Git使用培训

# 背景

公司项目将会使用git来统一管理

# 规范

* 大的需求需要创建功能分支
* 小的改动可以直接在开发分支上修改，也可以利用本地分支灵活开发。

# 目的

* 更快熟悉git的使用
* 根据公司代码管理规范去使用

# gitlab账户及权限

账户及权限申请需要找项目负责人及gitlab管理员去申请创建（潘钰元）。

Gitlab地址：

http://10.20.7.105/

# SVN到GIT迁移过程

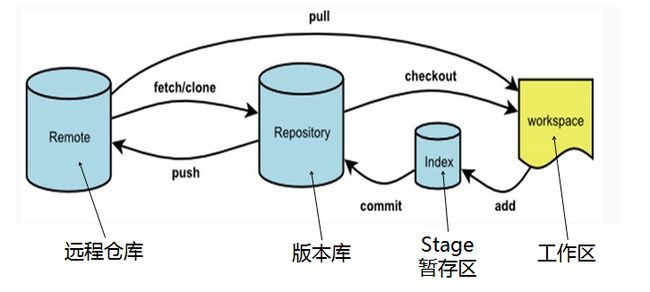
迁移会用svn的uat分支来初始化git的dev、test、uat、master 分支。

用svn的开发分支初始化git的svnDev分支。

注意：要保证测试跟验证的代码一致。

# 基本命令及流程

## 流程



### git clone

拷贝贝一个Git仓库到本地，让自己能够查看该项目，或者进行修改。

如果你需要与他人合作一个项目，或者想要复制一个项目，看看代码，你就可以克隆那个项目。

执行命令：

git clone [url]

### git add

git add 命令可将该文件添加到缓存

### git push

推送代码到远程仓库

### git pull

从远程获取最新版本到本地，并自动merge

### git fetch

拉取远程分支，从远程获取最新版本到本地

### git reset

一般使用在本地代码库的代码操作的回滚,以及暂存区代码的移除

### git revert

可以用户已经push到远程仓库的版本回滚

### git rm

如果只是简单地从工作目录中手工删除文件，运行 git status 时就会在 Changes not staged for commit 的提示。

要从 Git 中移除某个文件，就必须要从已跟踪文件清单中移除，然后提交。可以用以下命令完成此项工作

git rm <file>

如果删除之前修改过并且已经放到暂存区域的话，则必须要用强制删除选项 -f

git rm -f <file>

如果把文件从暂存区域移除，但仍然希望保留在当前工作目录中，换句话说，仅是从跟踪清单中删除，使用 --cached 选项即可

git rm --cached <file>

可以递归删除，即如果后面跟的是一个目录做为参数，则会递归删除整个目录中的所有子目录和文件：

git rm –r \*

进入某个目录中，执行此语句，会删除该目录下的所有文件和子目录。

## 忽略已加入版本控制的文件

1 将你的文件添加到忽略文件中.gitignore

2 命令行执行 git rm -r --cached <你要忽略的文件> 。如果是文件夹 可以用-r \*

3 执行git commit

## 注意

如没有push到远程的分支可以使用git reset（reset只会对本地版本库起作用,如果已经push的版本用这个是不会回退的）

如果已经提交代码可以使用git revert

# 分支管理

## 分支命名

master分支

uat分支

test分支

dev分支

feature分支（功能分支）

* 开发新的需求需要用dev分支创建feature分支；
* 分支命名：feature/8位日期(yyyyMMdd)/JIRA ID
* 如需求地址为：<http://10.20.7.104:8090/browse/REQ-2061>

分支就应该是：feature/20190108/REQ-2061

BUG分支（紧急bug分支）

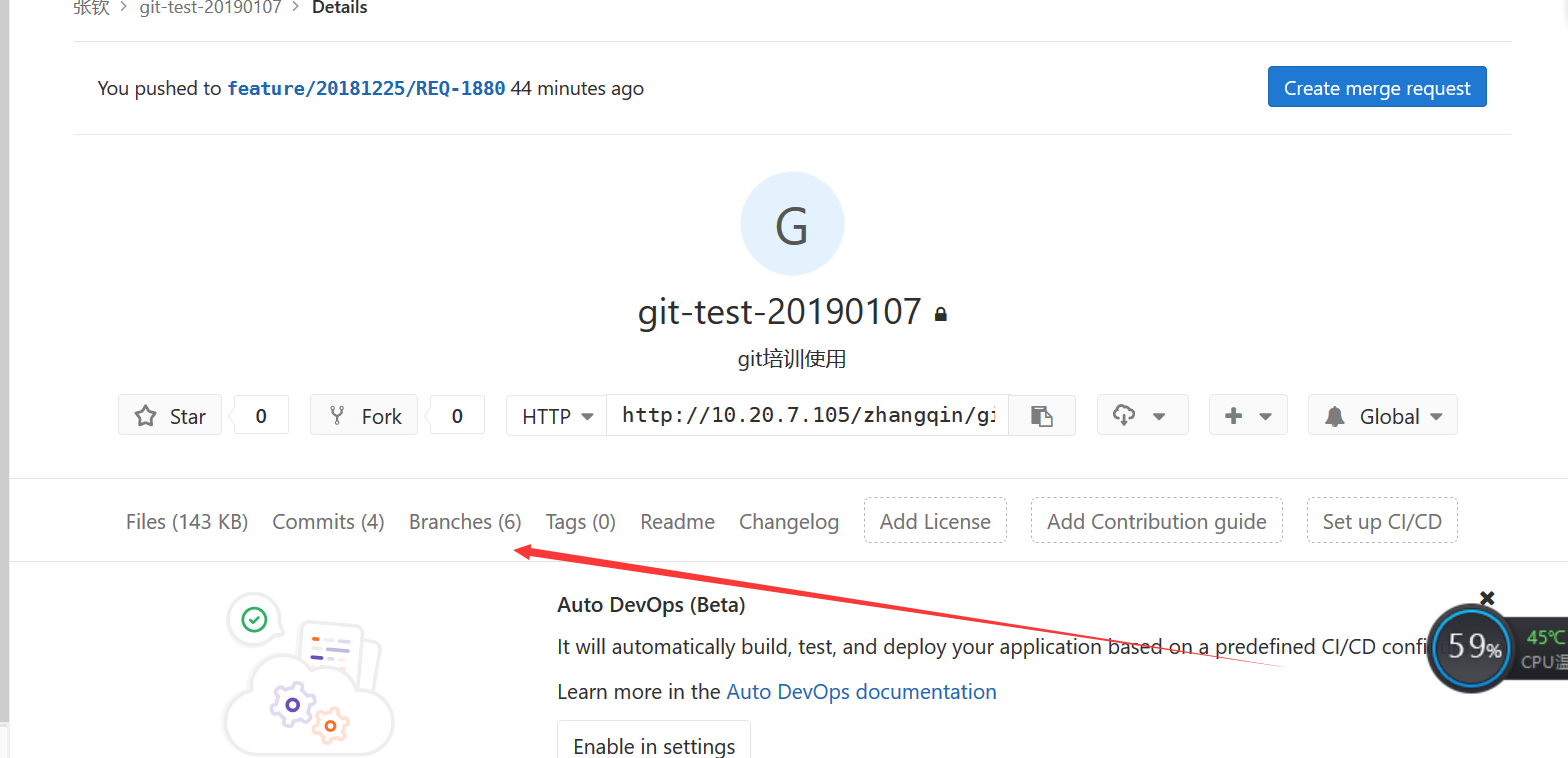
* 处理生产BUG时，基于master创建的BUG分支
* 分支命名：BugFix/8位日期(yyyyMMdd)/JIRA ID

(如: BugFix /20190107/REQ-0003)

## 分支创建

一般由开发小组长创建功能分支

创建位置在gitlab如下图



功能分支要从dev分支创建

紧急bug分支要从主分支创建

非紧急bug还是以dev分支创建，也就是功能优化

## 提交注意事项

禁止在master分支上直接修改提交代码，防止出现各个分支版本不一致问题。

每个人在提交Git时，提交日志应该按照：[分支名称]+message（代码说明）

## 分支合并

**注意：合并分支涉及多人开发，禁止拿整个分支合并，因为会把别人不想合并的代码也合并上去。应该找到自己的版本用cherry-pick命令去合并。**

1. feature分支开发完成并测试无误，才能往dev分支上进行合并;
2. dev分支测试通过，再往test分支上合并;
3. test测试通过，再往uat分支上合并;
4. uat测试通过，再往master分支上合并;。(小组长去合并)

## 发布

如果新建的git项目没有在自动化部署工具（jenkins）中，需要联系配置管理员添加。

## 注意

1切分支的时候如果你当前 有未提交到本地的版本库的代码，切换的时候会将你的代码带到新切换的分支如果有冲突会造成代码的丢失，所以切换分支建议将修改的代码提交或者暂存（stash）起来。

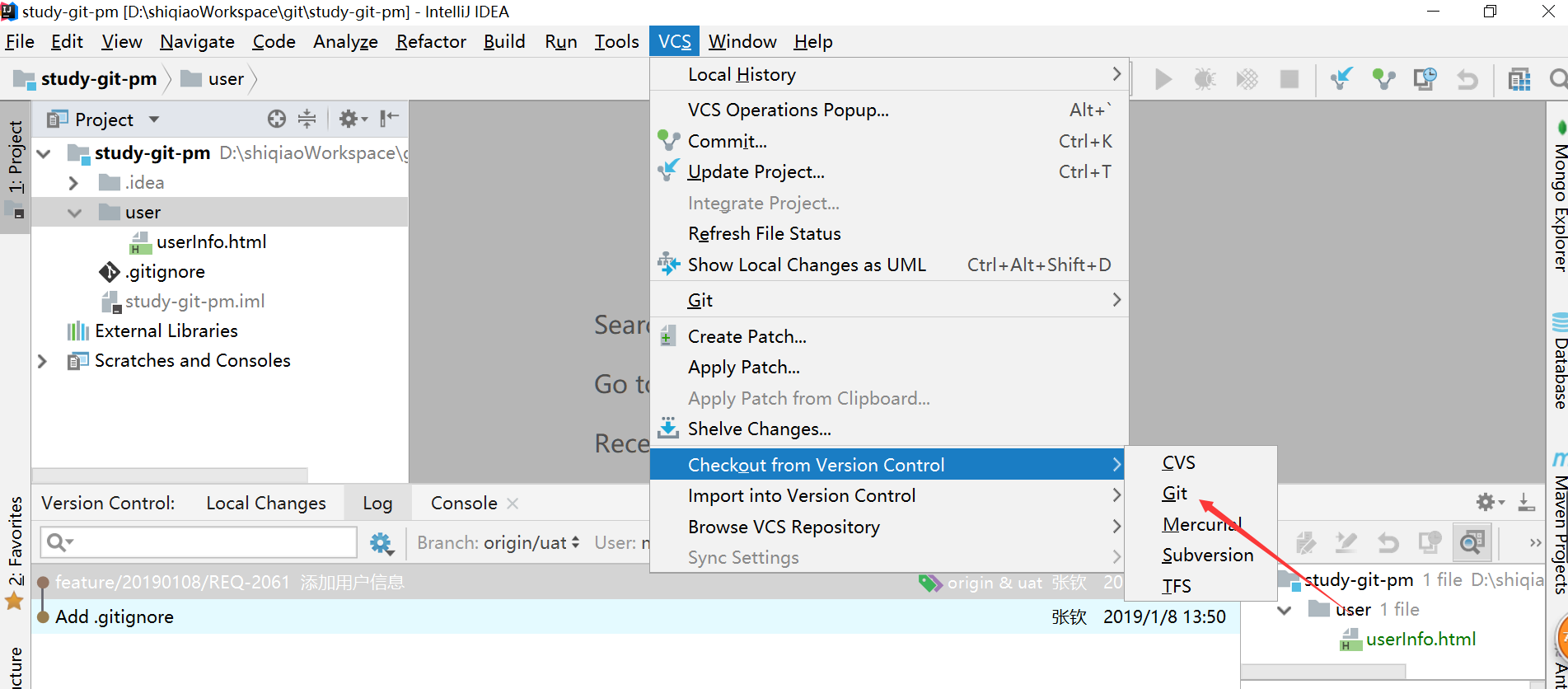
2 小的改动可以在开发分支上直接修改。

# Idea演示操作步骤

## 下载项目

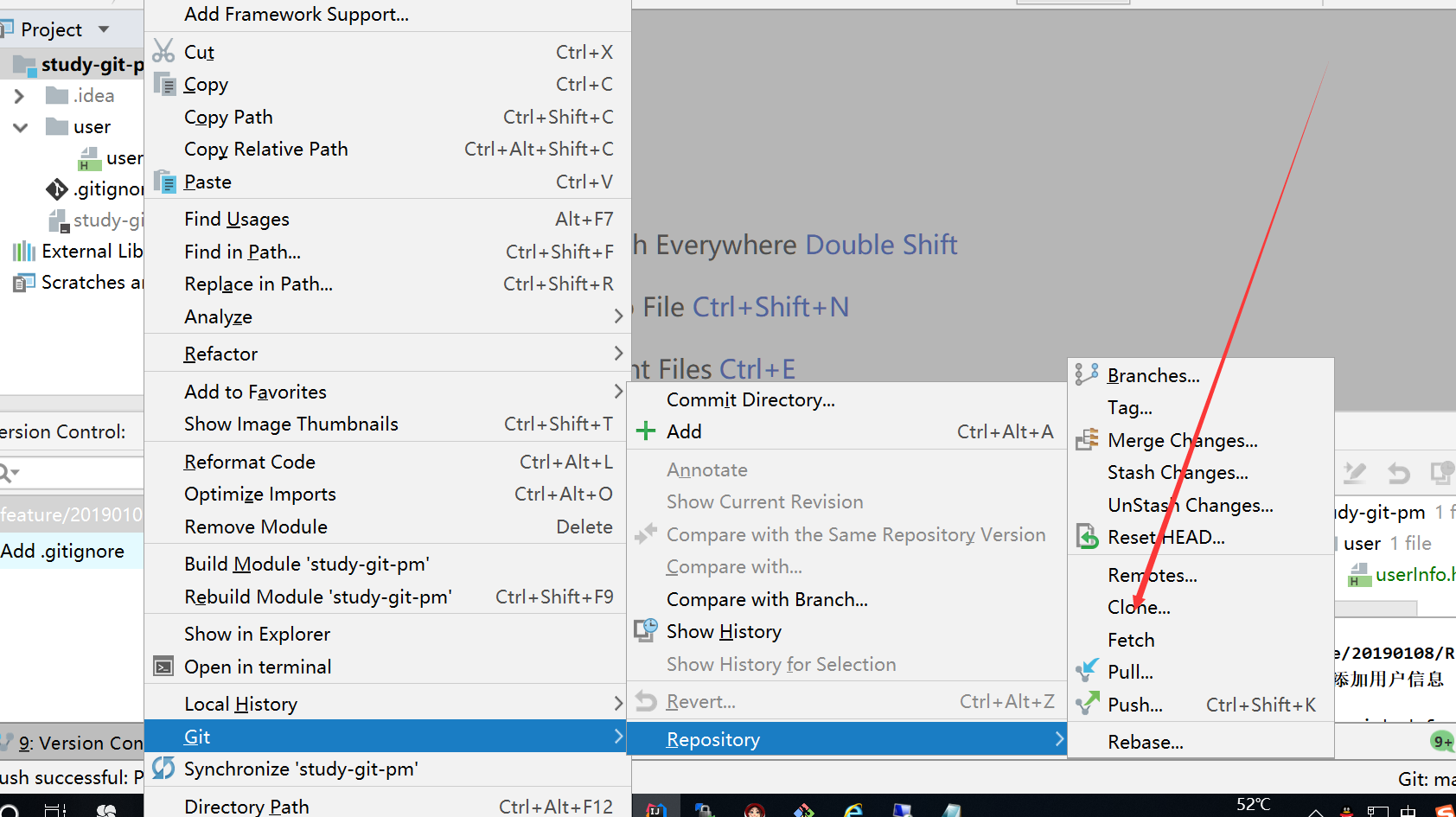
git clone 项目路径

vcs->checkout from version control->git



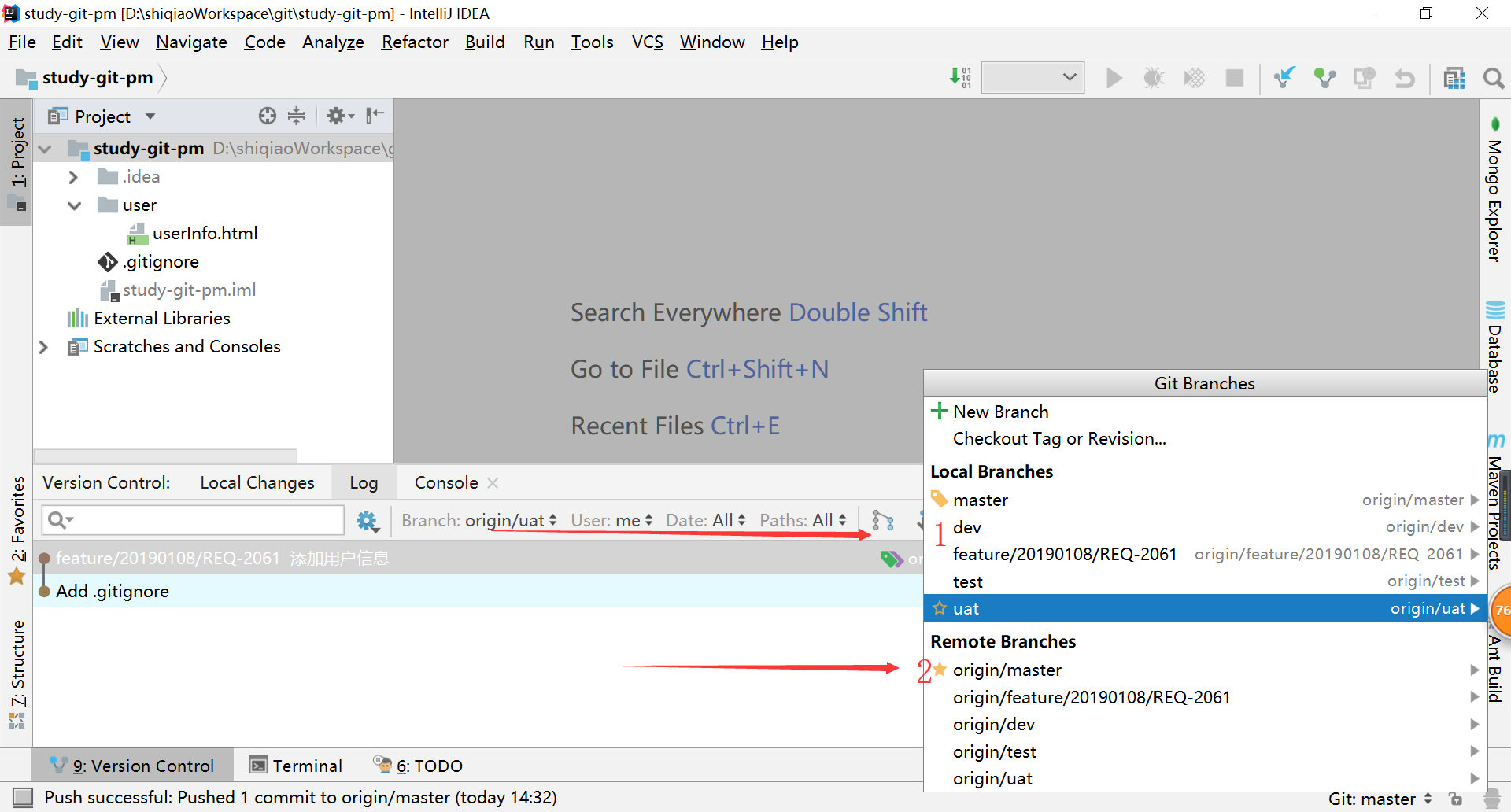
## 抓取分支

git fetch



## 切换分支

git checkout 分支名



图中指向红色“1”为已经存在的分支，如果存在直接切换即可

如果不存在需要从指向红色“2”的区域去检出，如果你想要检出的分支没有存在，则需要执行fetch命令去将分支从远程上抓取到本地代码库

## 代码提交

git add

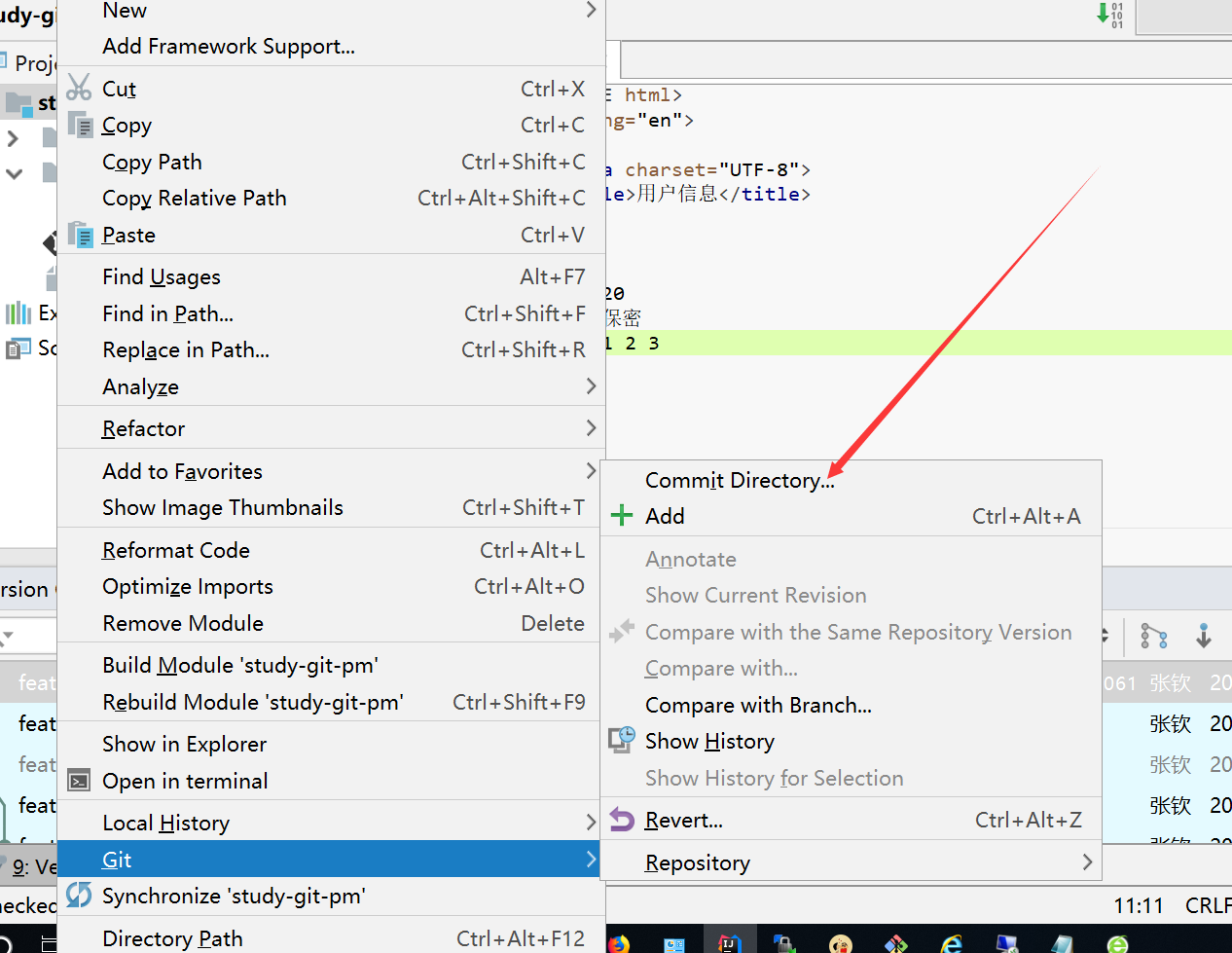
git commit

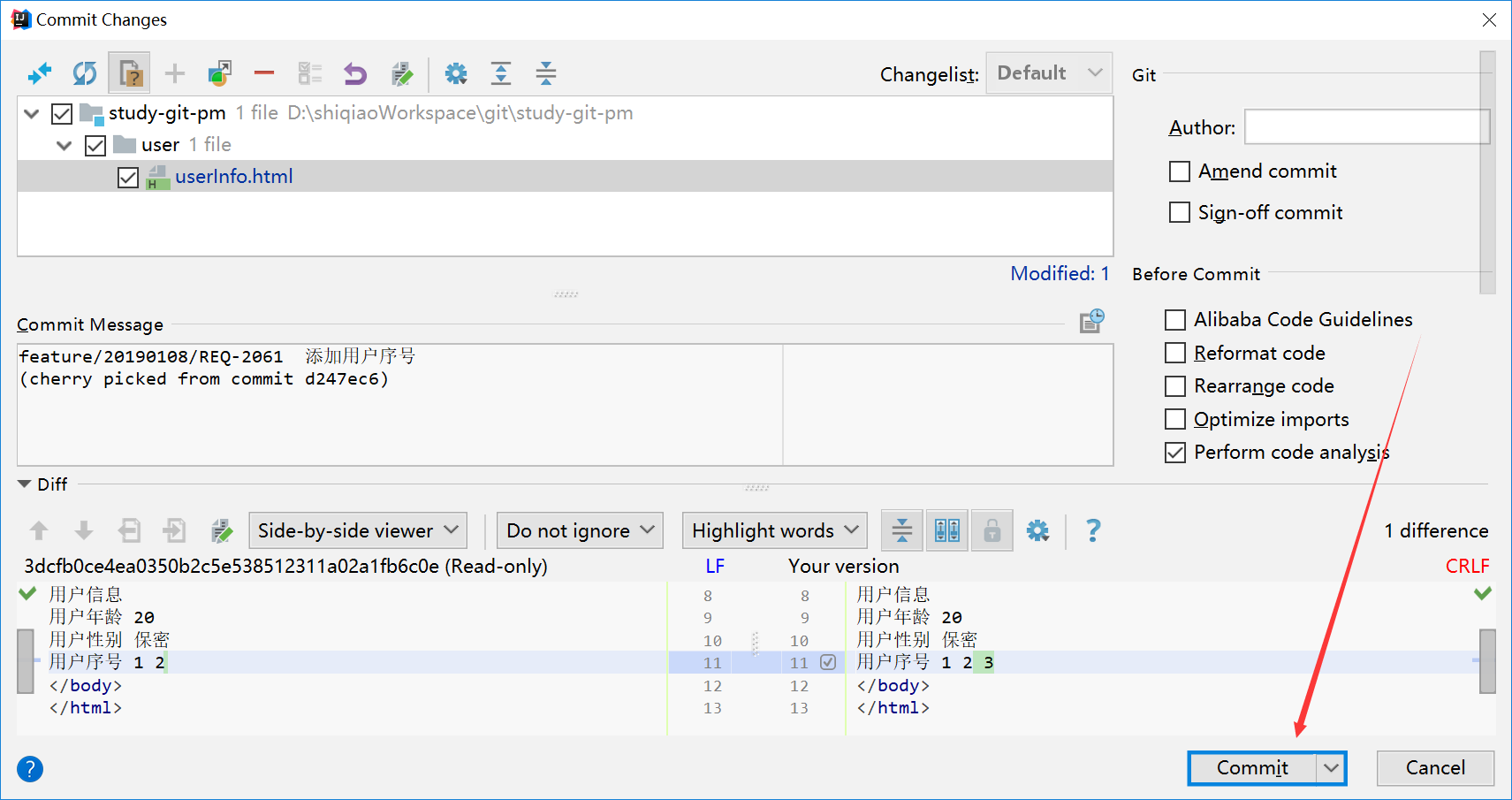
git push

如果没有添加到暂存区的代码需要先执行add操作，然后再执行commit

commit只是提交到本地代码库，如果想要推送到远程需要执行push操作

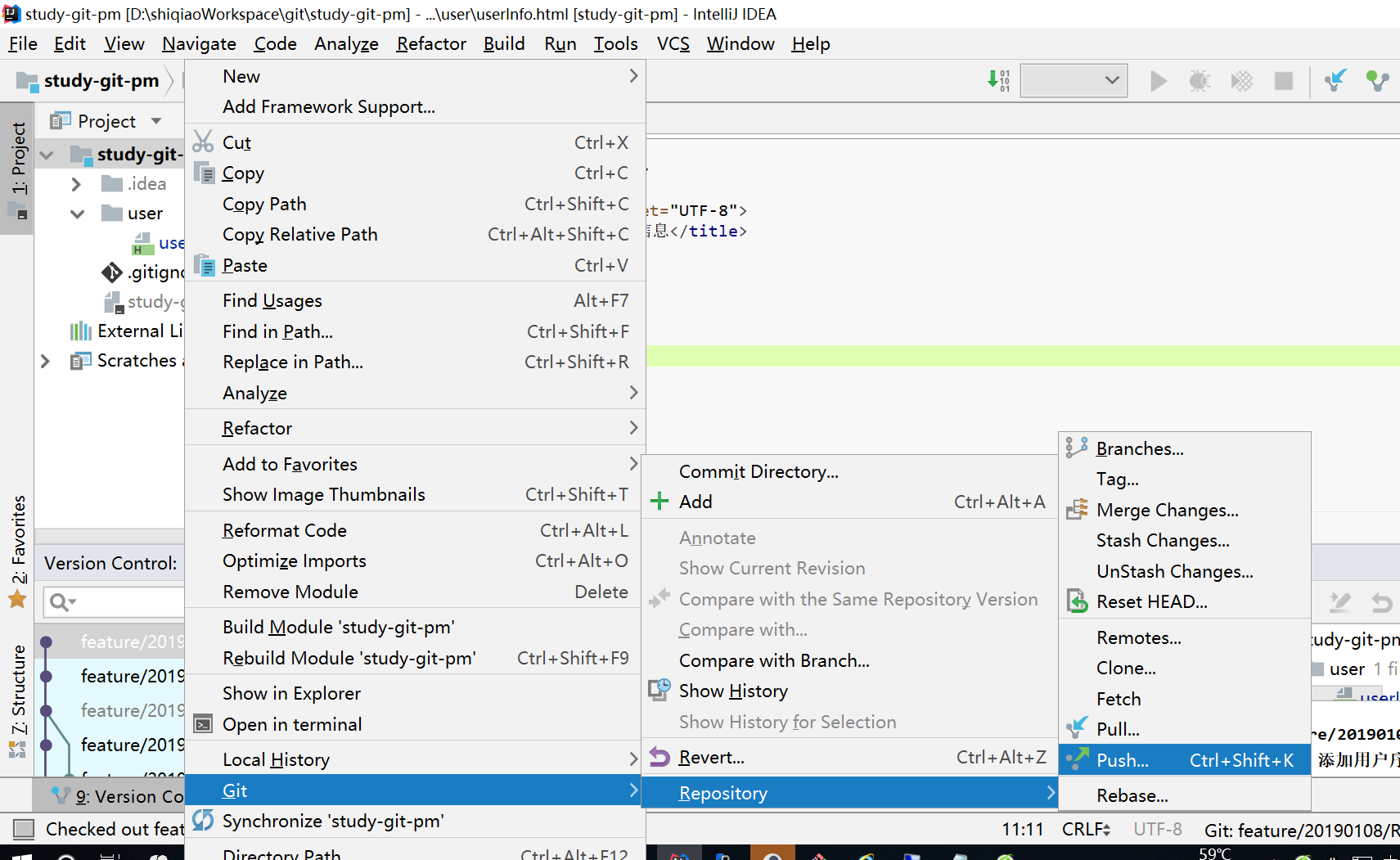
项目右键如下图：

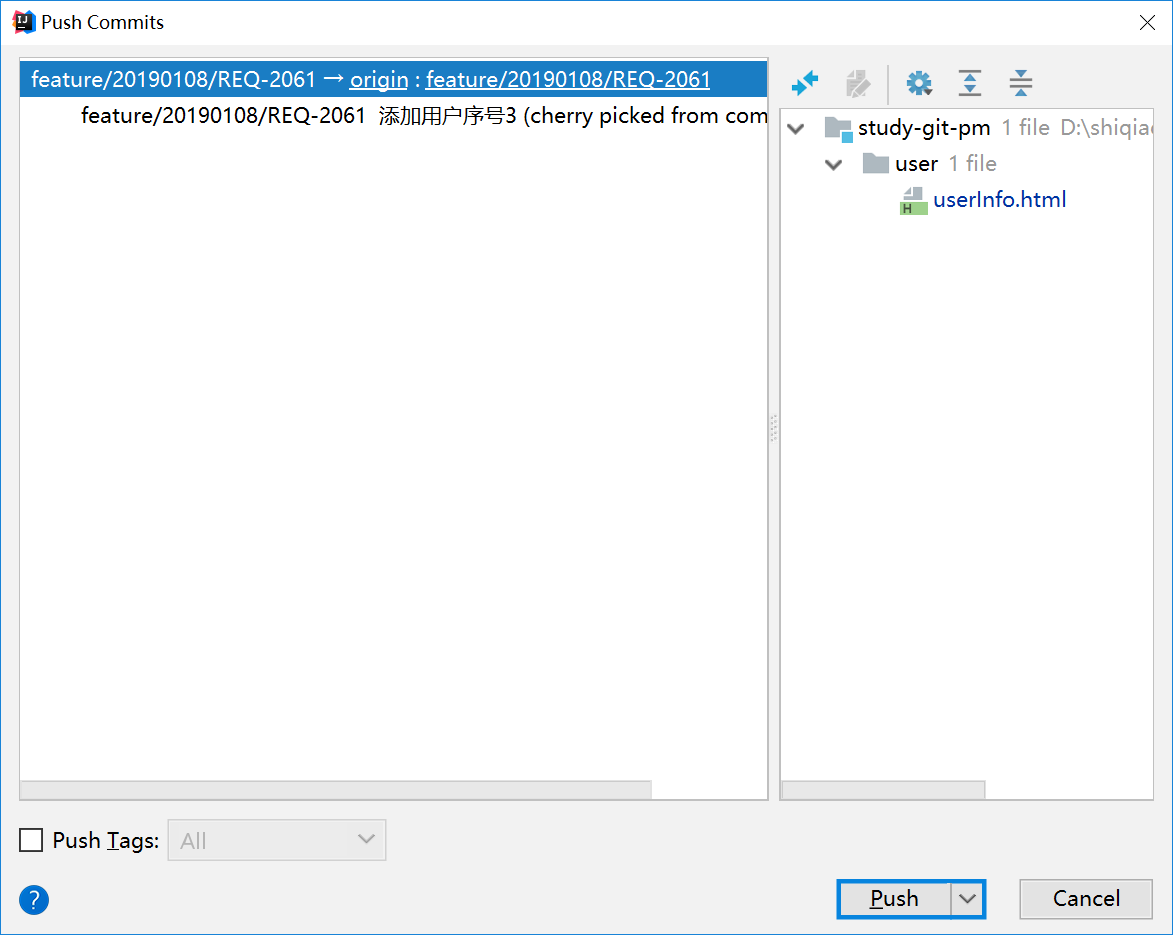




提交完成之后再执行push操作

如下图





## 版本撤回

### 注意

如没有push到远程的分支可以使用git reset（reset只会对本地版本库起作用,如果已经push的版本用这个是不会回退的）

如果已经提交代码可以使用git revert

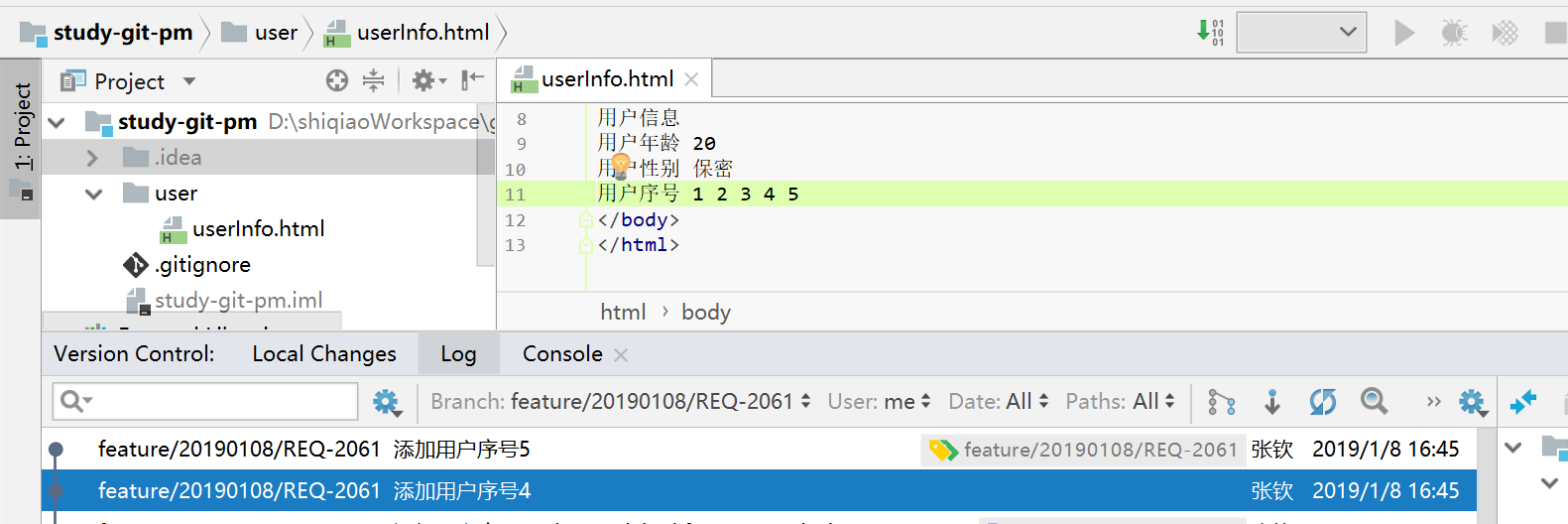
这两个的区别还有就是reset会删除提交记录revert不会

如果已经push的代码用reset操作再次提交会造成冲突需要去解决。

### Reset演示

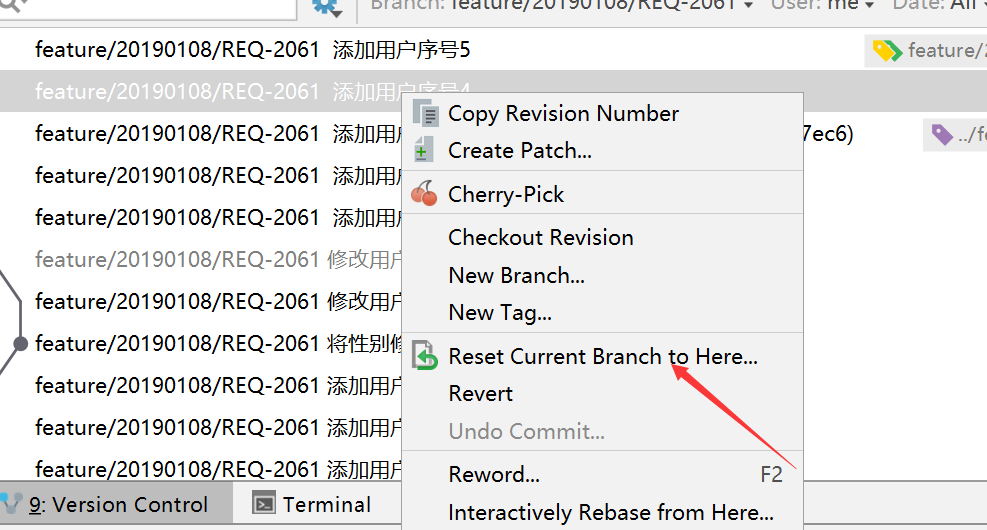
撤回某个版本之后的版本可以用它

我现在提交了两个版本如下图添加了4 5

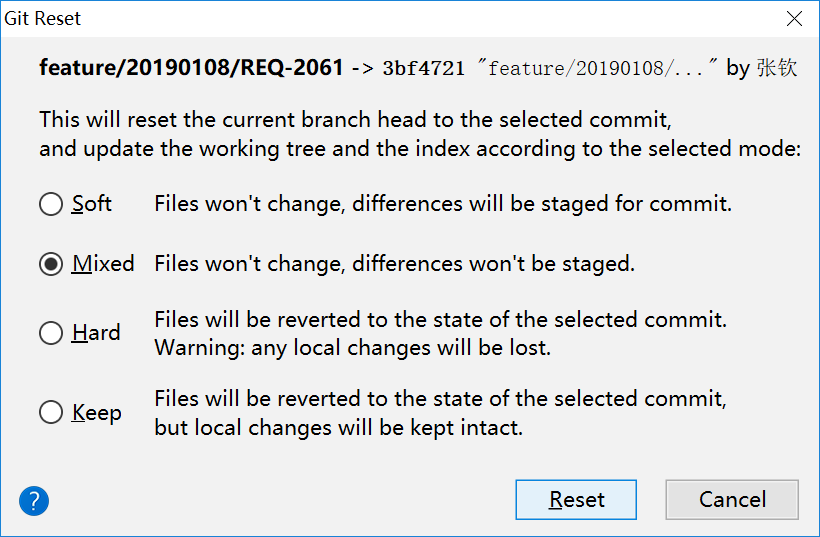


然后想把4以后的版本回退

在4版本上点击右键会弹出选择框如下图



会出现如下界面

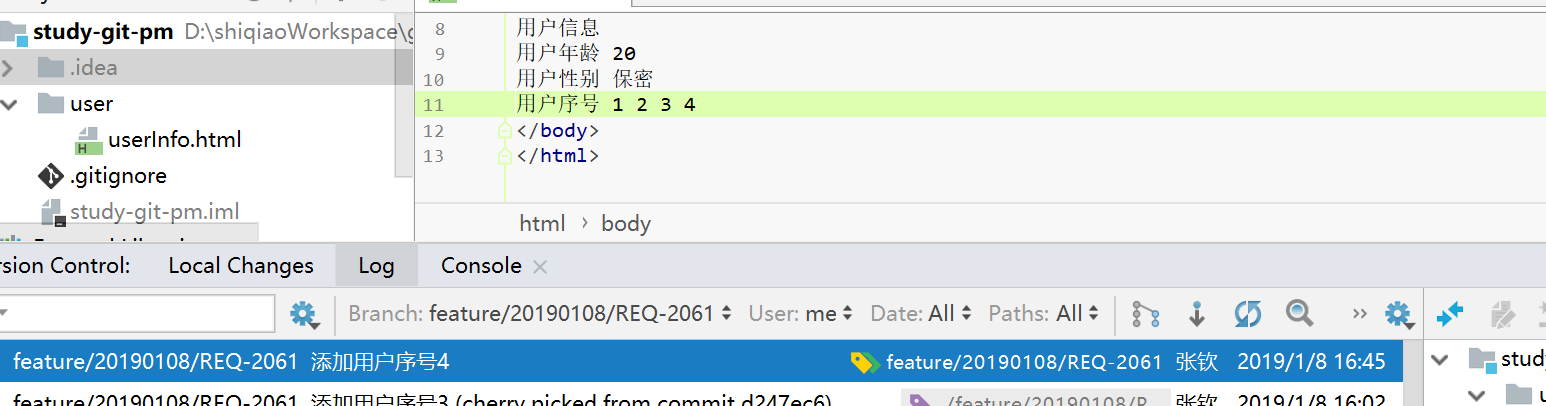


这几个有什么区别呢

soft mixed不会删除你本地的代码，只是将提交记录删除让代码还原到未提交或者未add的状态(新增文件)

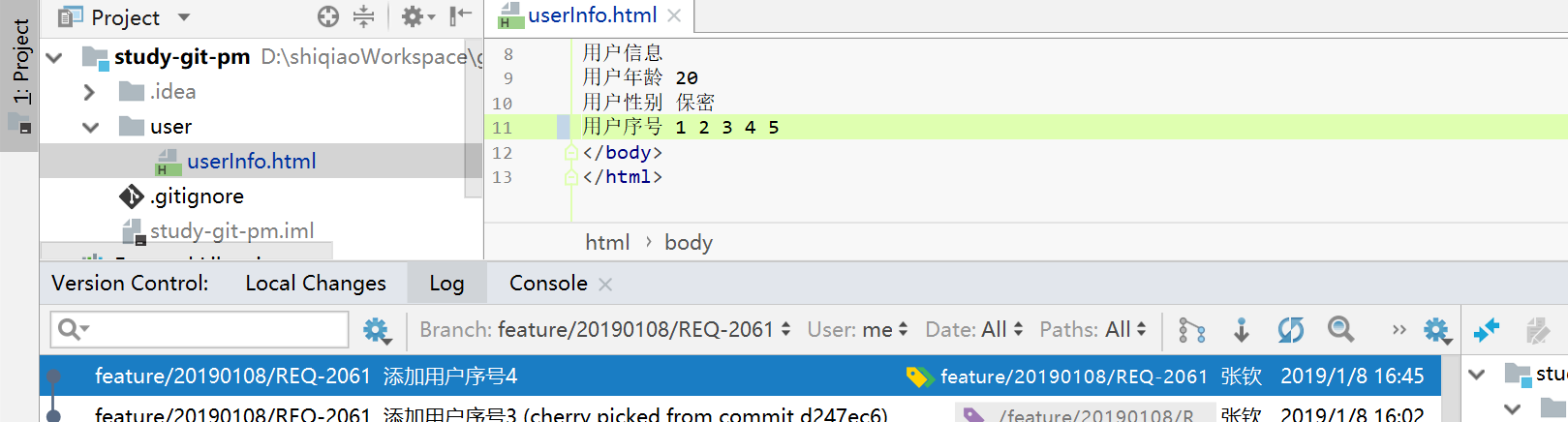
hard keep 则会删除你的代码（如果该版本有新的文件也会删除）一定要谨慎使用

执行hard结果如下图



你会发现提交记录没有了 你新增的5 没有了

然后我再演示下mixed

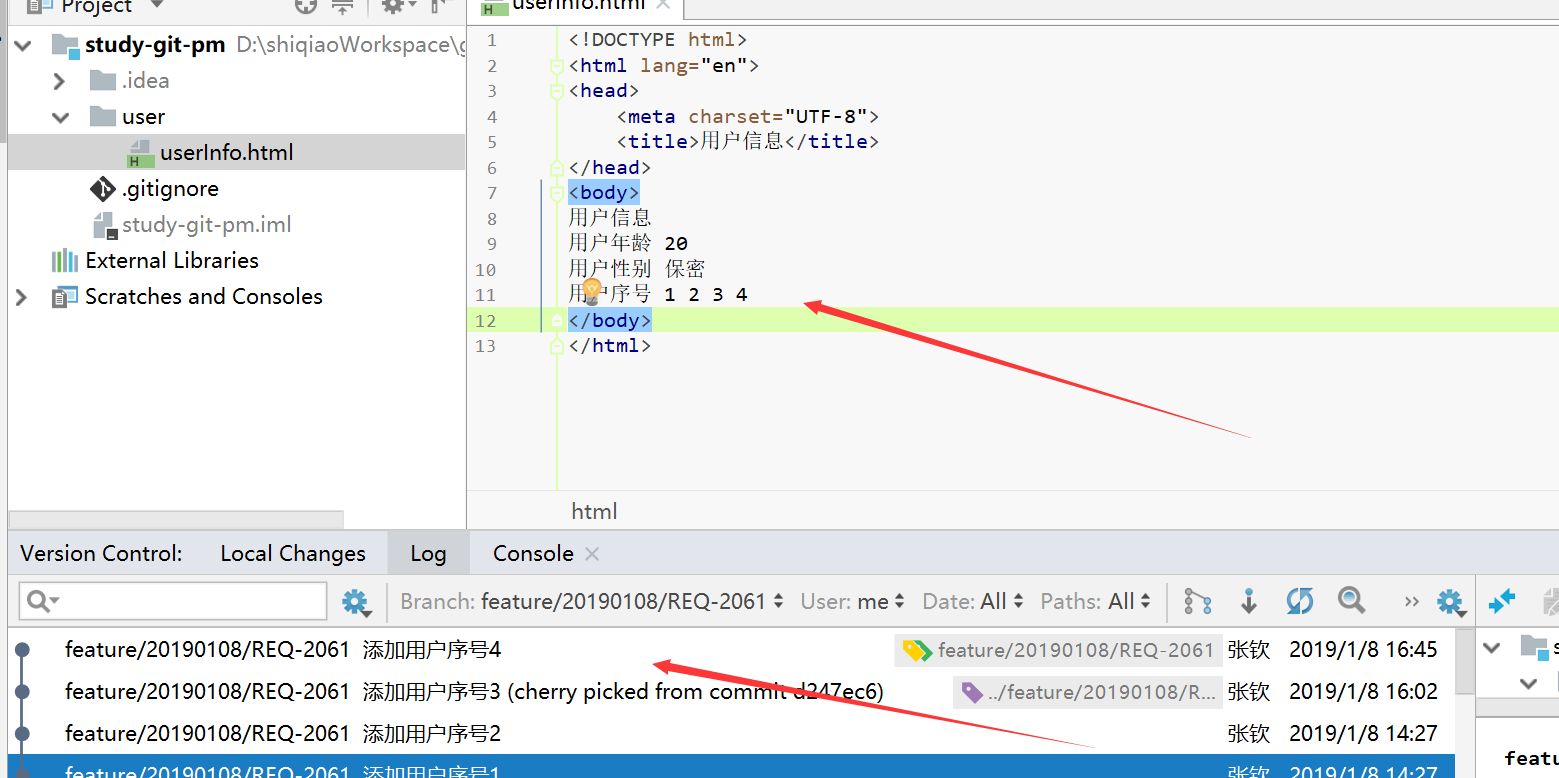


你会发现执行完5没有了但是 代码还在只是一个未commit的状态。具体区别可以自己实际操作下。

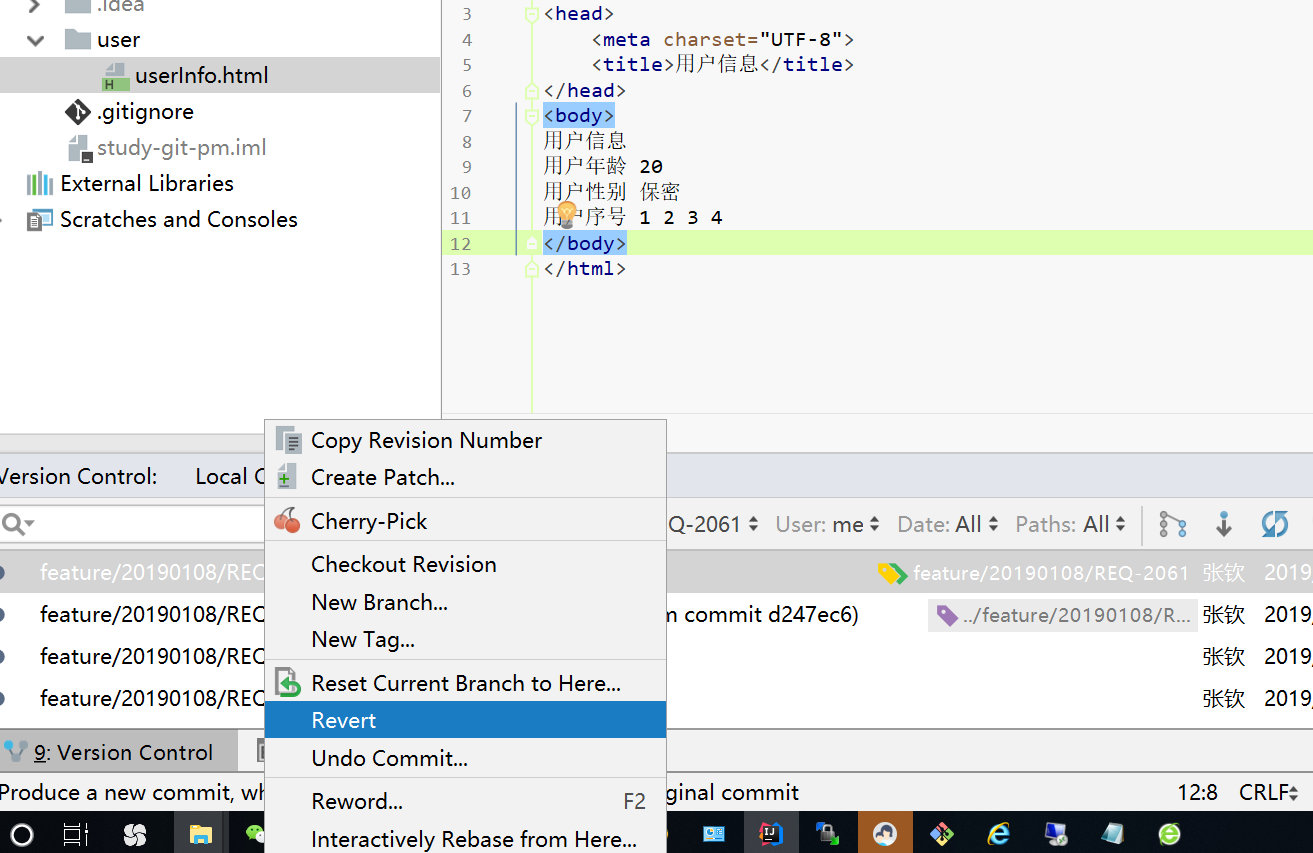
### Revert演示

如果你要撤回某个版本则可以用它

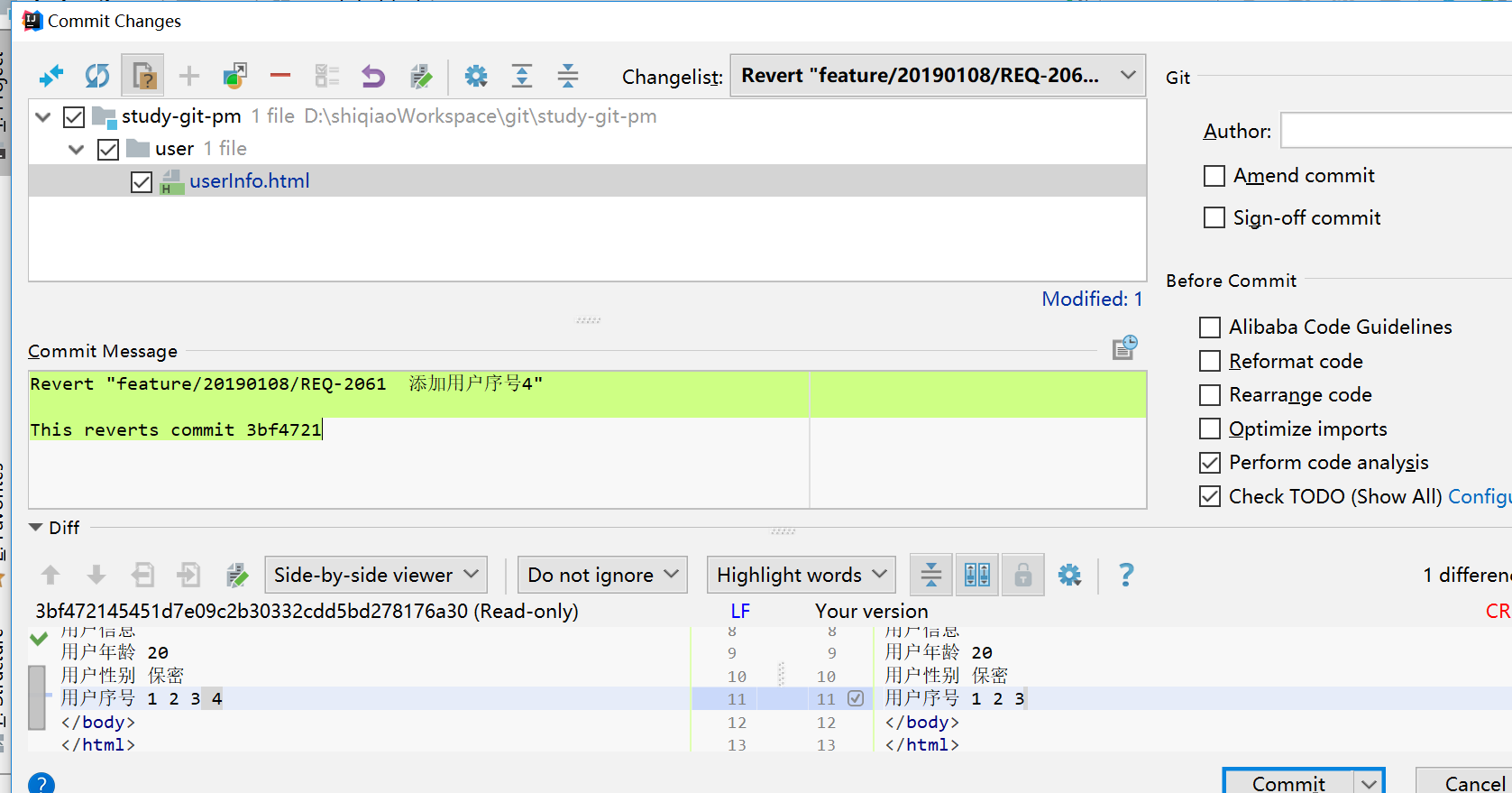
如图：我想将下图版本撤回



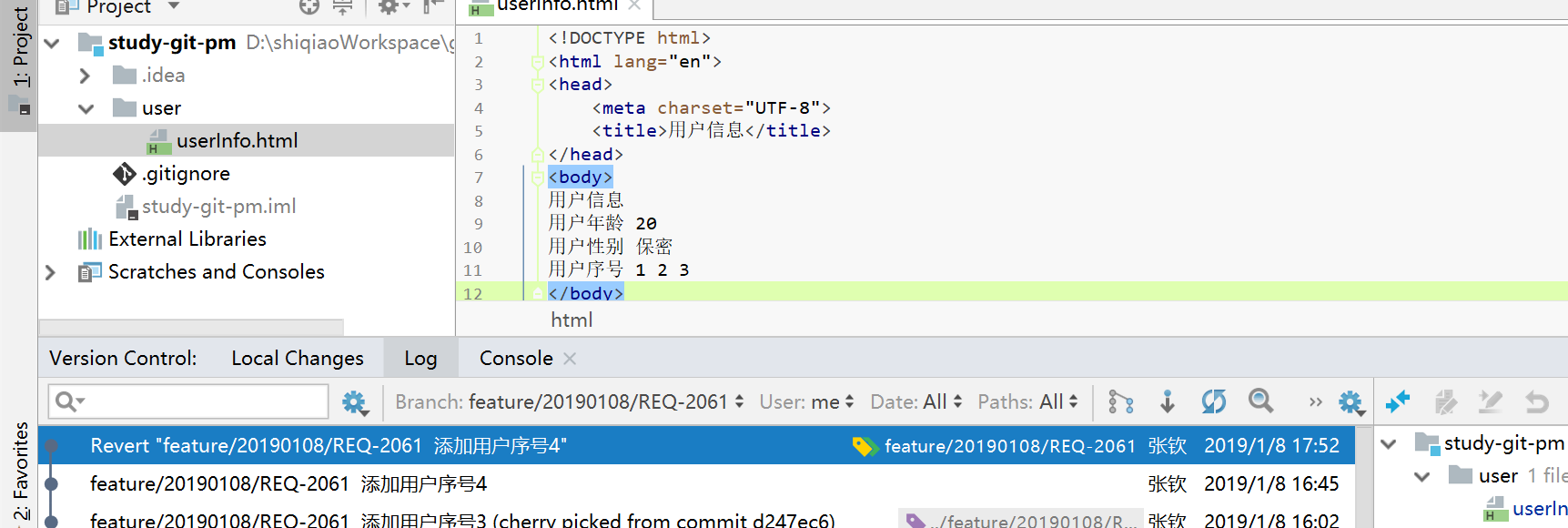
点击右键选择revert



然后会出现



你会看到新的代码用户序号里没有4 了，已经去掉了，然后commit看下提交记录

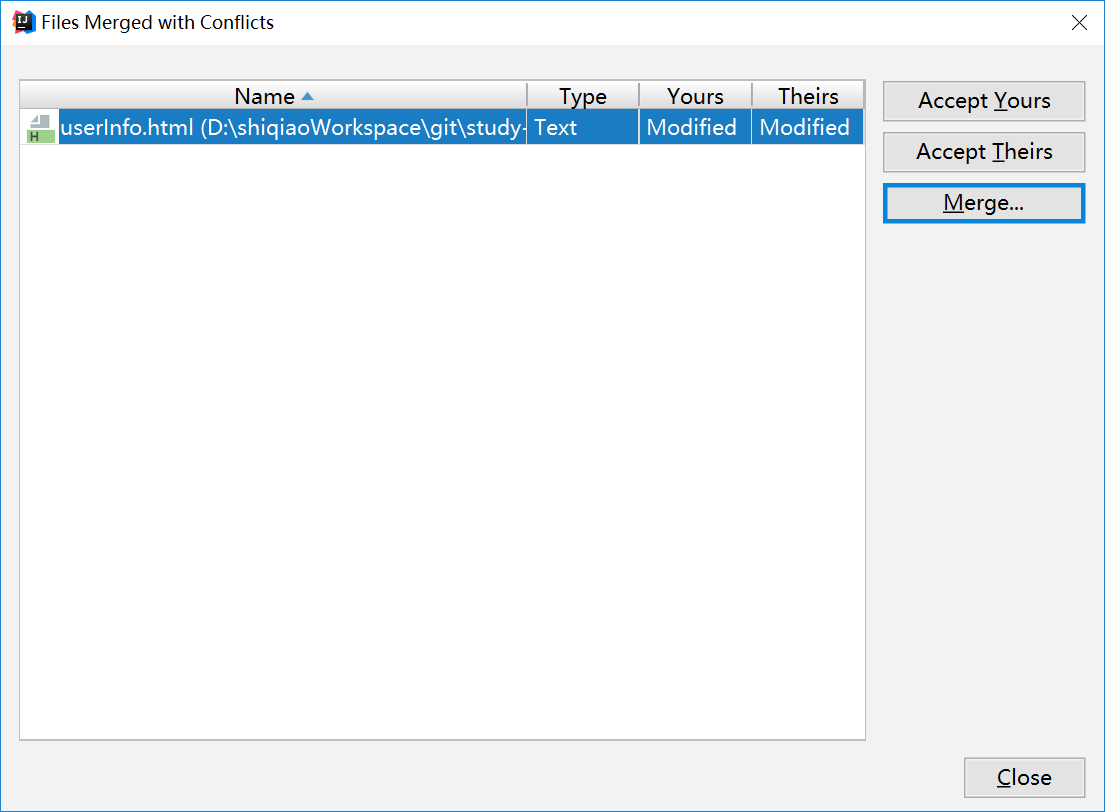


你会发现原来的版本也会存在，新的提交记录已经没有用户序号4了。

## 冲突解决展示

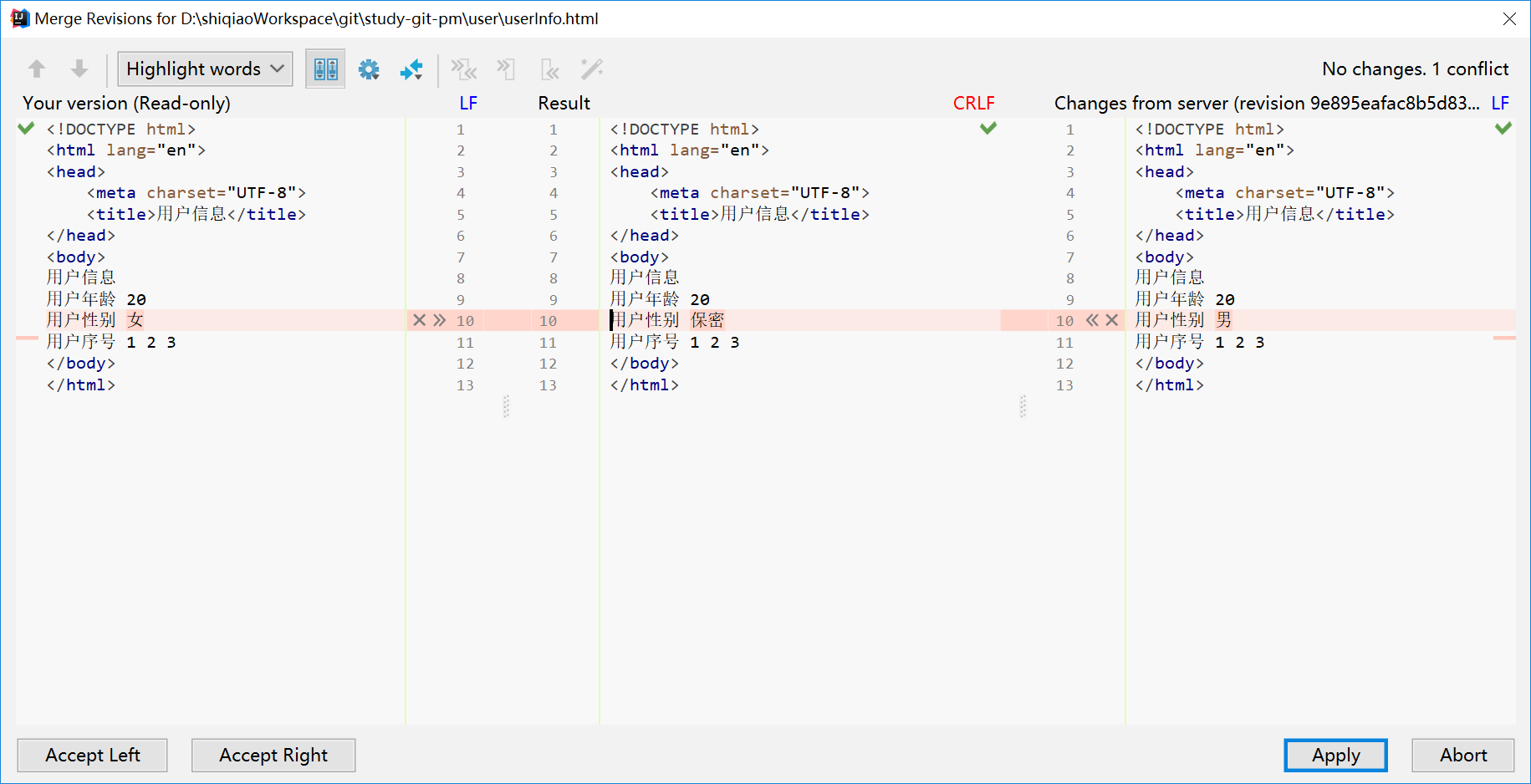
如果有冲突会产生冲突文件，需要自己去解决

Idea中情况如下



分别为接收你的版本接收远程仓库的版本以及自己去合并解决

一般情况都是需要自己去合并的 点击合并如下图



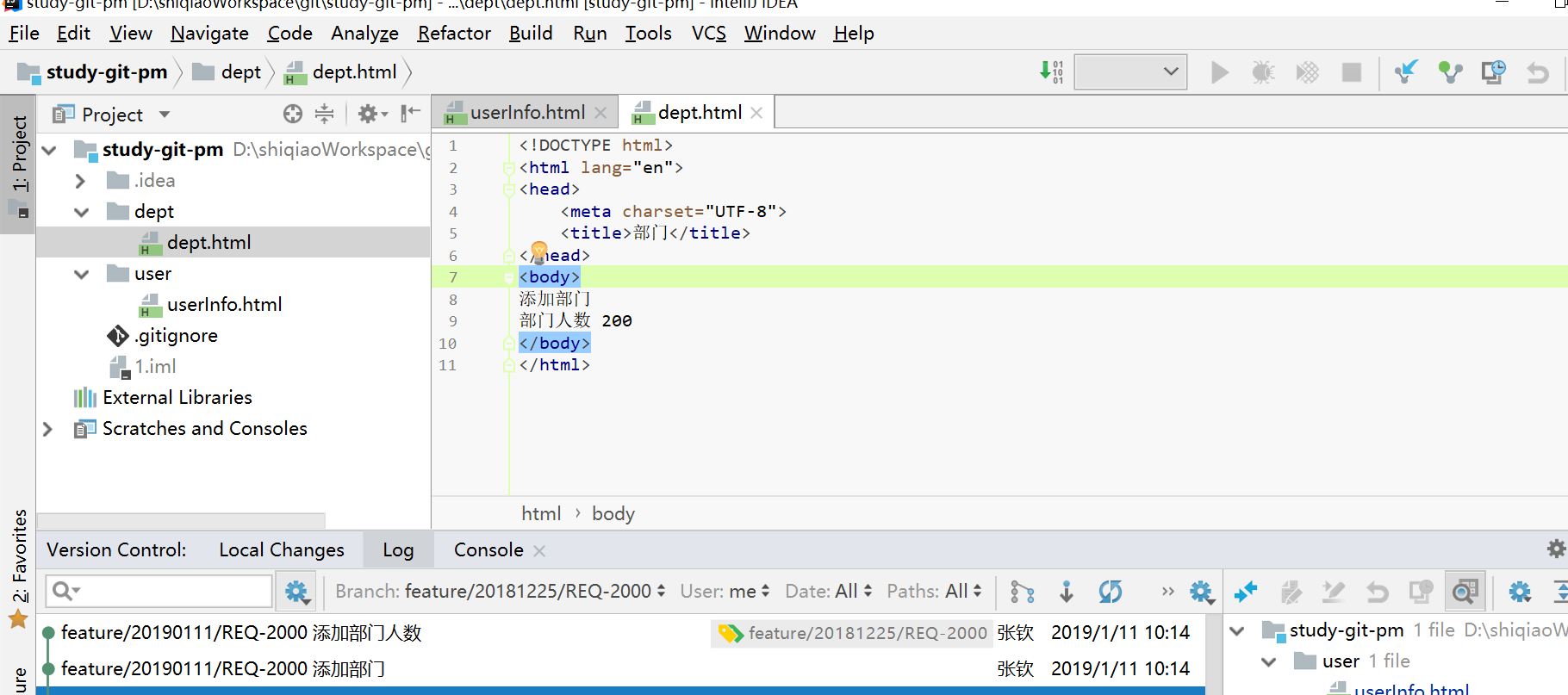
分别为你的版本 要使用的版本 以及远程服务器的版本

你可以根据实际情况去修改result代码然后点击应用，最后commint push到远程服务器

## 合并演示

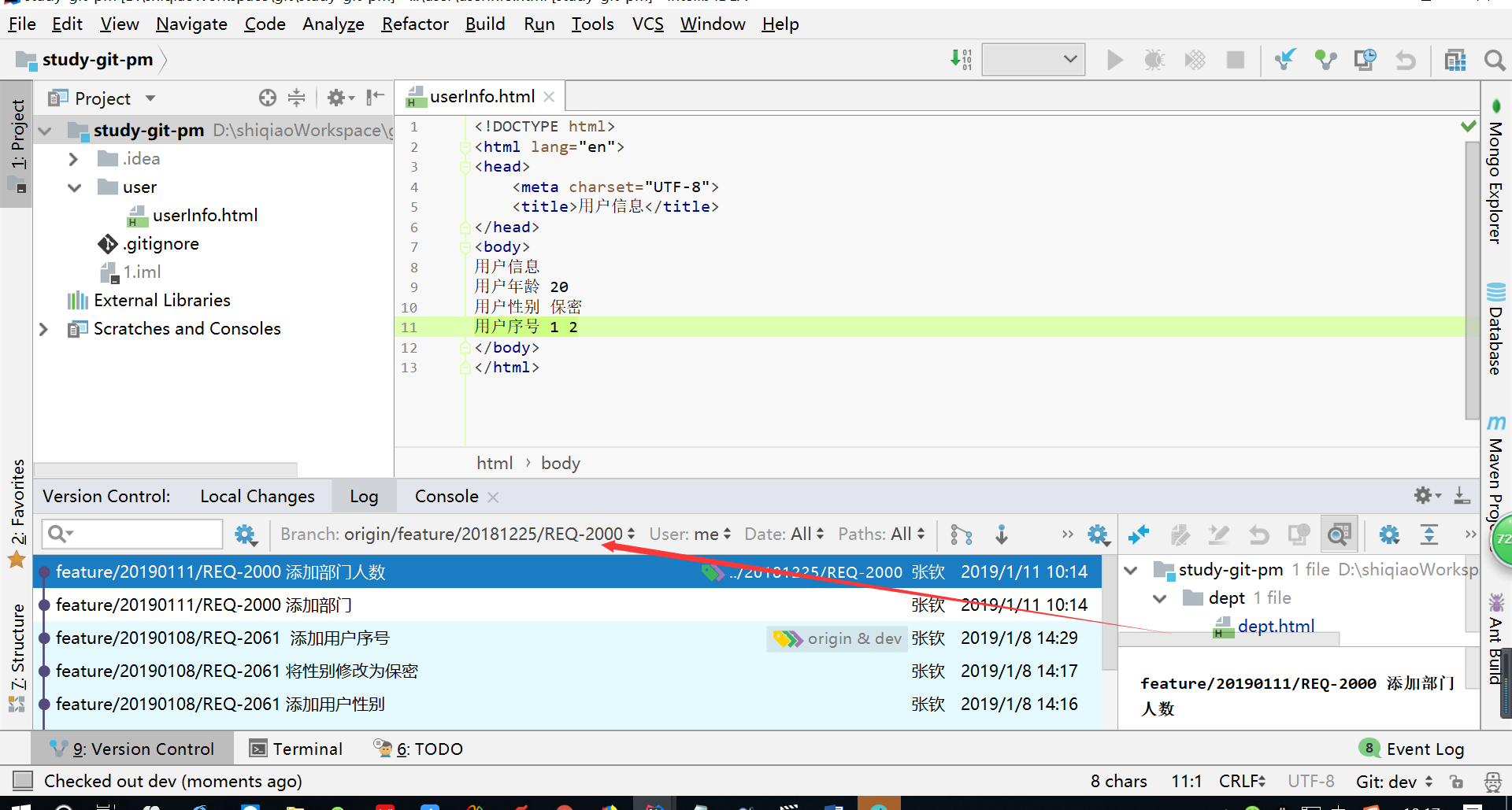
功能分支开发完成之后我们可以去合并到开发环境

比如我新建了个需求分支并开发完成有两次提交



我要将其合并开发分支首先得切换到开发分支

在log里面筛选你要合并的分支，建议如果有远程分支选择远程分支。

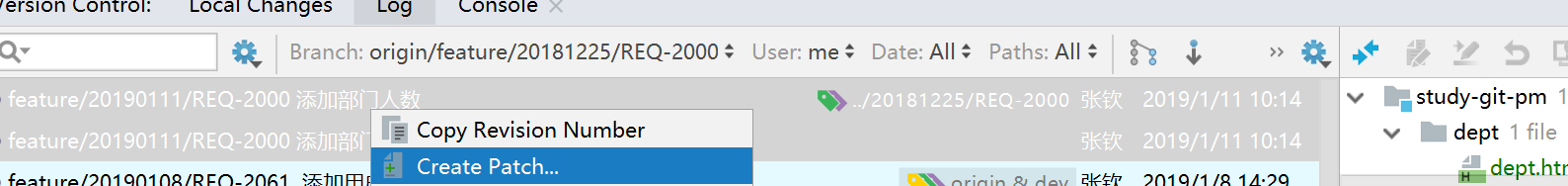


### 打补丁包合并（idea专属）

其他可以自行参考

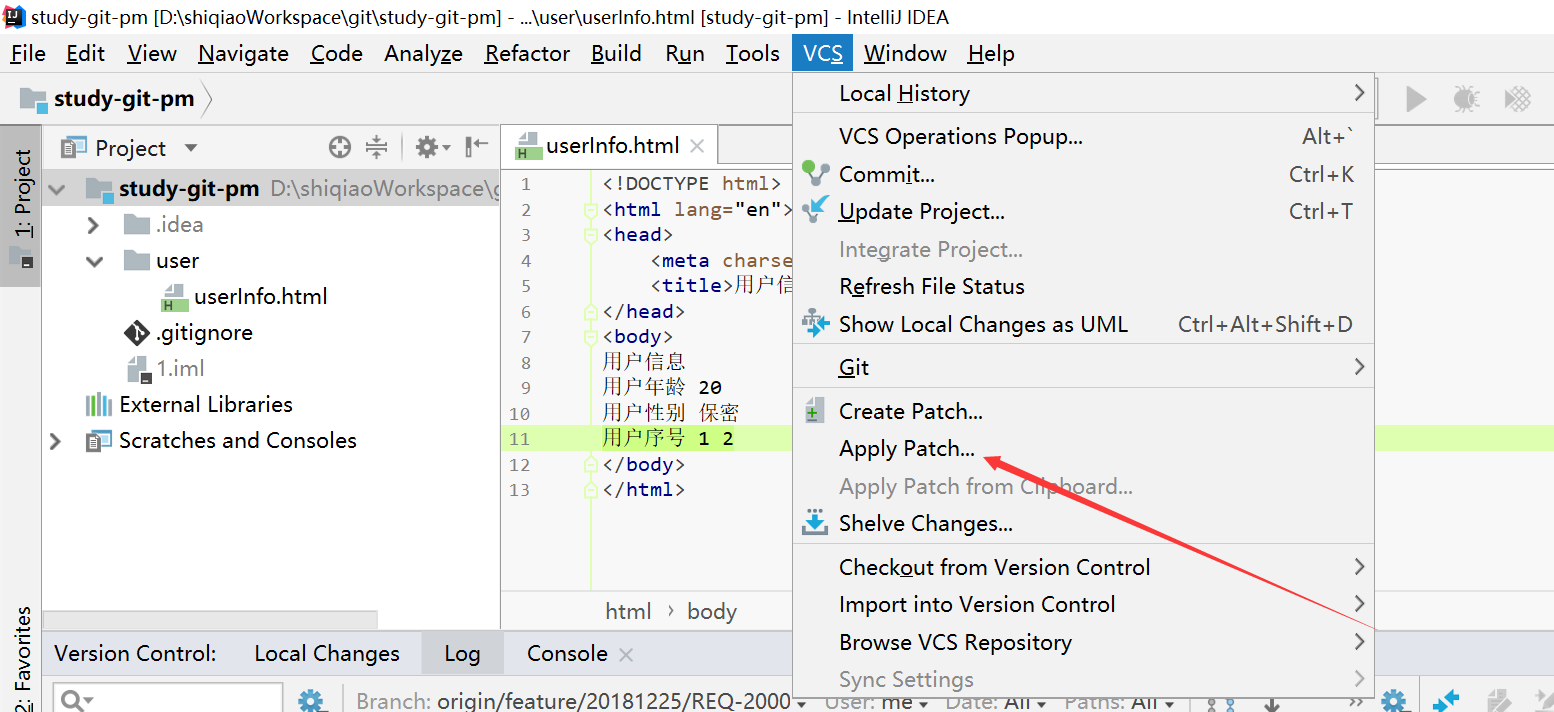
如果是打包合并

选择你要打包的版本，可以多选然后右键选择打包如图：



之后会生成个打包文件

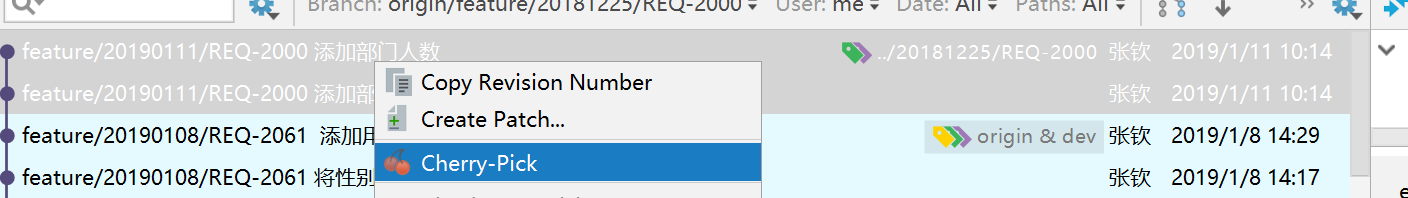
然后在



应用你打好包的文件就可以了，然后执行commit push就可以

### 保留提交记录合并（推荐）

使用荔枝如下图



选择多个版本cherry-pick

然后commit push