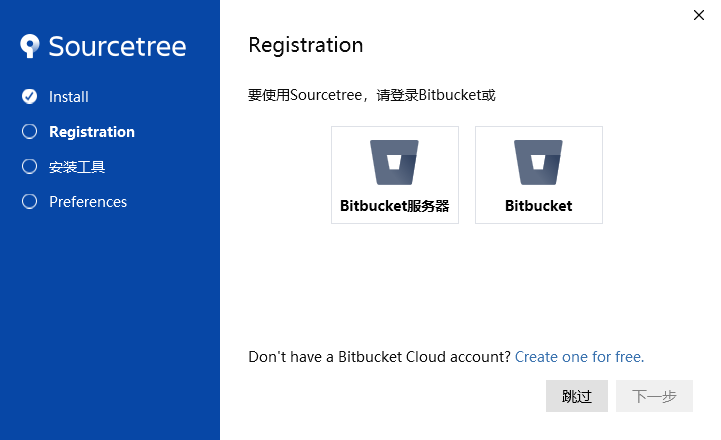
# SourceTree

## 工具的基本使用

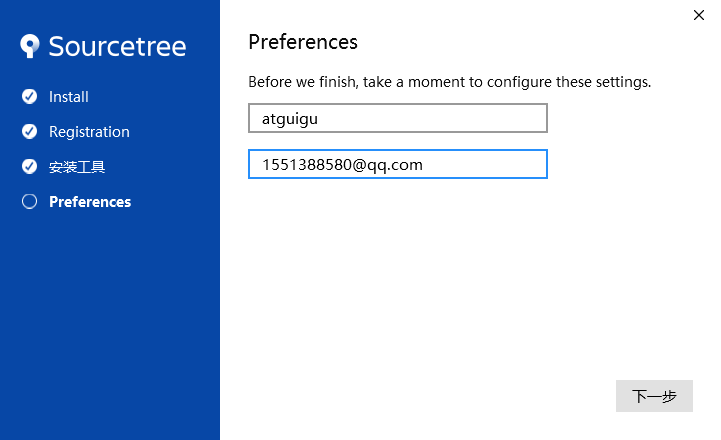
### 安装



跳过设置

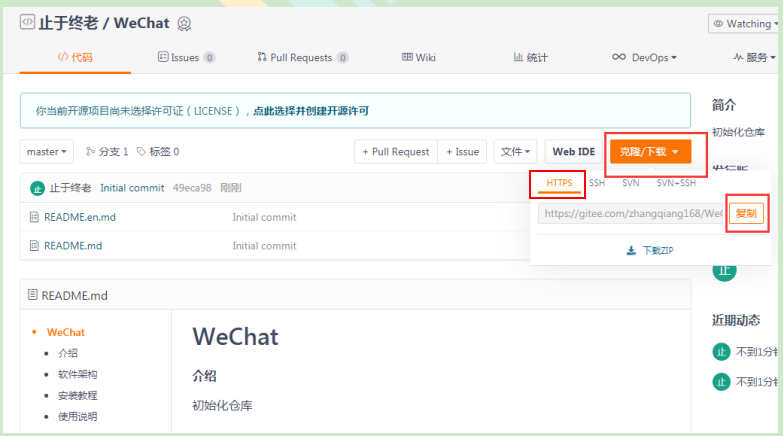


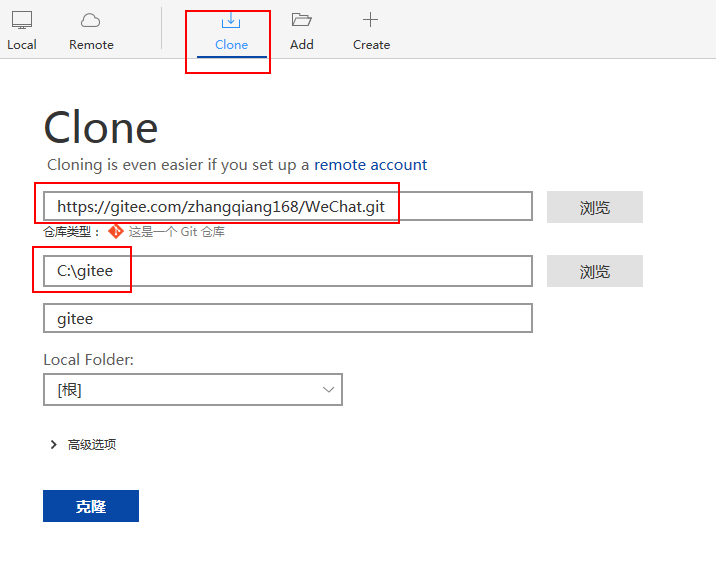
随便填一个正确的邮箱格式



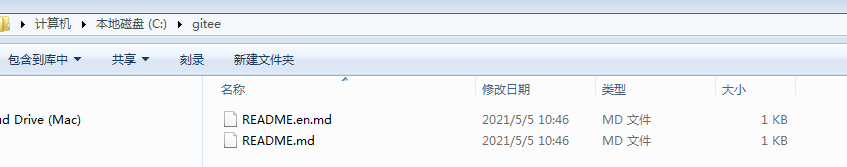
### 克隆



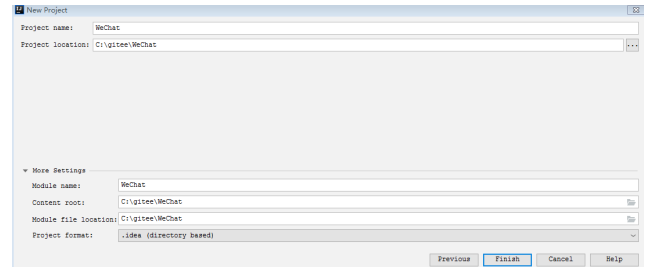




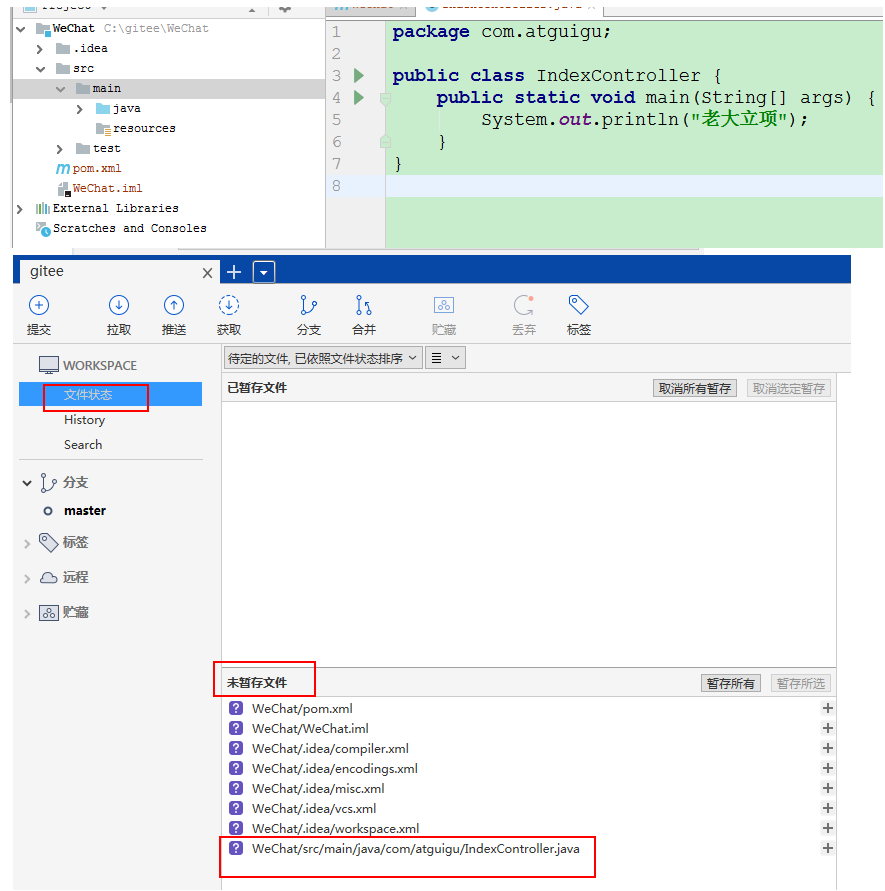
克隆完成之后有个这样的文件



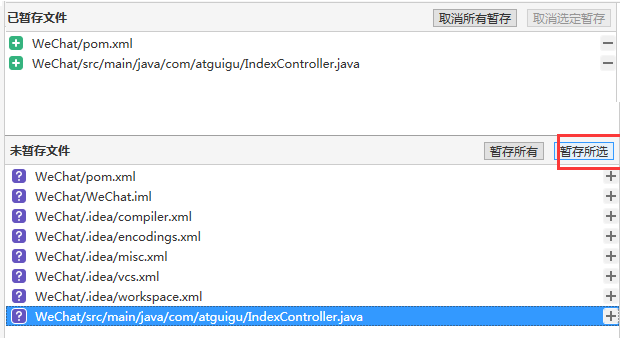
在仓库创建一个新的项目



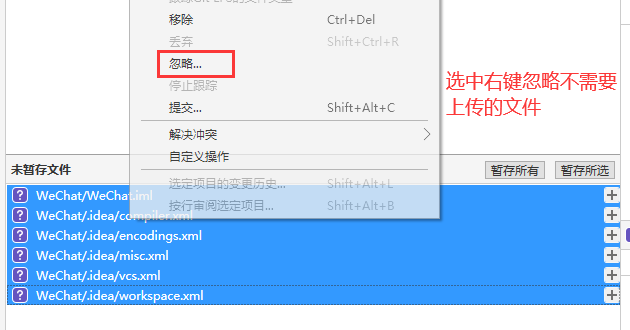
Idea创建项目之后sourcetree里面的内容变化



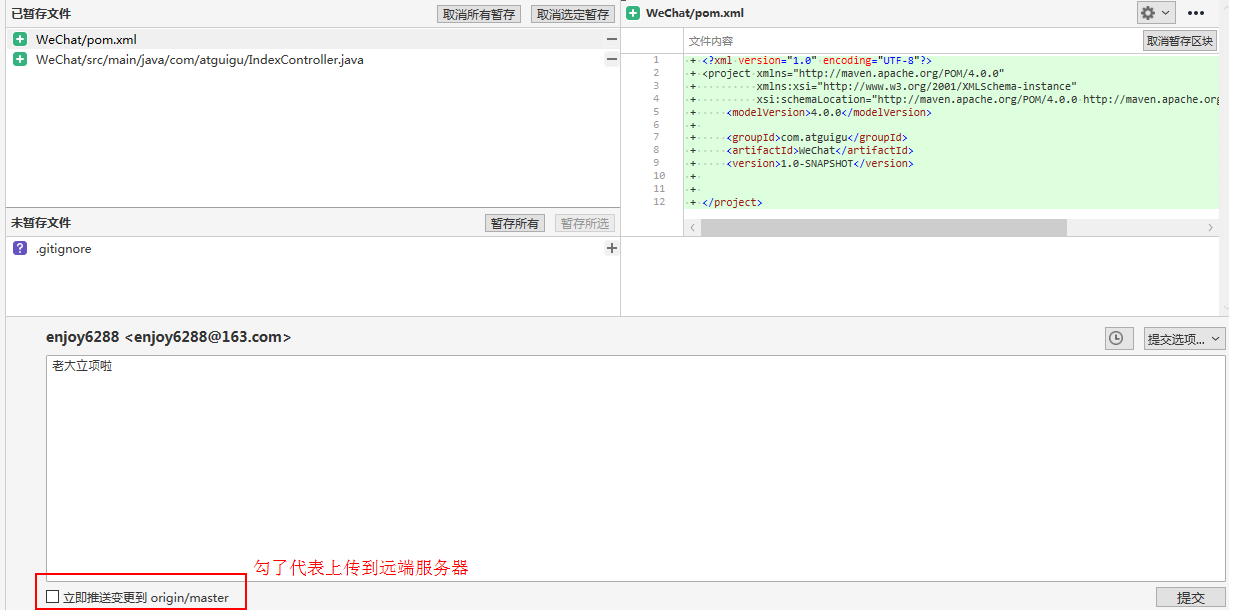
### 将修改的代码提交到暂存区



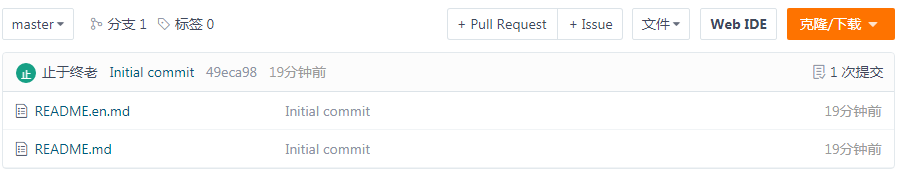
忽略不需要上传的文件



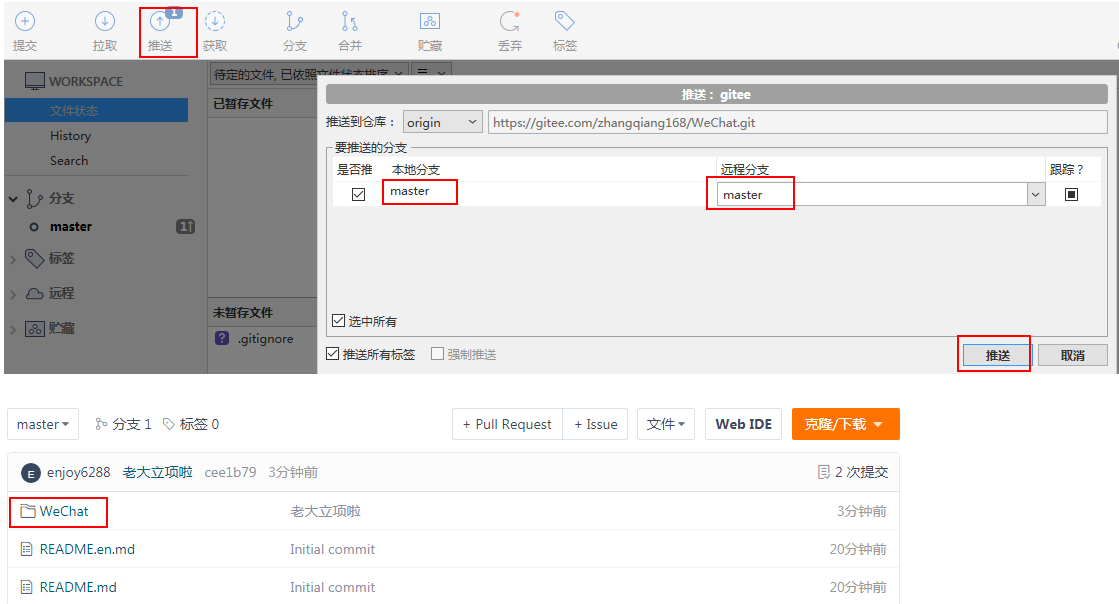
### 将暂存区中的代码提交到本地代码仓库



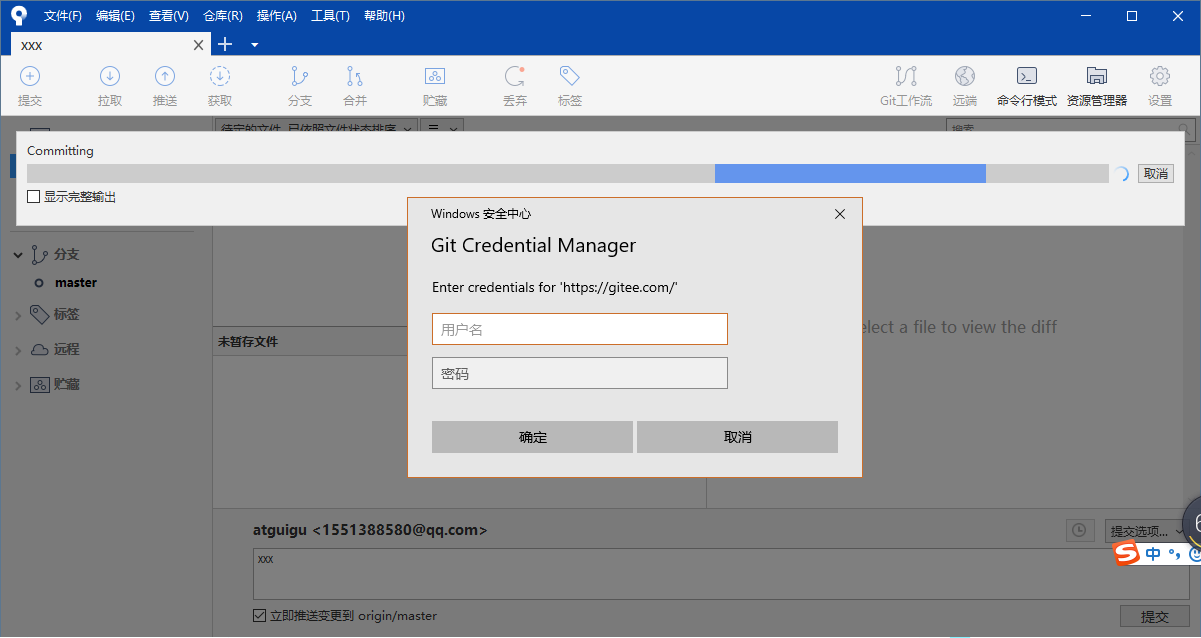
此时远端服务器是没有刚刚推送的文件

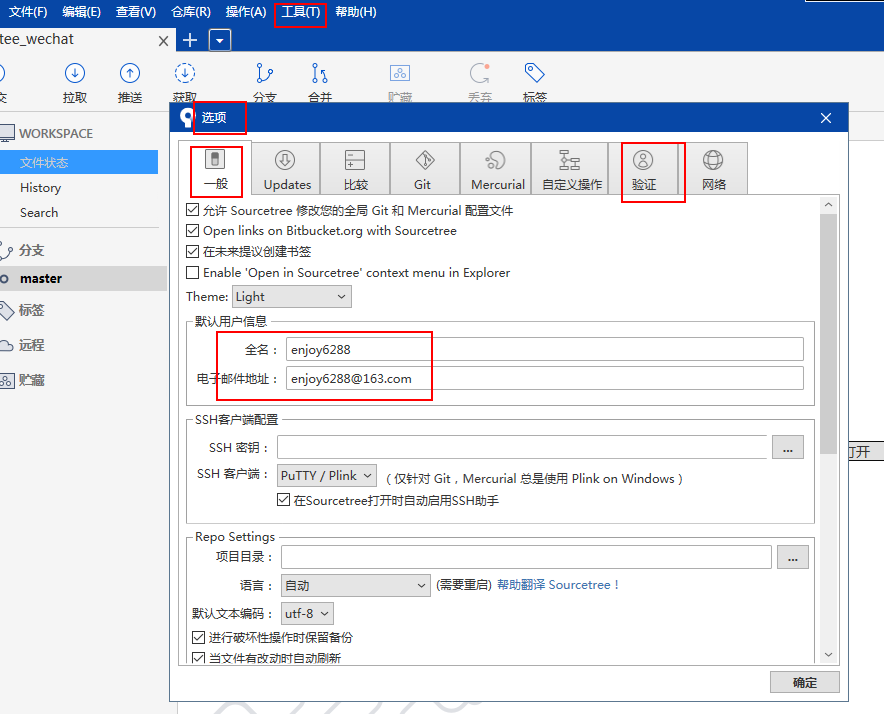


### 推送到远端



需要输入gitee或者github账号密码





## 分支

### 分支的概念

几乎所有的版本控制系统都以某种形式支持分支。 使用分支意味着你可以把你的工作从开发主线上分离开来，以免影响开发主线。Git 的master分支并不是一个特殊分支。 它跟其它分支没有区别。 之所以几乎每一个仓库都有 master 分支，是因为git init 命令默认创建它，并且大多数人都懒得去改动它。

### 实际开发流程

计算机生成了可选文字:
重亲斤
讠殳十
产品经理
内部
不通过专
评审
通过
需求
评审
通过
需求排期
分配任务
产品原型
开发人员参与
Bug
服务器
模块之间
相互调用
廾发
服务器
O冂0
开发工程师
版本
测试
服务器
《测试工程师》
反馈
最终用户
服务器

在使用版本控制工具开发的过程中，同时推进多个任务

计算机生成了可选文字:
hot
master
feature—blue
feature—game

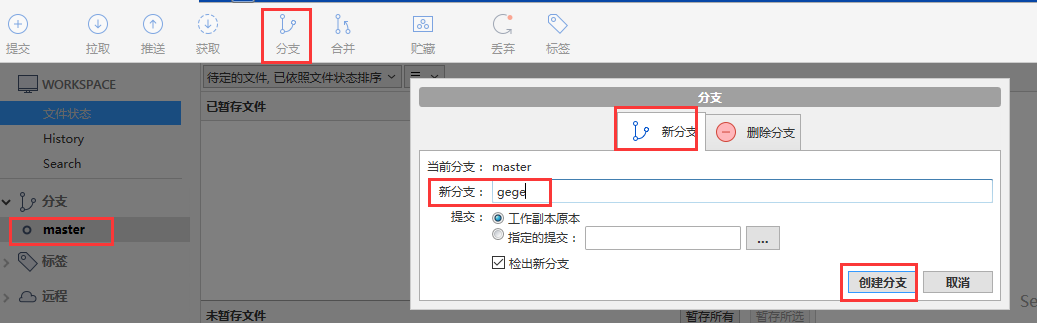
### 分支的好处

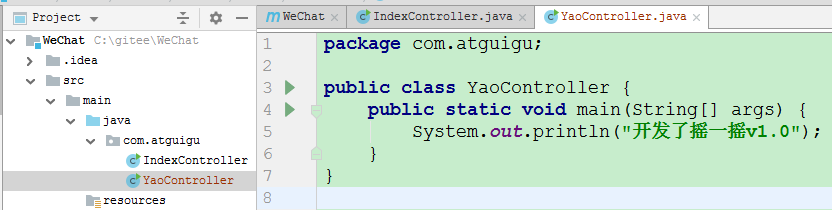
同时并行推进多个功能开发，提高开发效率各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响。失败的分支删除重新开始即可。

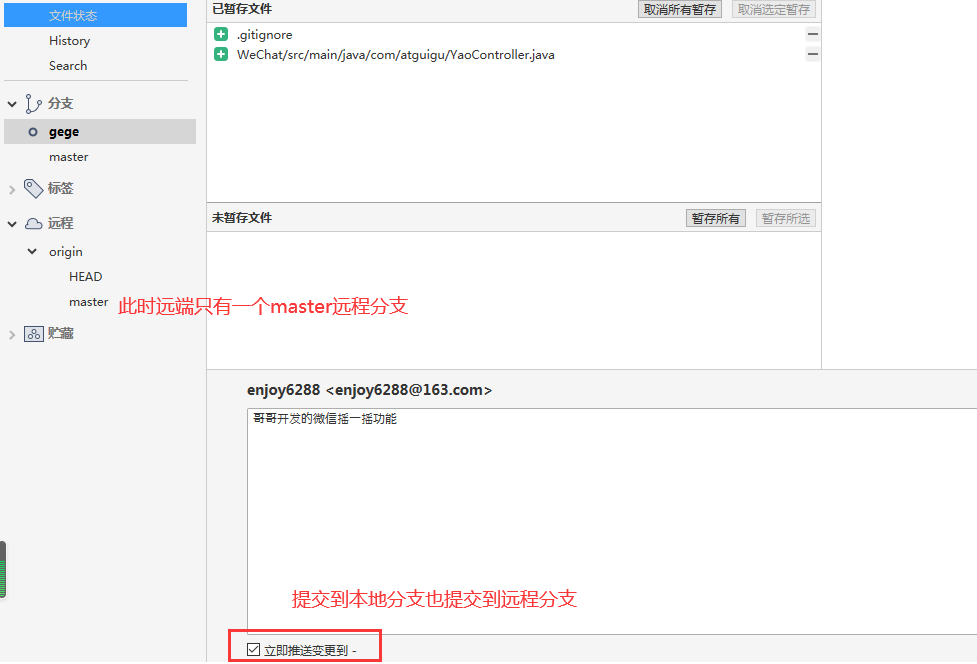
## 模拟真实项目开发过程

### 拉新分支(开发摇一摇功能)

程序员哥哥在自己电脑上克隆刚刚老大建立的项目，在这里我用建立新的分支来模拟克隆过程



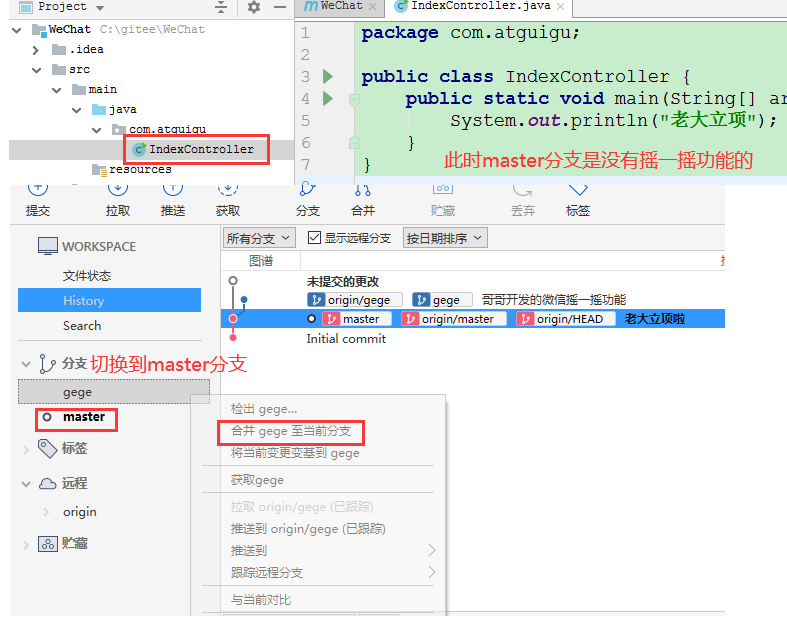




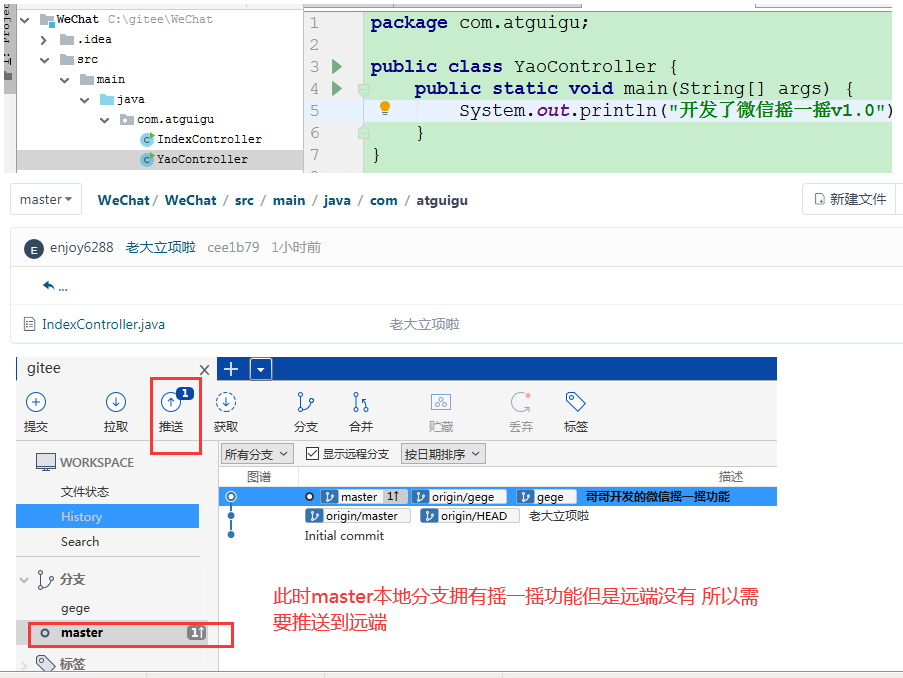


### 合并分支

先进行本地分支的合并

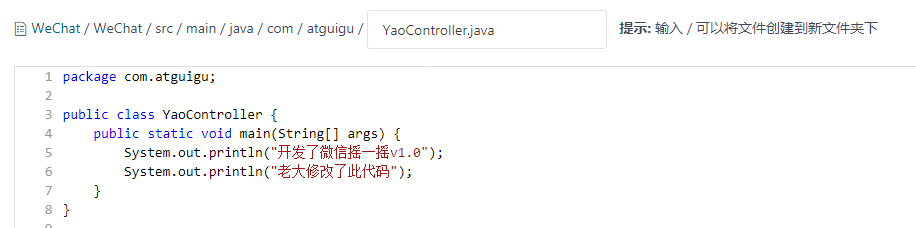


远端分支是没有刚刚提交分支的信息的

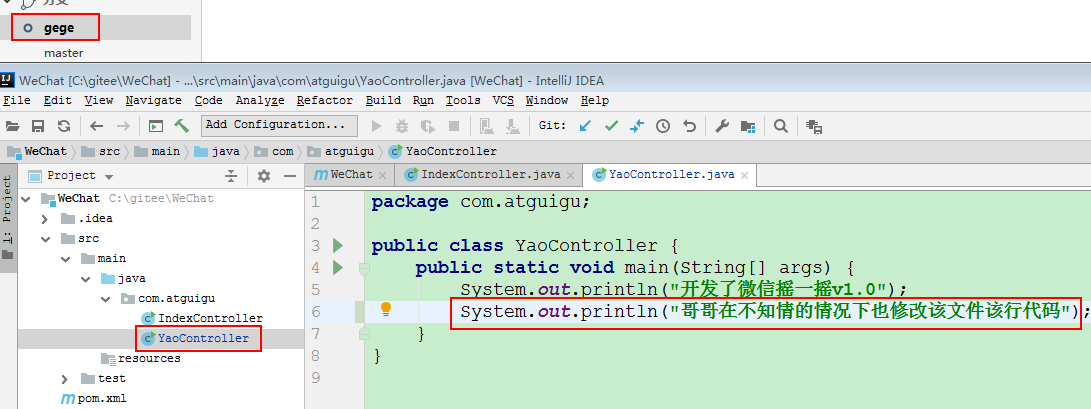


### 解决冲突

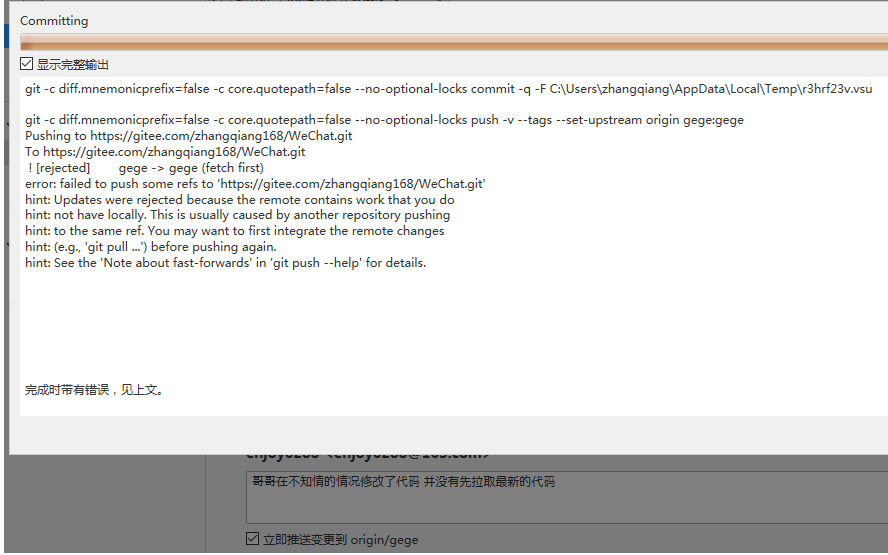
老大在gege分支下操作了YaoController



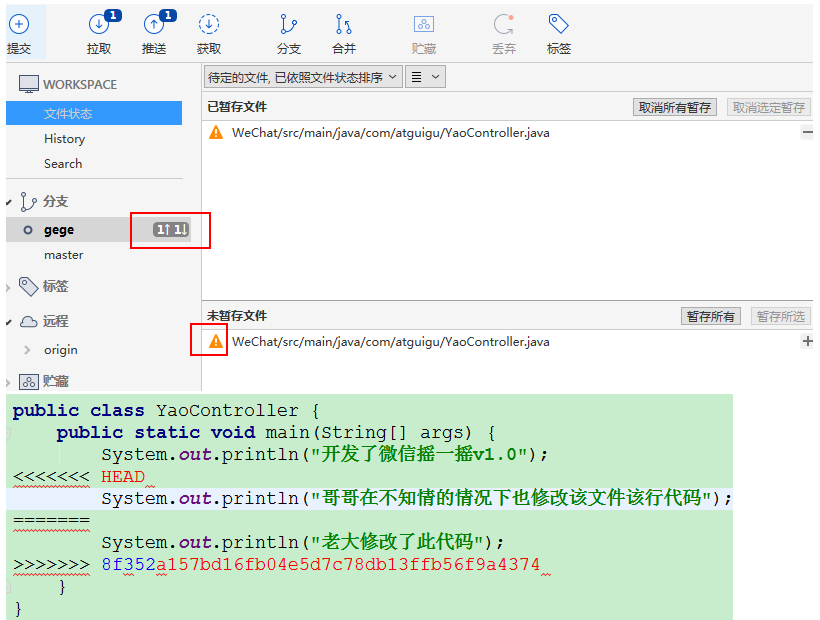
哥哥在不知情的情况下也修改该文件该行代码



修改之后他直接提交到远端去了



此时哥哥想把远端的代码再拉下来

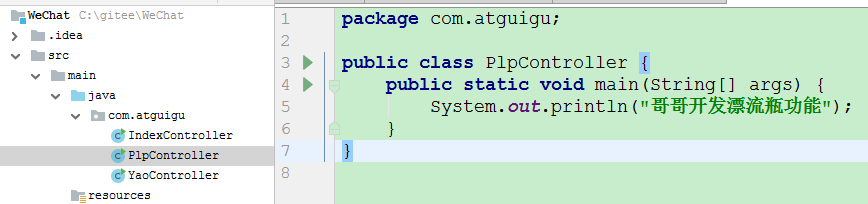


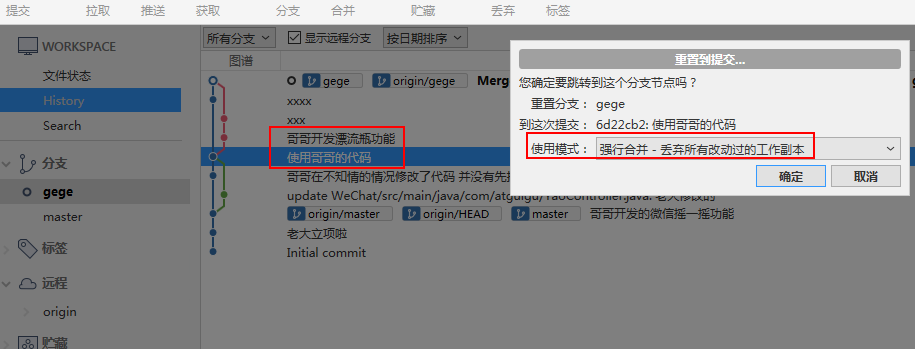




### 回滚操作

哥哥又开发了一个漂流瓶功能





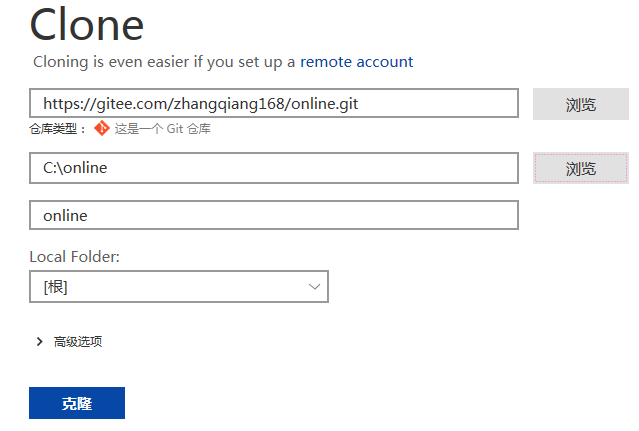
## 跨团队开发

### Fork项目

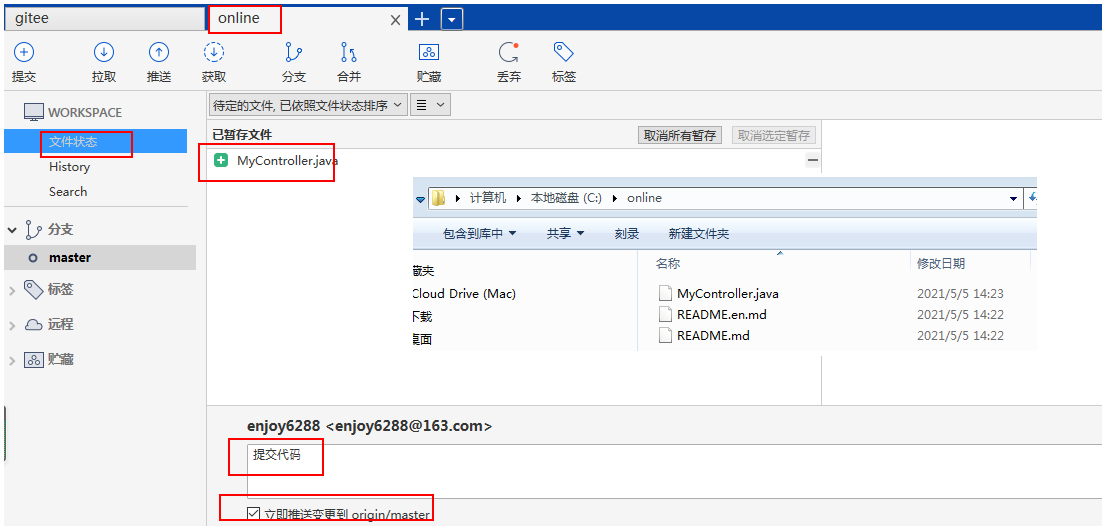
在仓库里面搜索网上开源的项目



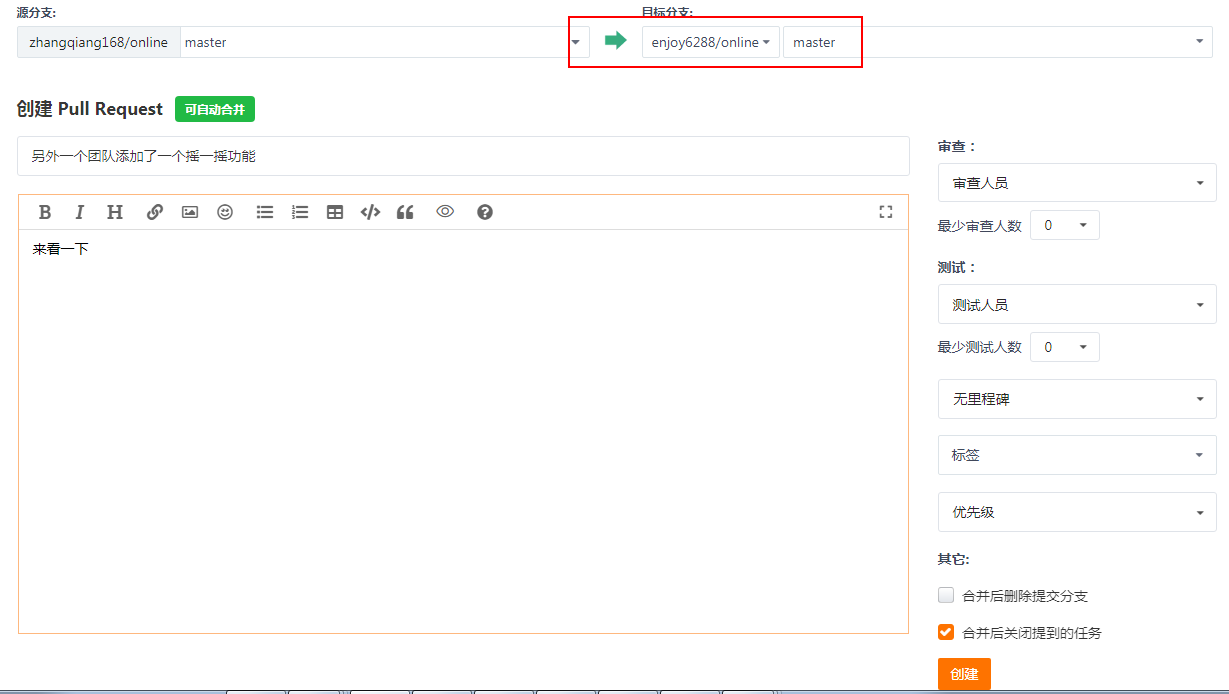
### 克隆项目

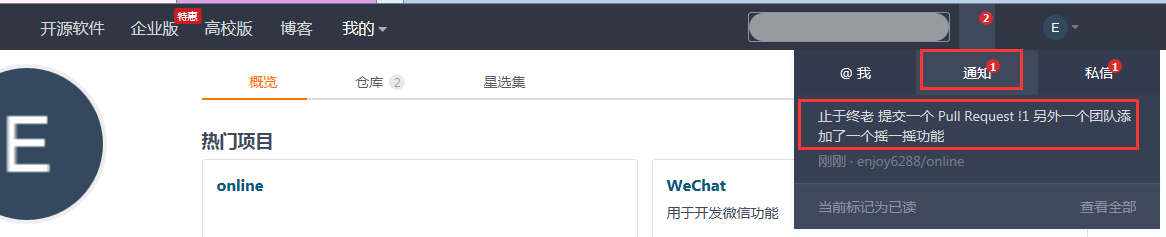


### 添加新的内容并提交







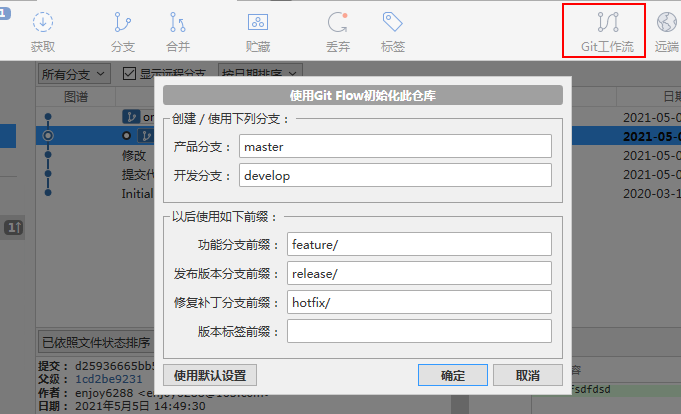




## git工作流

### 创建工作流

注意执行此操作的时候只留下master分支



### 分支类型

master 最终发布版本整个项目中有且只有一个

develop 项目的开发分支原则上项目中有且只有一个

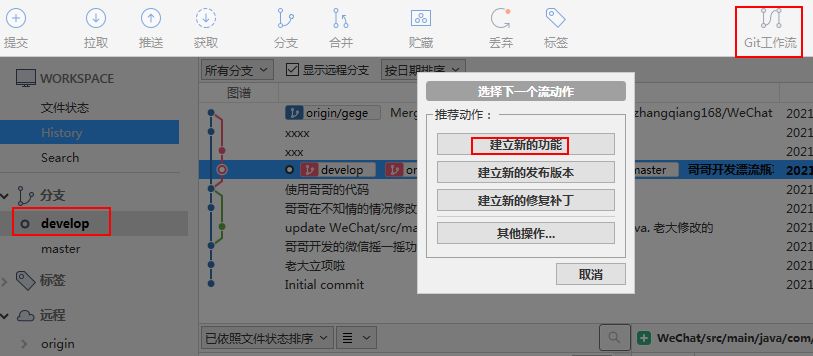
feature 功能分支用于开发一个新的功能

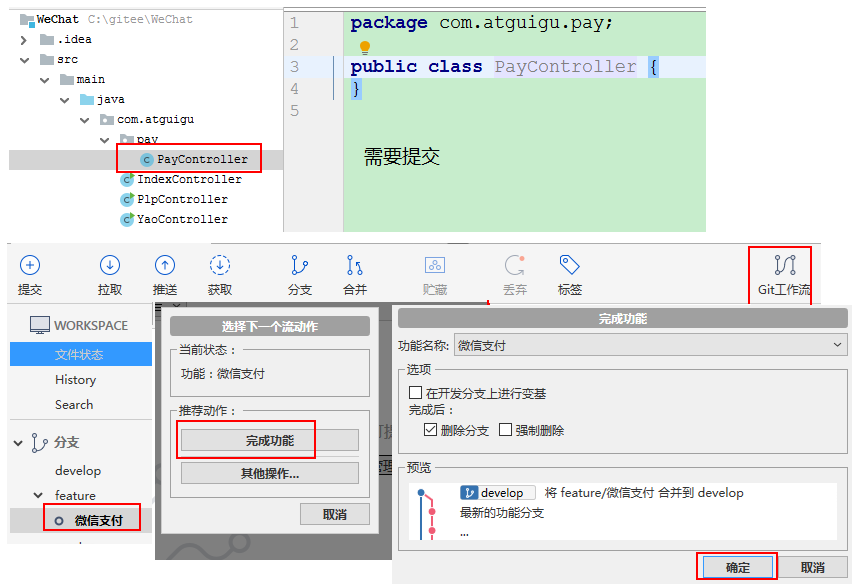
release 预发布版本介于develop和master之间的一个版本主要用于测试

hotfix 修复补丁用于修复master上的bug直接作用于master

### 开发新的功能

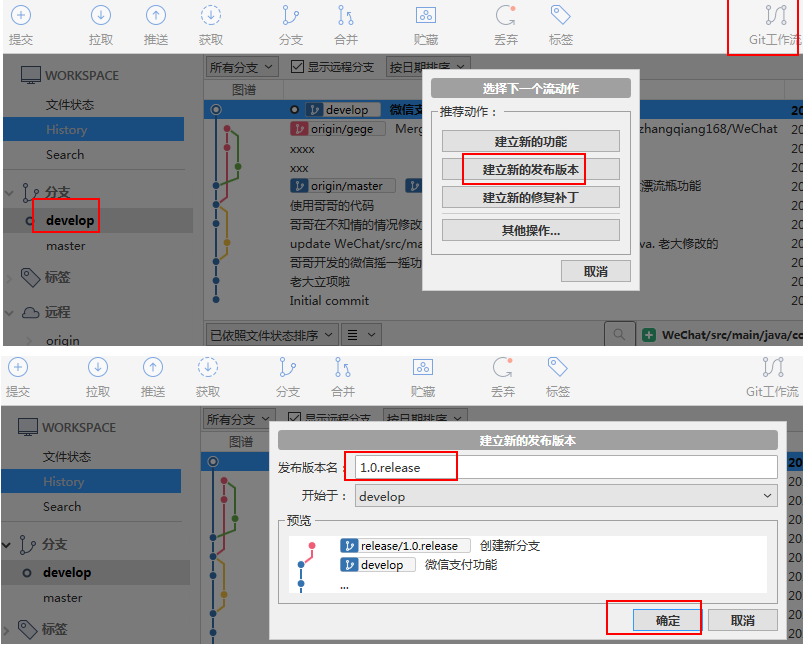
当开发中需要增加一个新的功能时，可新建feature分支，用于增加新功能，并且不影响开发中的develop源码，当新功能增加完成后，完成feature分支，将新功能合并到develop中，更新develop上的代码





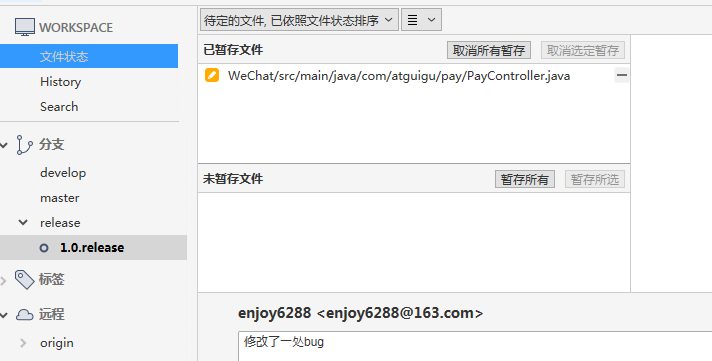
### 建立发布版本

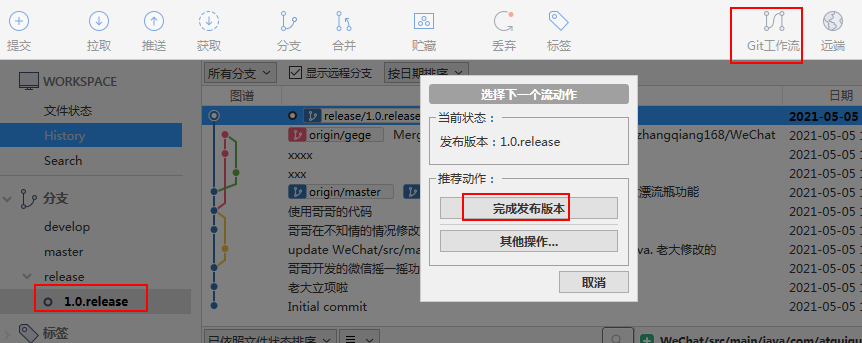
当开发到一定阶段，可以发布测试版本时，可以从develop分支，建立release分支，进入预发布测试阶段。点击"Git工作流"，选择"建立新的发布版本"

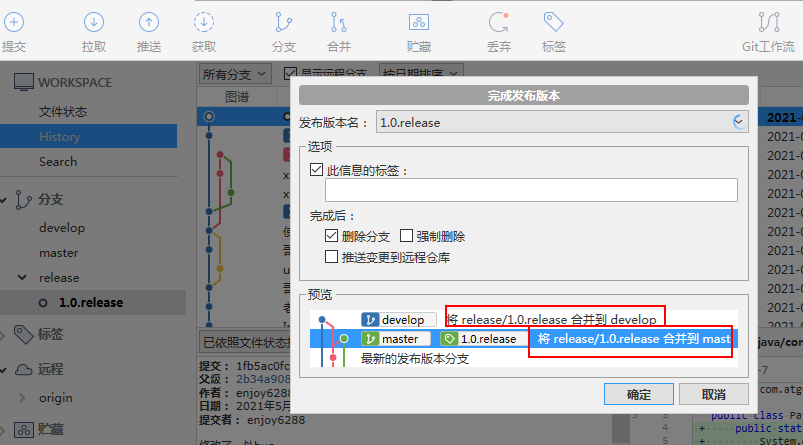


1.0.release为阶段性发布版本，主要用于发布前进行测试，后续的开发工作仍旧在develop上进行，如果在测试过程中发现问题，直接在release上进行修改，修改完成后进行提交。

对release分支1.0.release进行两次修改后，测试完成，可以进行正式发布，在当前分支指向1.0.release分支下，点击"Git工作流"，选择"完成发布版本"





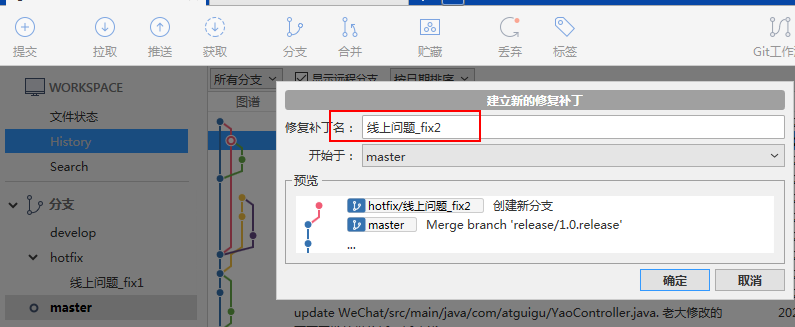


完成合并后，默认指向develop为当前分支，master增加多个版本更新，将master分支推送到origin，完成线上发布

### 补丁hotfix

正式版本发布后，develop可继续进行后续开发，当正式版本出现问题时，需要进行问题的修改，可以在master分支建立修改补丁hotfix。将当前分支切换到master，点击"Git工作流"，选择"建立新的修复补丁"





在该分支下进行master的问题修改，修改完成后进行提交。当所有补丁问题修改完成后，点击"Git工作流"，选择"完成修复补丁"

