大纲 git-scm.com

除了git分布式版本管理工具，还有SVN集中式管理工具

工作区---->暂存区---->本地库

**git基础命令**

**git和代码托管中心**

局域网环境下

* GitLab服务器

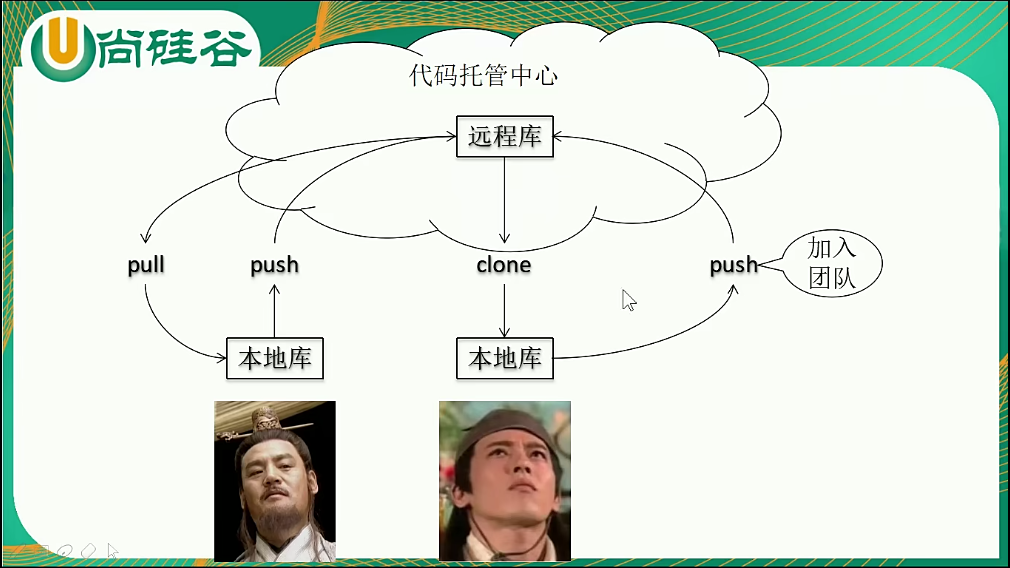
外网环境下

* GitHub
* 码云

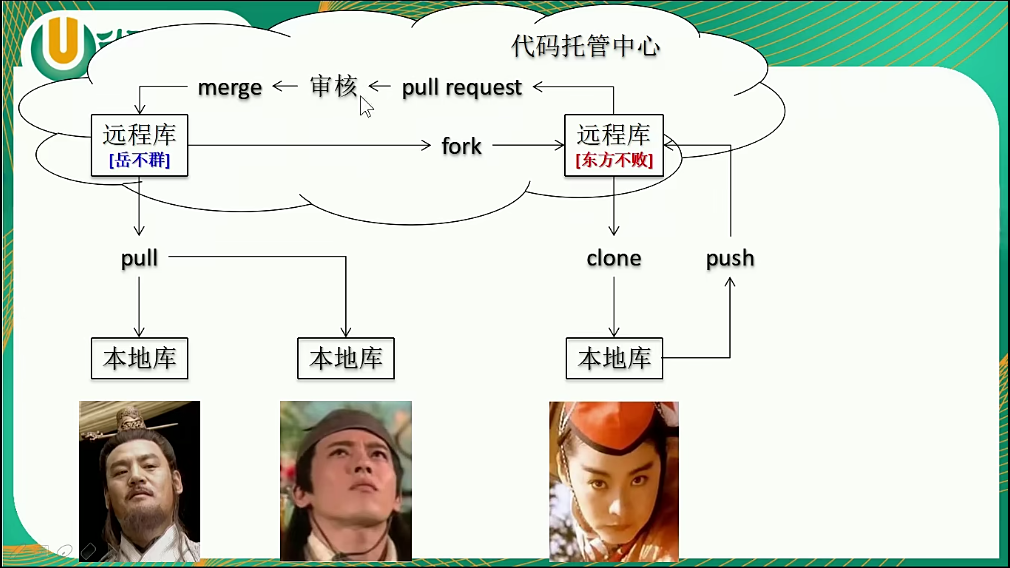
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**本地库和远程库的交互方式**

团队内协作



团队外协作



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**本地库初始化 git init**

就是创建.git 文件

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**设置签名**

用户名：Tom

Email地址：123456@qq,com

作用：区分不同开发人员的身份

辨析：这里设置的签名和登录远程仓库的账号、密码没有任何关系

命令：

* 项目级别/仓库级别：仅在当前本地库范围有效
  + git **config** user.name Tom\_pro
  + git **config** user.email [goodMoring\_pro123456@qq.com](mailto:goodMoring_pro123456@qq.com)
  + 信息保存位置：.git/config 文件



* 系统用户级别：登录当前操作系统的用户范围
  + git **config --global** user.name Tom\_pro
  + git **config --global** user.email [goodMoring\_pro123456@qq.com](mailto:goodMoring_pro123456@qq.com)
  + 信息保存位置：~/.gitconfig



* 级别优先级：
* 就近原则：项目级别优先系统用户级别，二者都有时采用项目级别
* 如果只有系统级别，就系统级别
* 不允许二者都没有

通常实际情况

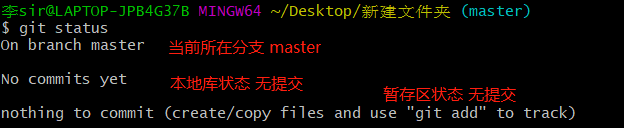
只设置系统级别就够了，所有项目都用息系统级别

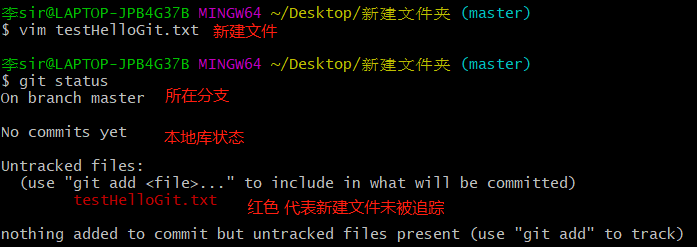
如果某个项目单独需要在设置

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**添加提交以及查看状态操作**

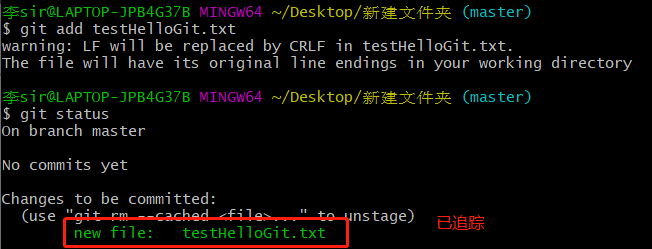
git status 查看状态



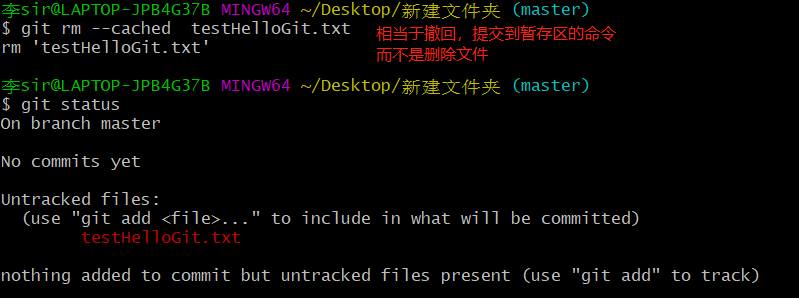


**git add filenames 追踪文件**

* **红色代表为追踪**
* **绿色代表已追踪**



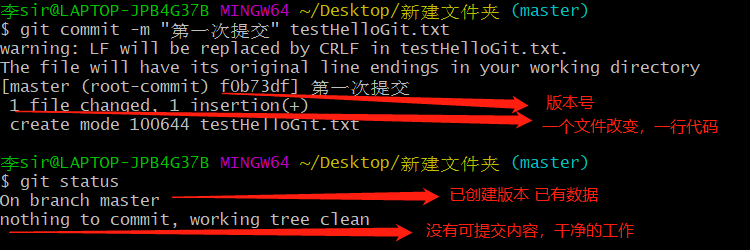
git rm --cached filenames 撤回提交追踪文件



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

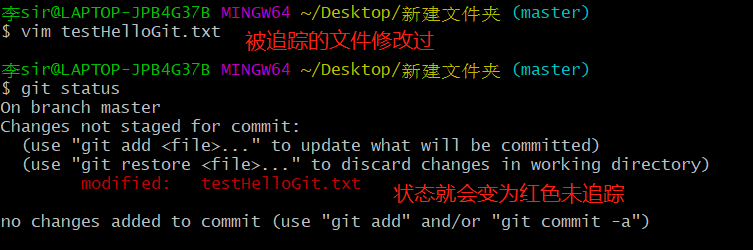
**提交到本地库**

git commit -m “日志信息” filenames 提交文件到本地库



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**修改文件**

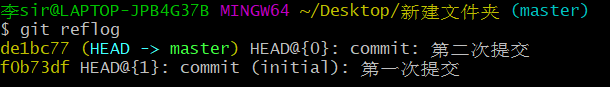


----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**历史版本**

基本语法

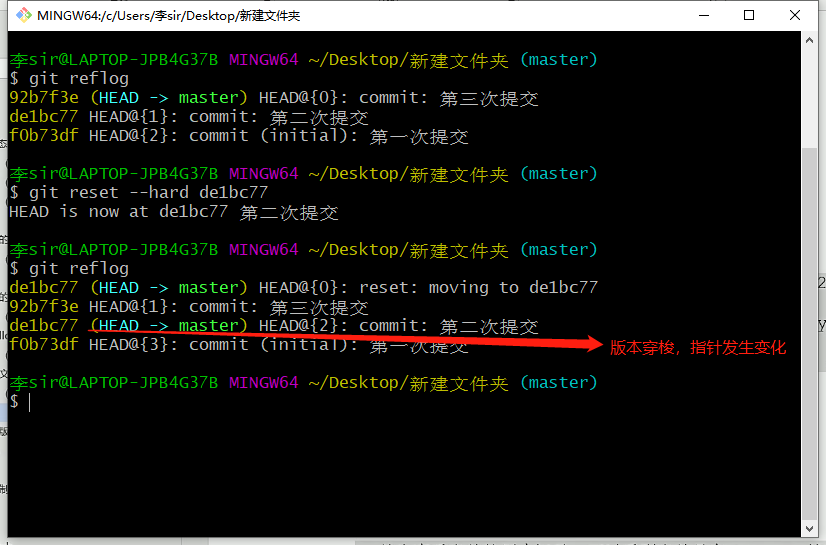
git reflog 查看精简版本信息



git log 查看详细版本信息



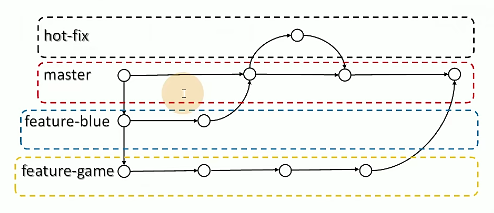
git reset --hard 版本号 版本穿梭



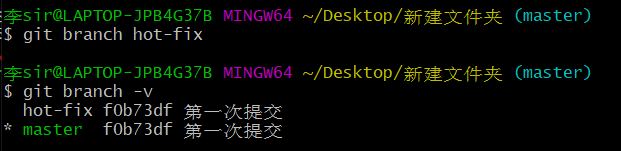
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

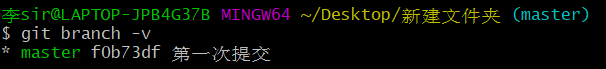
**git分支操作**



创建分支 ：git branch 分支名

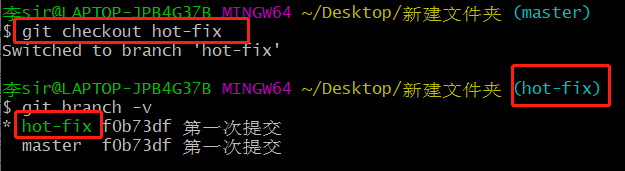


查看分支 ：git branch -v



切换分支 ：git checkout 分支名

拉取的那个主分支版本 也不要搞错（就是要用的那个版本，新建分支）



修改过后一样，git add gitcommit -m

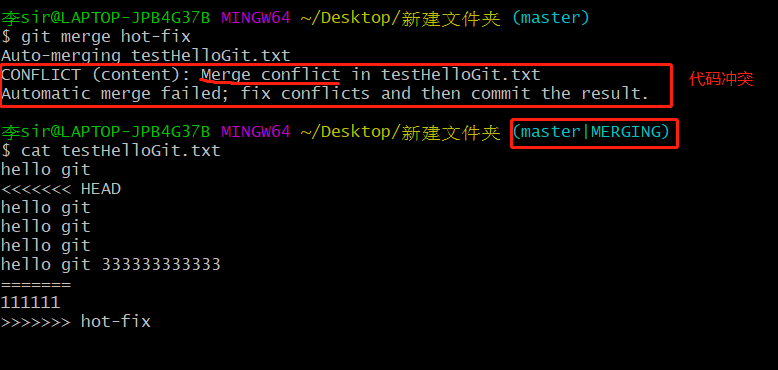
版本信息中指针会变成分支，显示分支的历史版本

把指定分支合并到当前分支上 ：git merge 分支名（指定）

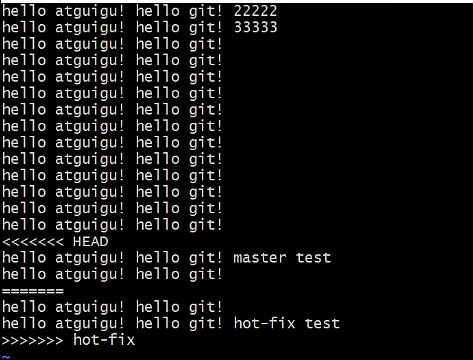
要合并B，先切换回A，B会覆盖A

合并冲突

冲突的原因：两个分支在同一个文件的同一个位置，有两个不同的改动，需要人为决定新代码的内容

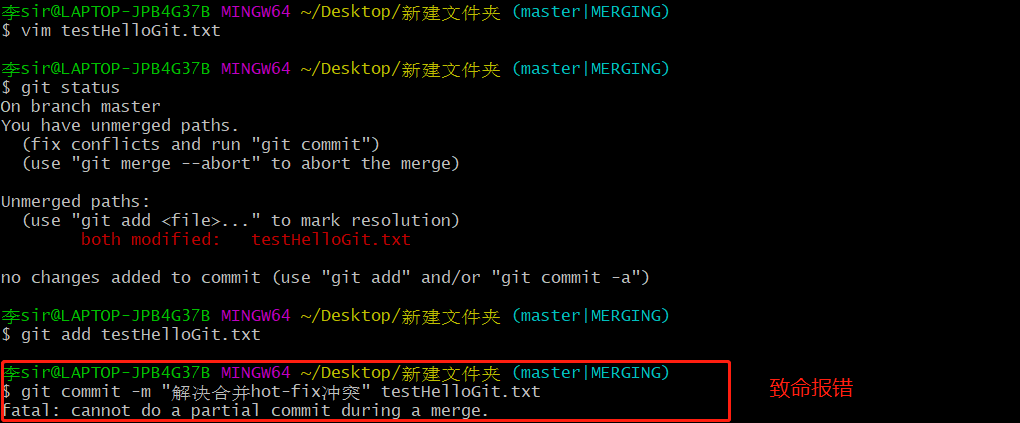


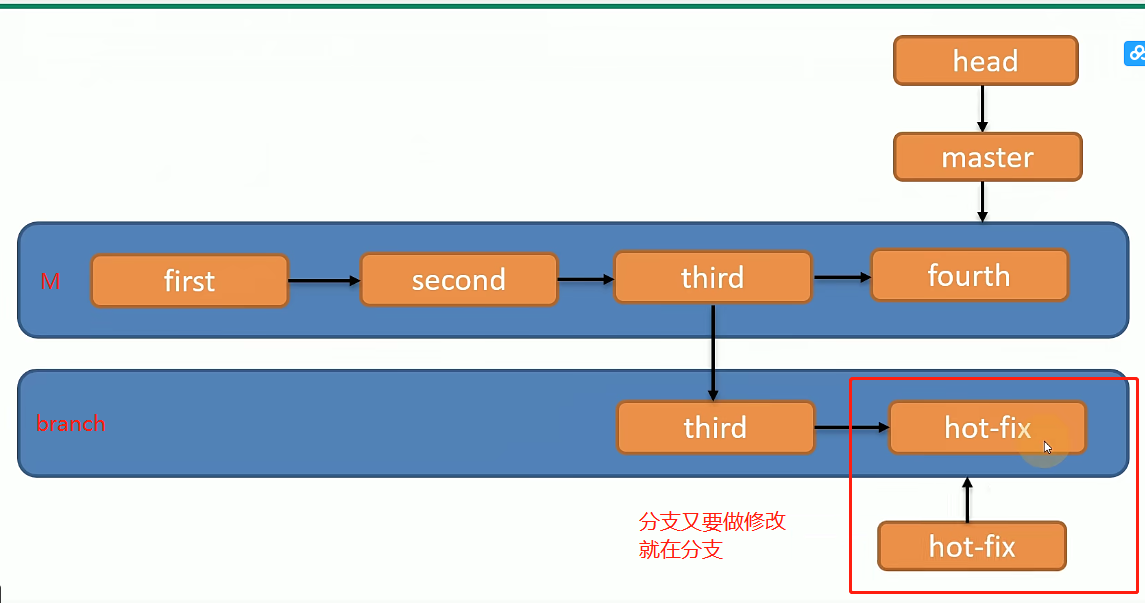
人为修改代码



注意：

* 要手动人为解决冲突
* 解决冲突代码之后，要add commit
* 但是commit 不能加文件名不然会报错





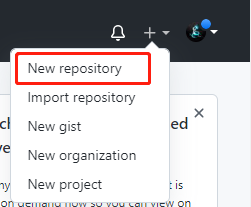
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**GitHub操作**

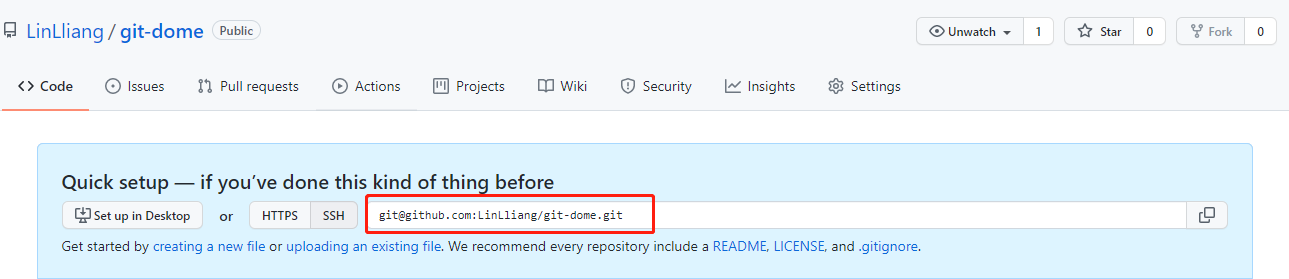
创建远程仓库





创建别名 remote

取别名，就是为了方便记住链接



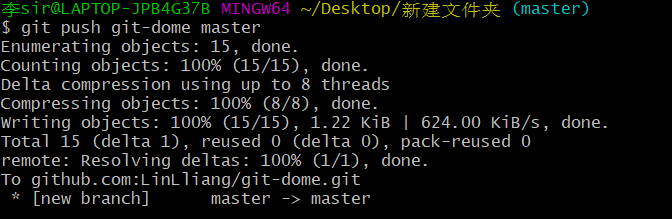
git remote -v 查看当前所有别名

git remote add 别名 远程库链接



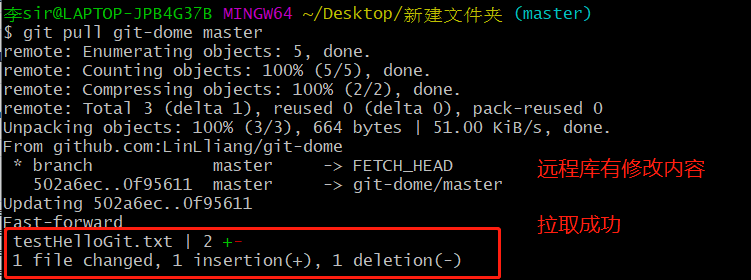
推送本地库至远程库 push

git push 别名(链接) 分支



拉取远程库至本地库 pull

git pull 别名（链接） 分支

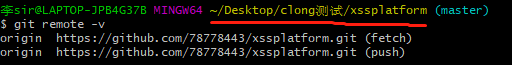


克隆远程仓库 clone

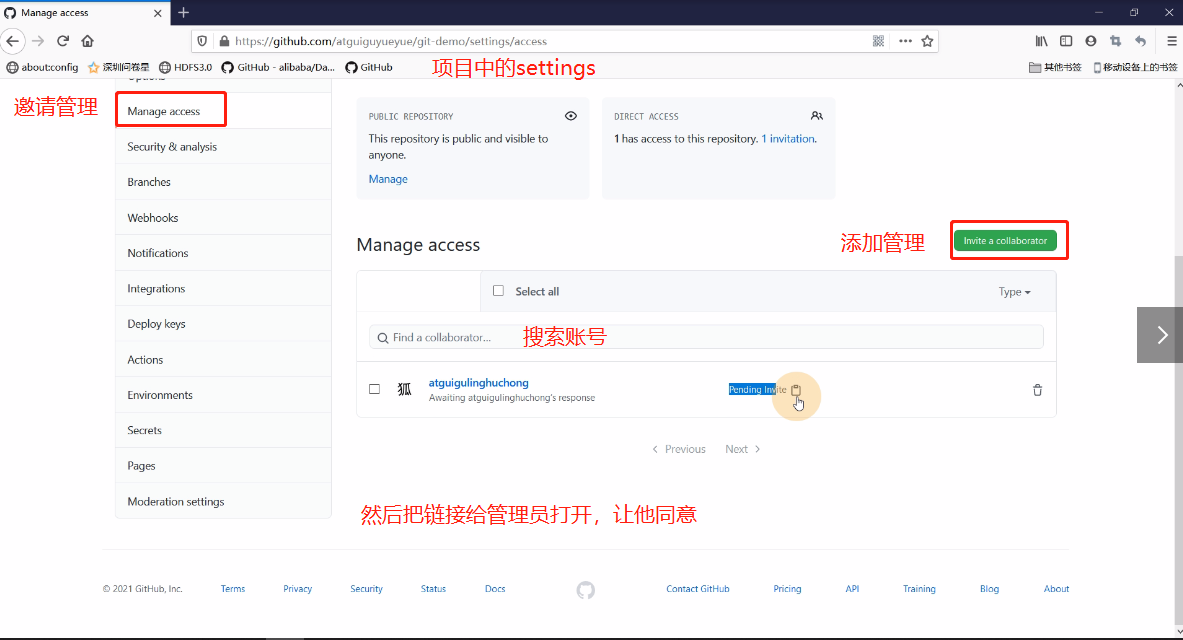
git clone 链接

clone完成了三件事

1. 拉取代码
2. 初始化本地仓库
3. 创建别名



团队协作添加管理成员



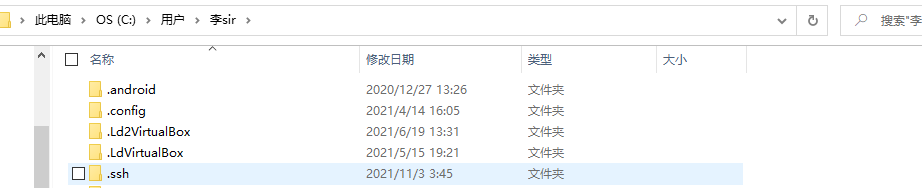
跨团队协作



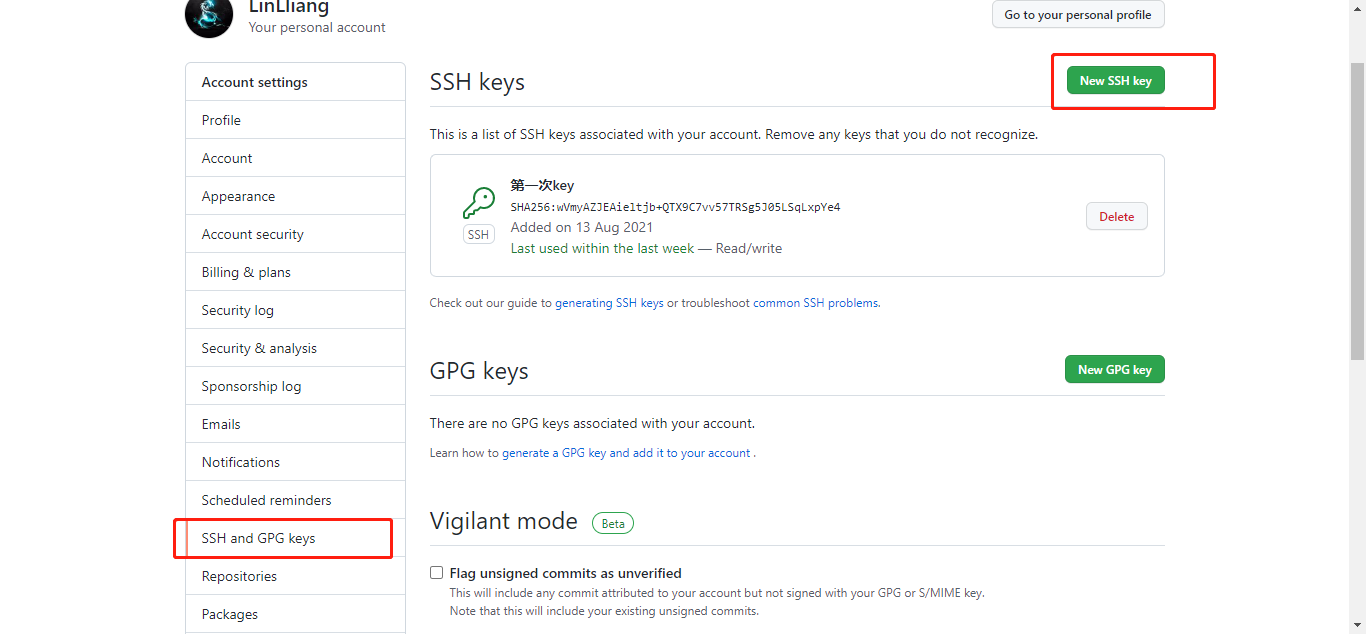
ssh免密登录设置

ssh-keygen -t rsa -C git登录邮箱

生成秘钥 存在C盘 Windows 用户 .ssh文件里面



GitHub里面添加公钥



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**IDEA集成Git**

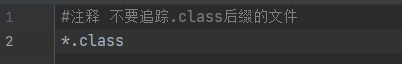
配置忽略文件 .ignore （也可以配置在项目里面）

xxx.ignore 放在家目录下面

内容规则

#注释 后缀class的不要

.class



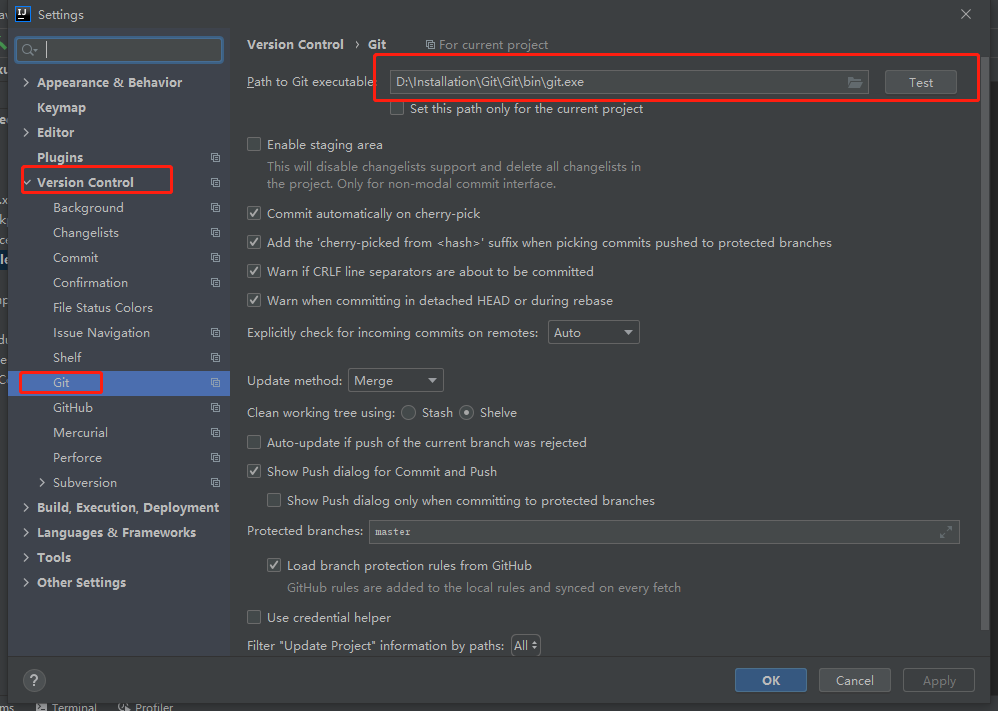
.gitconfig 配置引入文件

[core]  
 excludesfile = C:/Users/李sir/git.ignore

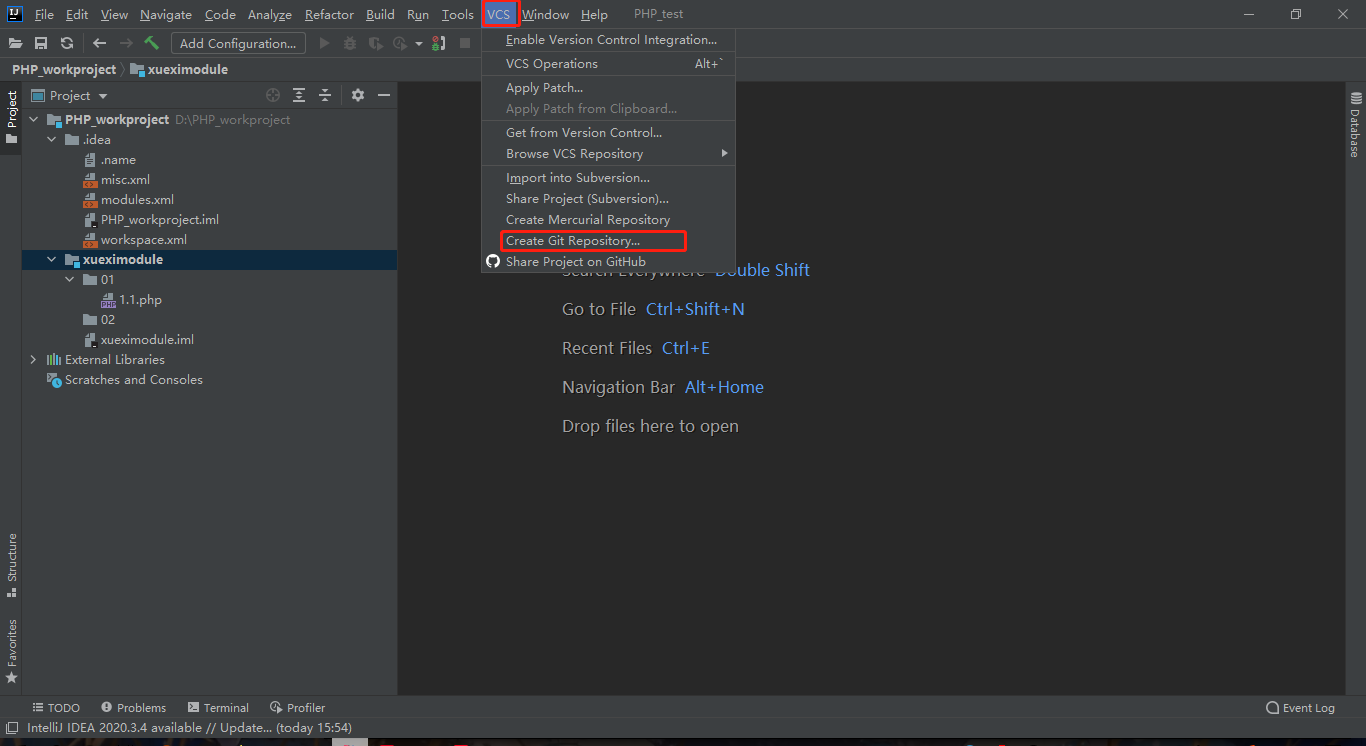
注意：

路径和反斜线

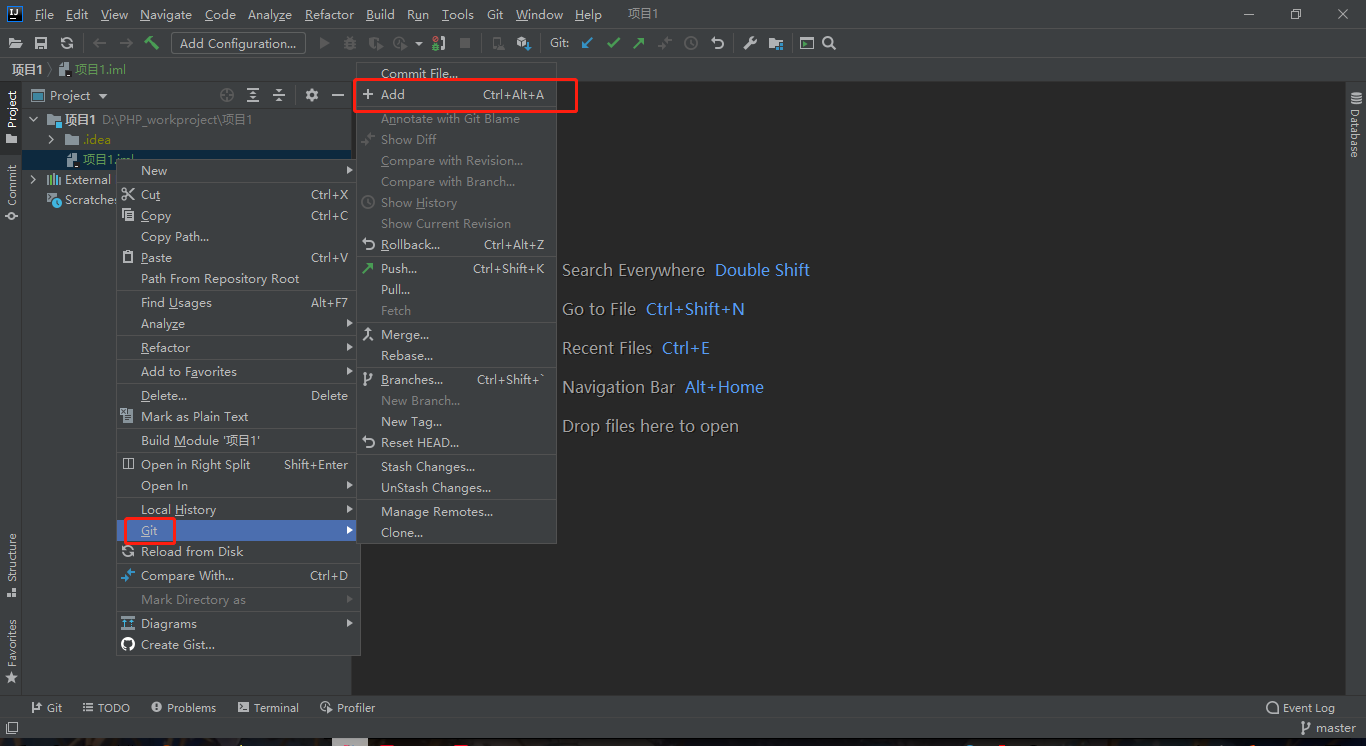
定位Git程序

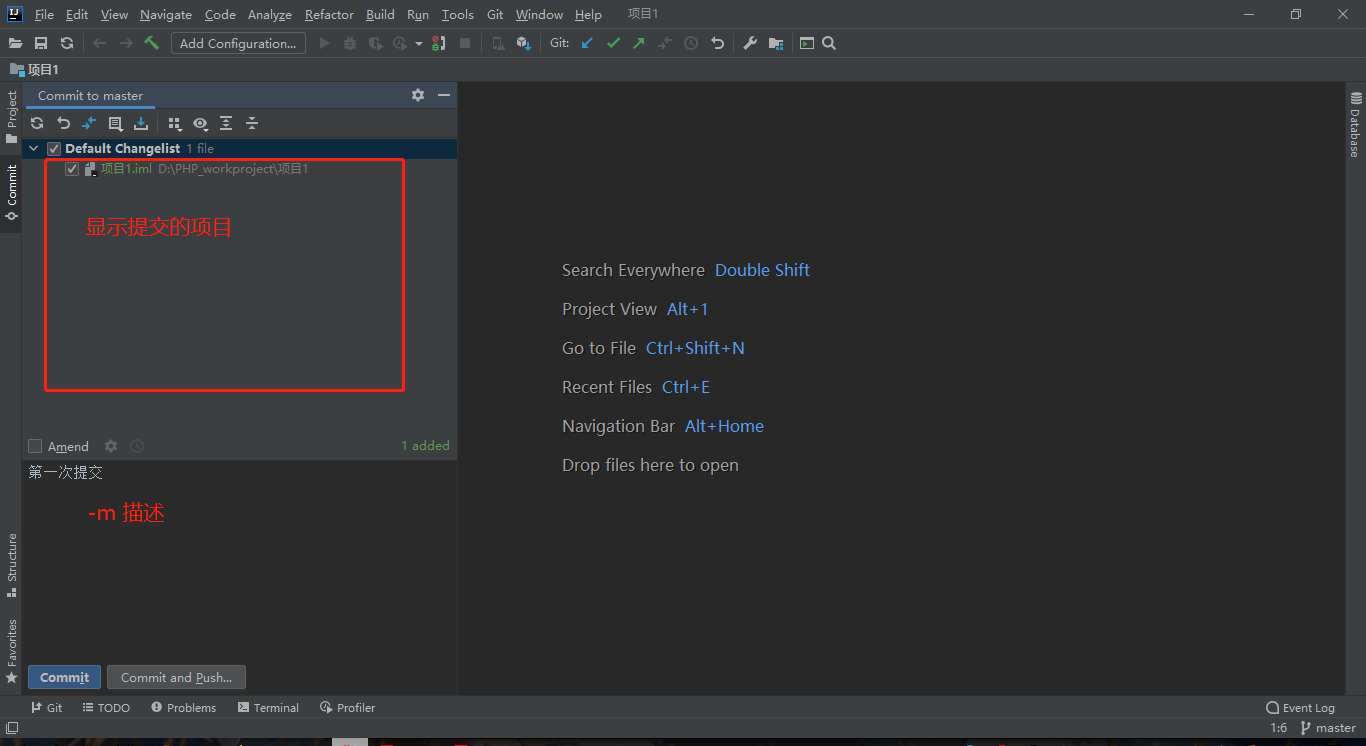


IDEA初始化本地库

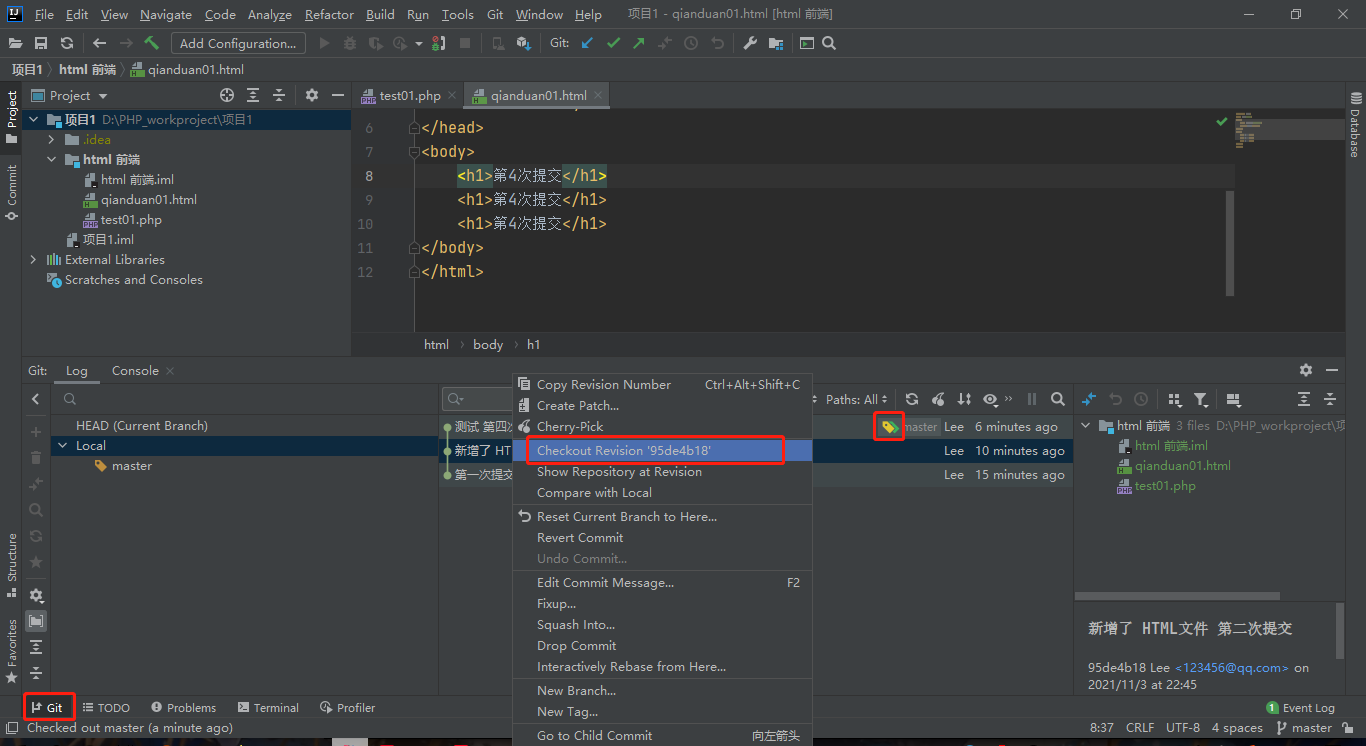


IDEA集成 git add commit



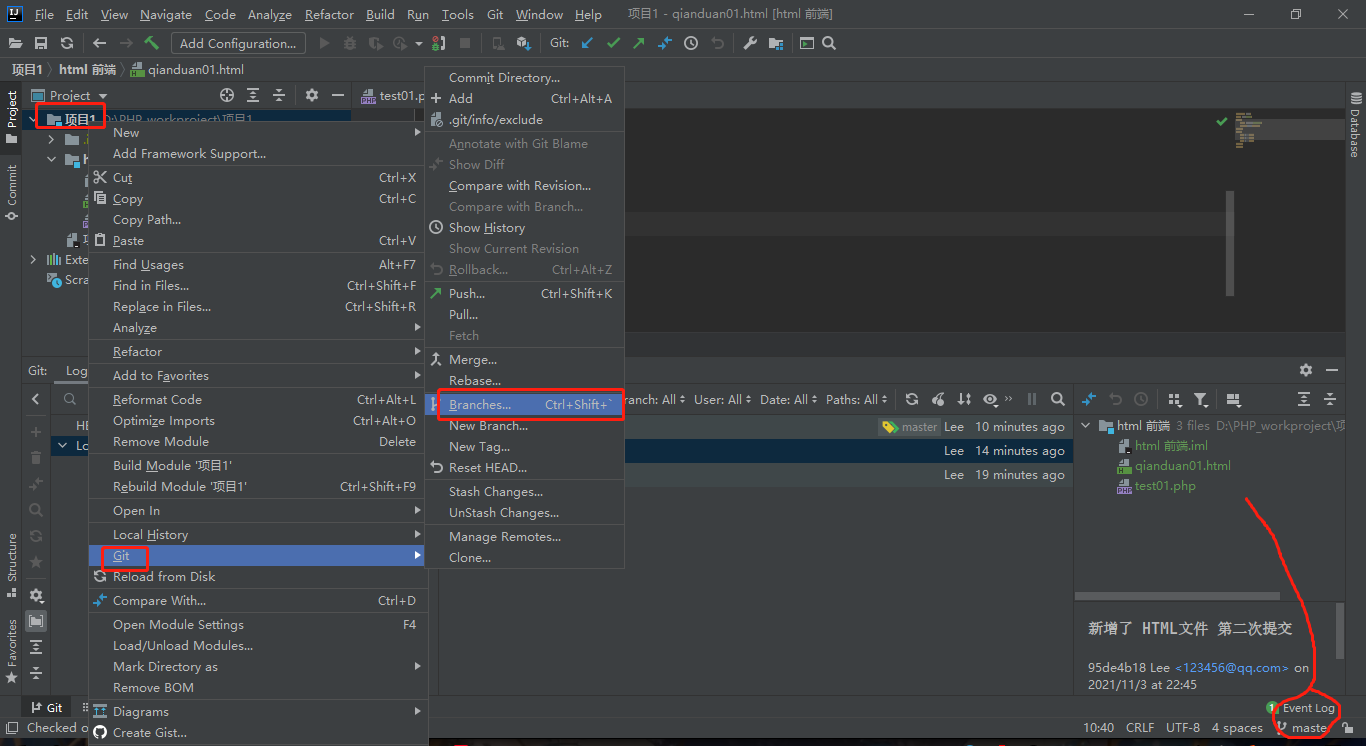


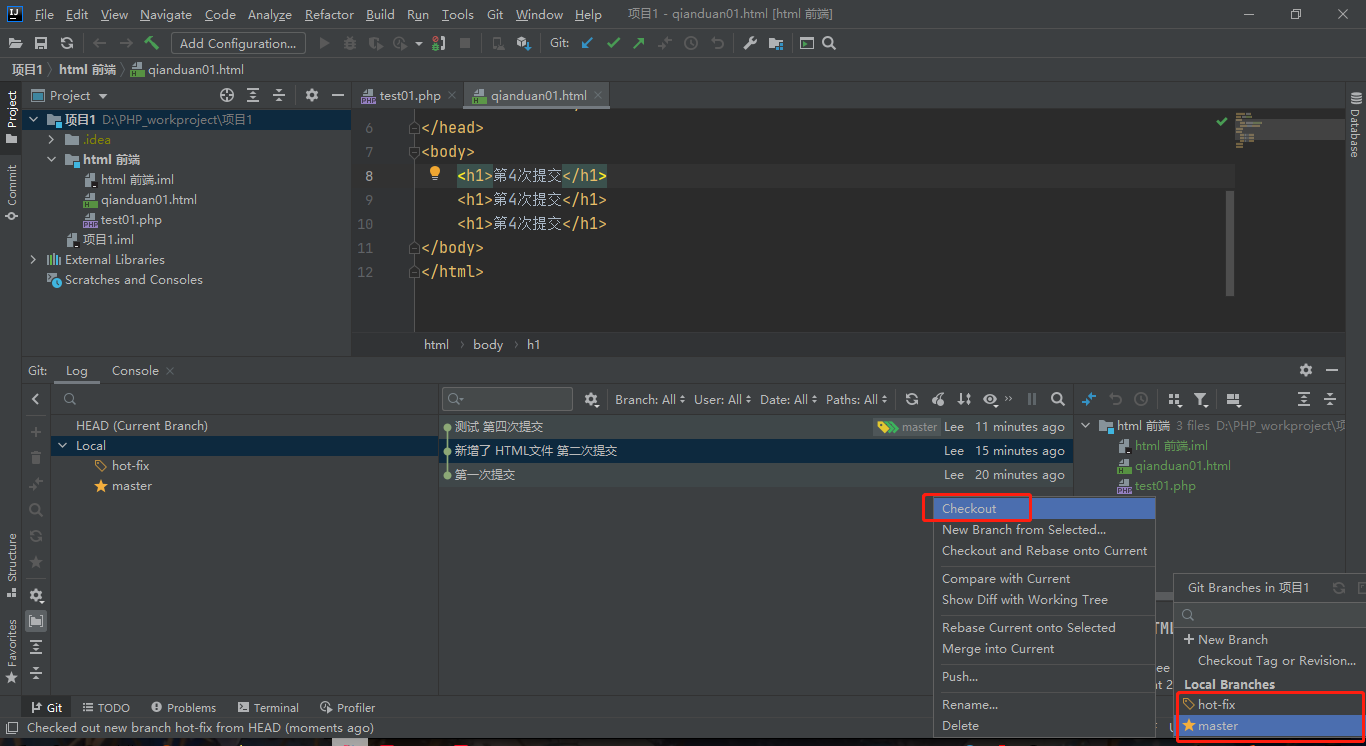
IDEA集成 版本穿梭



IDEA集成 创建分支 切换分支

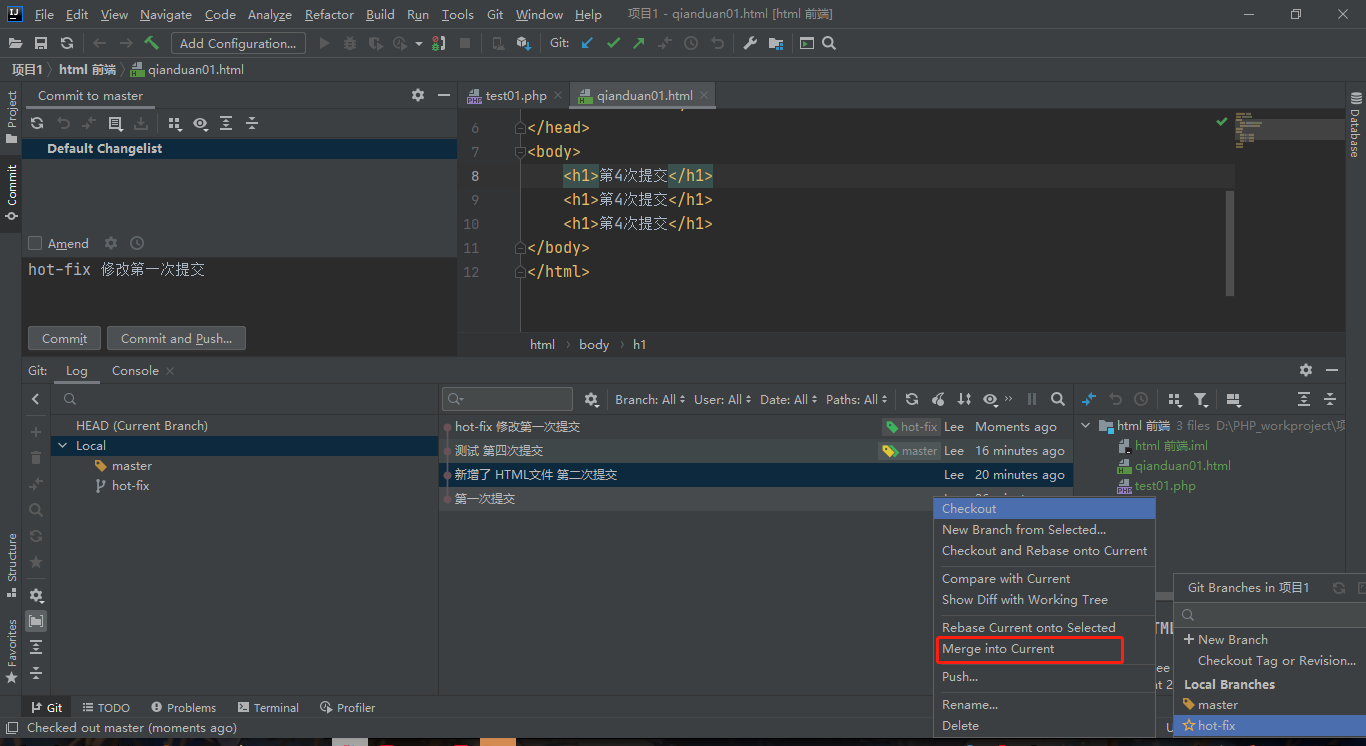
两种方式



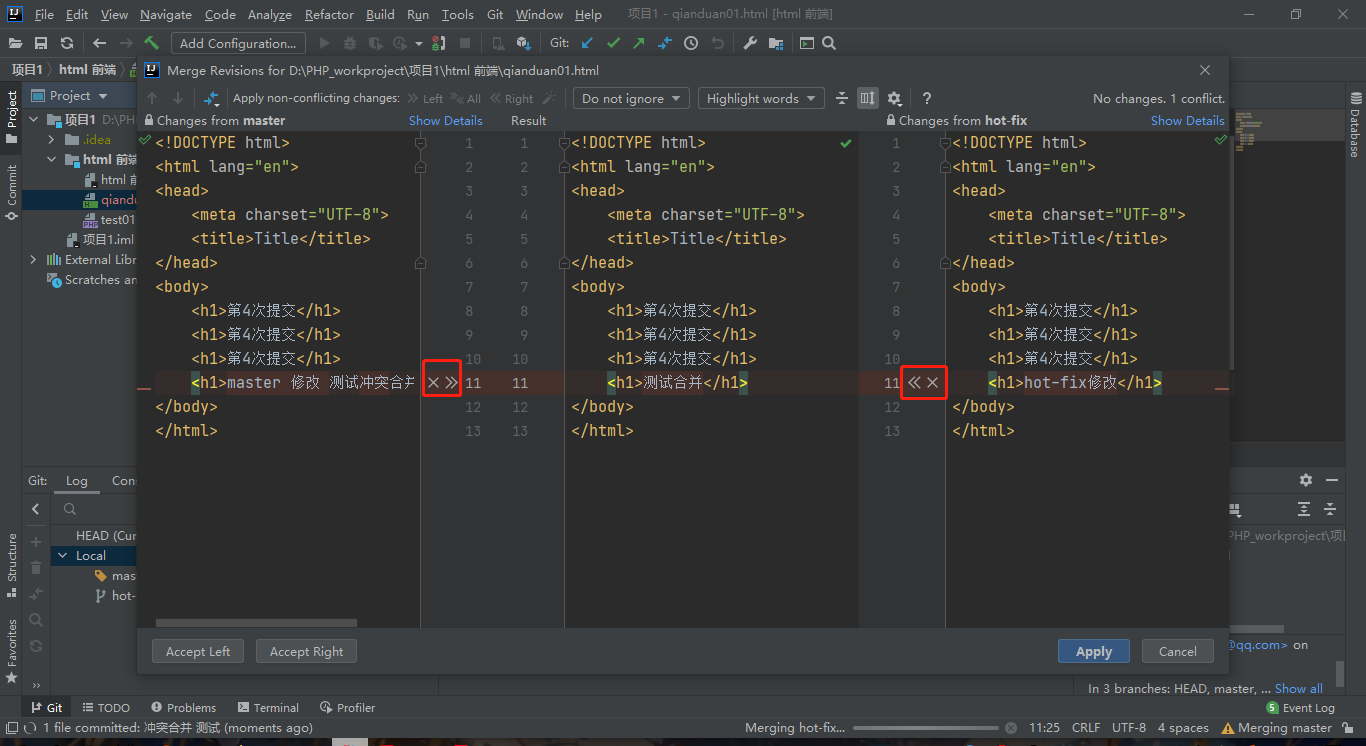


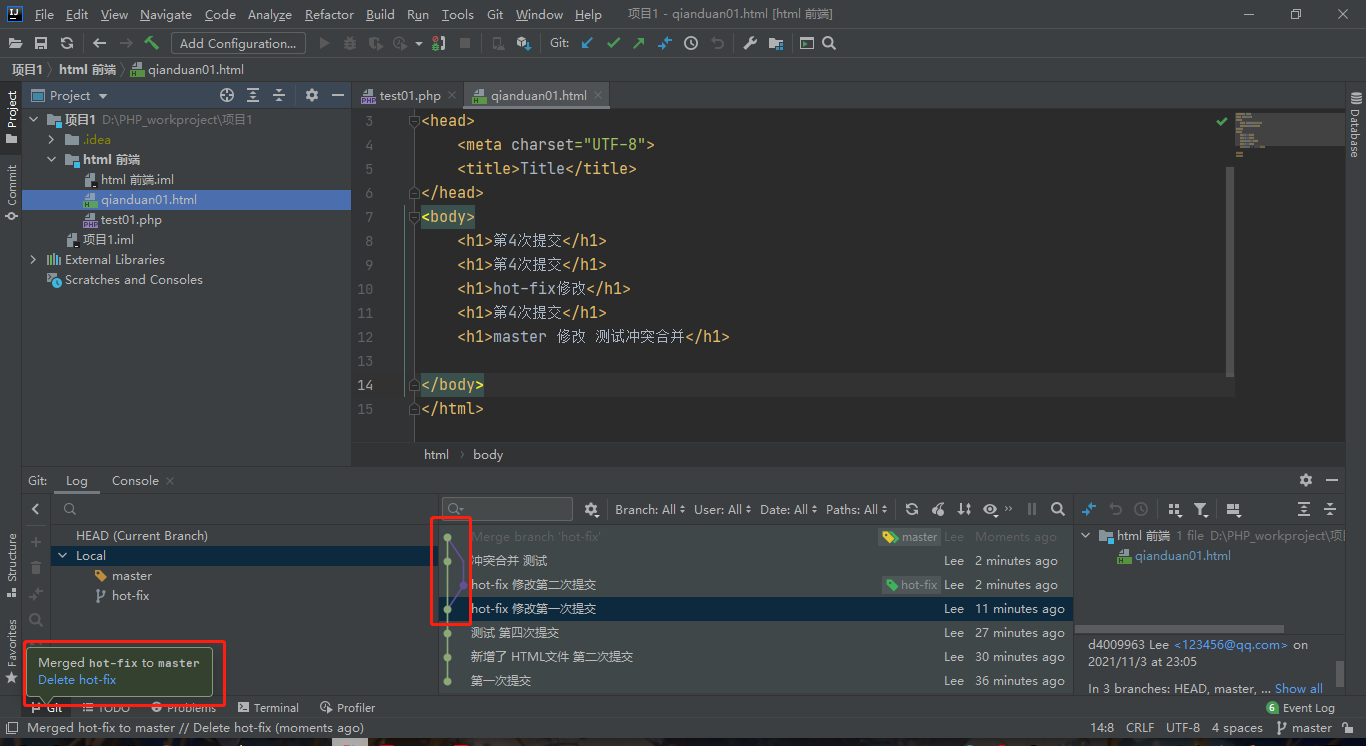
IDEA集成git 合并 合并冲突

分支还是要先提交commit

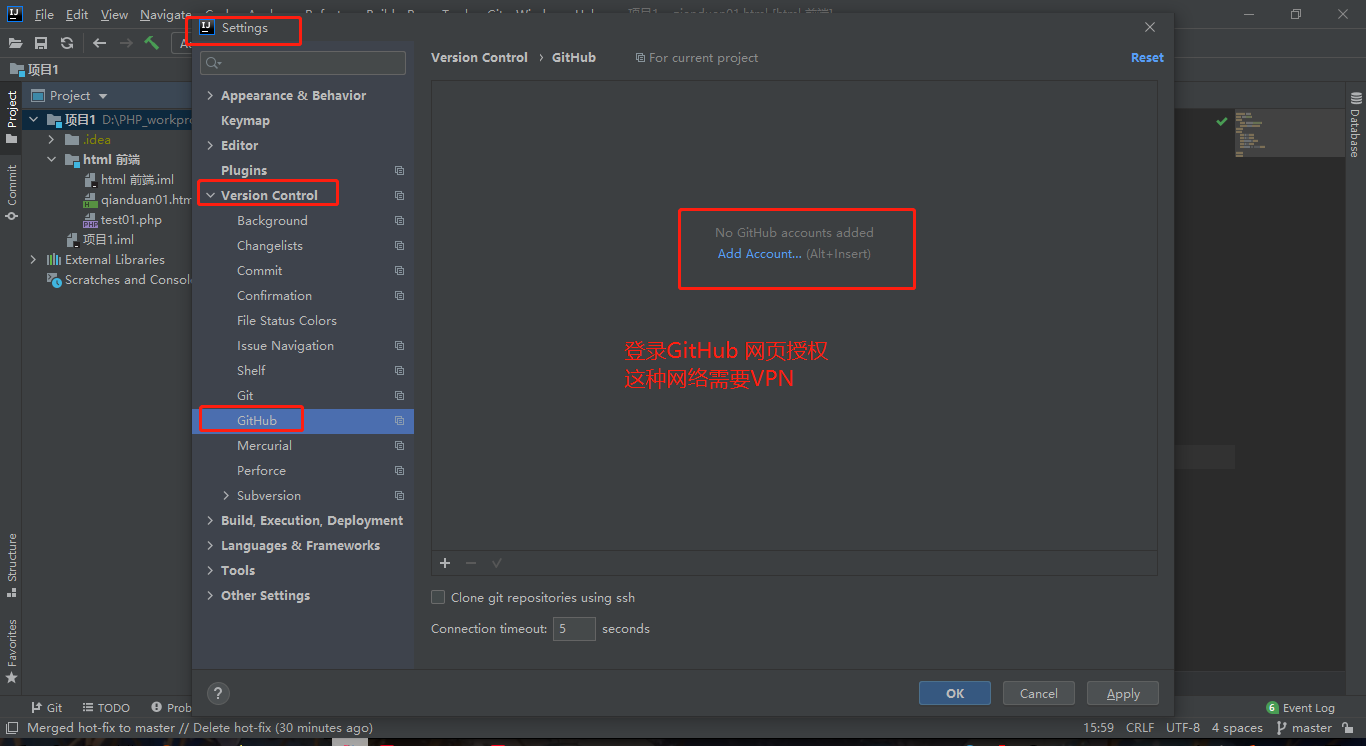


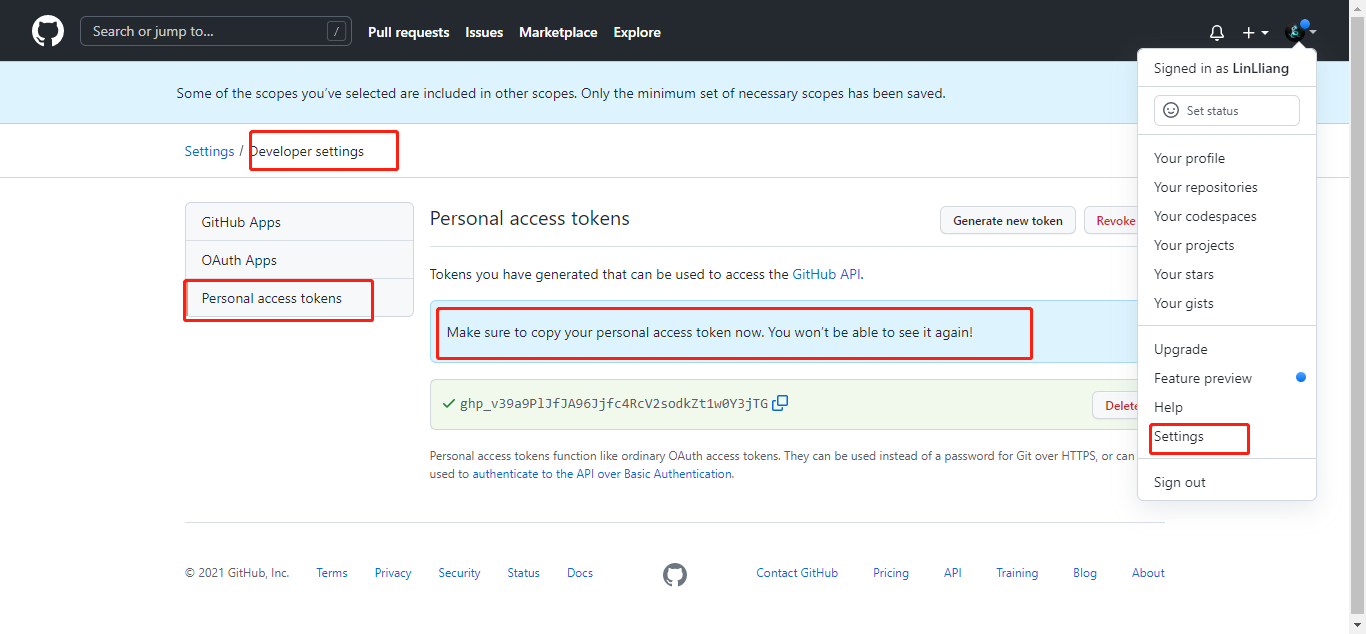
冲突



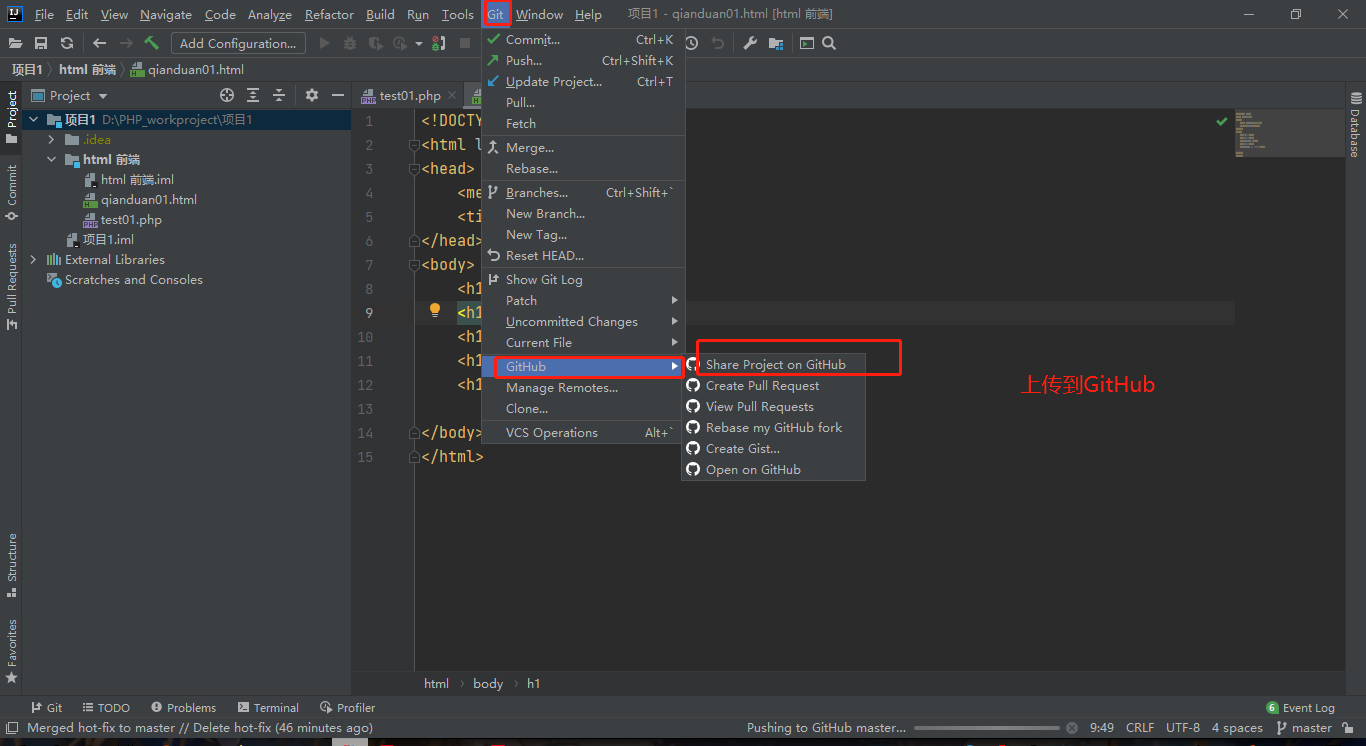


IDEA集成 设置GitHub账号

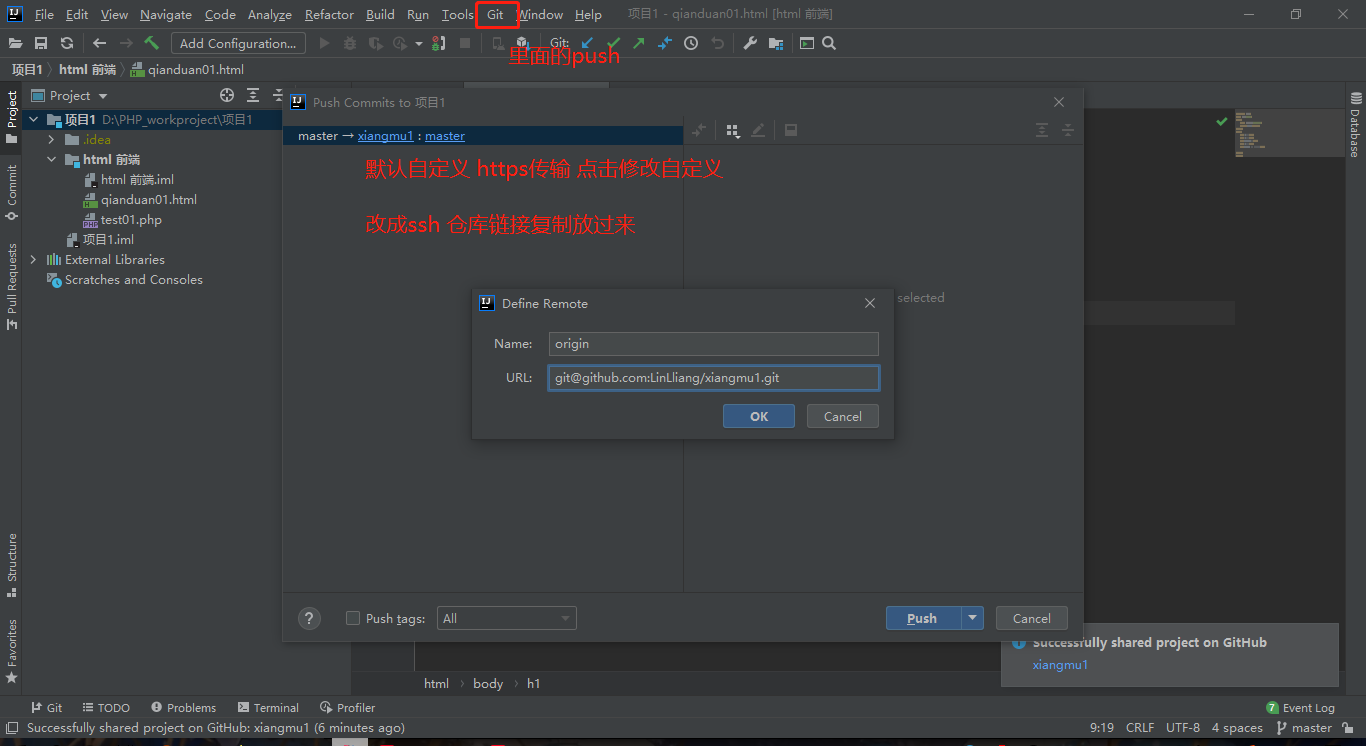




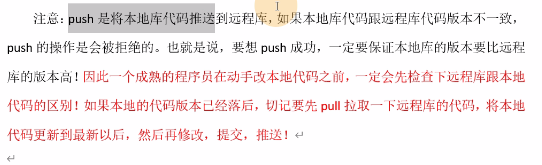
IDEA集成 分享项目到GitHub（属于新建）

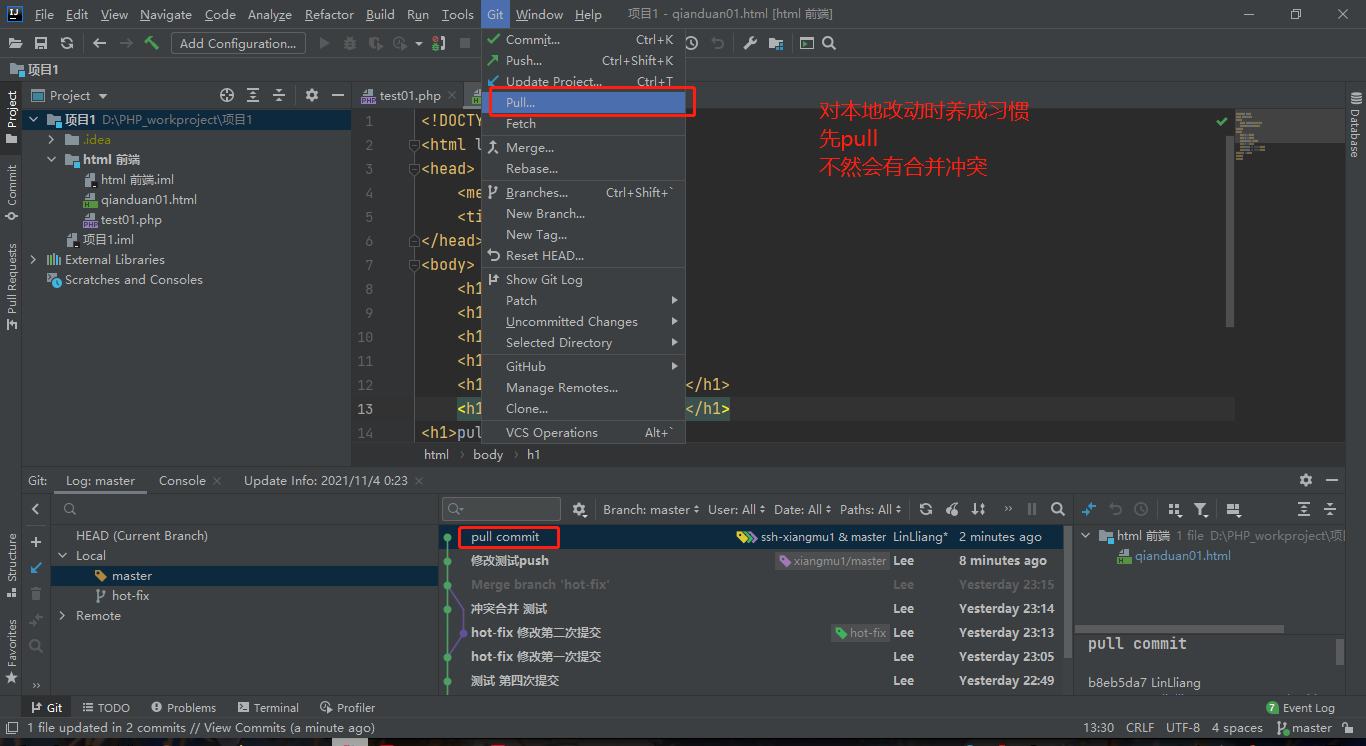


IDEA集成 push代码到GitHub

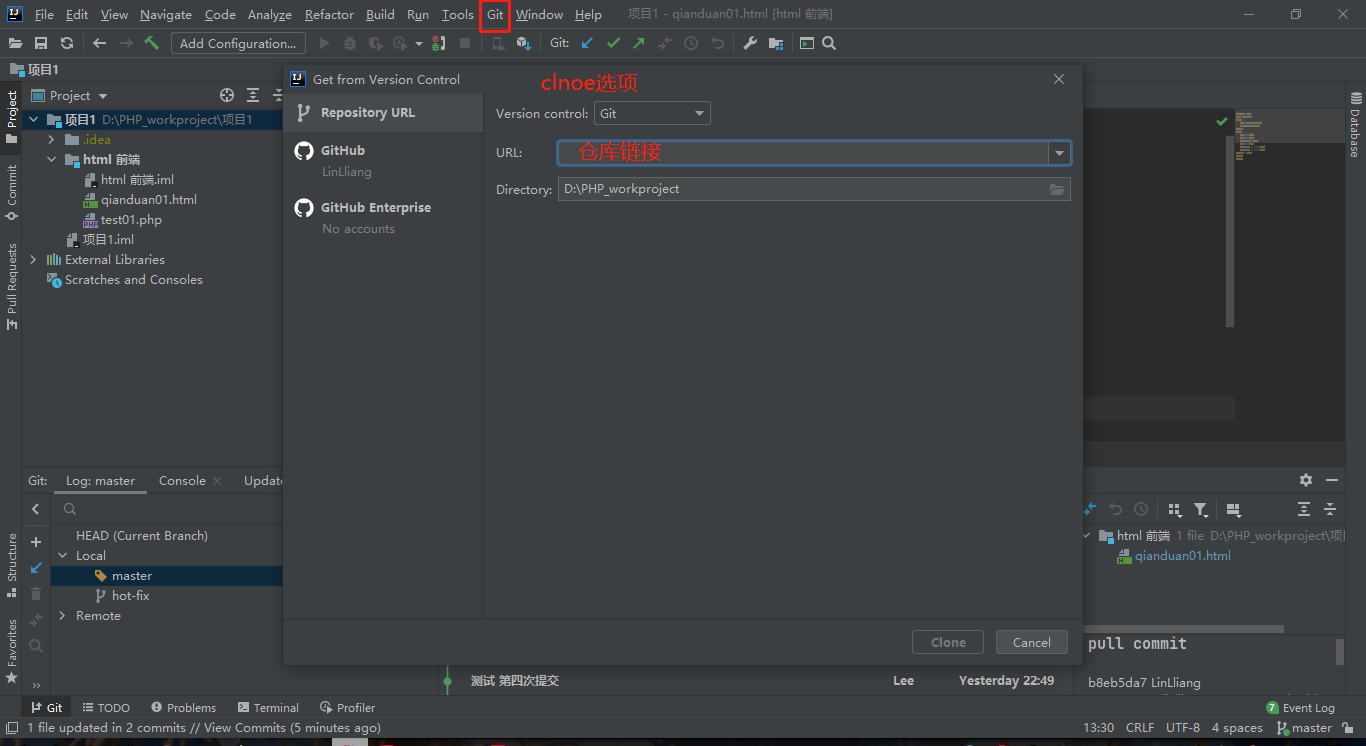


IDEA集成 pull拉取





IDEA集成 clone 代码



Gitlab 暂时用不上