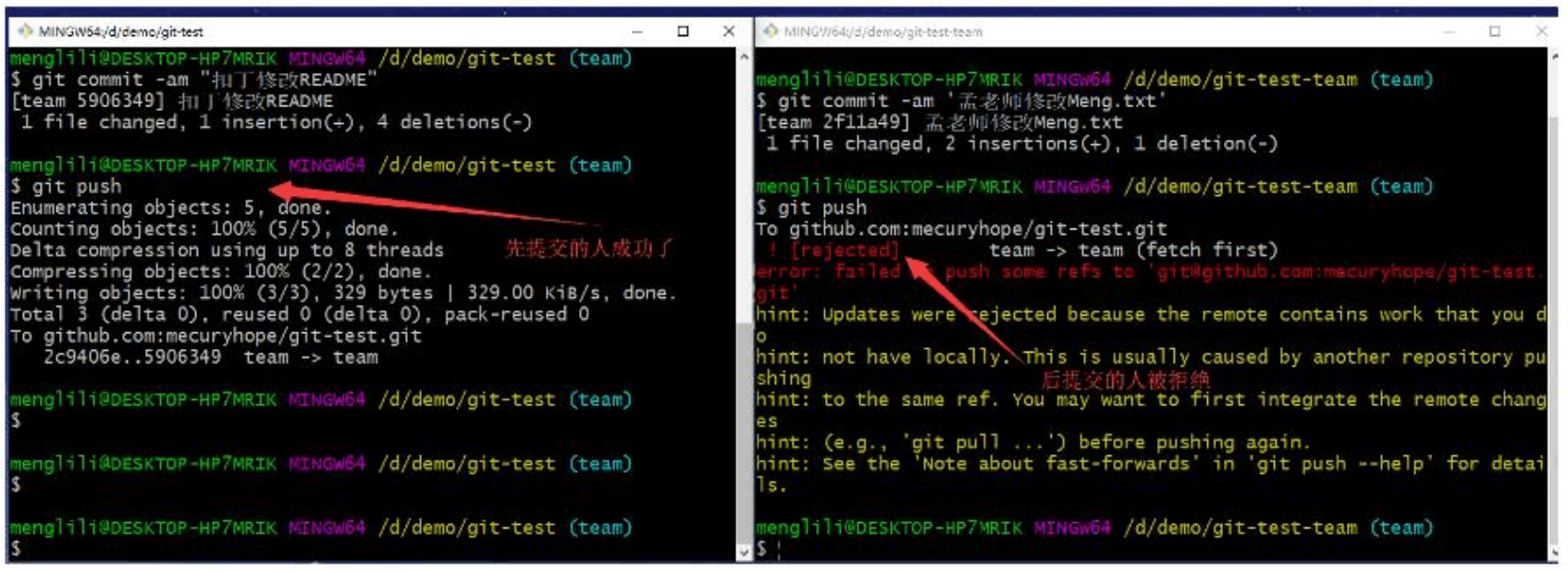




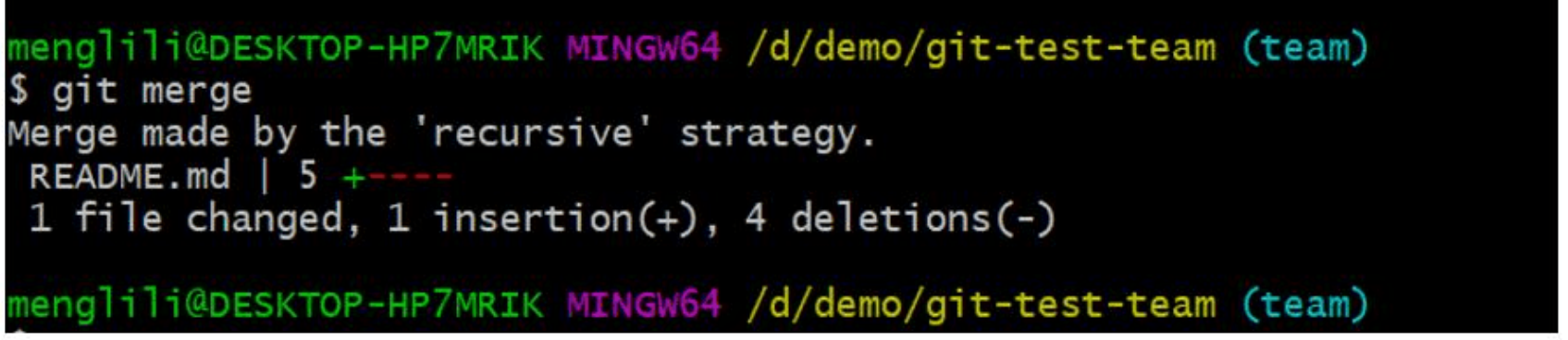
7.2.1 修改不同文件

> 当团队中的人修改不同文件时，会出现如下情况：
>
> 用户扣丁修改README.md
>
> 用户孟老师修改Meng.txt
>
> 只要其中一个人提交到远程，另一个人肯定是提交失败的：

> 让扣丁先提交，孟老师后提交，后提交的人是被拒绝的状态：



在孟老师端执行git fetch命令，再merge（输入提交信息，或默认）：



> 之后孟老师再提交，就可以成功了：

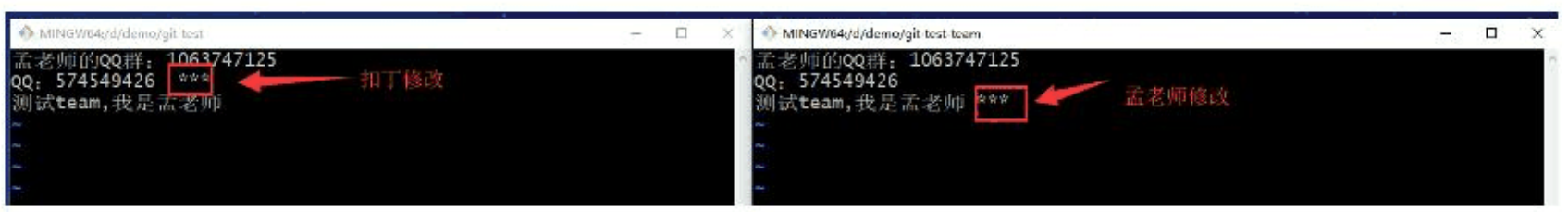
```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 678 bytes | 169.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mecuryhope/git-test.git
   5906349..94c81f3  team -> team

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team)
```

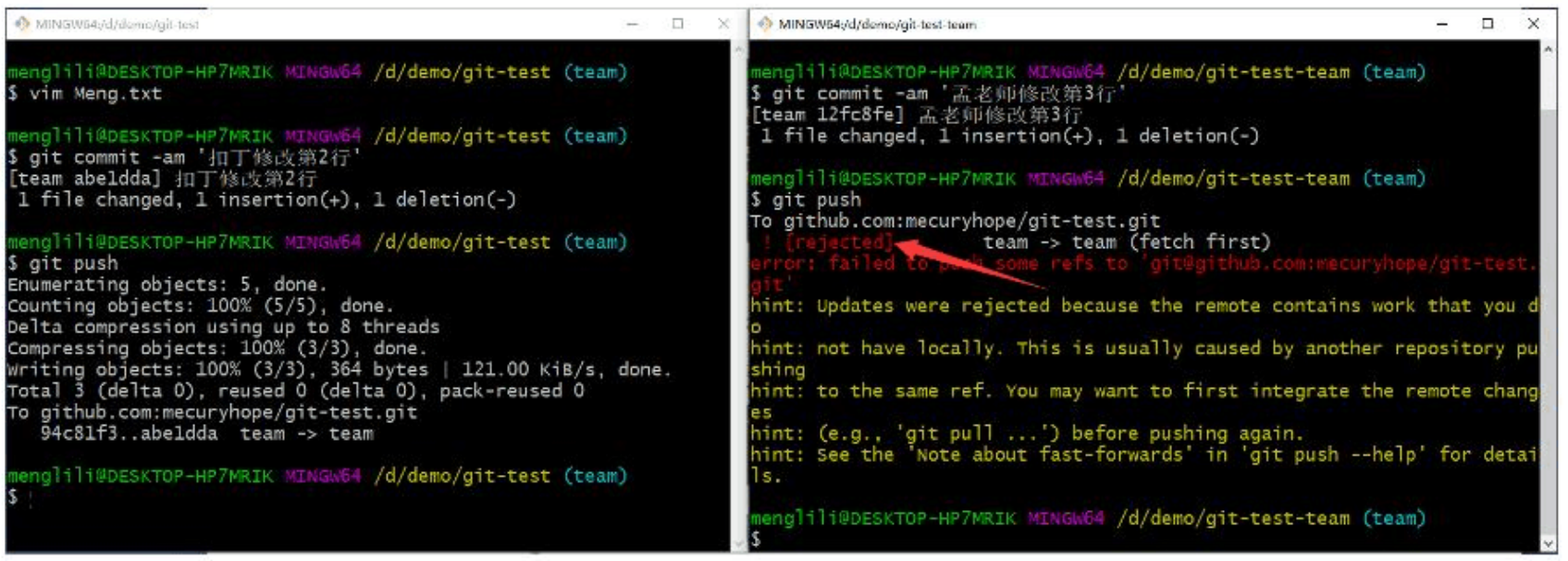
> 因为两个人修改的是不同的文件，所以同步的时候不会出现冲突，用户扣丁直接pull文件即可：

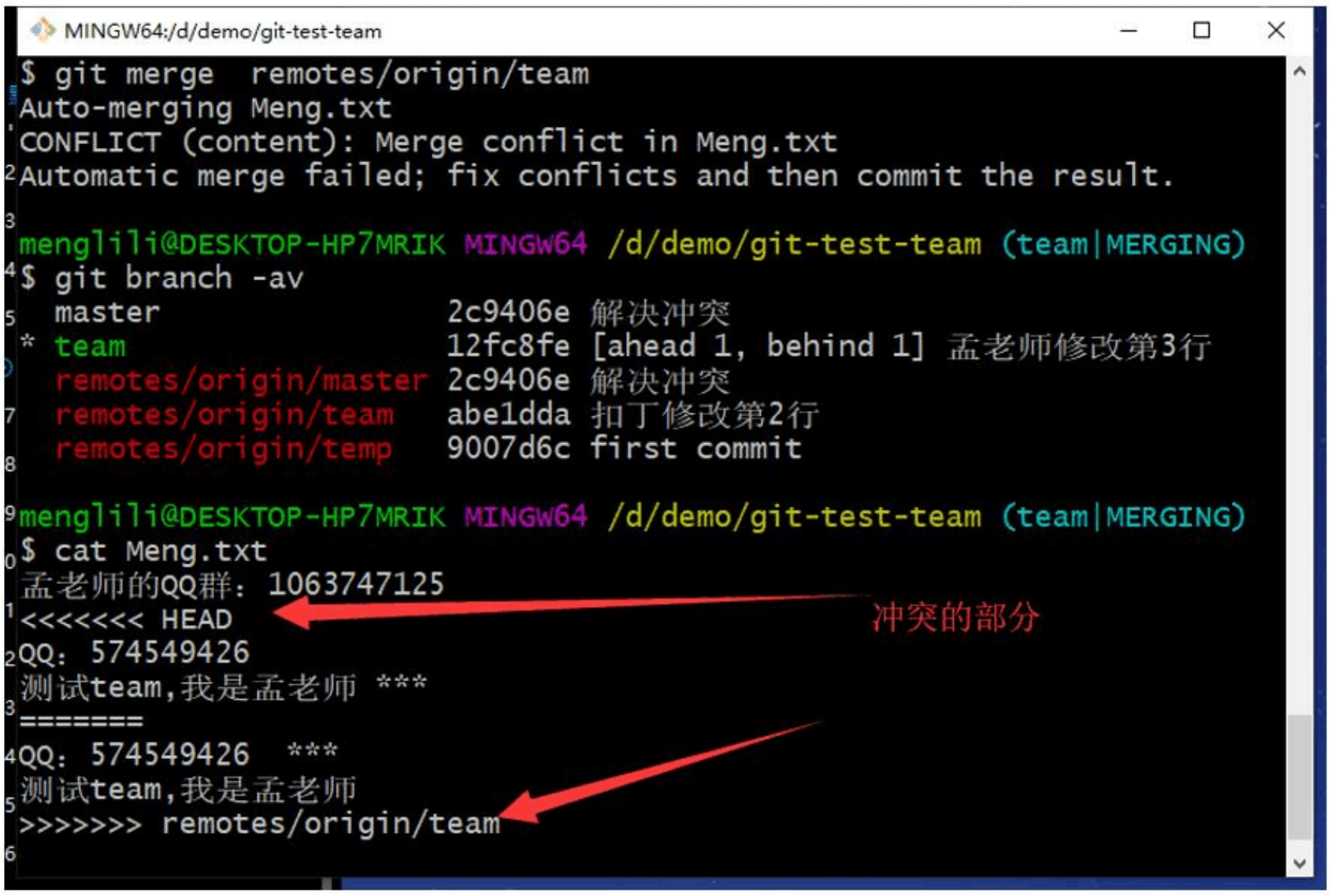
```
menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (team)
$ git pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), 658 bytes | 25.00 KiB/s, done.
From github.com:mecuryhope/git-test
   5906349..94c81f3  team       -> origin/team
 * [new branch]      temp       -> origin/temp
Updating 5906349..94c81f3
Fast-forward
 Meng.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (team)
$ cat Meng.txt
孟老师的QQ群：1063747125
QQ：574549426
测试team,我是孟老师

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test (team)
$
```

7.2.2 修改相同文件

> 修改相同文件，有两种情况，一种是修改的区域相隔较远，一种是修改的区域相同或邻近，比如下面的情况就是邻近的情况

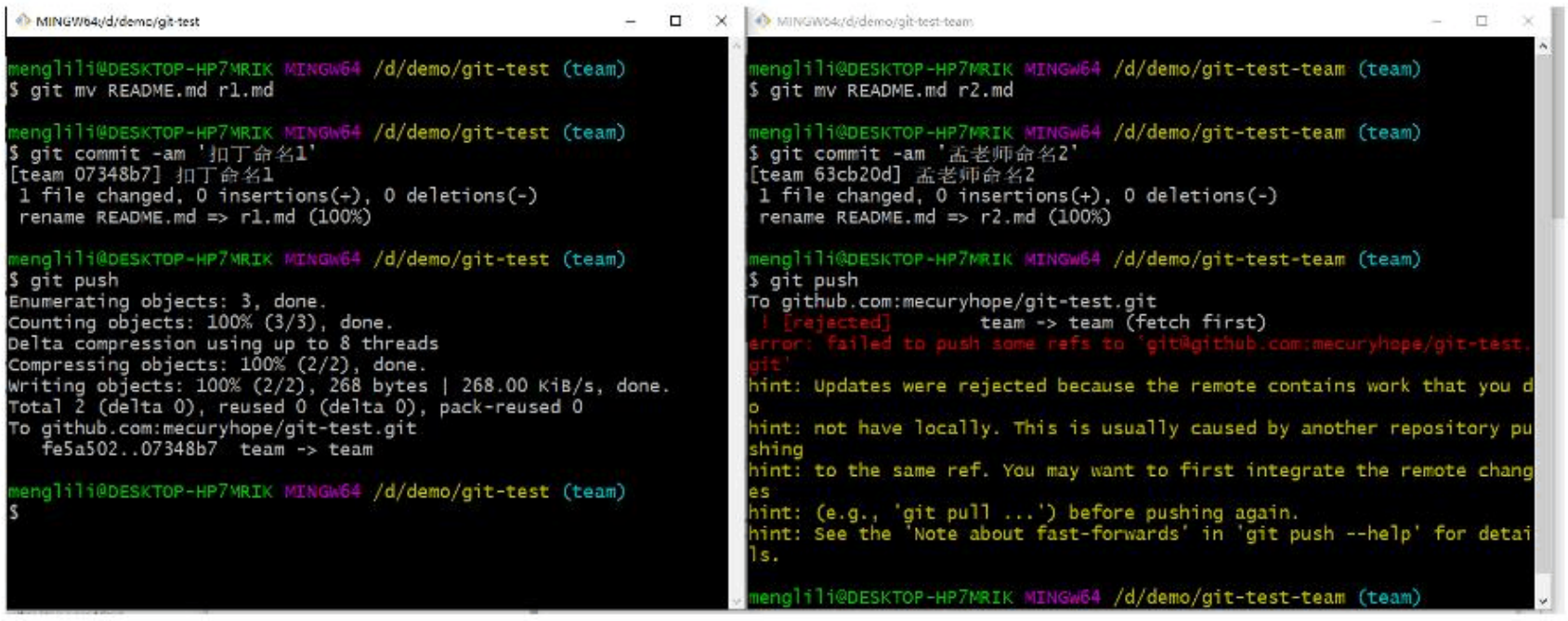




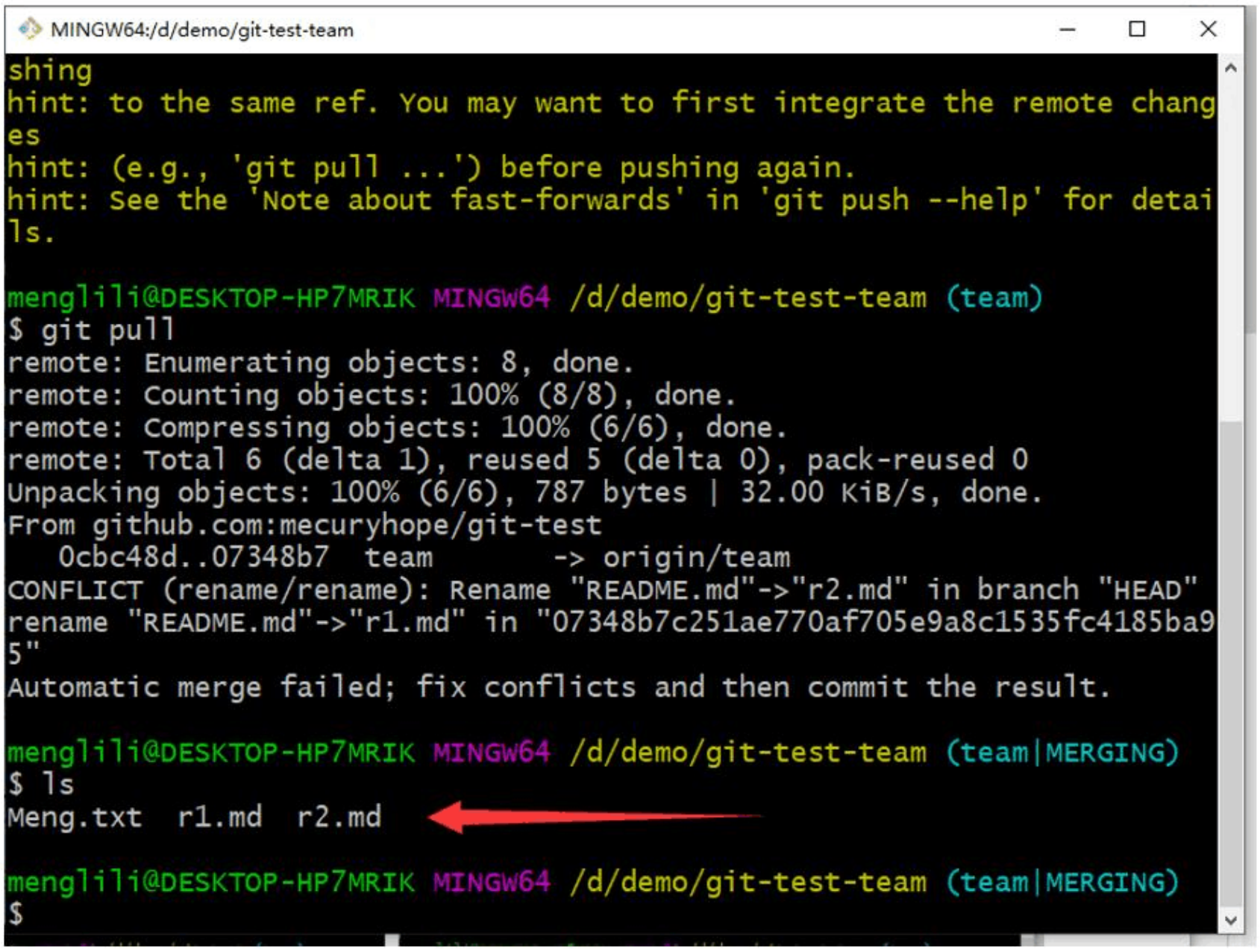


> 此时就需要手动编辑，人为决定保留哪部分，如果取消merge,可以使用`git merge --abort`
>
> 如果修改的区域相隔较远，一般merge的时候就会直接合并，不会出现冲突。
>
> merge之后，确认冲突已解决，就可以继续提交了。

7.2.3 同时修改文件名



此时 冲突的一方去pull,会的到对方修改后的文件：



```
MINGW64:/d/demo/git-test-team
Automatic merge failed; fix conflicts and then commit the result.

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ ls
Meng.txt  r1.md  r2.md

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git status
On branch team
Your branch and 'origin/team' have diverged,
and have 1 and 4 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add/rm <file>..." as appropriate to mark resolution)
        both deleted:    README.md
        added by them:   r1.md
        added by us:     r2.md

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

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$
```

> 需要做的是rm不需要的文件，add需要的文件，最后commit

```
MINGW64:/d/demo/git-test-team
no changes added to commit (use "git add" and/or "git commit -a")

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git rm r1.md
rm 'r1.md'

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git add r2.md

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git commit -m '解决文件名冲突'
error: Committing is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
U       README.md

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git rm README.md
rm 'README.md'

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team|MERGING)
$ git commit -m '解决文件名冲突'
[team 6e838e4] 解决文件名冲突

menglili@DESKTOP-HP7MRIK MINGW64 /d/demo/git-test-team (team)
$ git push
```

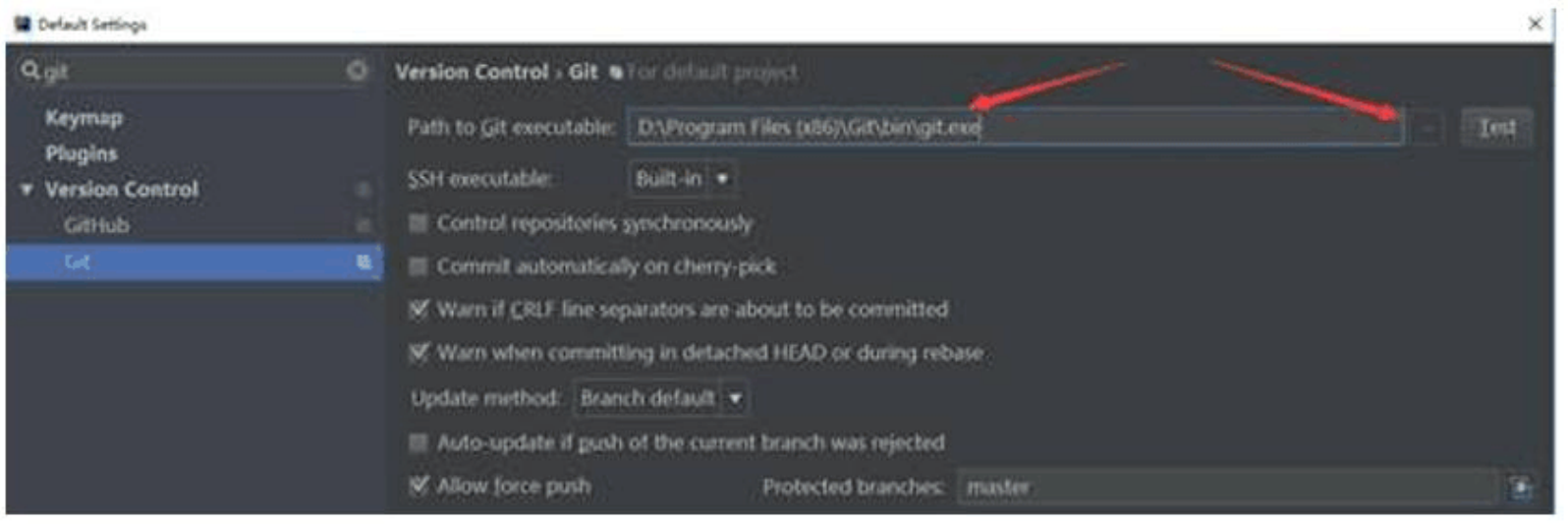
8. 合并分支

> 把team分支合并进master,先切换到master上，再执行merge

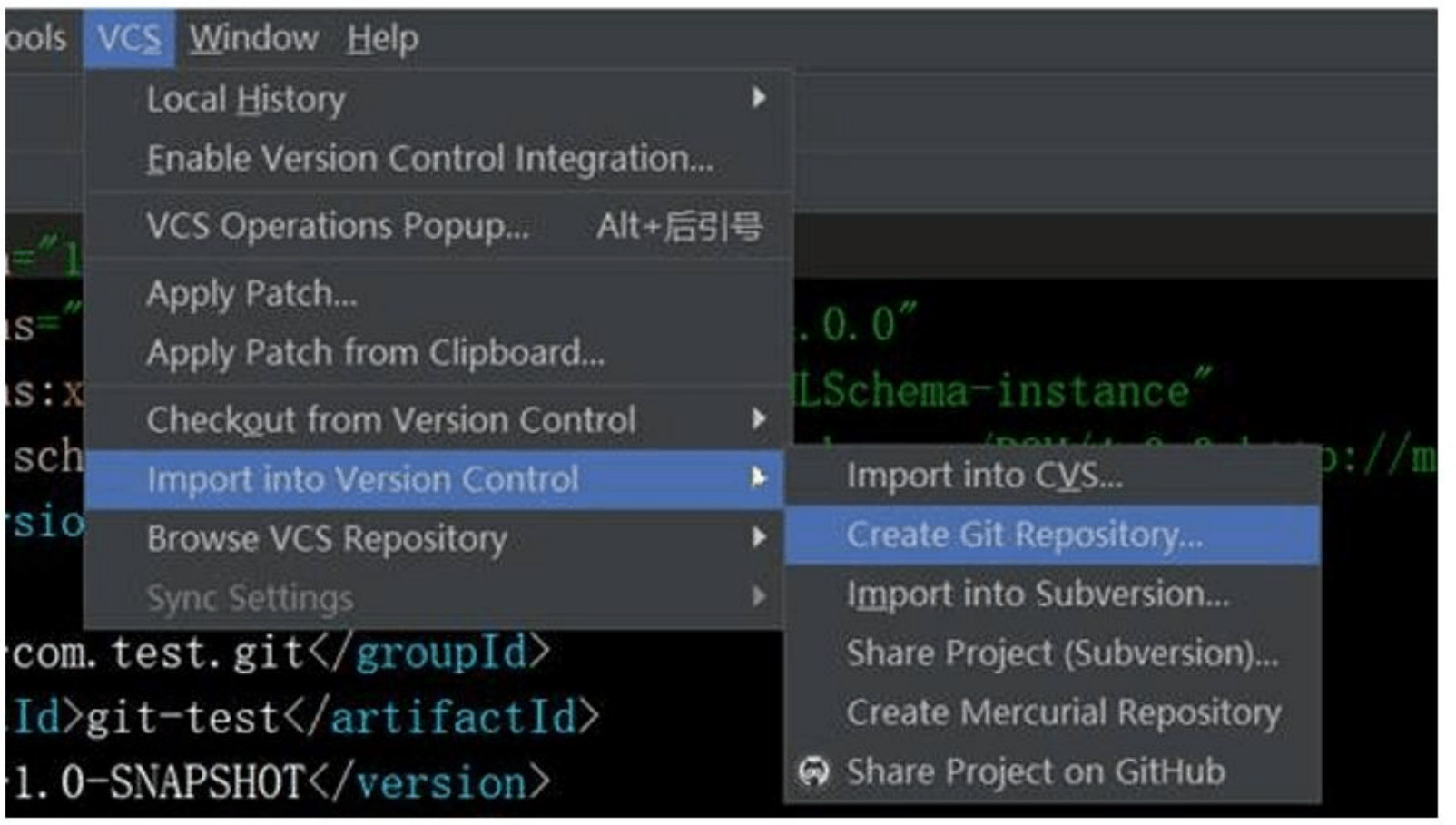
9. IntelliJ使用git

9.1 配置Git客户端

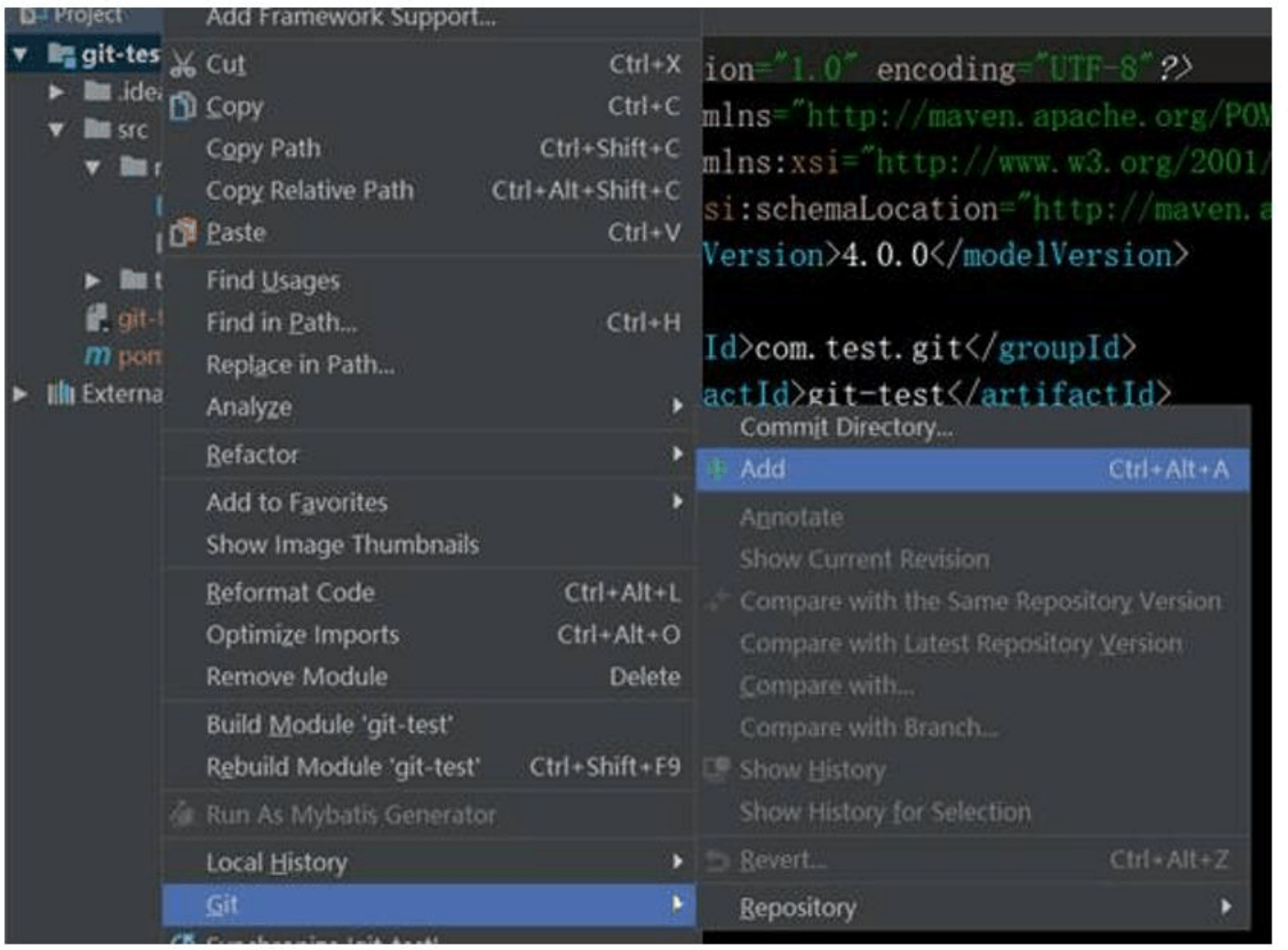
> 打开idea的settings，配置安装好的git地址

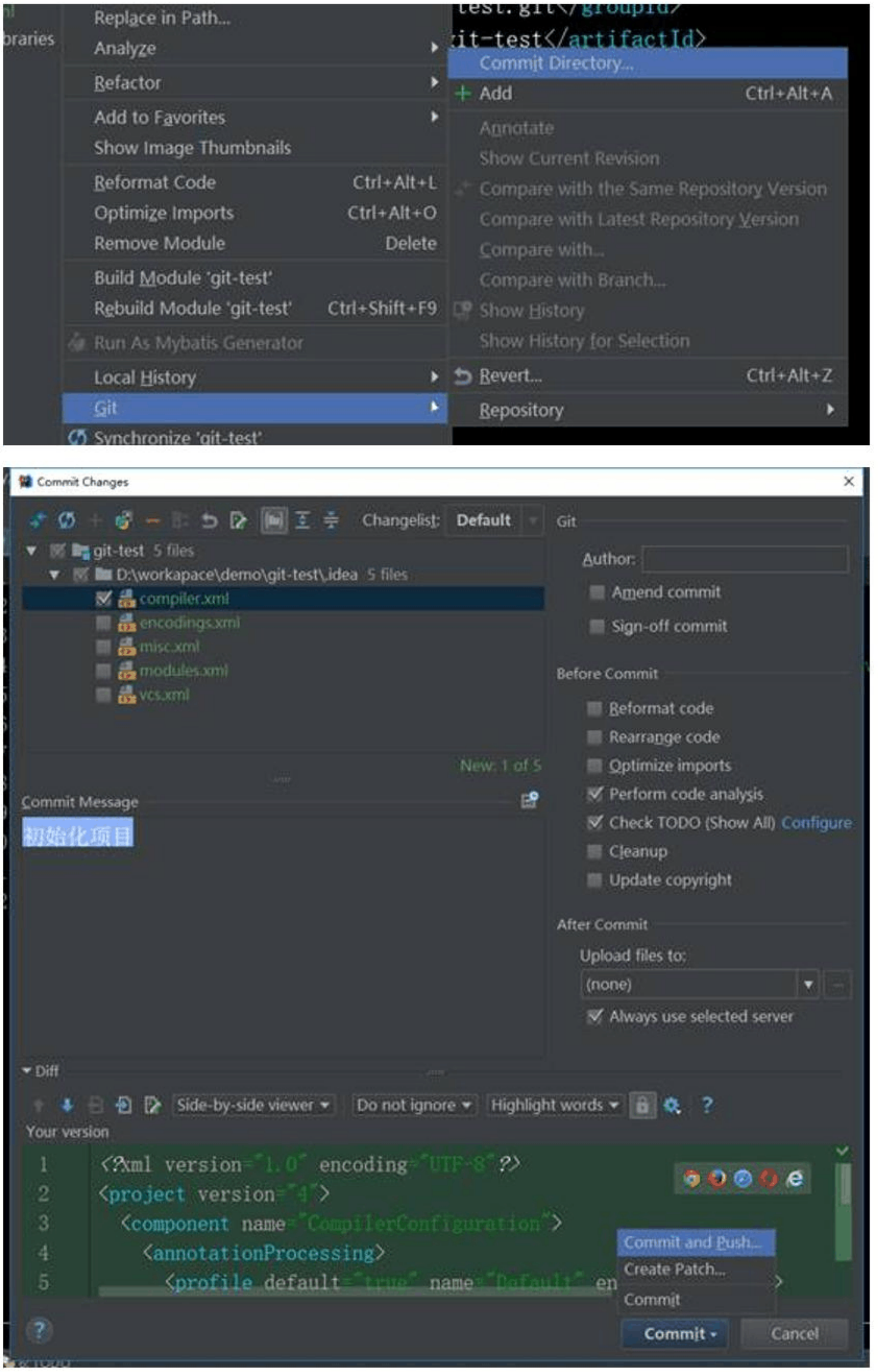


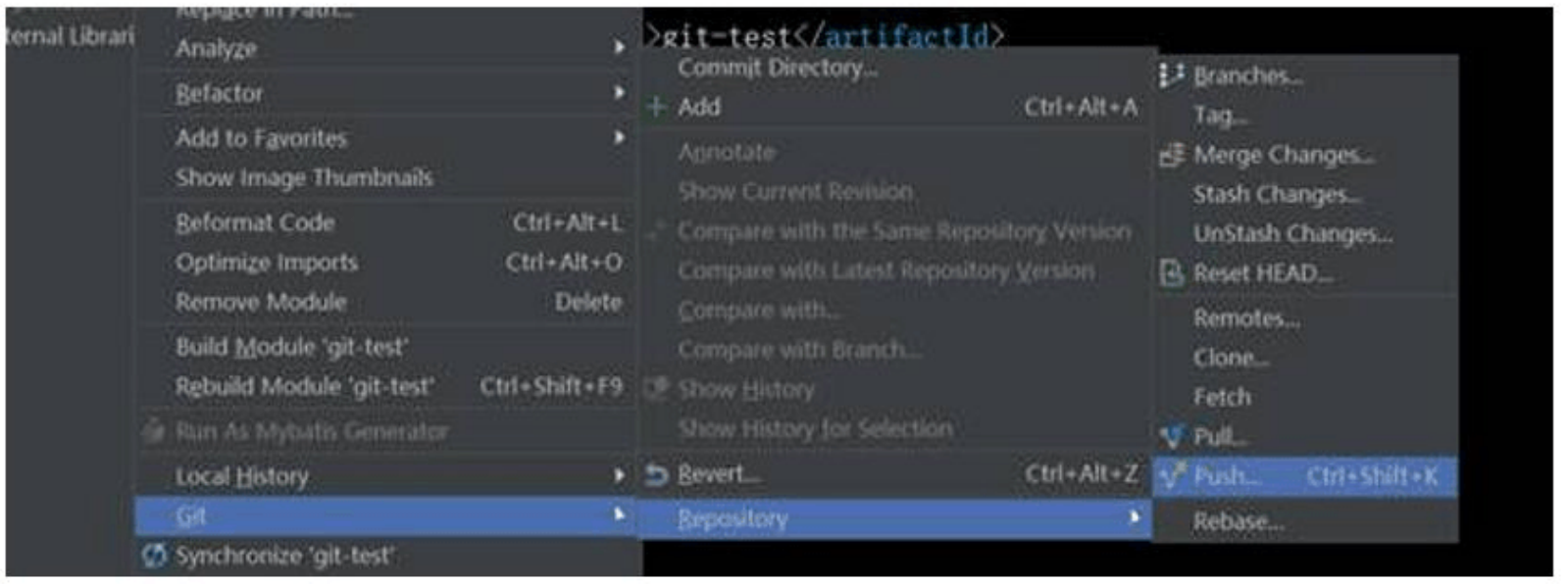
9.2 上传项目

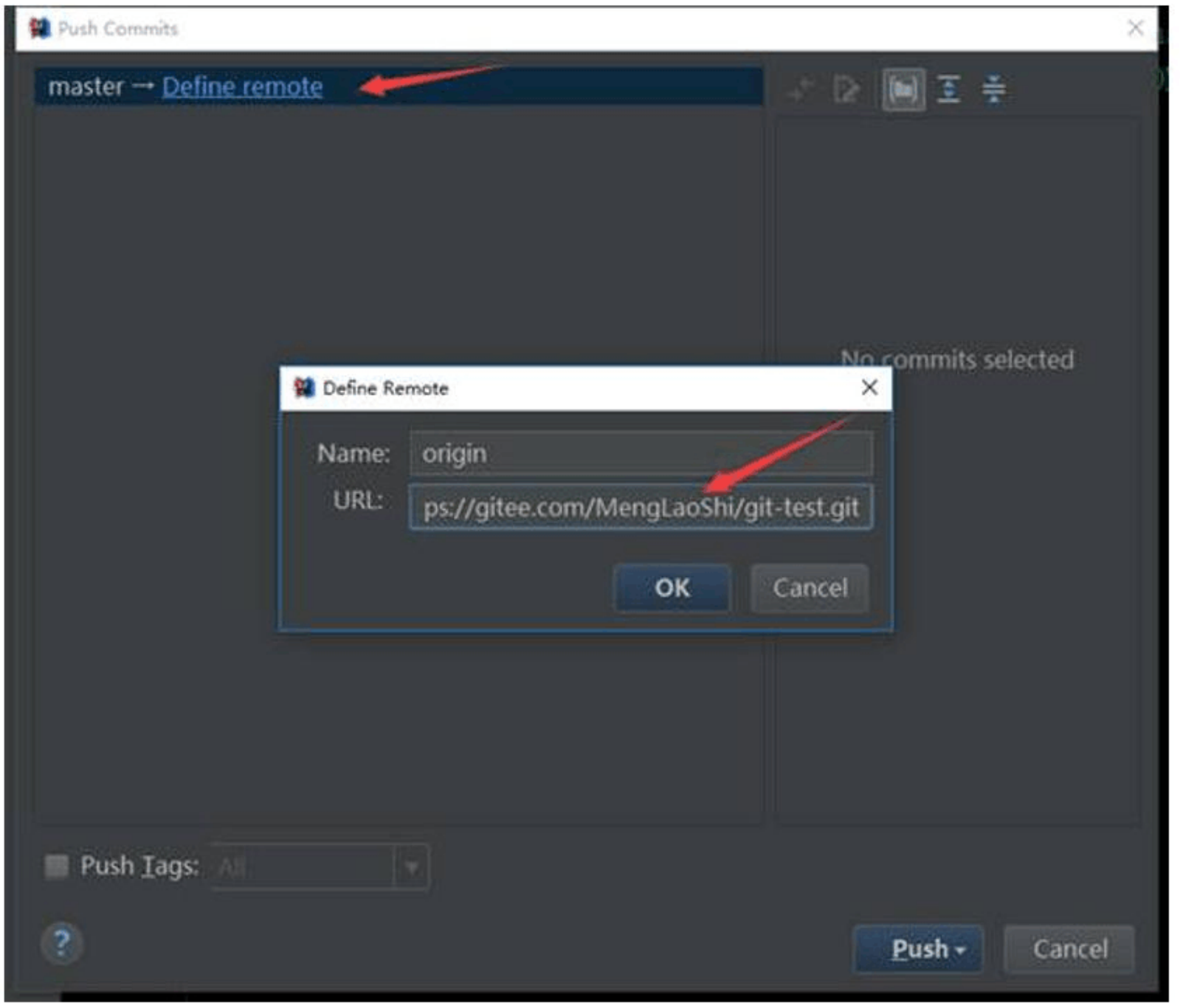

ools VCS Window Help
Local History
Enable Version Control Integration...
VCS Operations Popup... Alt+后引号
Apply Patch...
Apply Patch from Clipboard...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Sync Settings
Import into CVS...
Create Git Repository...
Import into Subversion...
Share Project (Subversion)...
Create Mercurial Repository
Share Project on GitHub
.0.0"
LSchema-instance"
com.test.git</groupId>
Id>git-test</artifactId>
1.0-SNAPSHOT</version>

右键项目

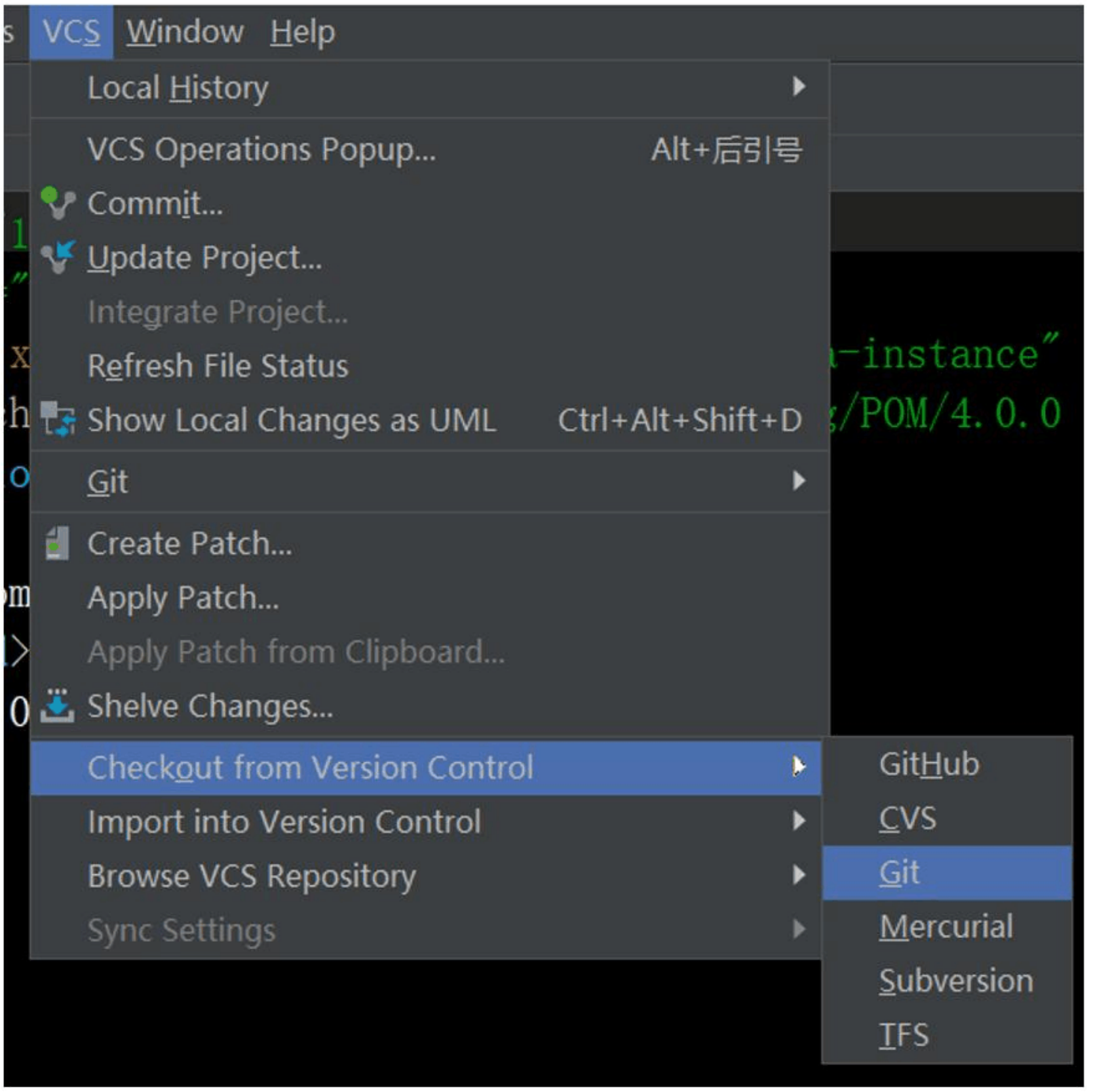

Add Framework Support...
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages
Find in Path... Ctrl+H
Replace in Path...
Analyze
Refactor
Add to Favorites
Show Image Thumbnails
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Remove Module Delete
Build Module 'git-test'
Rebuild Module 'git-test' Ctrl+Shift+F9
Run As Mybatis Generator
Local History
Git
Commit Directory...
Add Ctrl+Alt+A
Annotate
Show Current Revision
Compare with the Same Repository Version
Compare with Latest Repository Version
Compare with...
Compare with Branch...
Show History
Show History for Selection
Revert... Ctrl+Alt+Z
Repository
ion="1.0" encoding="UTF-8"?>
mlns="http://maven.apache.org/POM
mlns:xsi="http://www.w3.org/2001/
si:schemaLocation="http://maven.a
Version>4.0.0</modelVersion>
Id>com.test.git</groupId>
actId>git-test</artifactId>

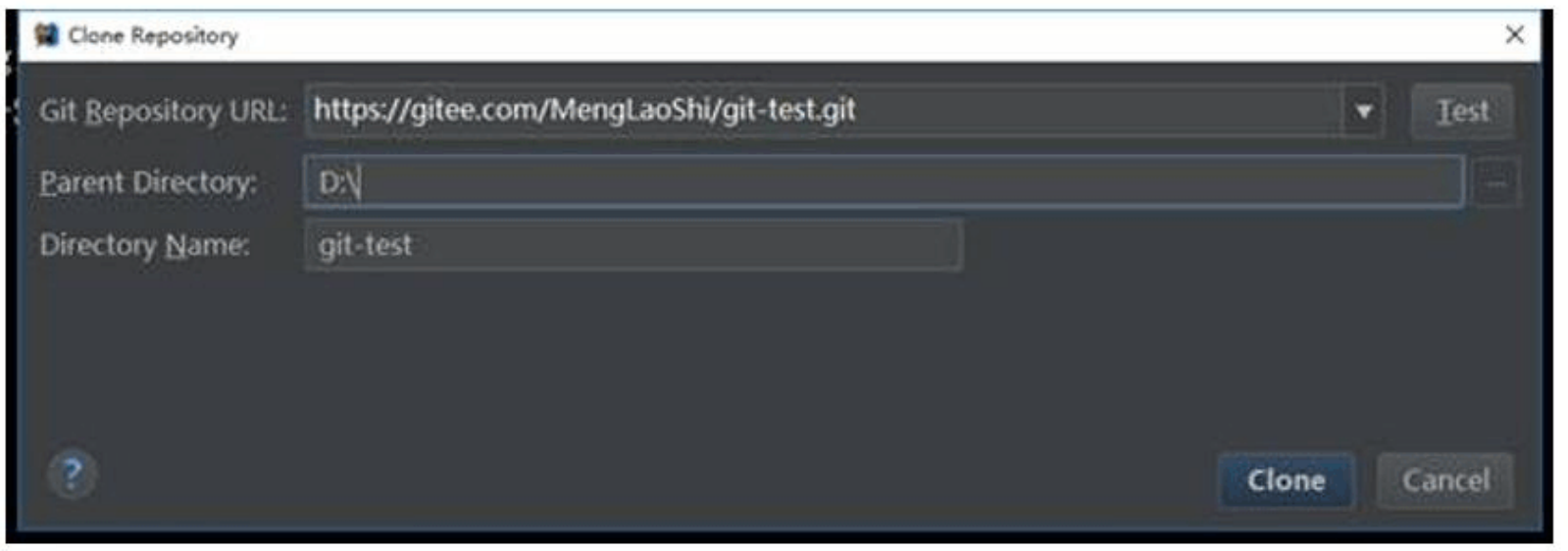

Replace in Path...
Analyze
Refactor
Add to Favorites
Show Image Thumbnails
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Remove Module Delete
Build Module 'git-test'
Rebuild Module 'git-test' Ctrl+Shift+F9
Run As Mybatis Generator
Local History
Git
Synchronize 'git-test'
Commit Directory...
Add Ctrl+Alt+A
Annotate
Show Current Revision
Compare with the Same Repository Version
Compare with Latest Repository Version
Compare with...
Compare with Branch...
Show History
Show History for Selection
Revert... Ctrl+Alt+Z
Repository
Commit Changes
Changelist: Default
git-test 5 files
D:\workapace\demo\git-test\.idea 5 files
compiler.xml
encodings.xml
misc.xml
modules.xml
vcs.xml
New: 1 of 5
Commit Message
初始化项目
Git
Author:
Amend commit
Sign-off commit
Before Commit
Reformat code
Rearrange code
Optimize imports
Perform code analysis
Check TODO (Show All) Configure
Cleanup
Update copyright
After Commit
Upload files to:
(none)
Always use selected server
Diff
Side-by-side viewer
Do not ignore
Highlight words
Your version
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project version="4">
3 <component name="CompilerConfiguration">
4 <annotationProcessing>
5 <profile default="true" name="Default" en
Commit and Push...
Create Patch...
Commit
Commit
Cancel

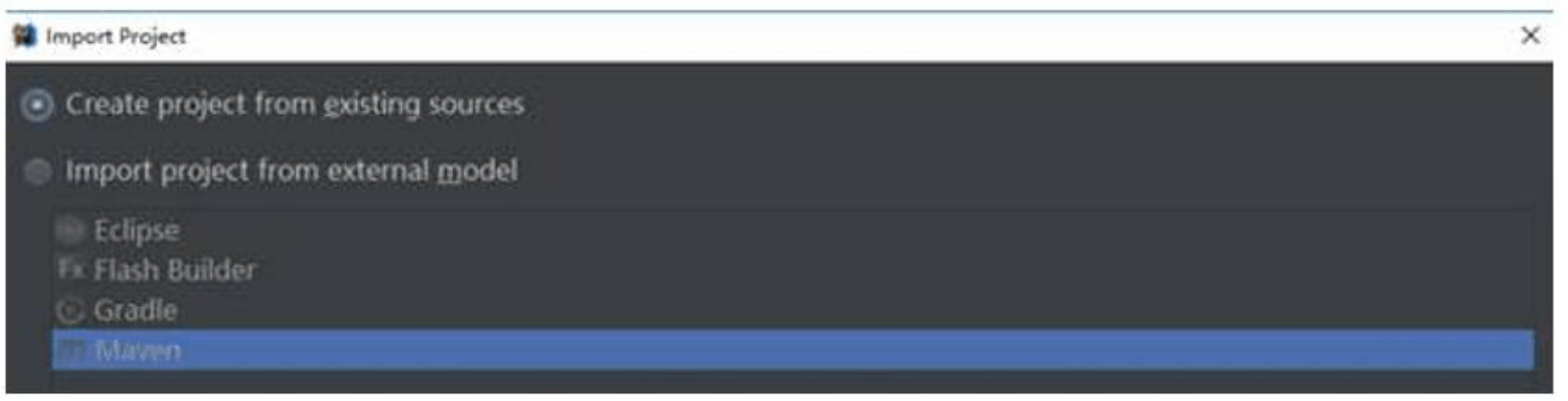

Analyze
Refactor
Add to Favorites
Show Image Thumbnails
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Remove Module Delete
Build Module 'git-test'
Rebuild Module 'git-test' Ctrl+Shift+F9
Run As Mybatis Generator
Local History
Git
Synchronize 'git-test'
Commit Directory...
Add Ctrl+Alt+A
Annotate
Show Current Revision
Compare with the Same Repository Version
Compare with Latest Repository Version
Compare with...
Compare with Branch...
Show History
Show History for Selection
Revert... Ctrl+Alt+Z
Repository
Branches...
Tag...
Merge Changes...
Stash Changes...
UnStash Changes...
Reset HEAD...
Remotes...
Clone...
Fetch
Pull...
Push... Ctrl+Shift+K
Rebase...


Push Commits
master → Define remote
No commits selected
Define Remote
Name: origin
URL: ps://gitee.com/MengLaoShi/git-test.git
OK
Cancel
Push Tags: All
Push
Cancel

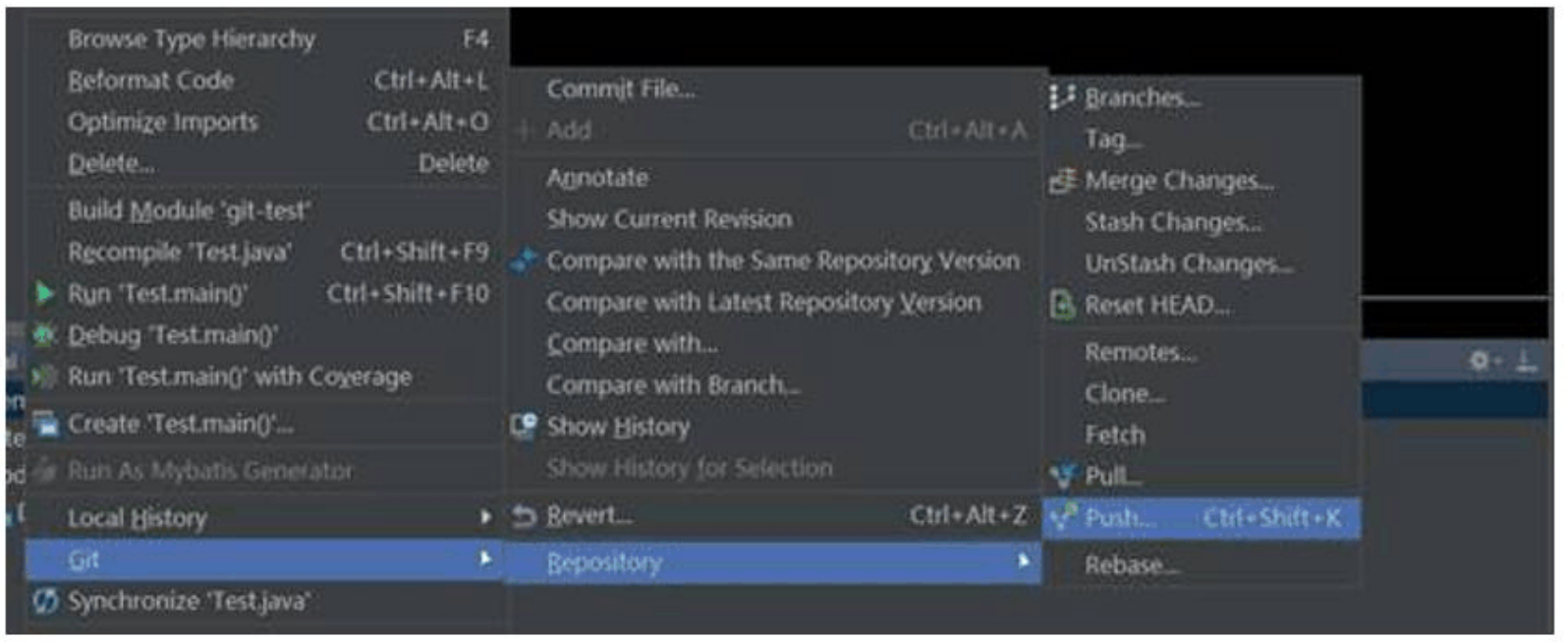
9.3 下载项目


VCS Window Help
Local History
VCS Operations Popup... Alt+后引号
Commit...
Update Project...
Integrate Project...
Refresh File Status
Show Local Changes as UML Ctrl+Alt+Shift+D
Git
Create Patch...
Apply Patch...
Apply Patch from Clipboard...
Shelve Changes...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Sync Settings
GitHub
CVS
Git
Mercurial
Subversion
TFS


Clone Repository
Git Repository URL: https://gitee.com/MengLaoShi/git-test.git
Test
Parent Directory: D:\
Directory Name: git-test
Clone
Cancel


Import Project
Create project from existing sources
Import project from external model
Eclipse
Flash Builder
Gradle
Maven

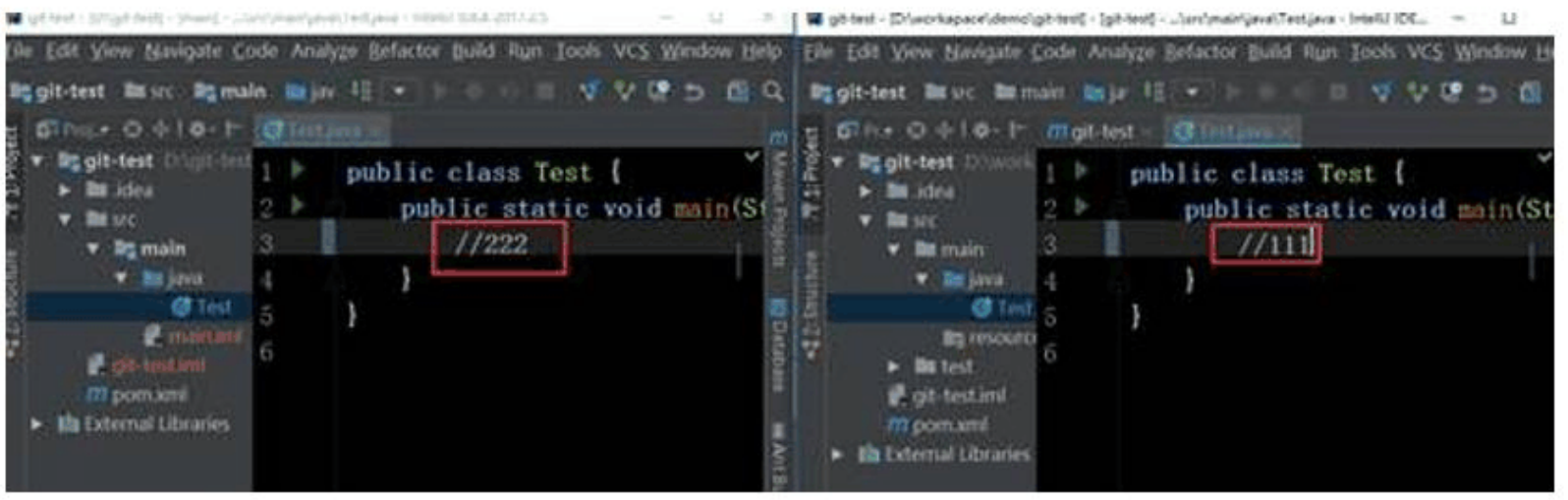
9.4 同步代码



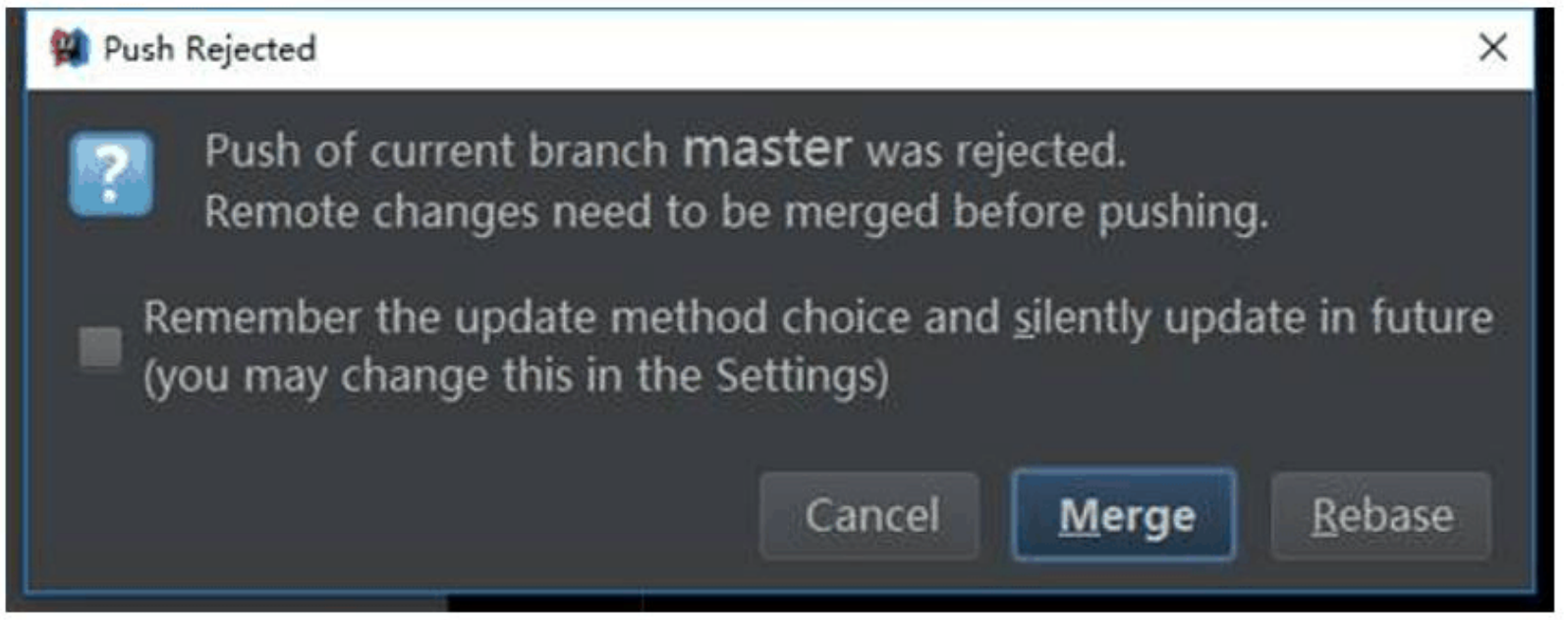
> Pull是下载代码，Push是上传代码

9.5 解决冲突

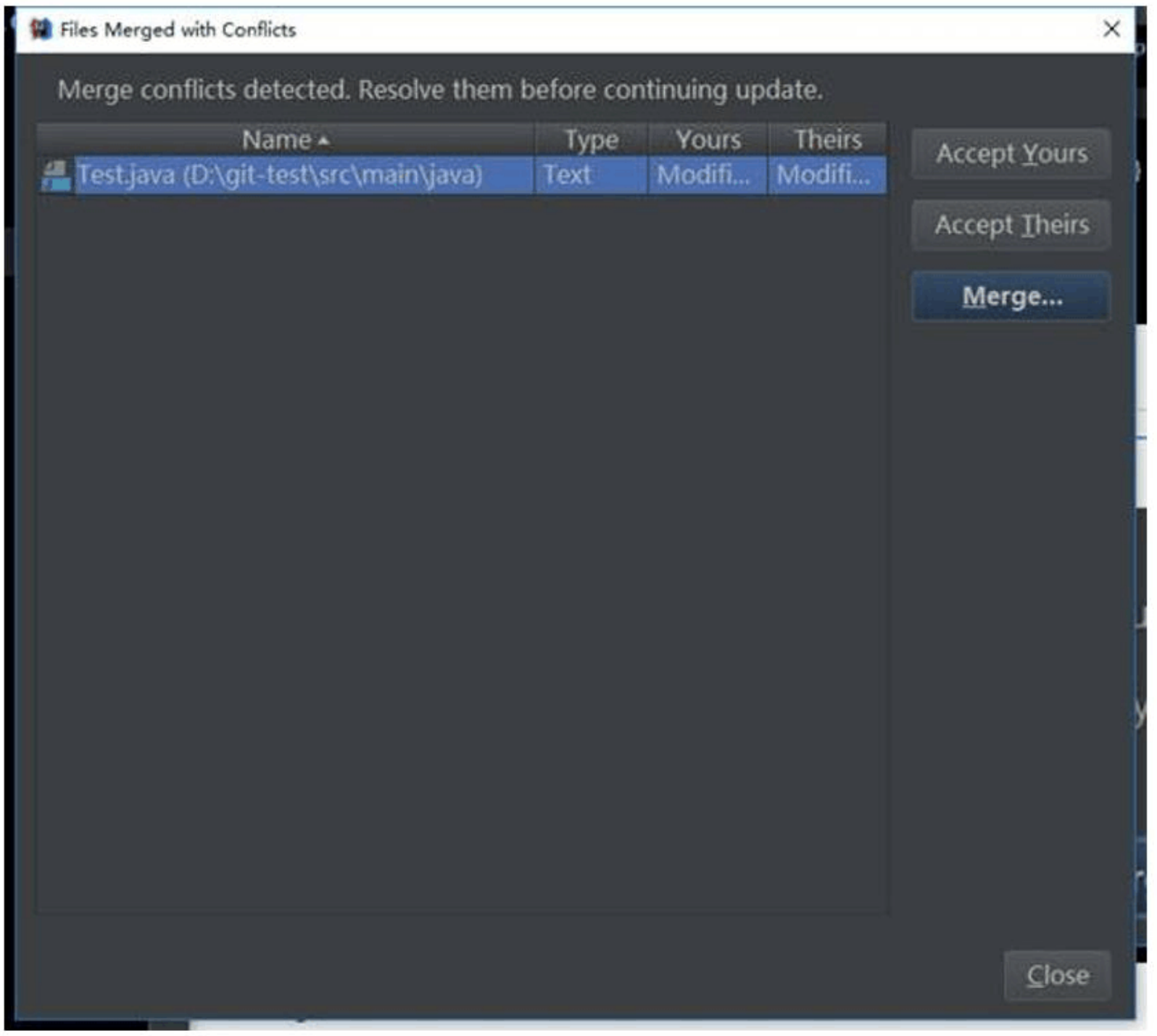
> 如果两个人改了同一段代码，一个人先提交，另一个人就无法提交



> 1先提交，2再提交会出现如下结果：



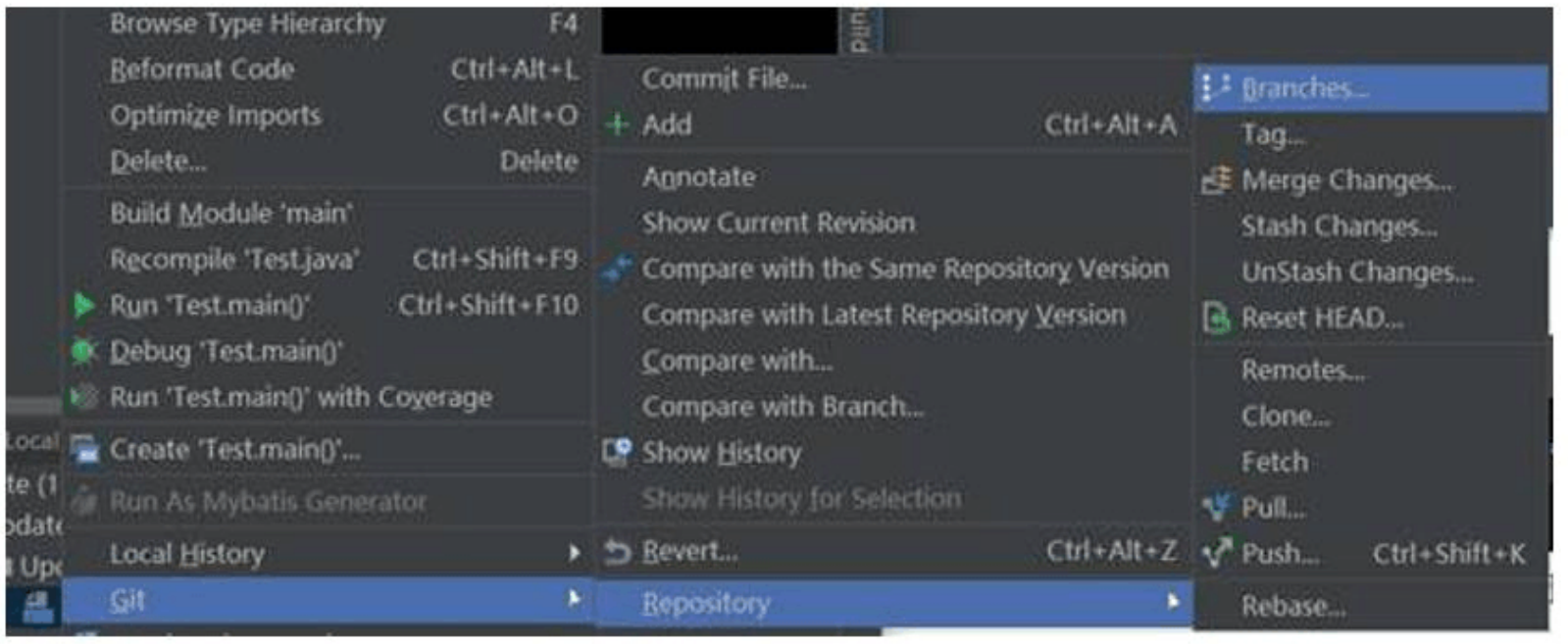
> 选择Merge



> Merge

> 点击这两个箭头可以选择合并代码的位置，调整代码之后，再重新Push代码即可。

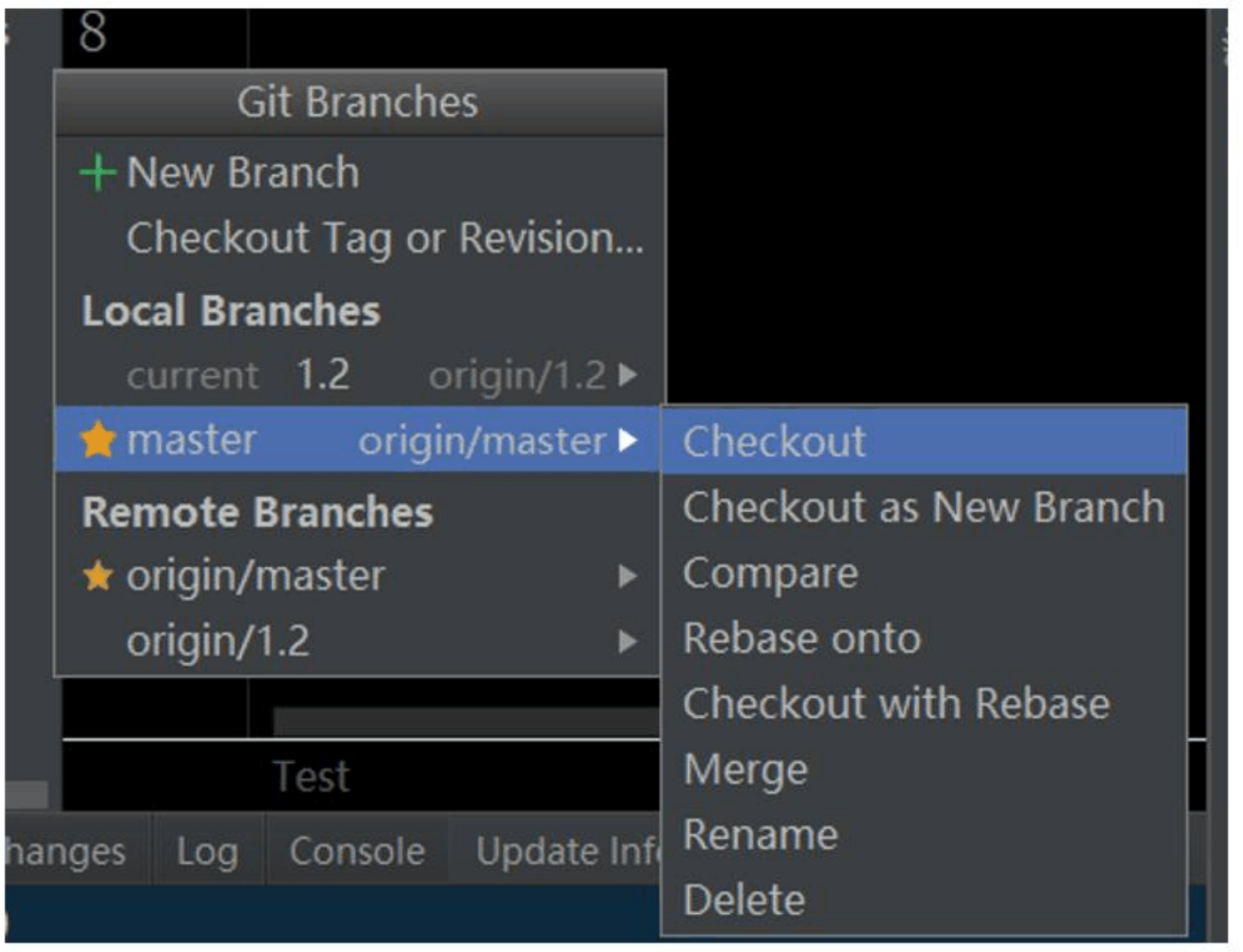
9.6 创建分支



创建新分支之后，提交代码，不会影响原分支

9.7 切换分支

切换分支之后，代码会变成该分支的样子

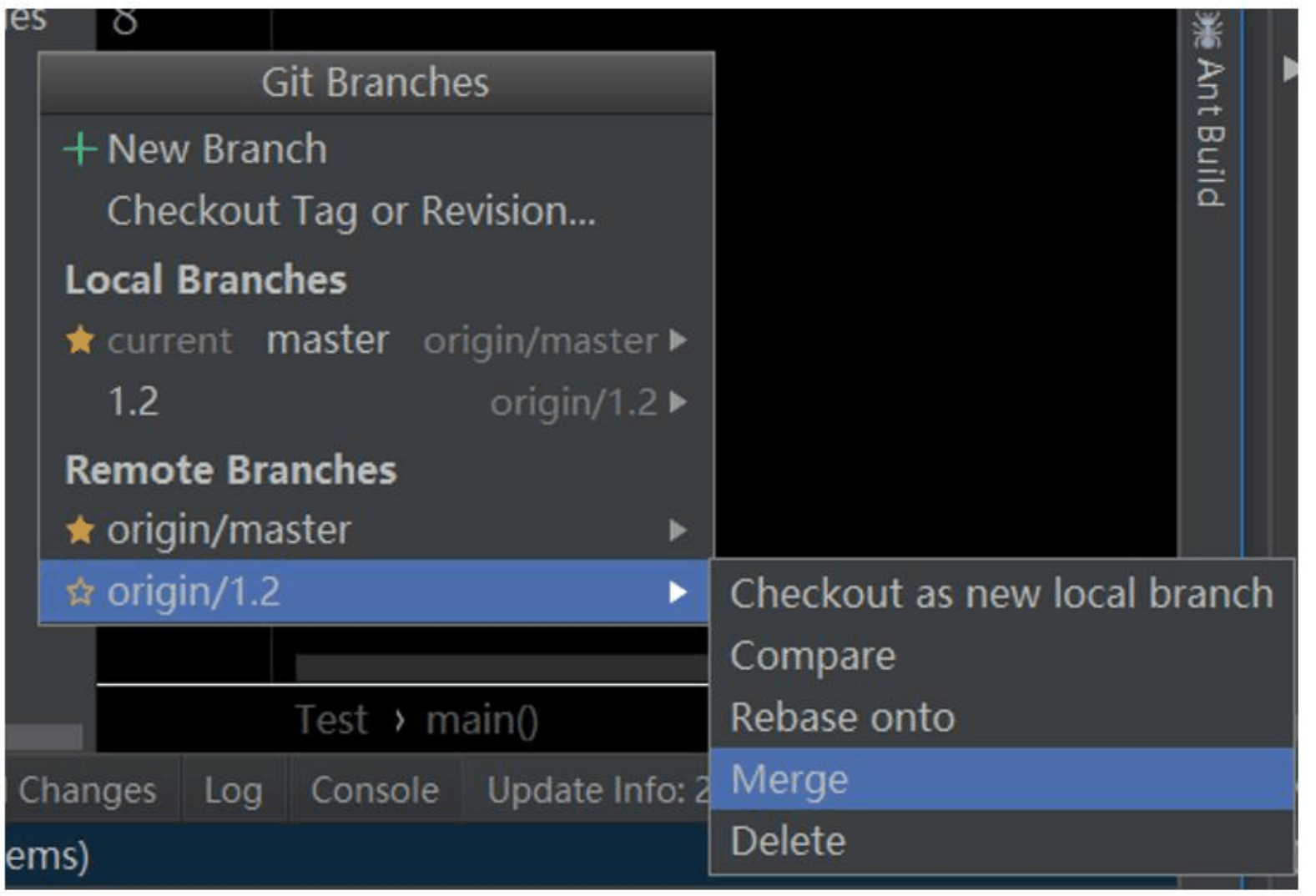


9.8 获取新分支

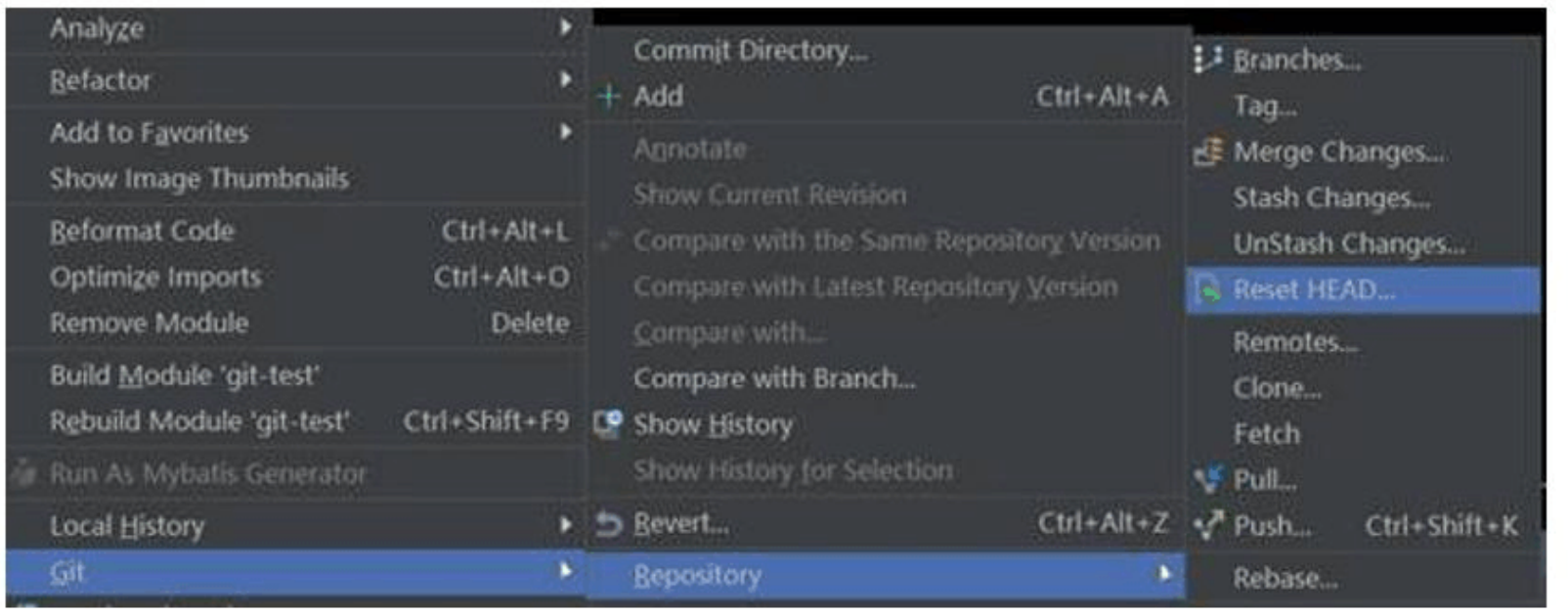
右键项目->Repository->Fetch，之后再分支中可以看到新分支了。

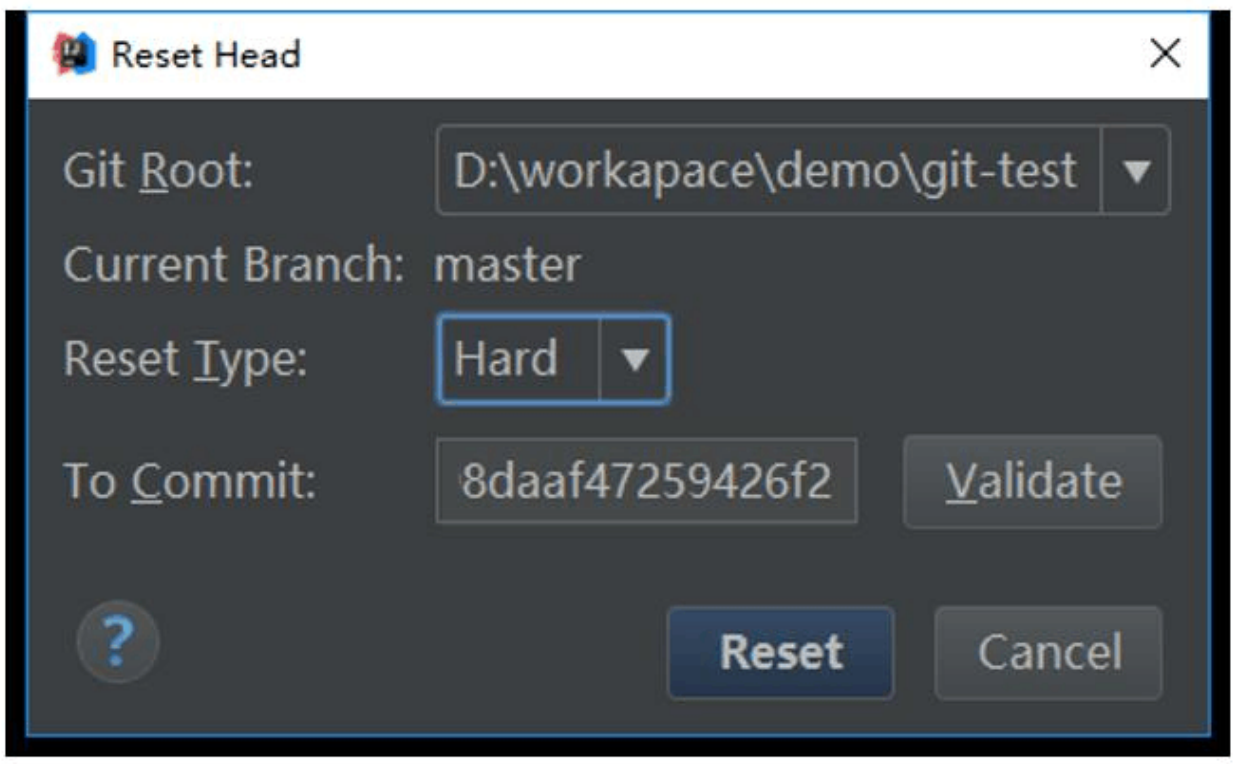
9.9 合并分支

> 把1.2合并到master
>
> 先切换到master,保证当前分支是master，在需要合并的1.2分支上点击merge,merge成功之后是在本地合并了，需要再push到远程仓库。在提交过程中如果遇到冲突需要解决冲突之后再Push



9.10 版本回退




Reset Head
Git Root:
D:\workapace\demo\git-test
Current Branch:
master
Reset Type:
Hard
To Commit:
8daaf47259426f2
Validate
Reset
Cancel