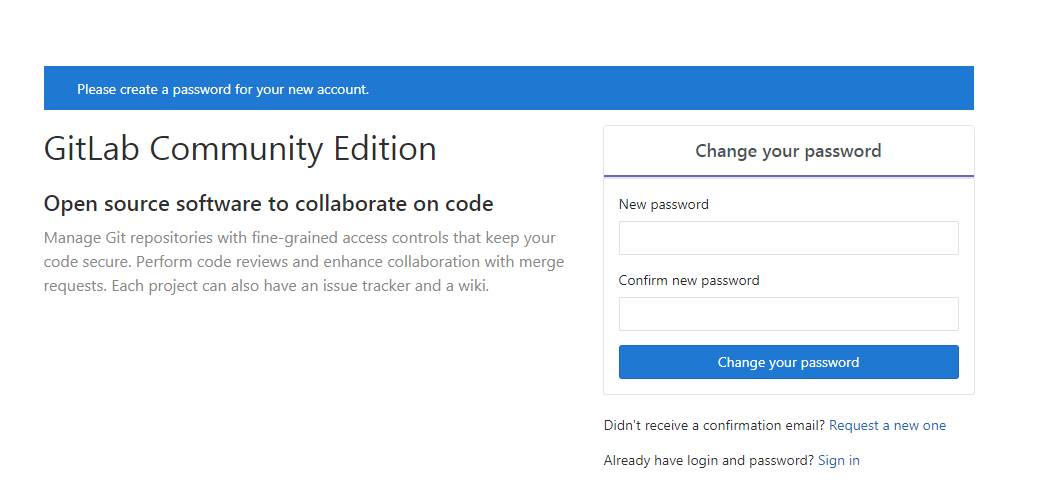
Gitlab使用手册

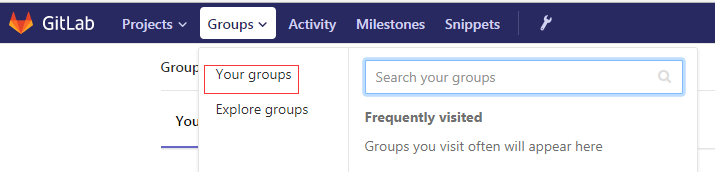
# 一、管理员使用手册

## 1.1使用root登录，在第一次gitlab启动时会强制设置一个root密码

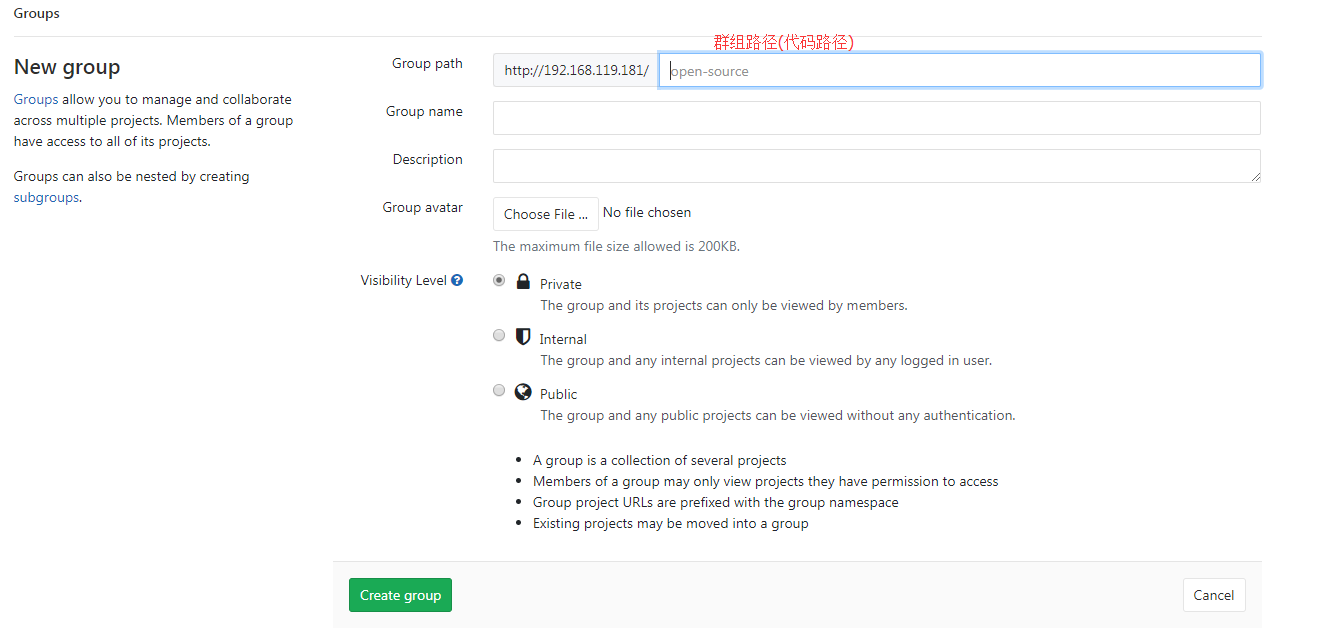


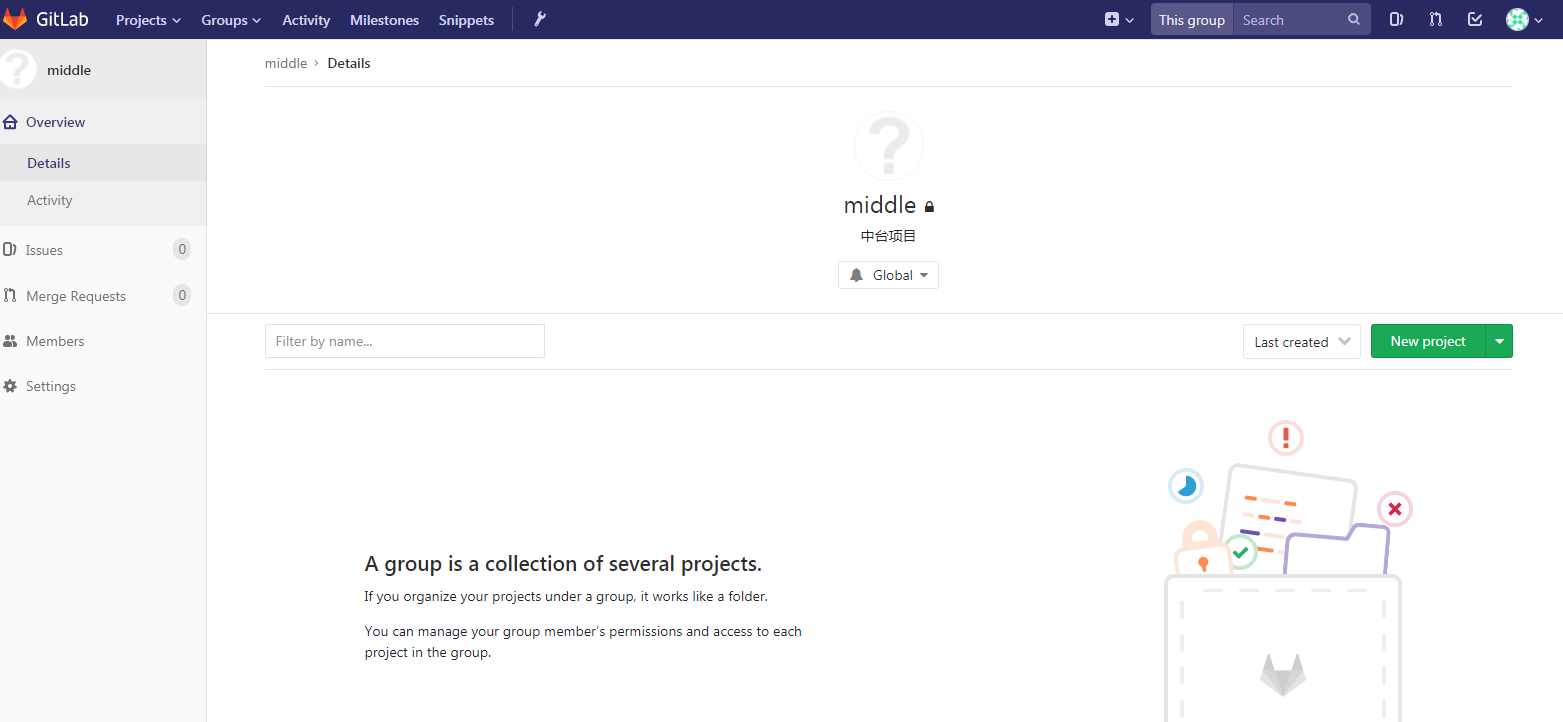
本案例设置：root/admin123

## 1.2 创建群组

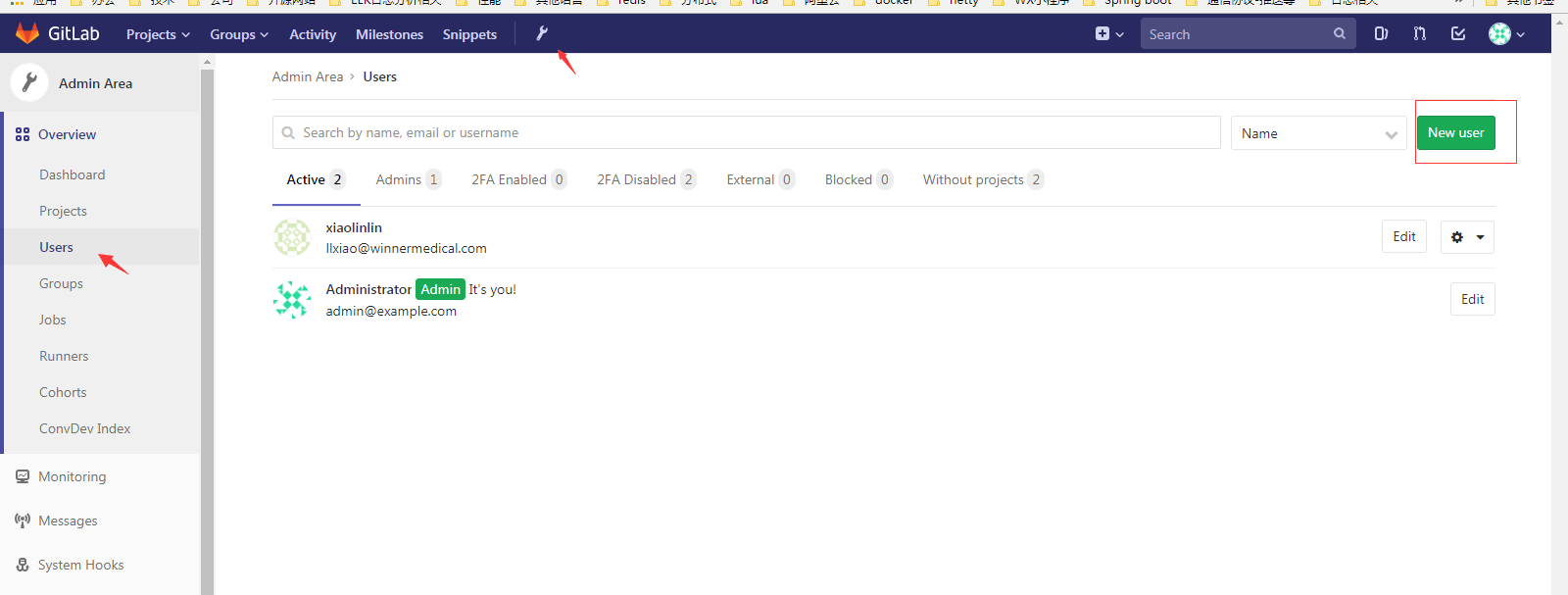


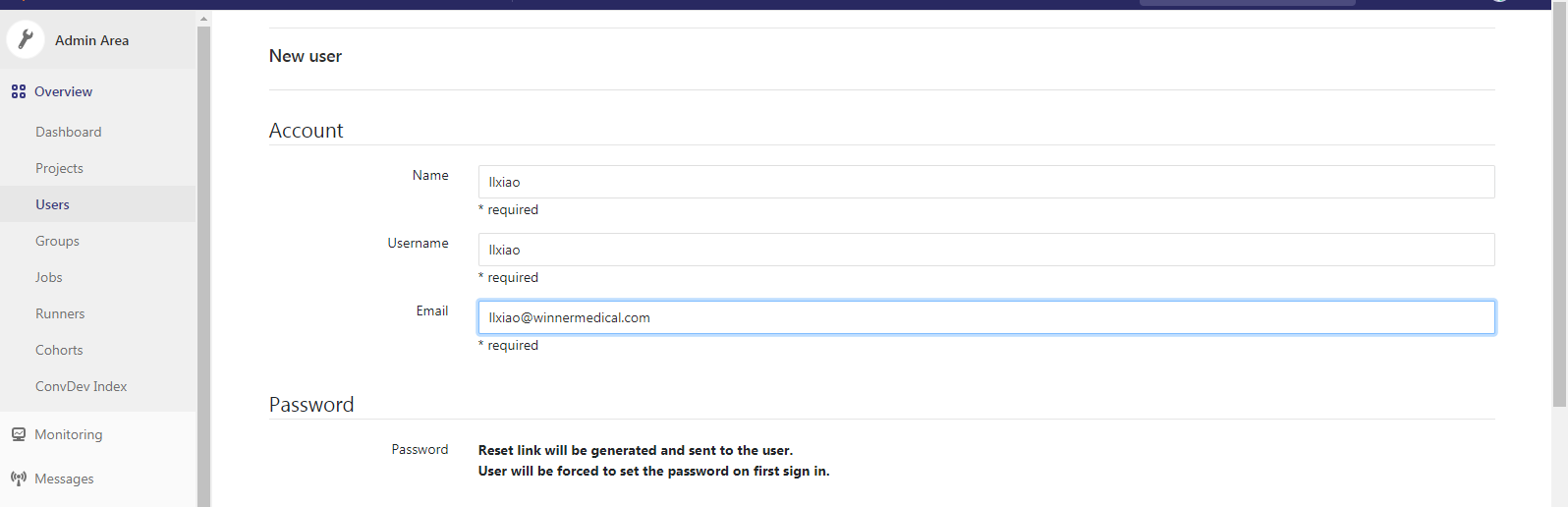




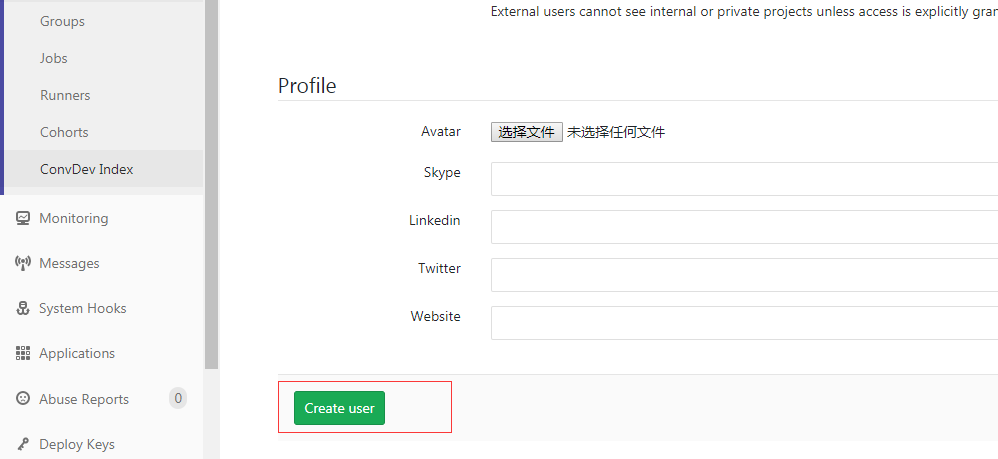


## 1.3创建用户(另可通过注册方式创建用户1.4章节)

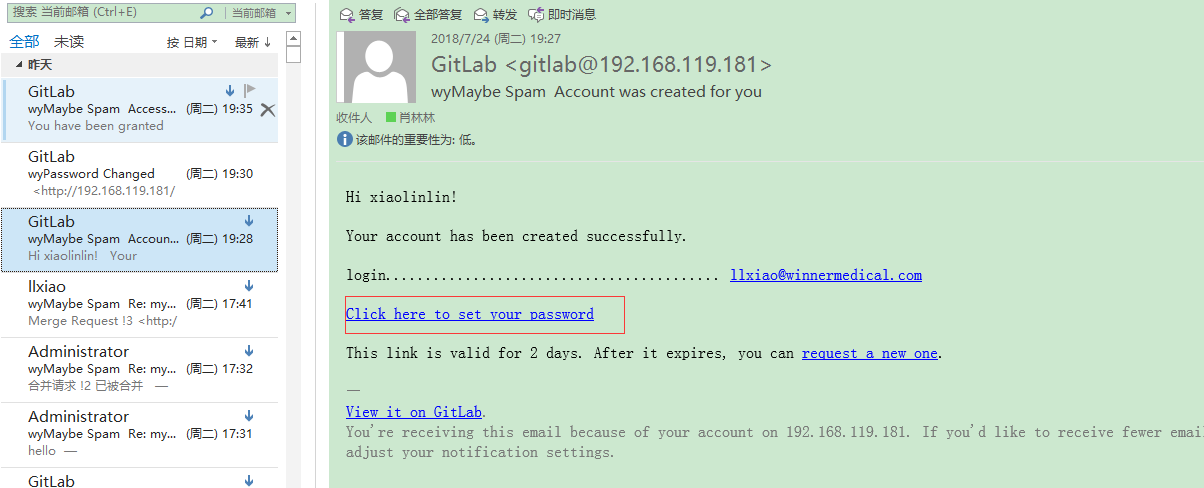




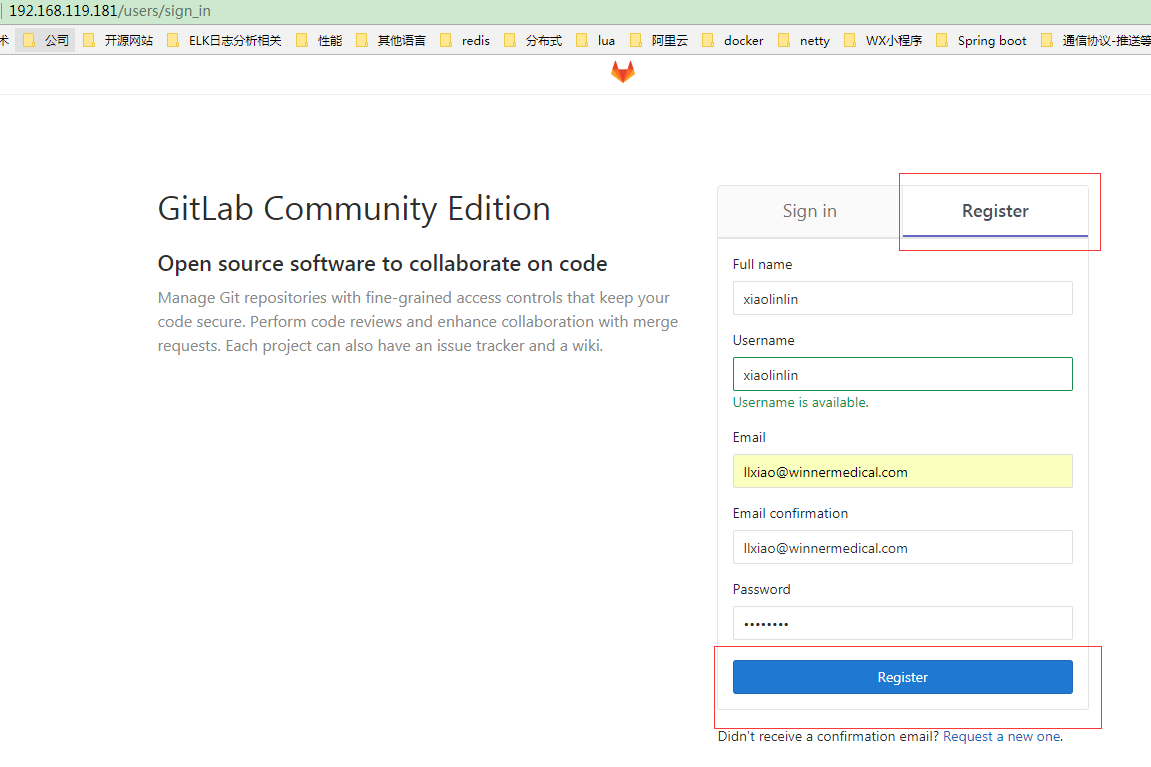
请务必填写正确的邮件地址，系统会发邮件来激活该账号。其他默认



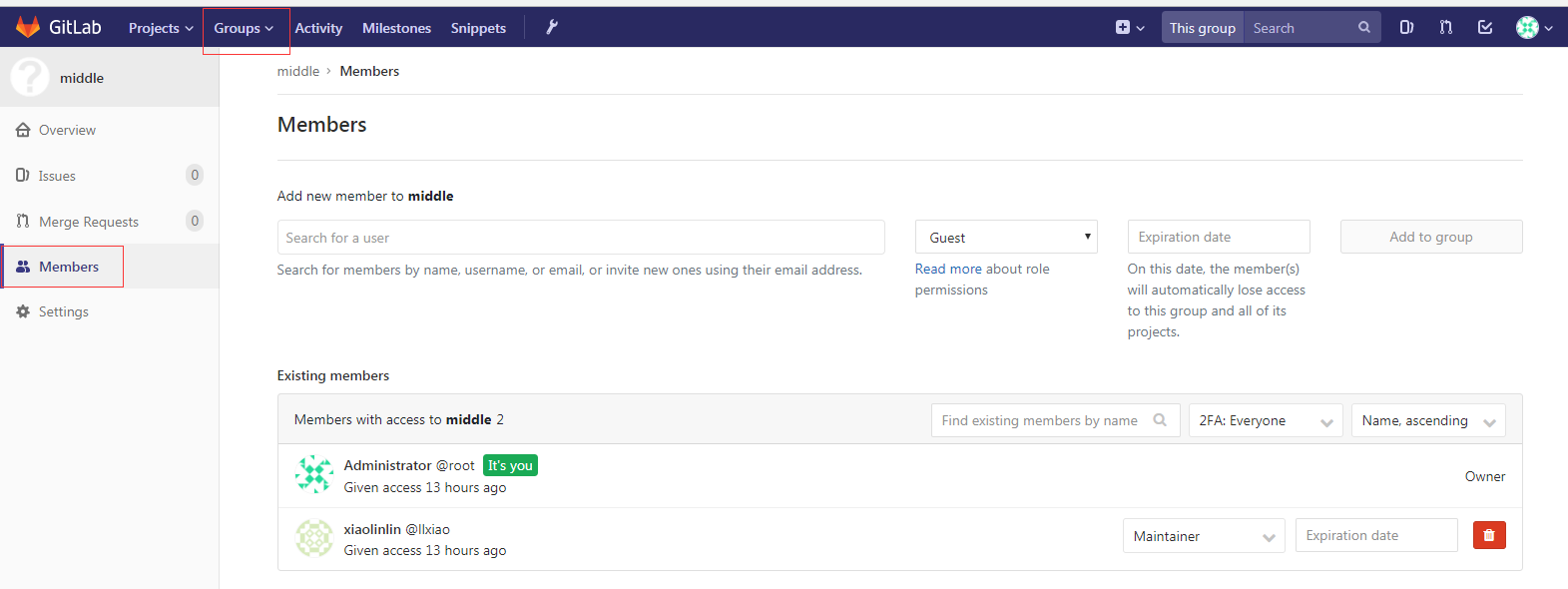
收到邮件，点击连接进行设置密码激活



## 1.4 用户注册



## 1.5 群组添加成员

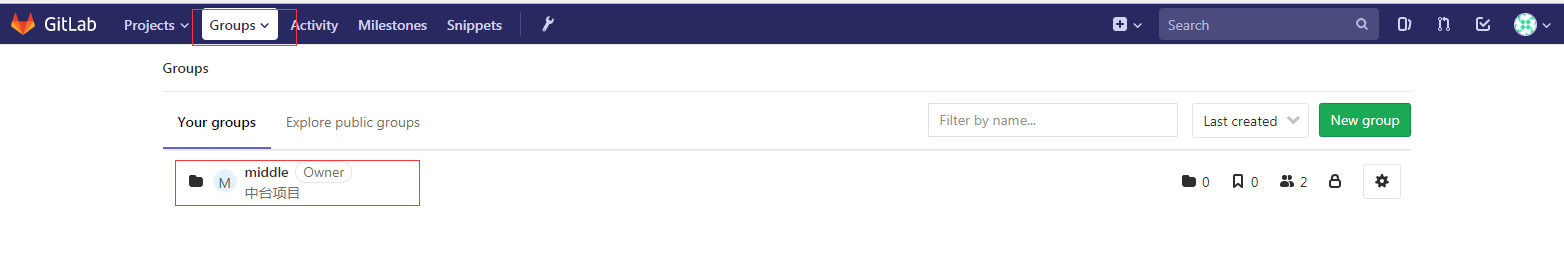


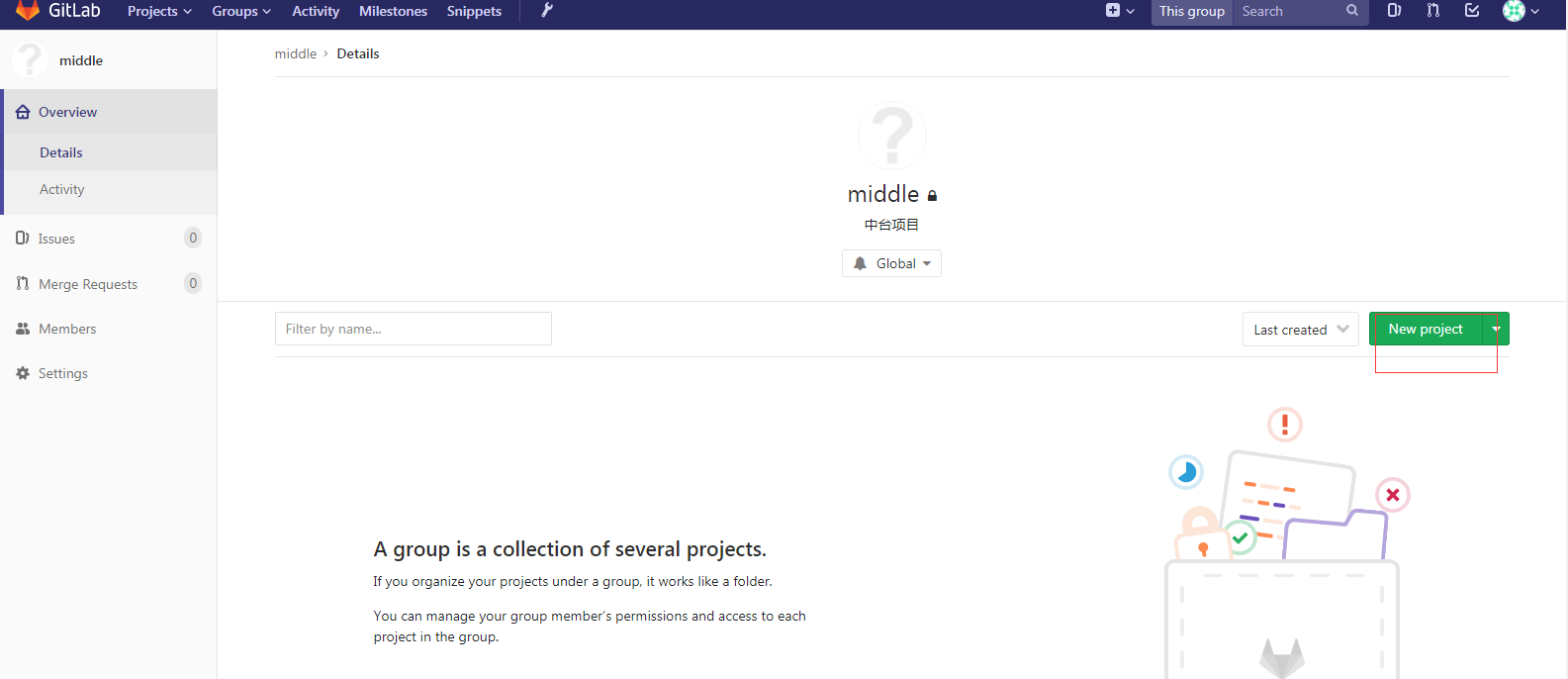


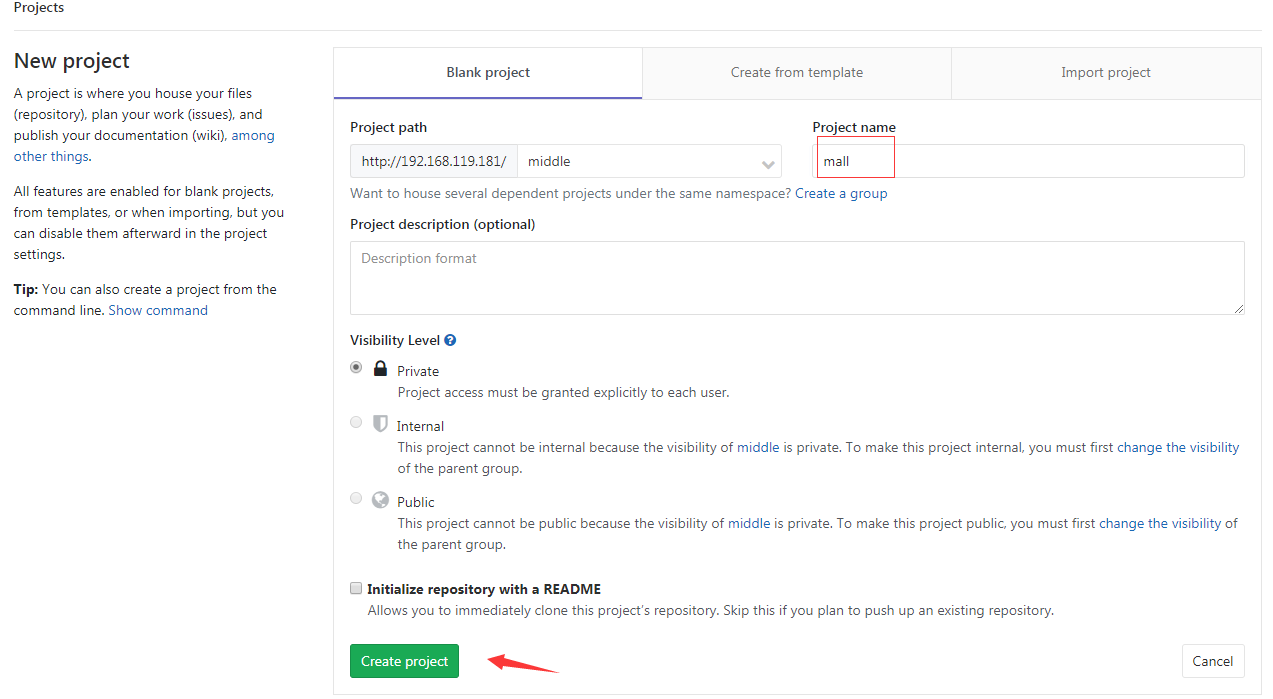
*权限说明：*

*Guest(匿名用户) - 创建项目、写留言薄Reporter（报告人）- 创建项目、写留言薄、拉项目、下载项目、创建代码片段Developer（开发者）- 创建项目、写留言薄、拉项目、下载项目、创建代码片段、创建合并请求、创建新分支、推送不受保护的分支、移除不受保护的分支 、创建标签、编写wikiMaster（管理者）- 创建项目、写留言薄、拉项目、下载项目、创建代码片段、创建合并请求、创建新分支、推送不受保护的分支、移除不受保护的分支 、创建标签、编写wiki、增加团队成员、推送受保护的分支、移除受保护的分支、编辑项目、添加部署密钥、配置项目钩子Owner（所有者）- 创建项目、写留言薄、拉项目、下载项目、创建代码片段、创建合并请求、创建新分支、推送不受保护的分支、移除不受保护的分支 、创建标签、编写wiki、增加团队成员、推送受保护的分支、移除受保护的分支、编辑项目、添加部署密钥、配置项目钩子、开关公有模式、将项目转移到另一个名称空间、删除项目***tips: 一般只给开发者权限，代码合并到主干只有管理者才有权限**.

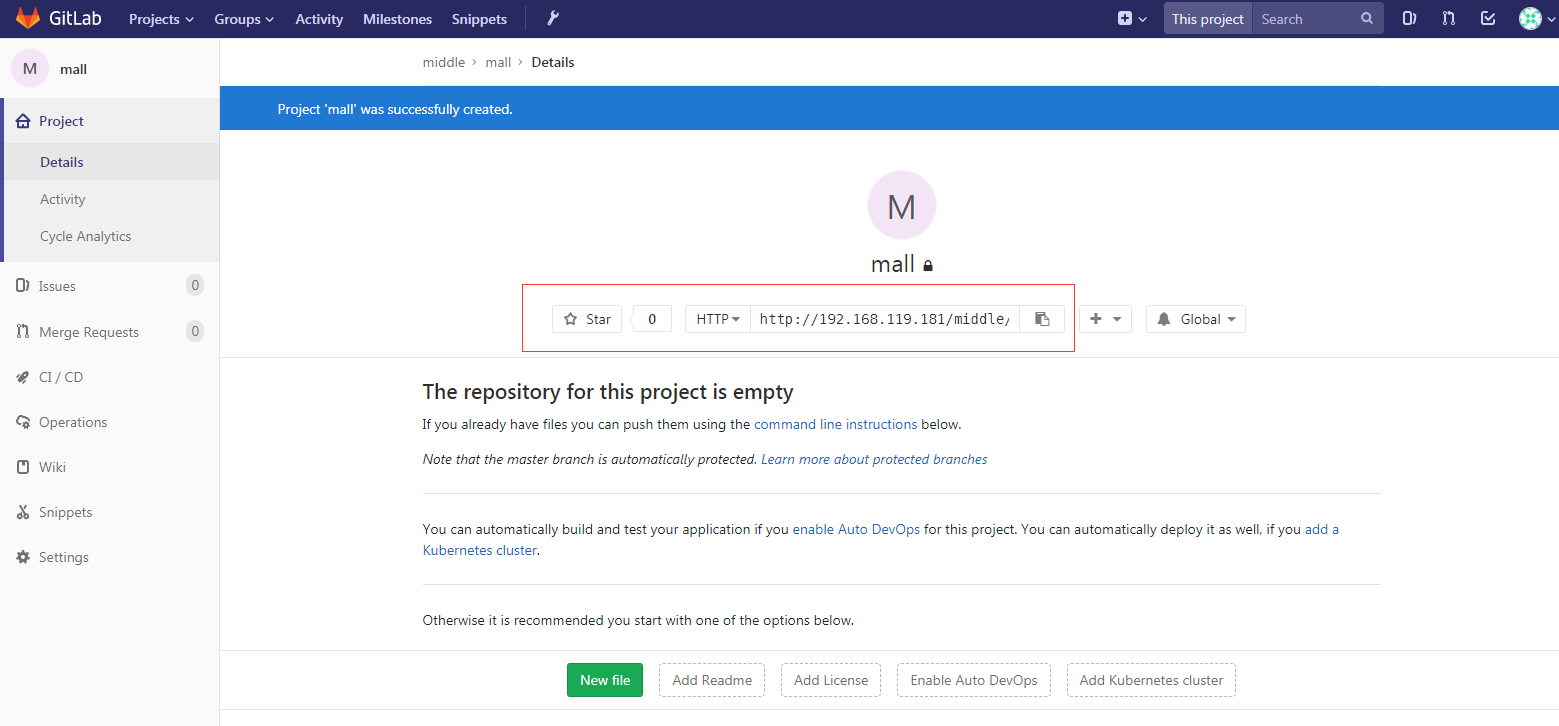
## 1.6 在群组中创建一个项目

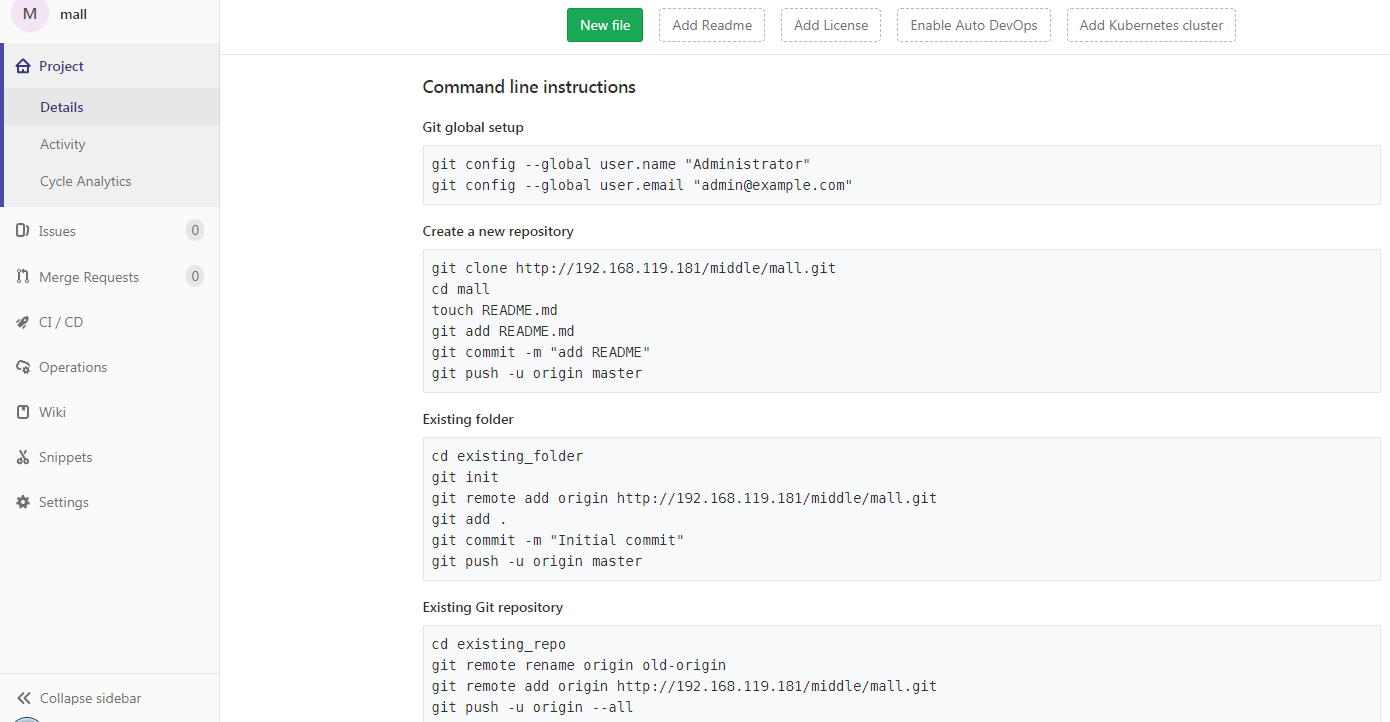




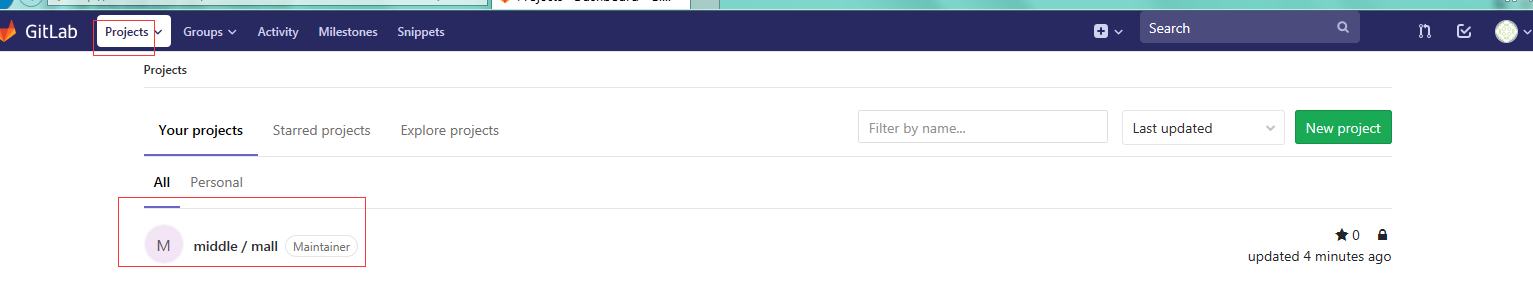


创建完成项目后如下图所示：

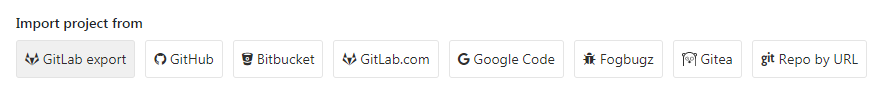


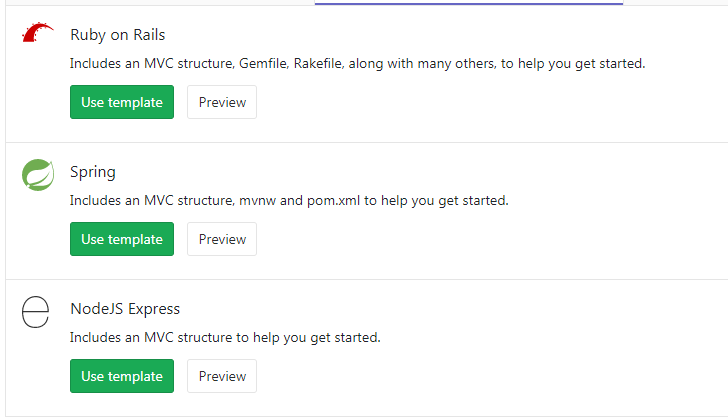


使用成员账号登录后，可以看到所属组的创建的项目



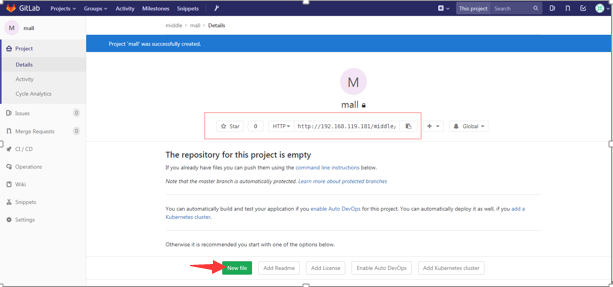
tips:*除此外还支持从模板创建和github等导入项目*



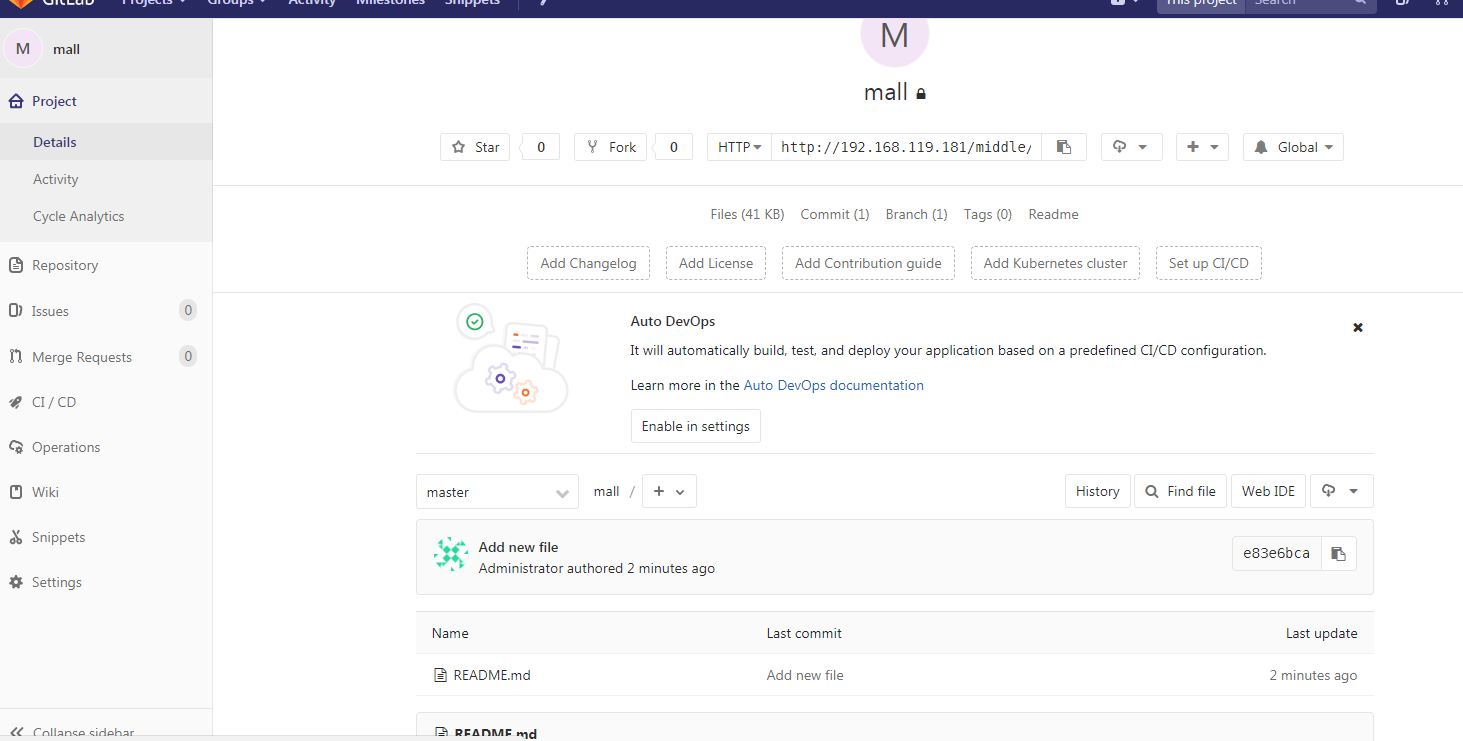


## 1.7 创建文件&创建分之

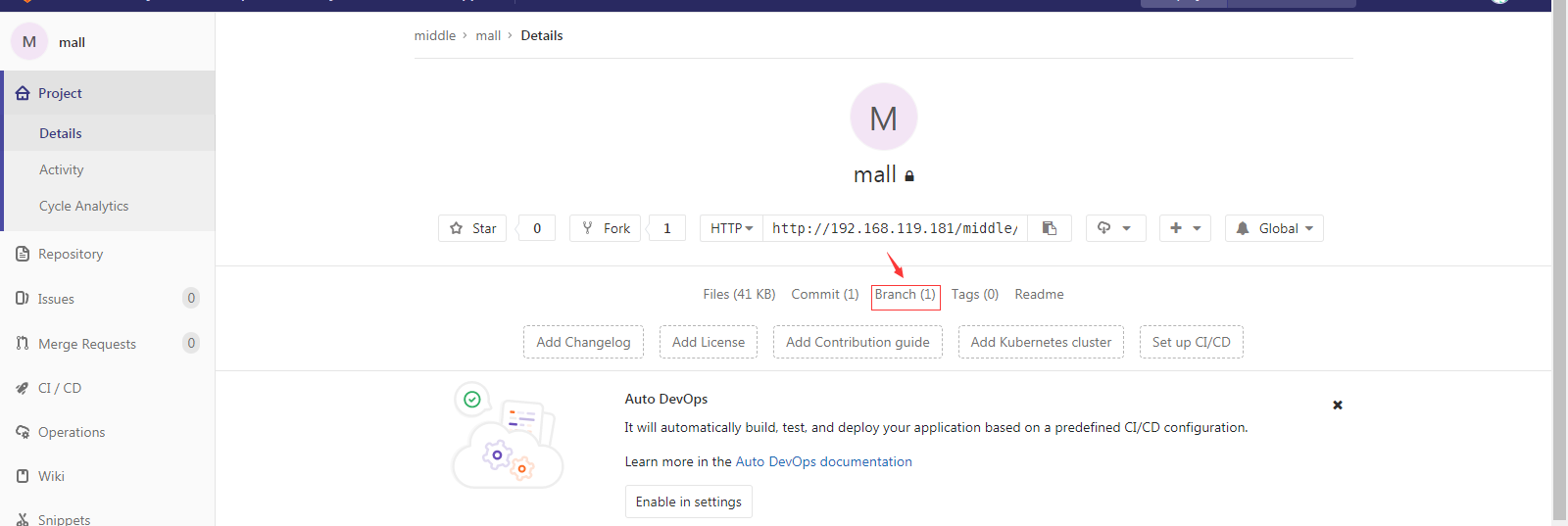
直接new file，创建一个readme.md文件

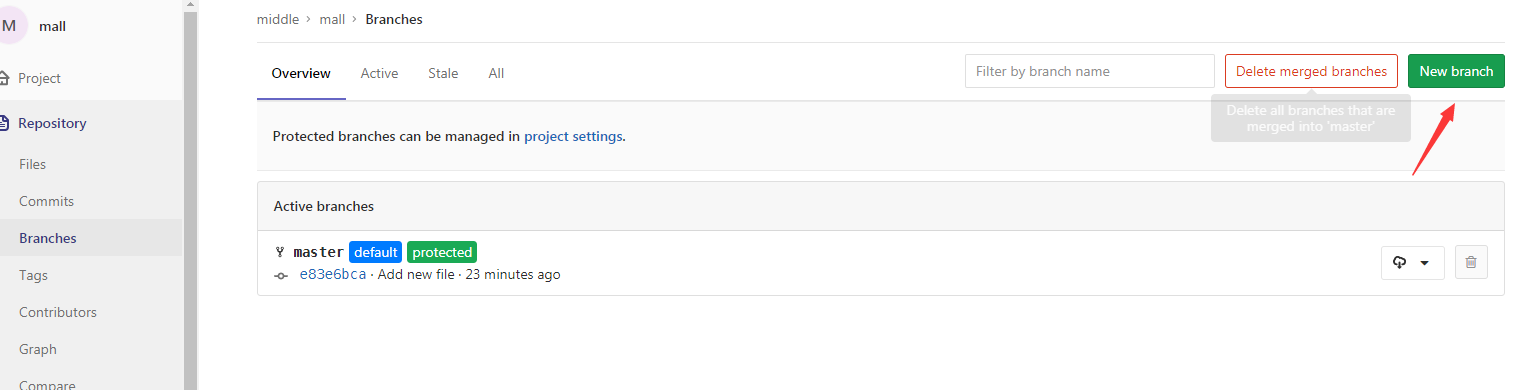


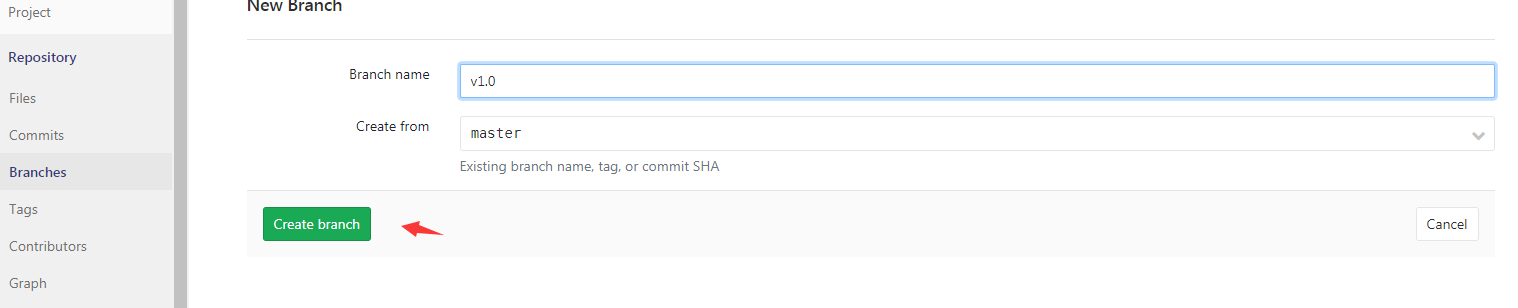
提交后显示如下：



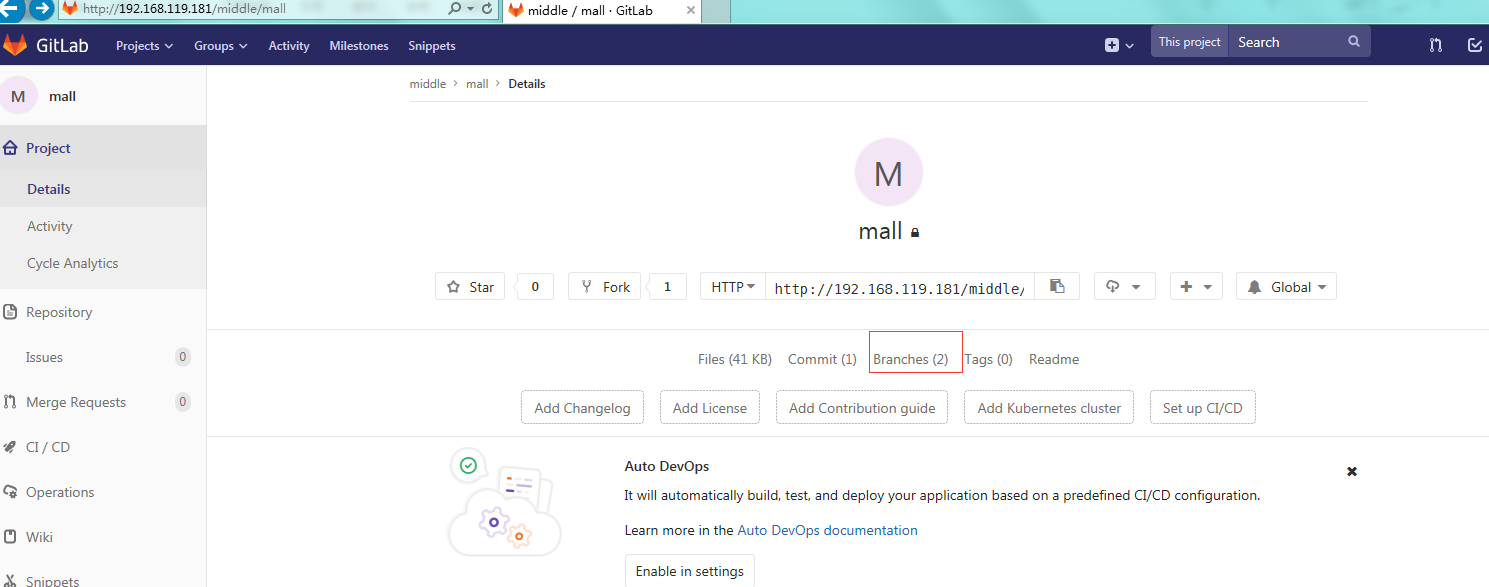
创建分之：

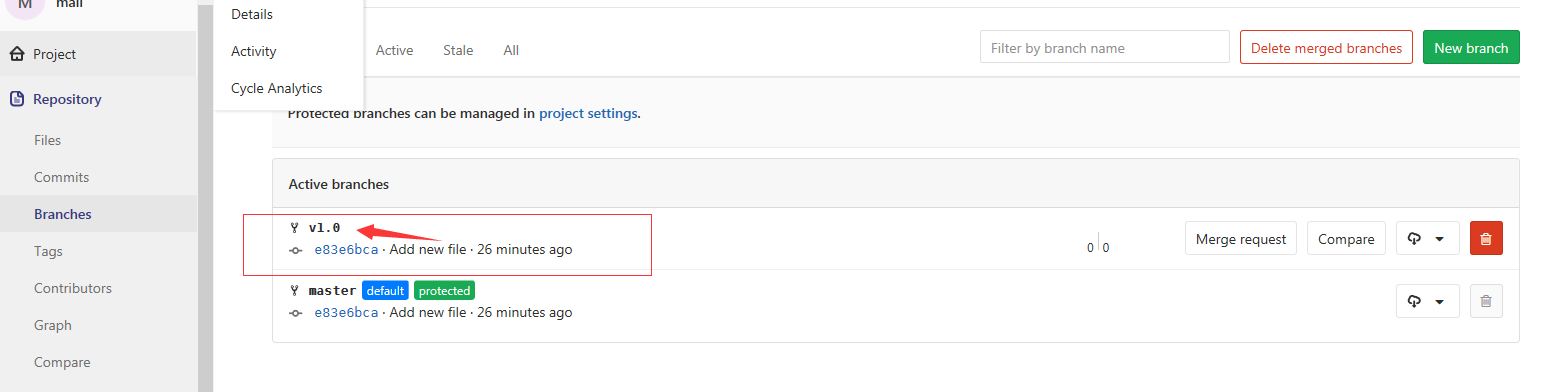




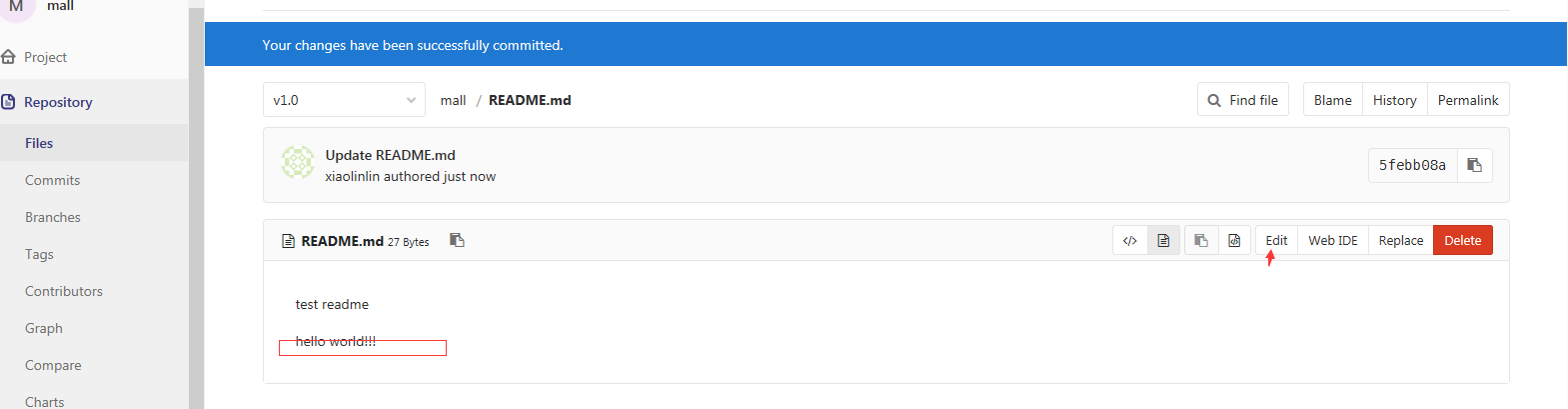


成员进入查看分之



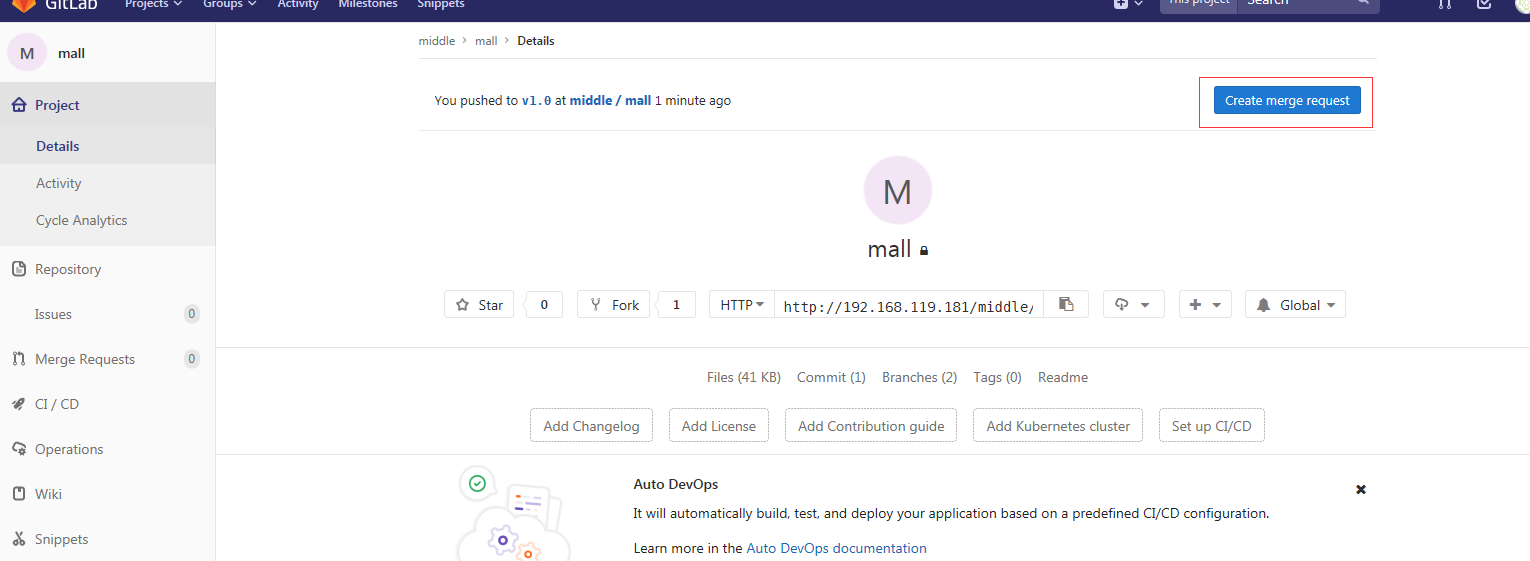


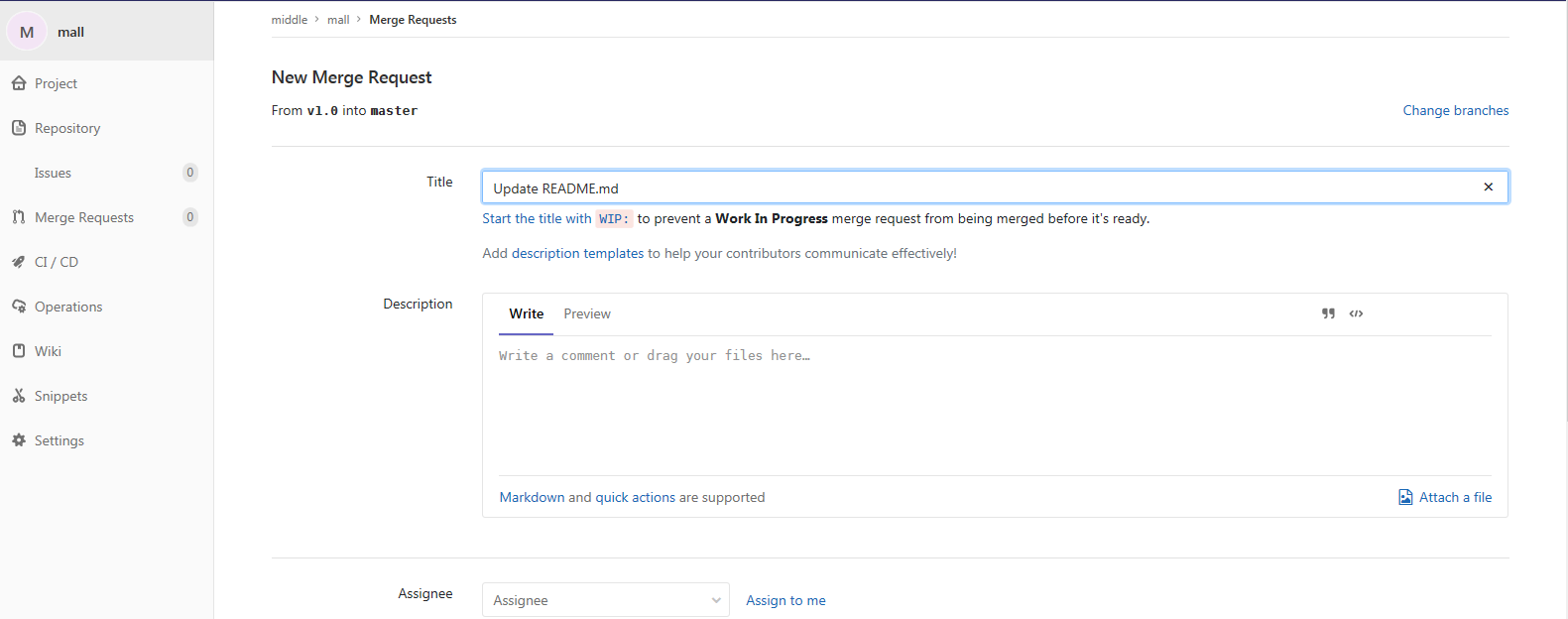
修改分之文件

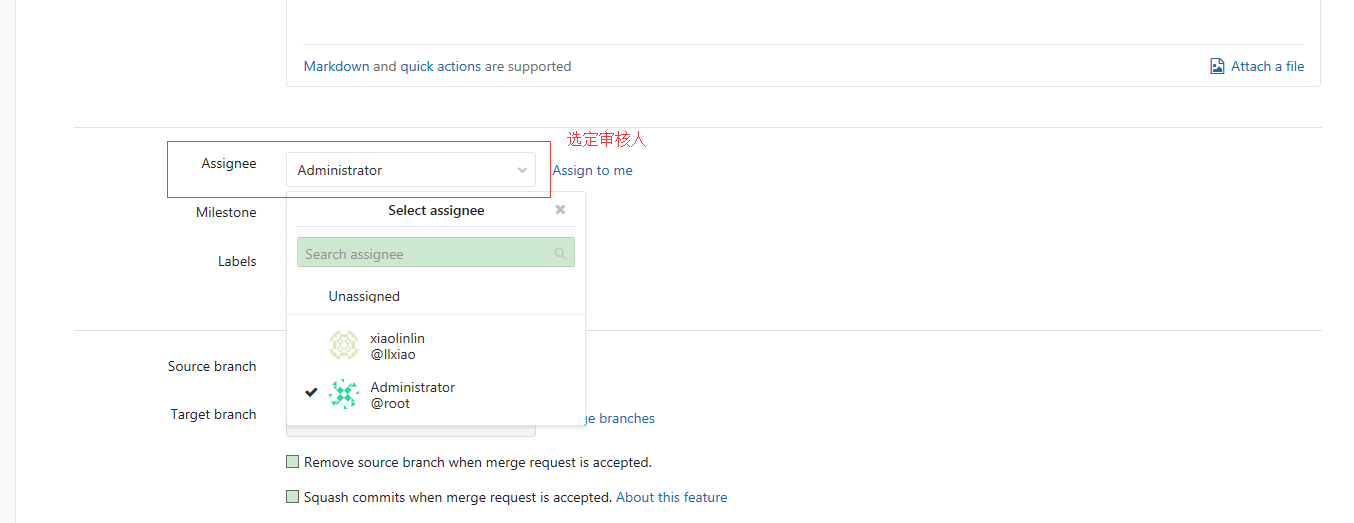


## 1.8合并分之到主干

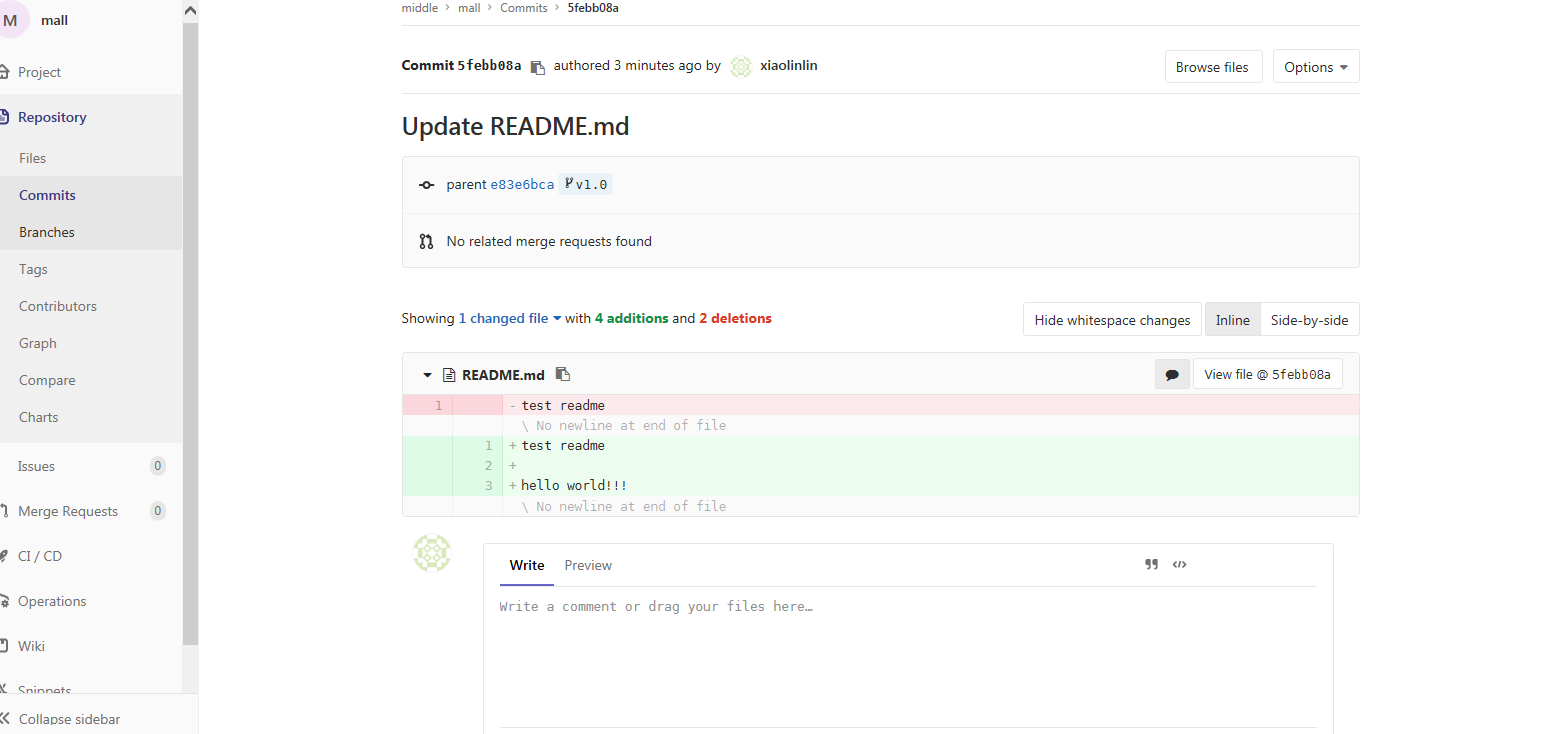
在上一步后修改文件后，回到项目列表，会有一个创建一个合并请求



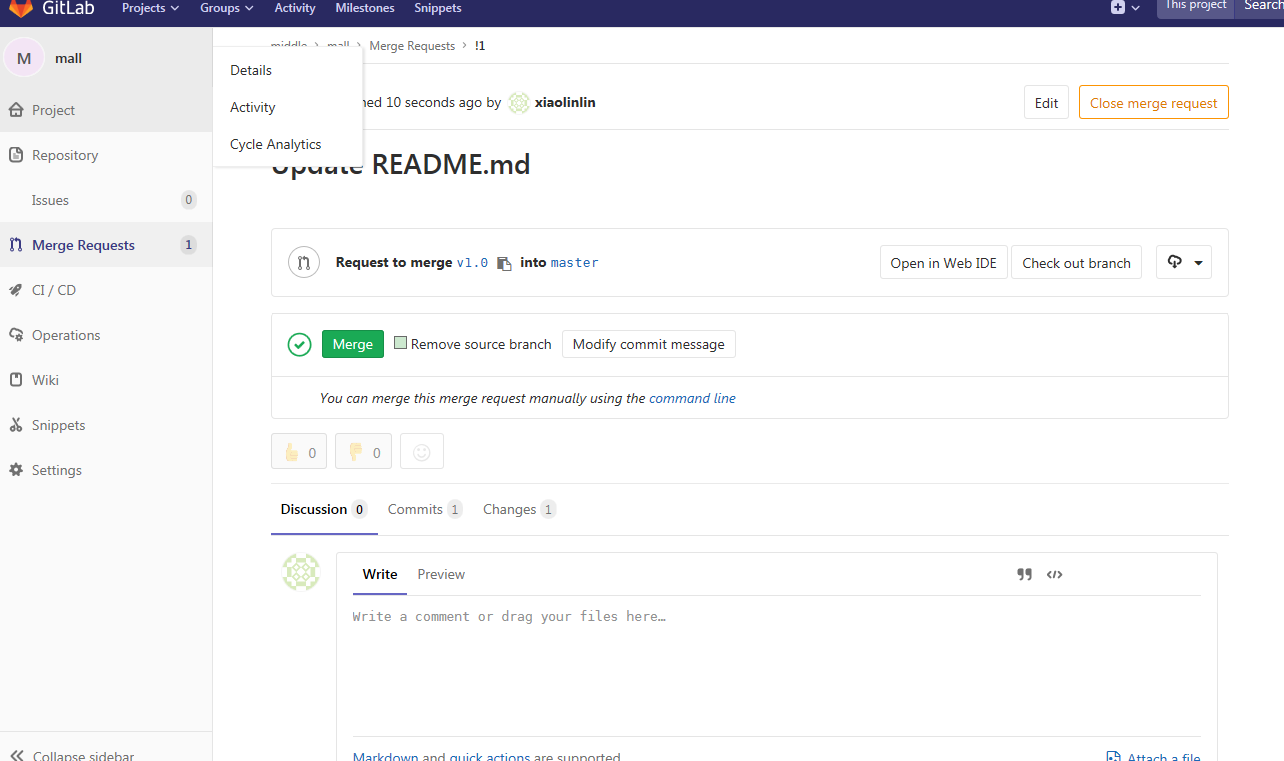




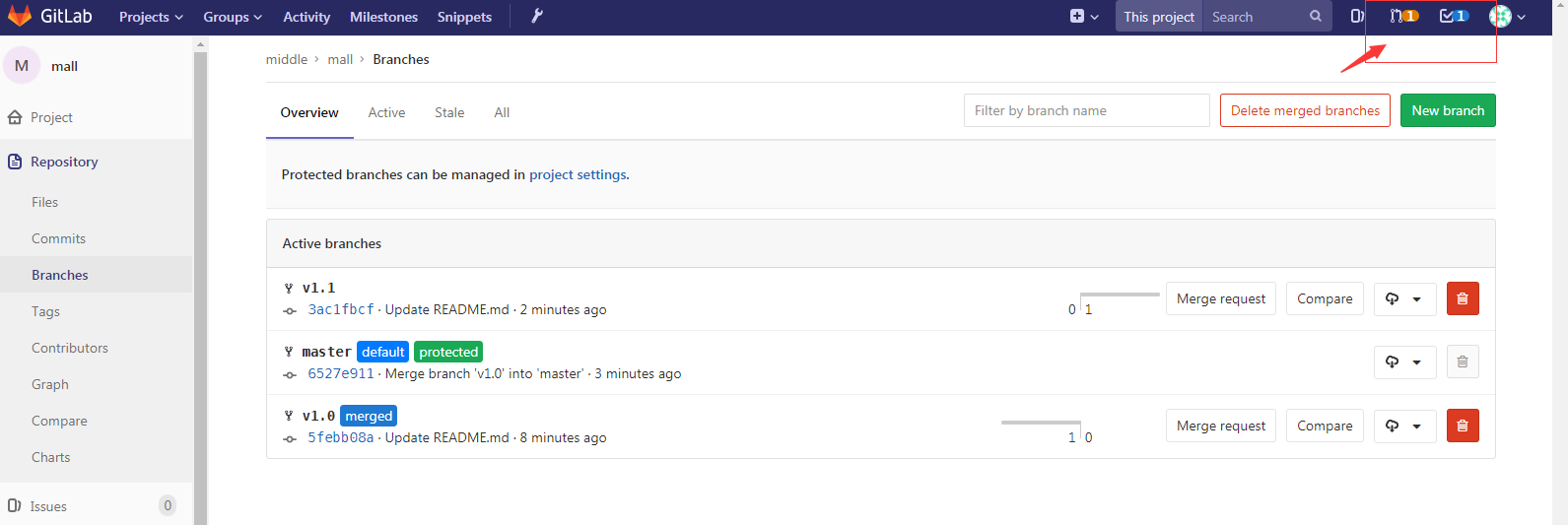
查看修改的文件内容



提交请求后：

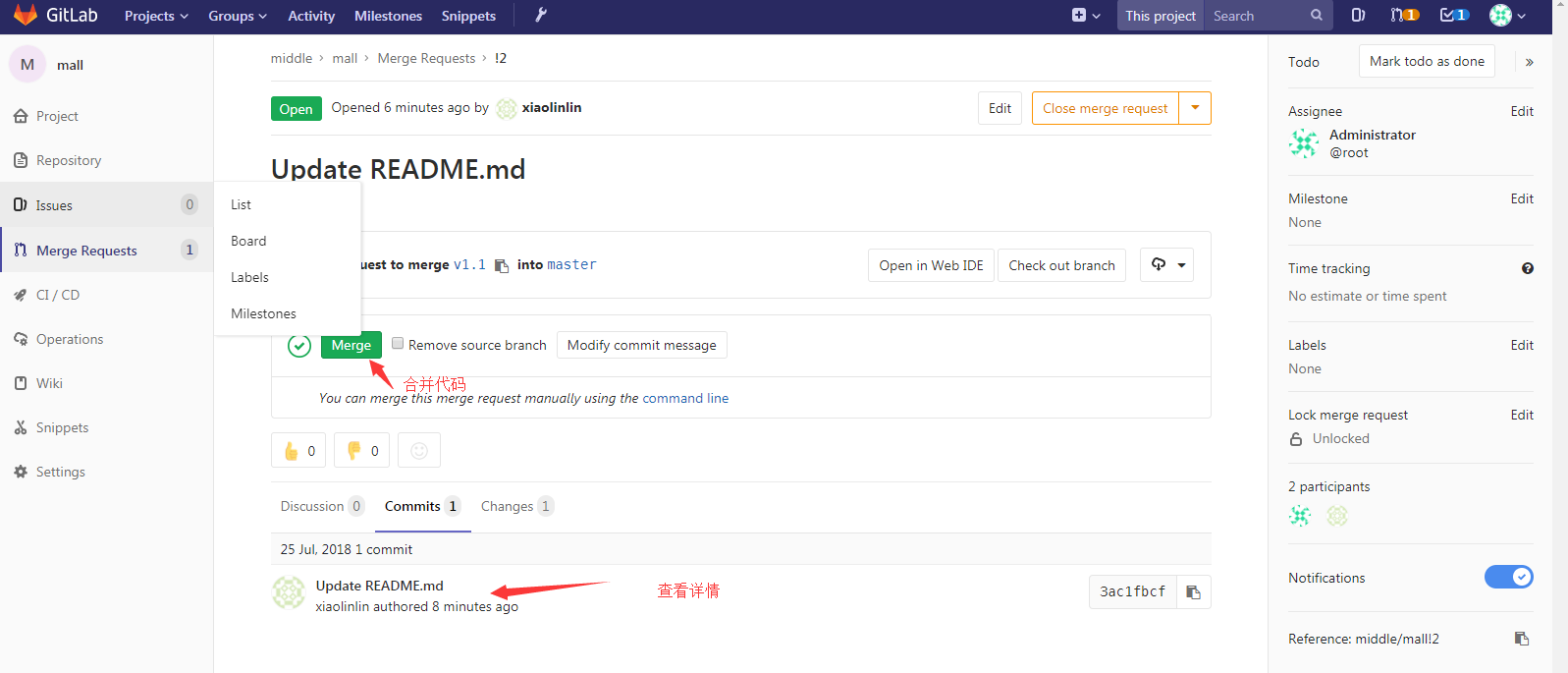


审核人会收到marge 请求

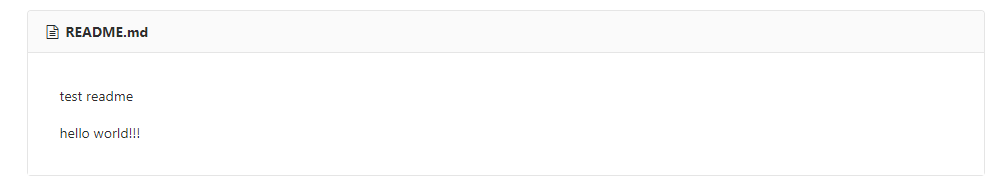


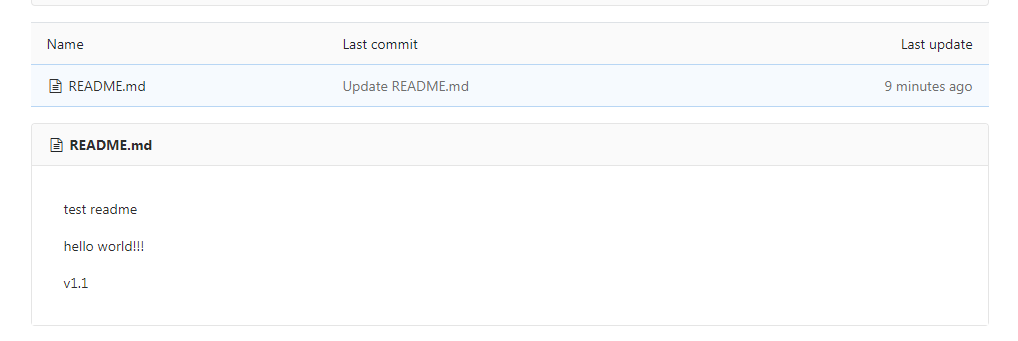
点击marge请求





合并前和合并后的内容对比：





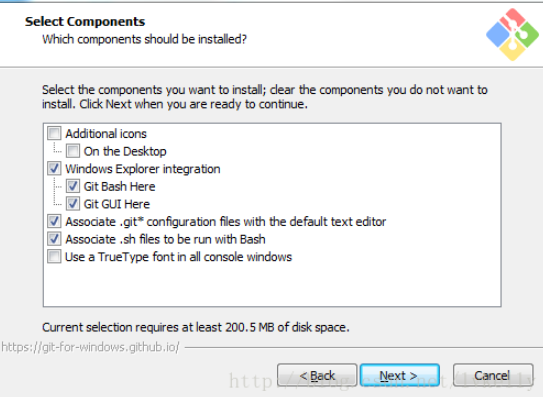
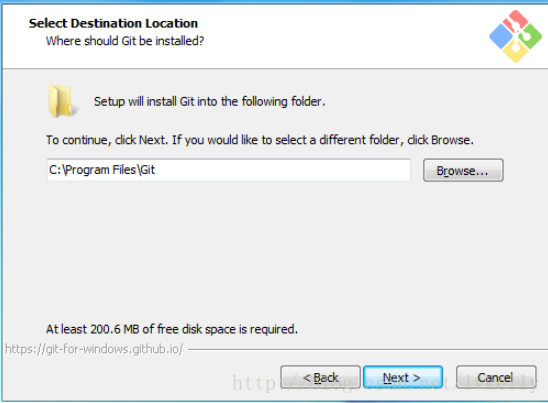
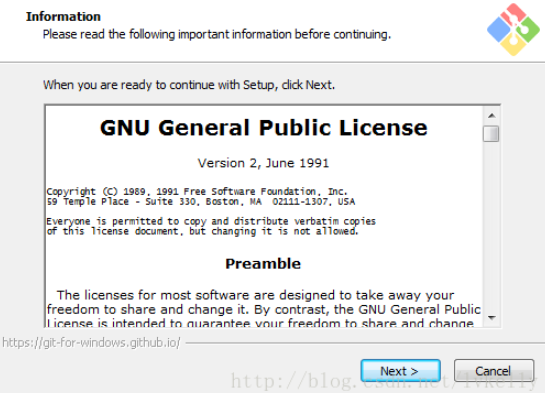
以上介绍完 分之和合并。另外还有一种fork方法，待续

# 二、eclipse+git客户端+gitlab使用

## 2.1 windows安装git 客户端

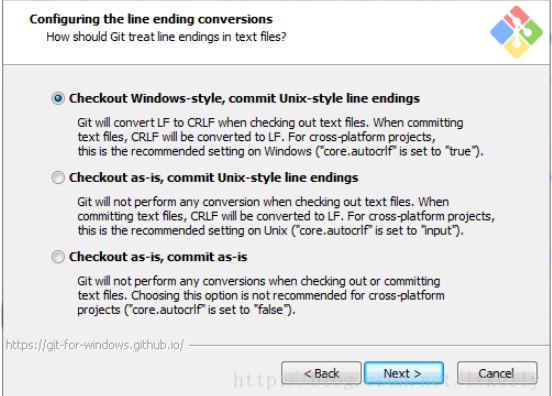
<https://gitforwindows.org/>

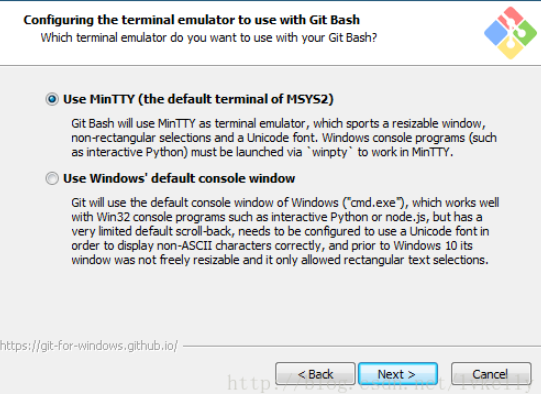


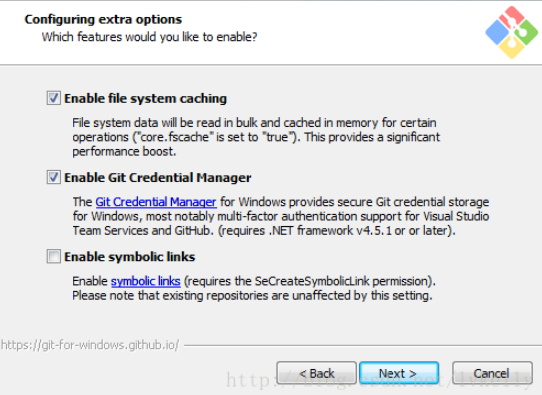


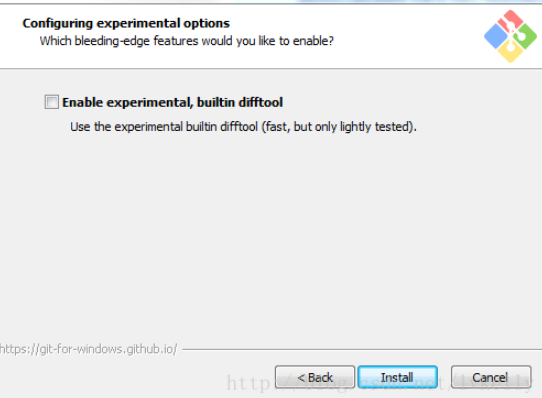


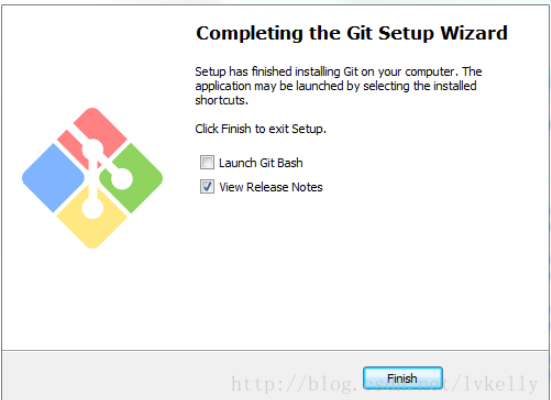


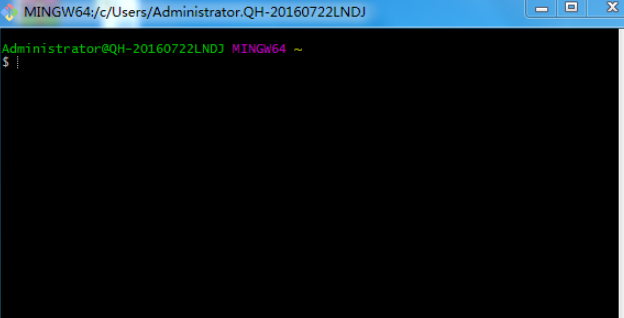


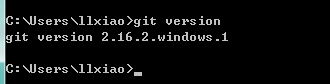








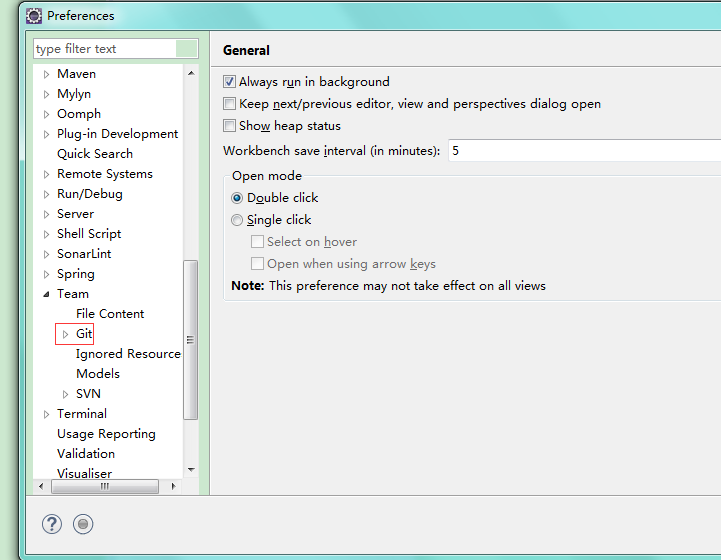




## 2.2 Eclipse使用git

### 2.2.1 eclipse git插件

Eclipse(4.7.3a)高版本已自己带git插件，请检查是否有git插件



如没有请在线安装egit插件:Help-eclipse marketplace

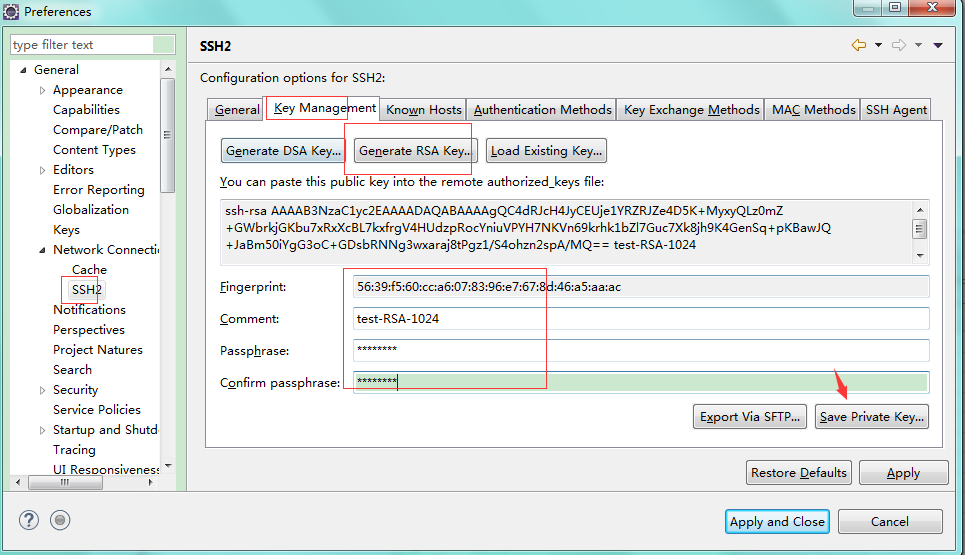


### 2.2.2 配置SSH证书，使用ssh登录gitlab(账号密码登录可忽略该步骤)

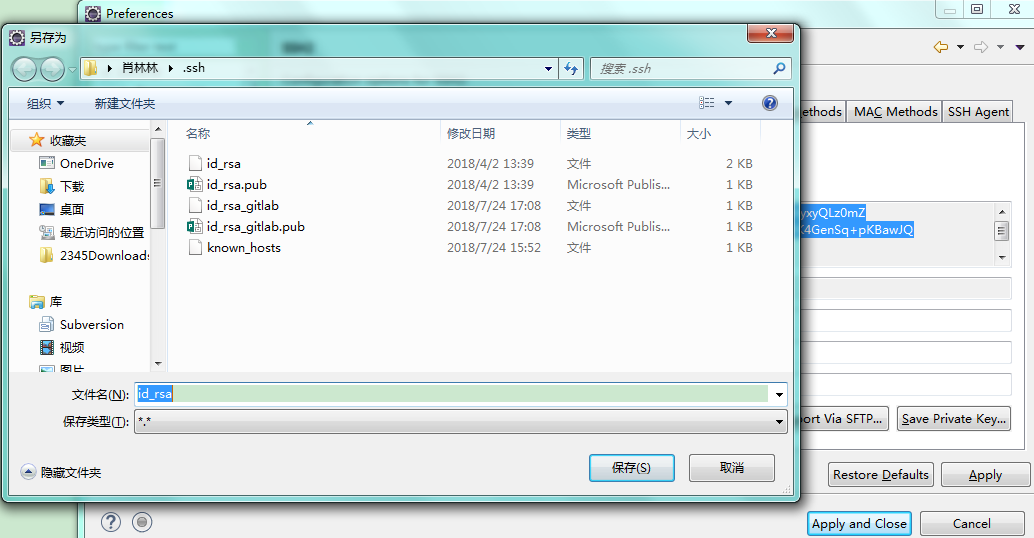
SSHkey证书登录，可以不用每次进行账号密码登录

1).生成证书

我们用的是eclipse自带的生成key的工具，windows->preferences->General->Network Connections->SSH2，点击SSH2

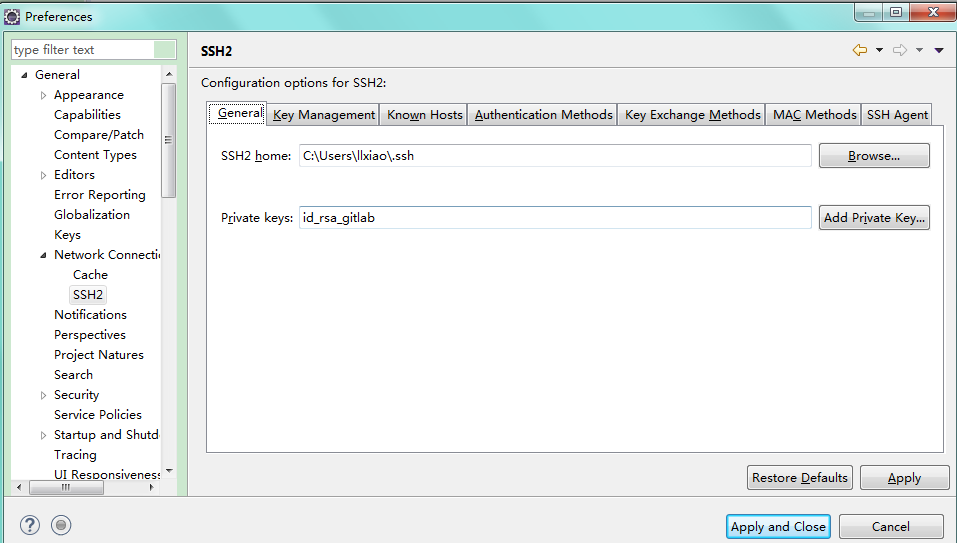


2)保存到用户.ssh目录下



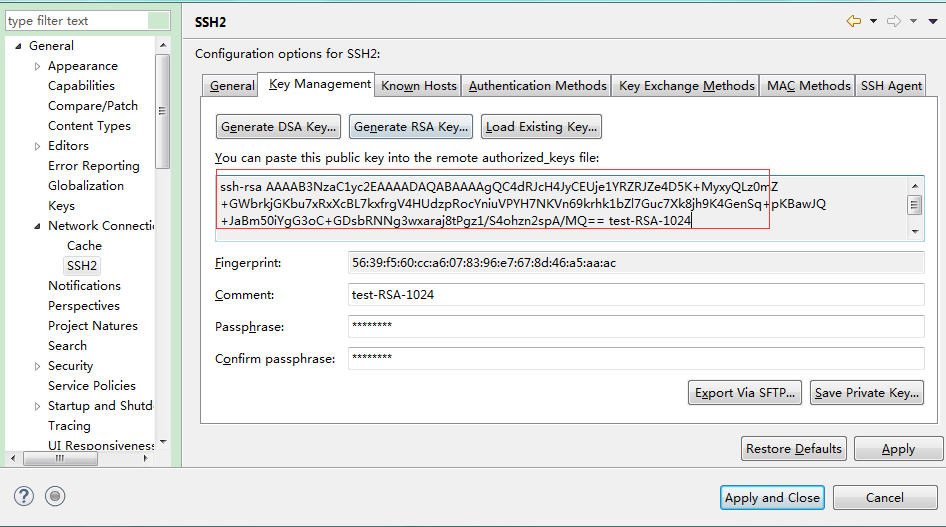
默认id\_rsa和id\_rsa.pub，我本地有github和gitlab的，所以重命名一个gitlab的

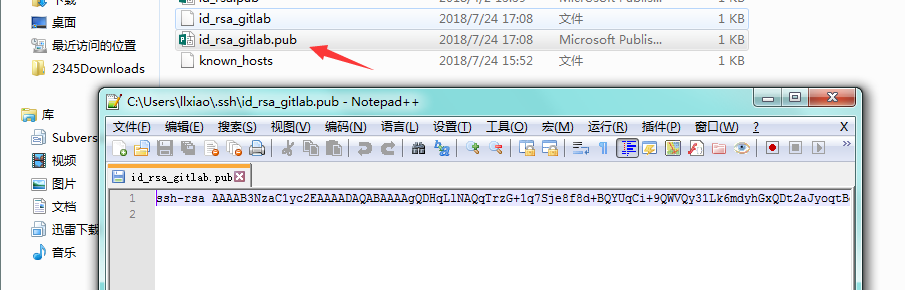
3)配置SSH home和证书，此步骤主要是区分多个证书的问题



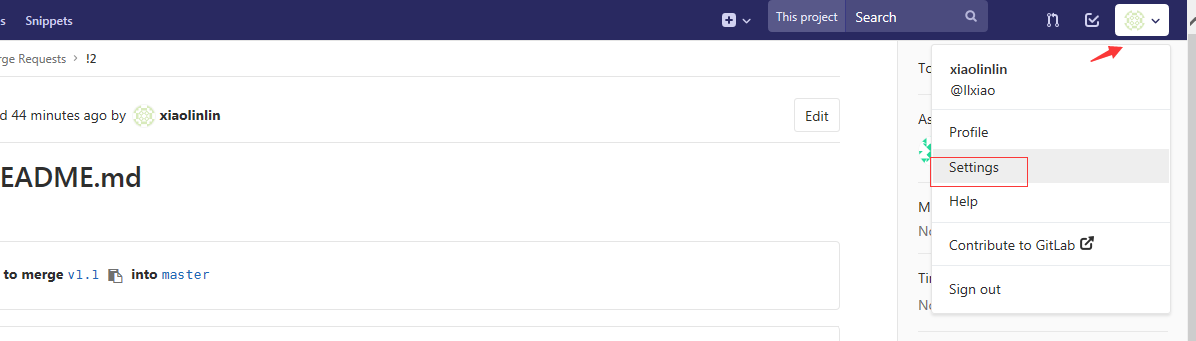
4)在gitlab上添加SSH key

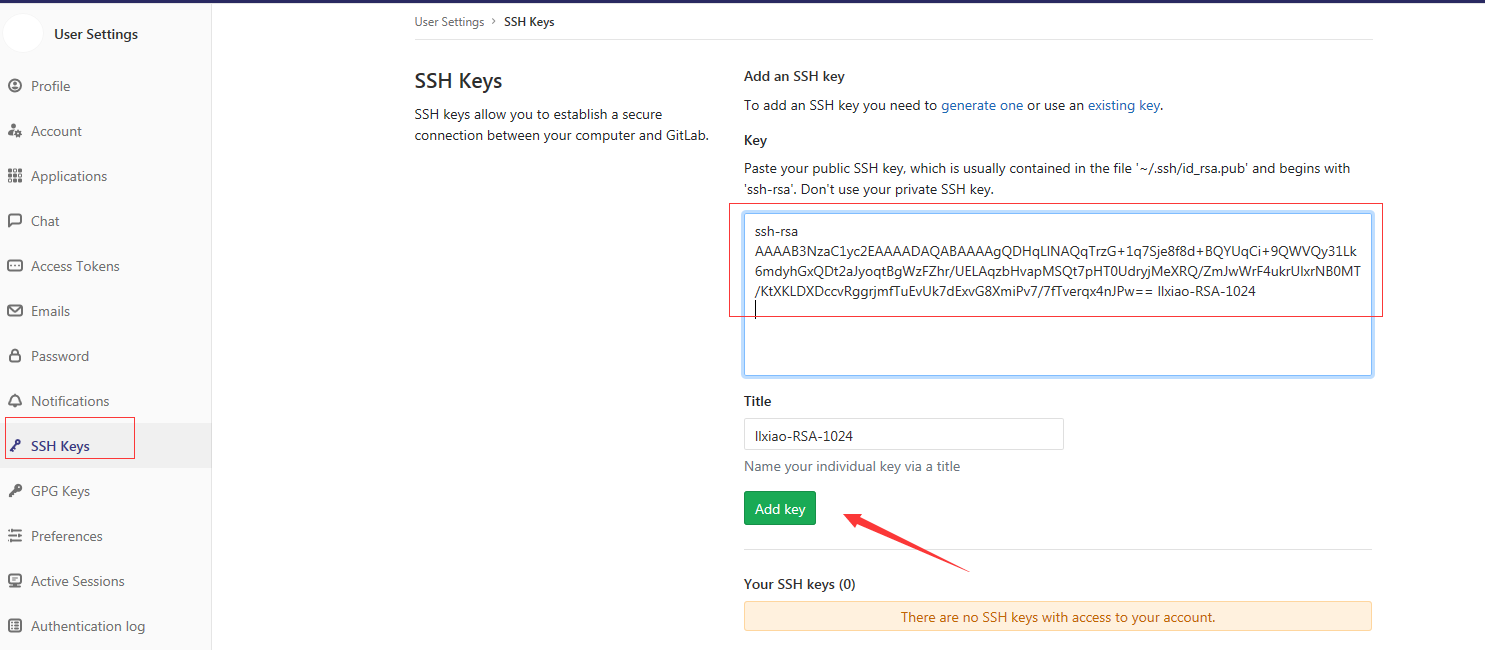
打开生成的id\_rsa.pub文件或者复制eclpse生成的那一串 key

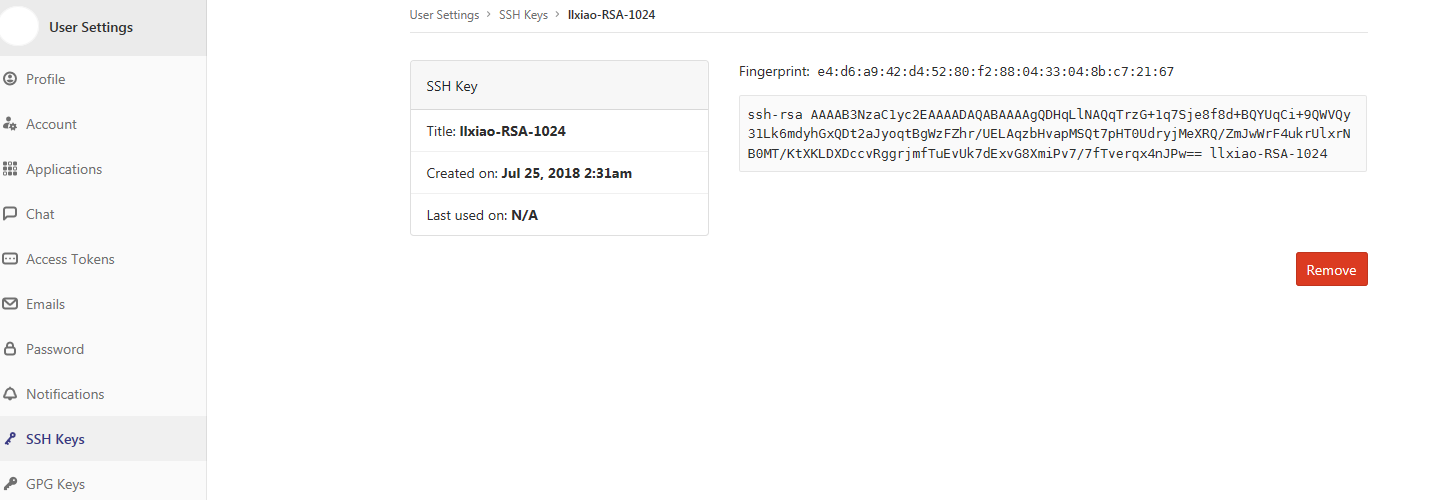




在gitlab上使用自己的账号登录配置

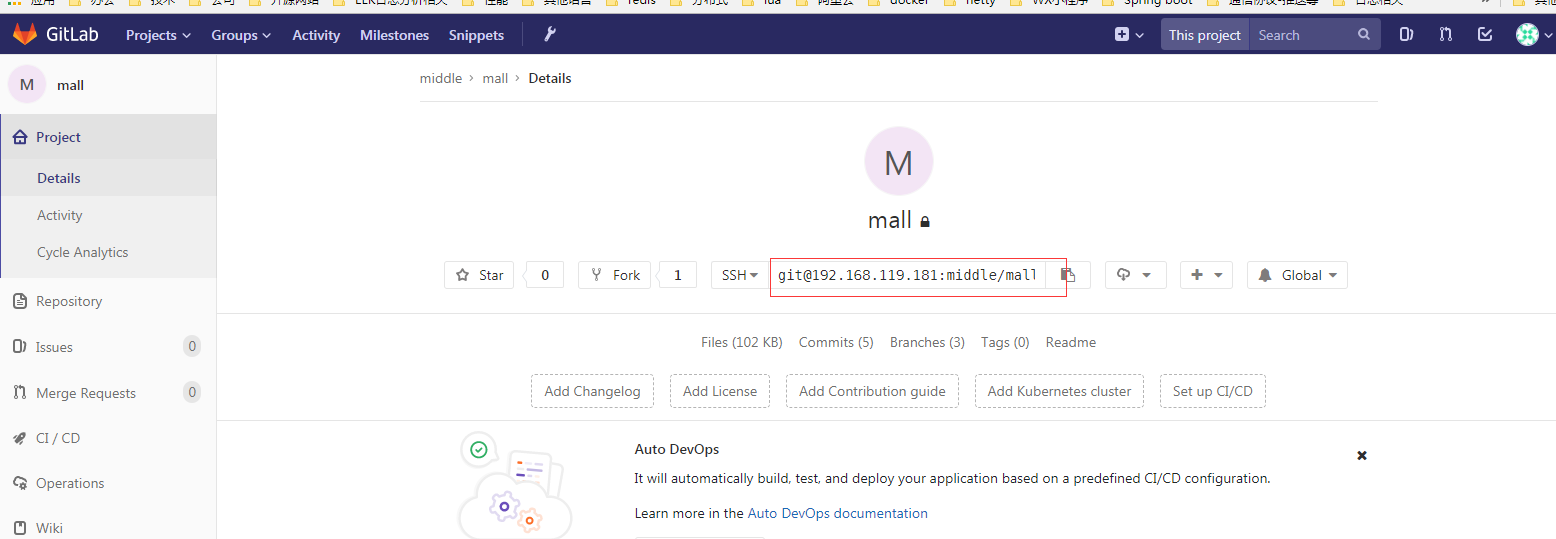






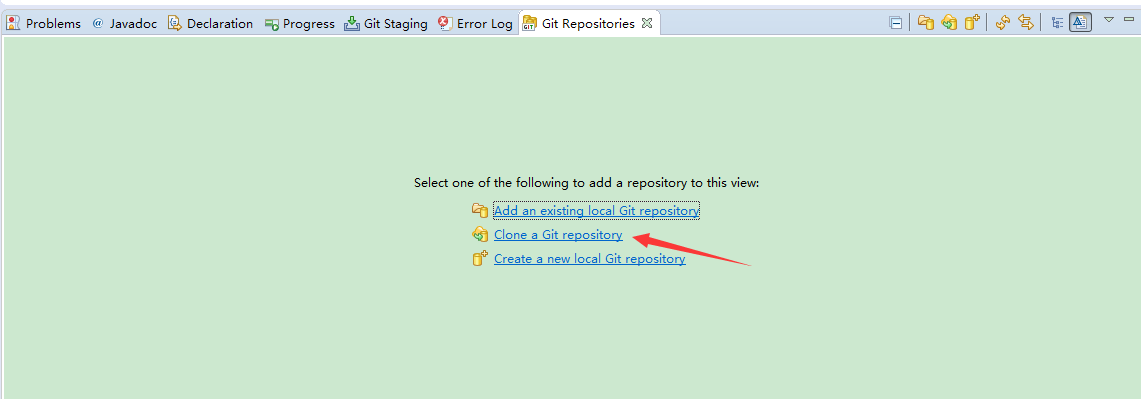
## 2.3 eclipse clone远程库代码

### 2.3.1 到gitlab复制主目录地址

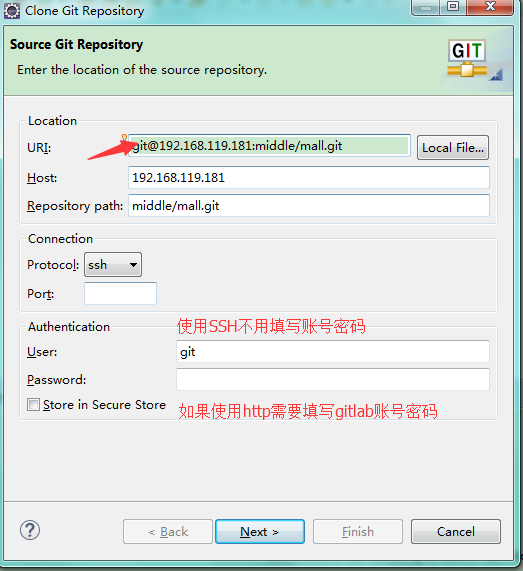


### 2.3.2 eclipse打开 git Repositories视图

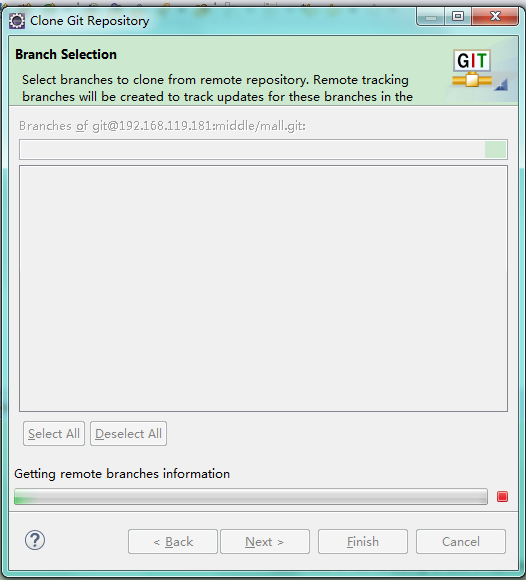
选择clone a git repository



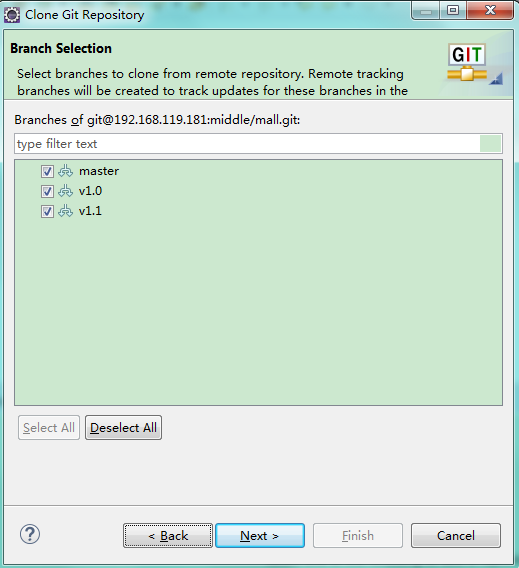
填入刚刚复制的gitlab仓库地址

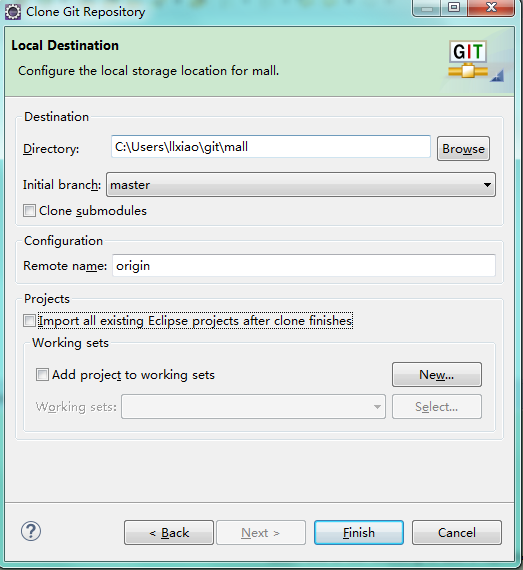


Next 等待结果

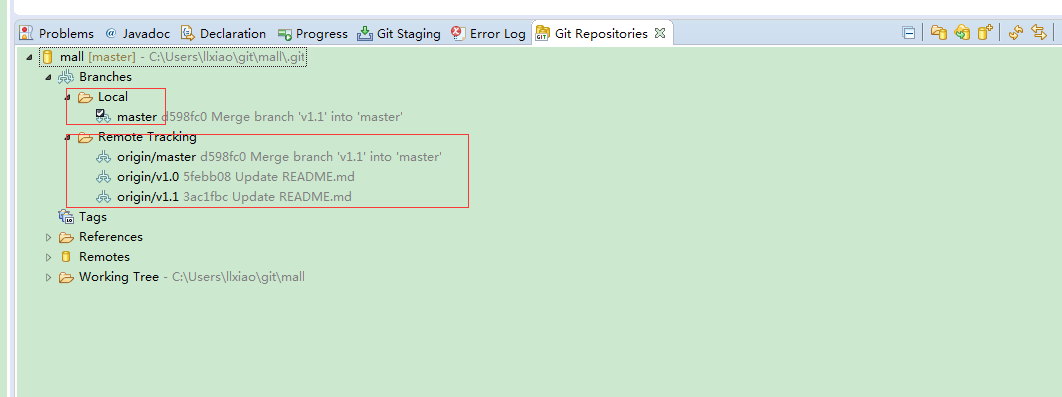


会有多个版本展示，可全部选择，异可选择某个分之代码



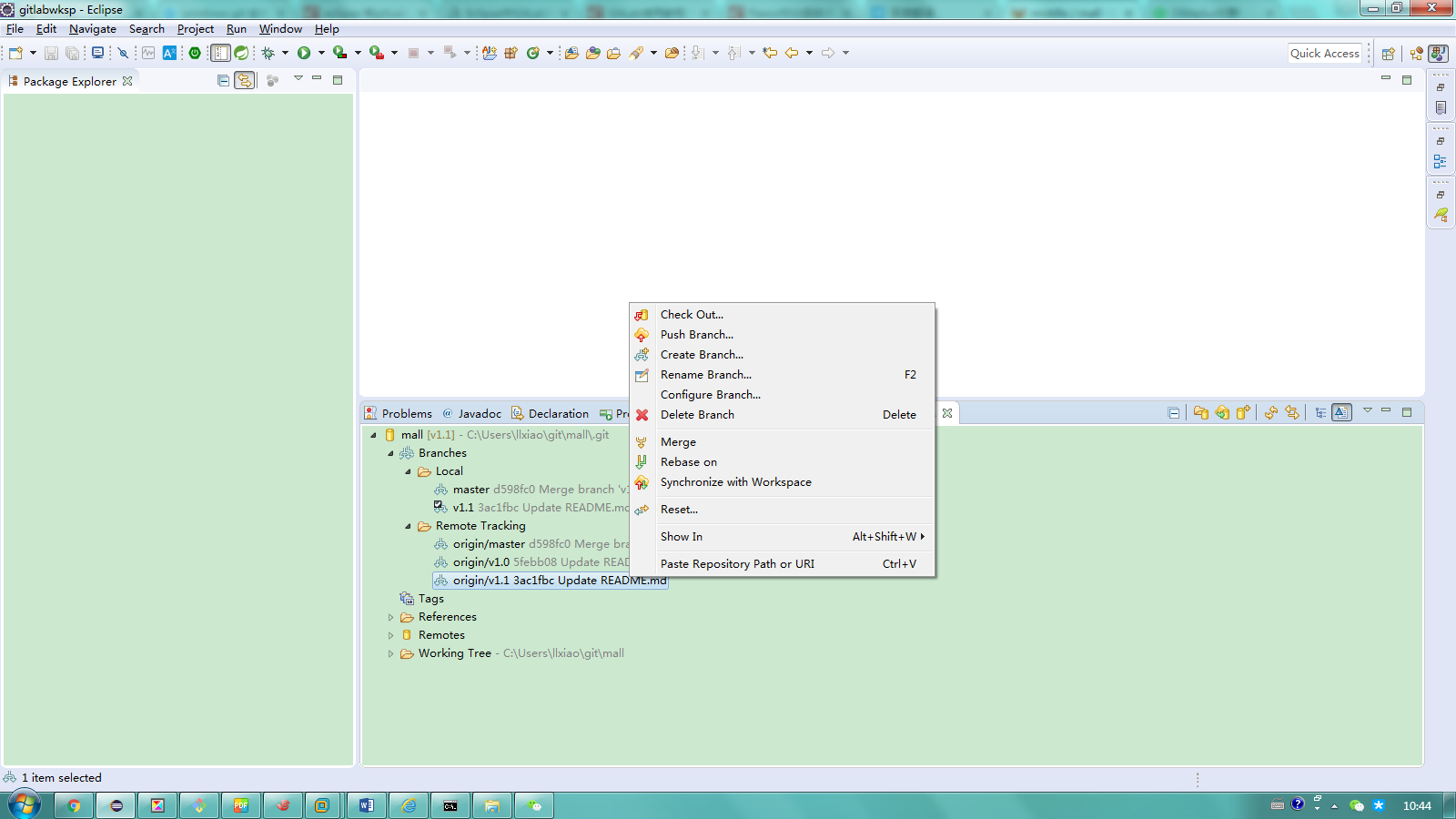


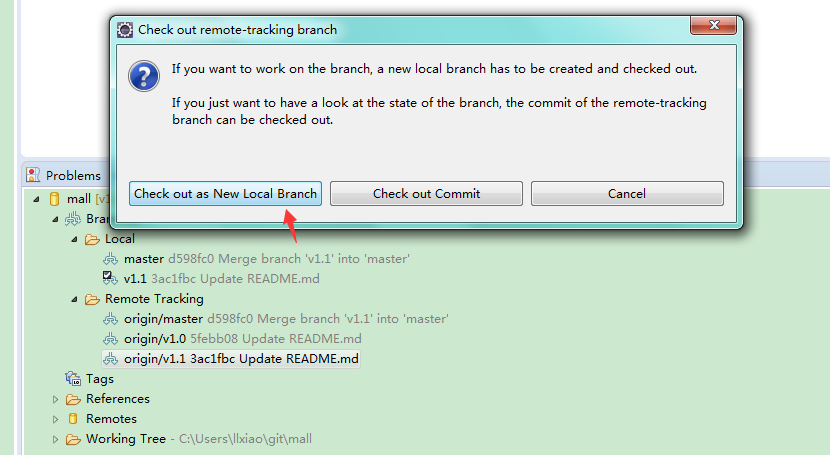
Finish等待加载



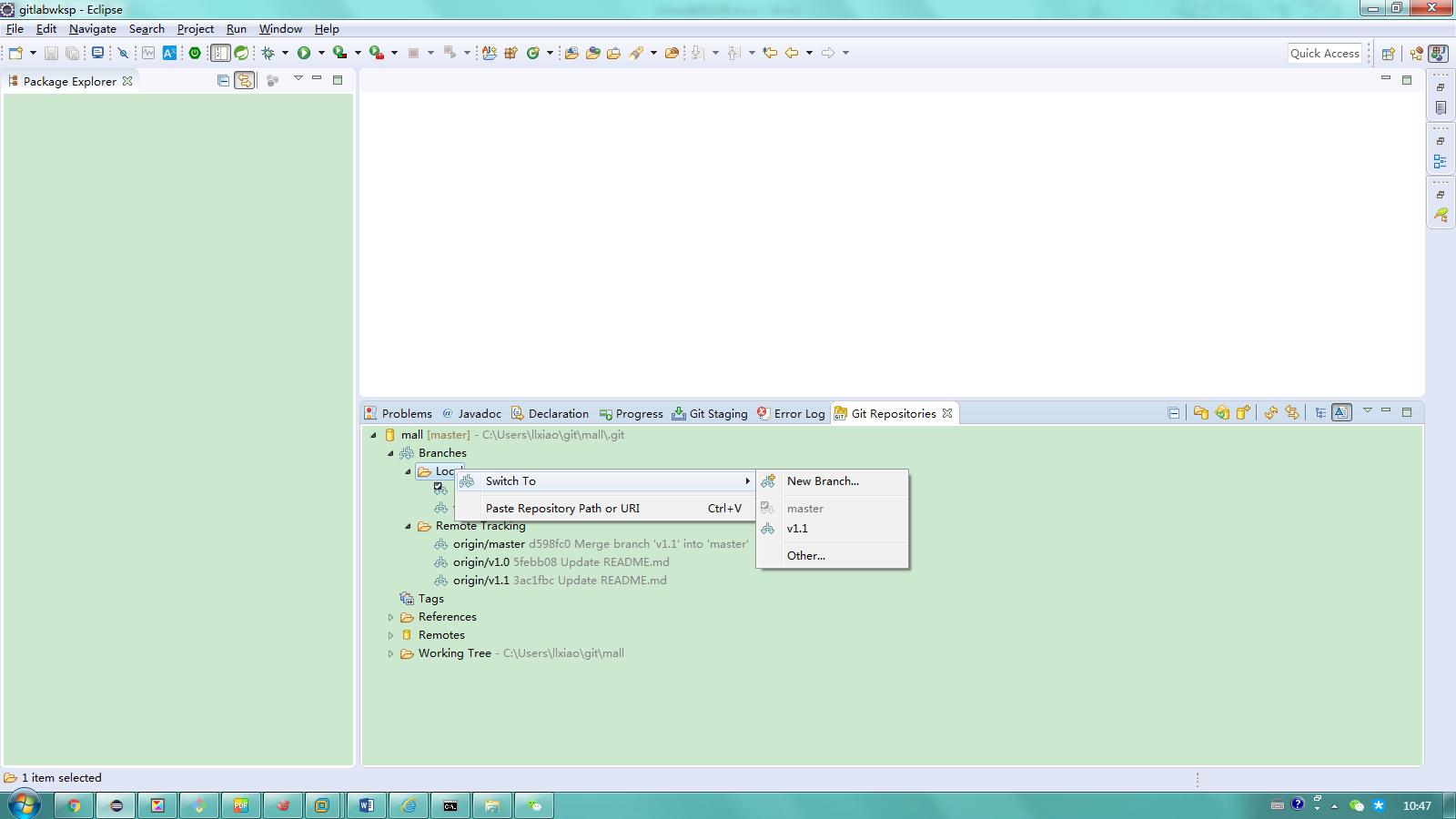
可以看到本地只有master，远程有三个分之

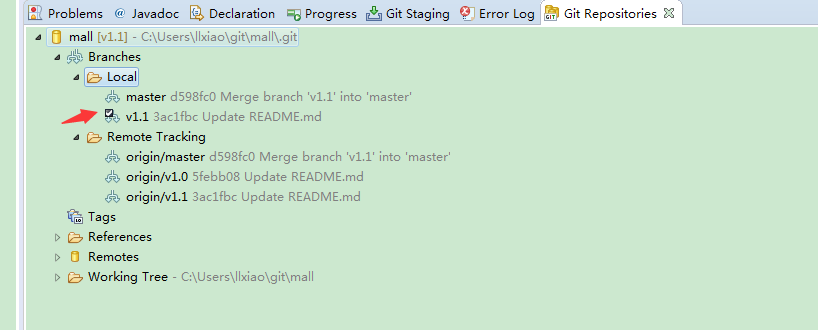
### 2.3.3 检出代码



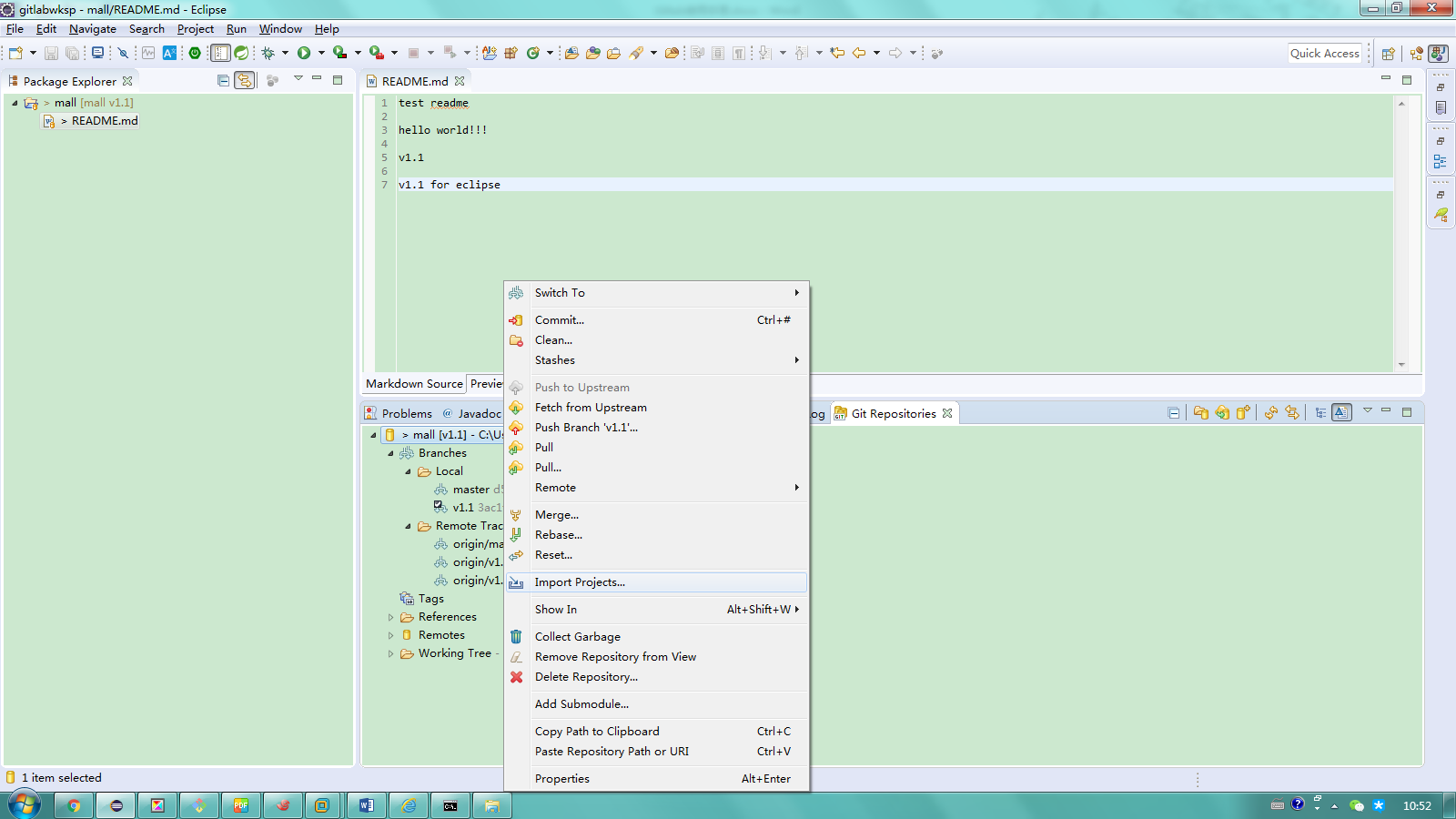


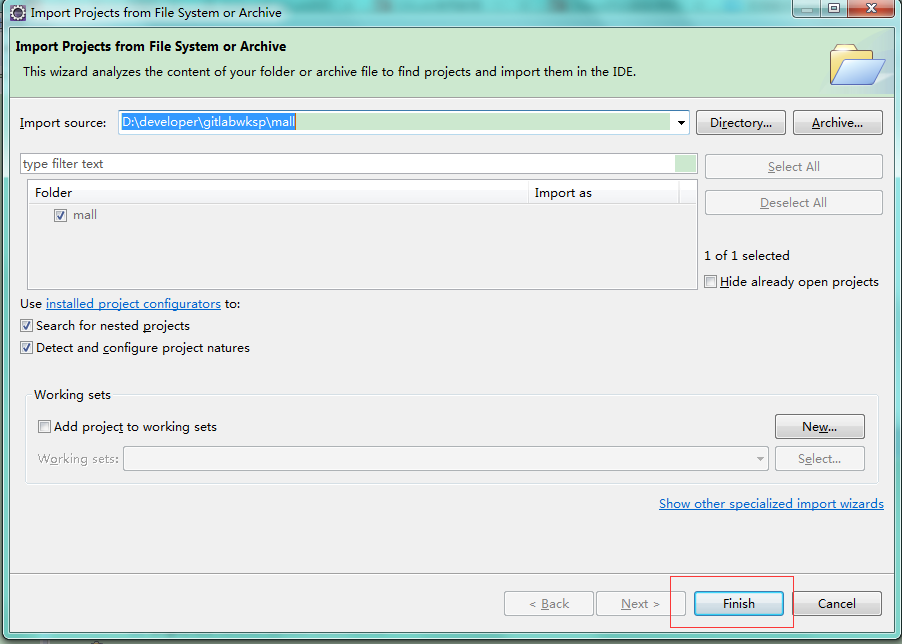
本地切换仓库



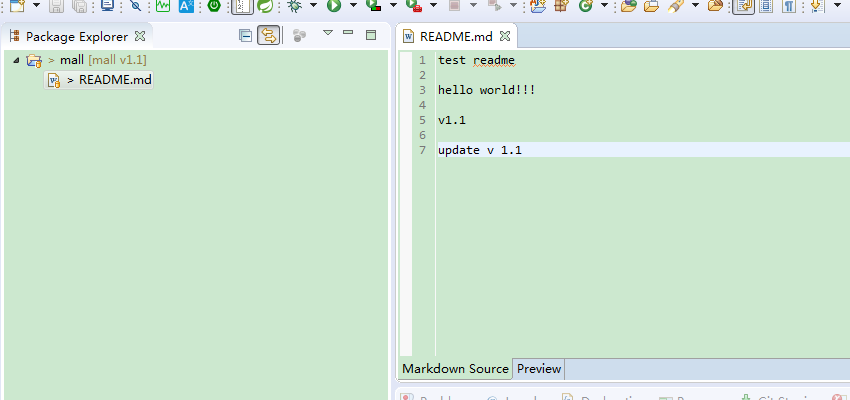


导入V1.1V的代码

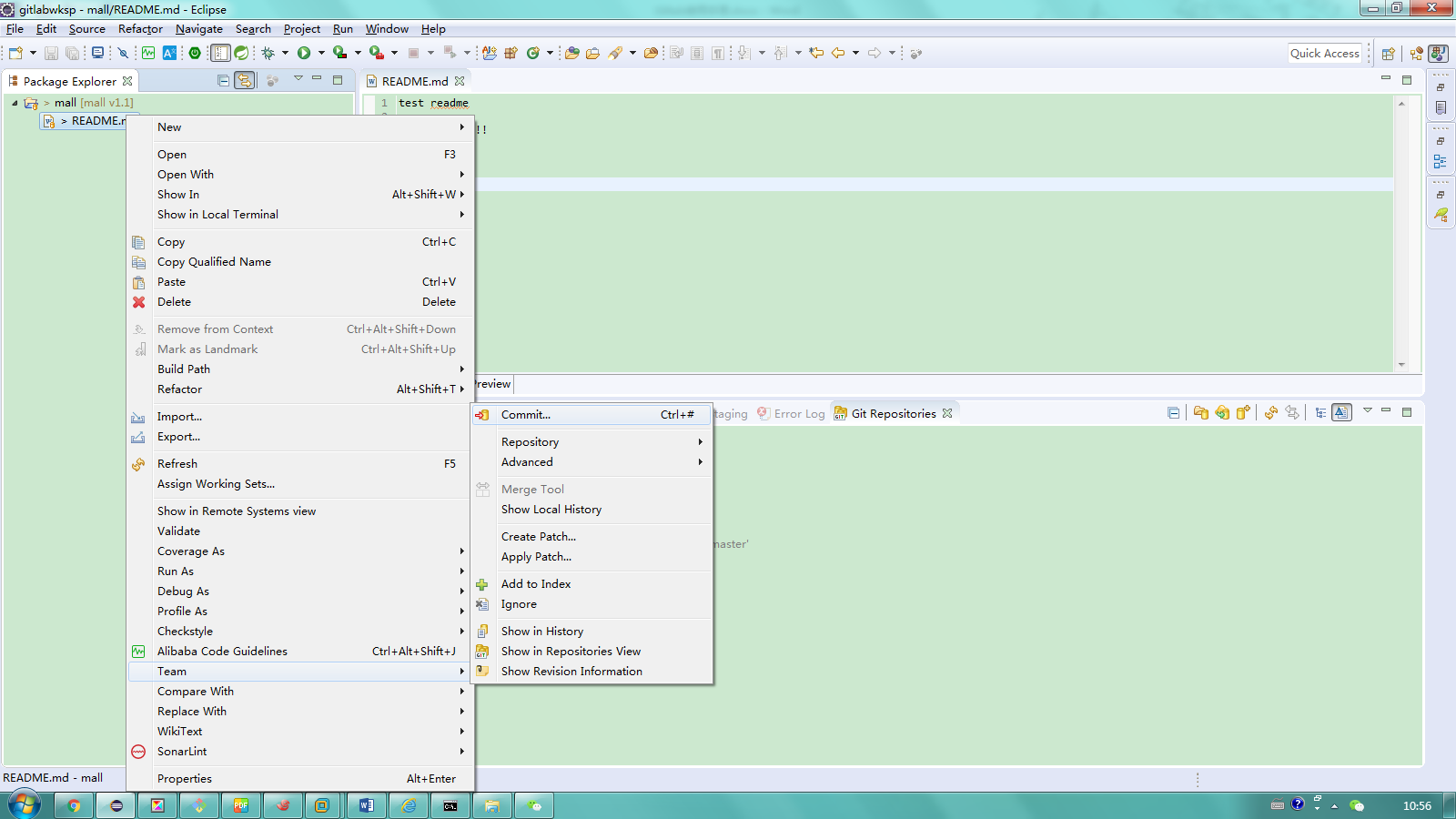


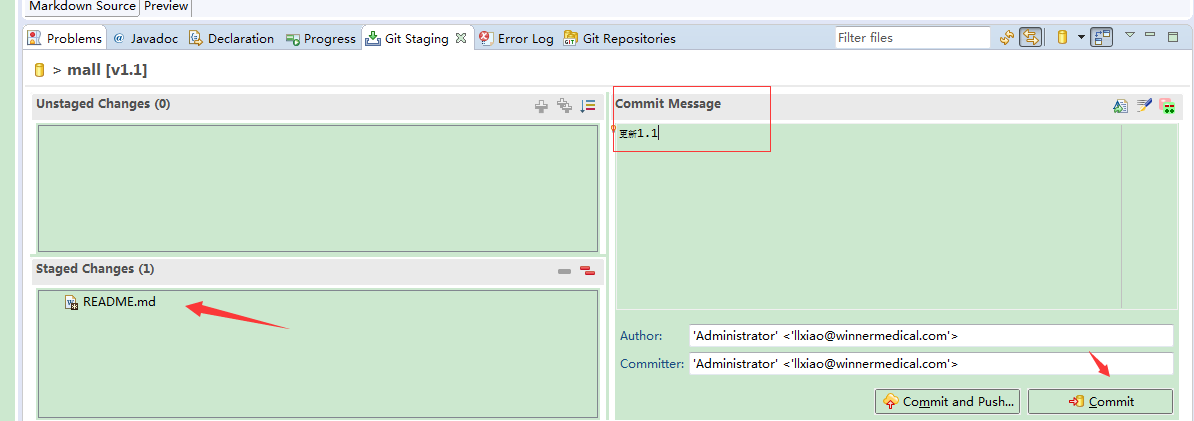


修改文件



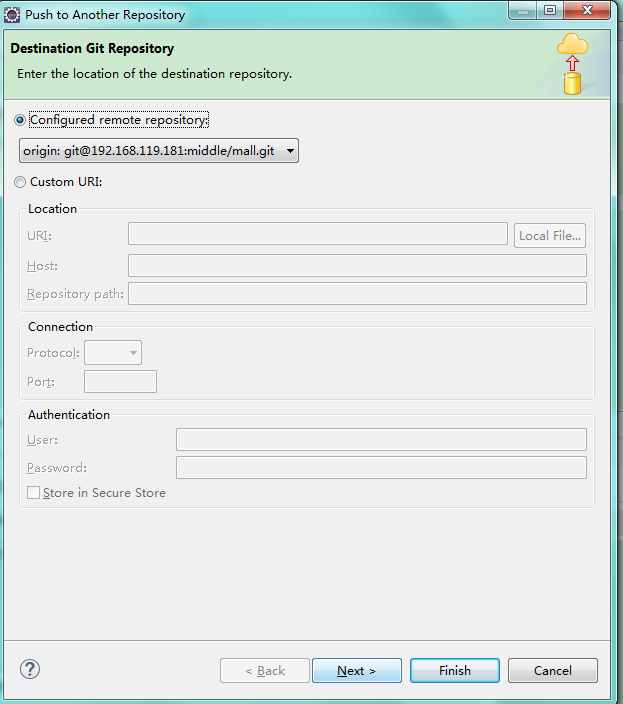
### 2.3.4 Commit提交修改



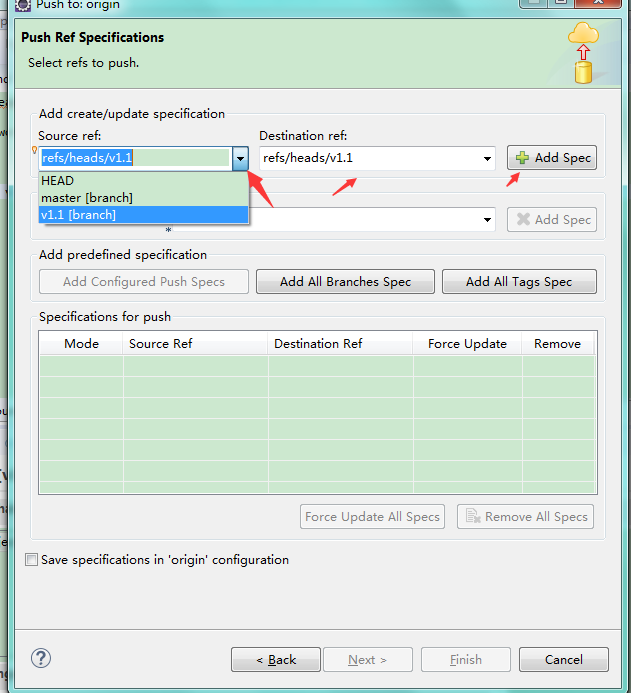


### 2.3.5 Push到远程

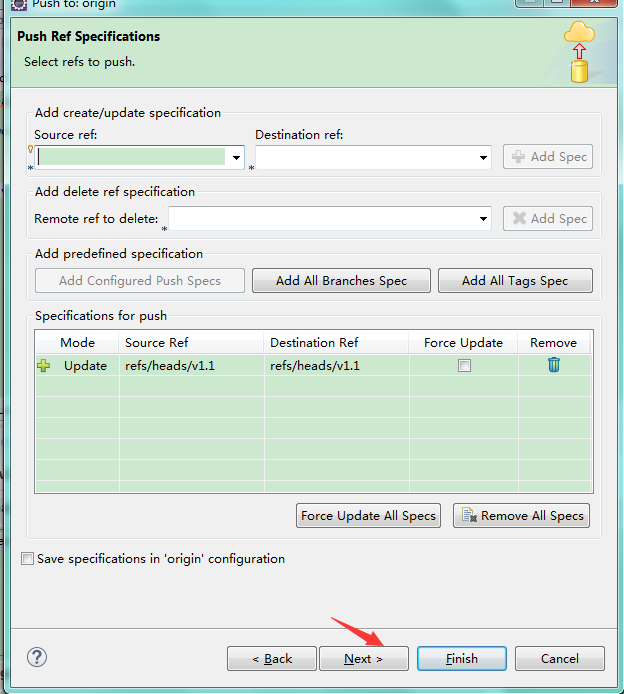




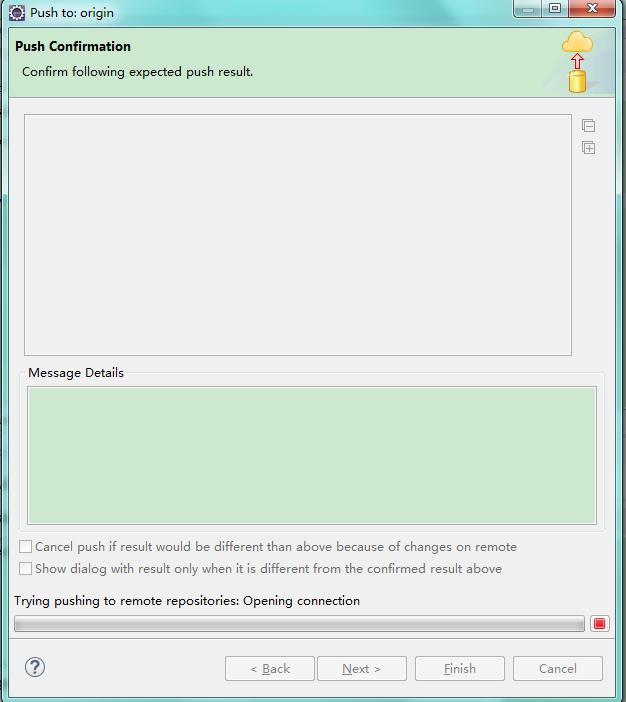
选择本地和远程的代码



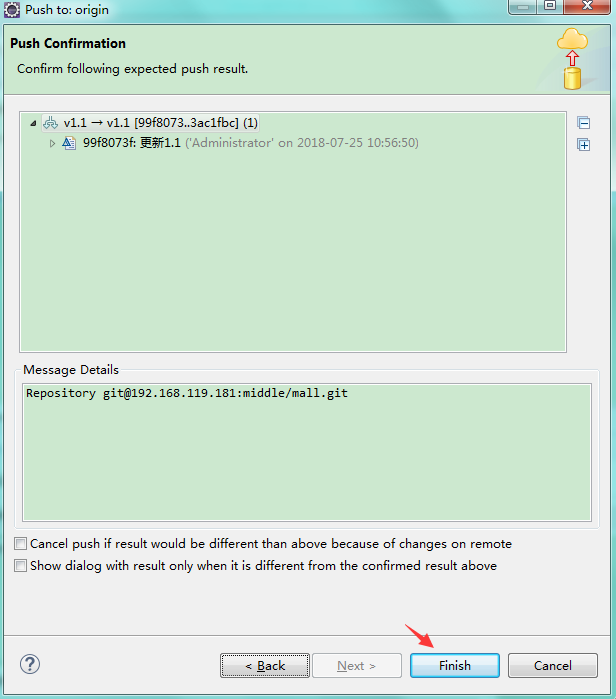
点击add spec



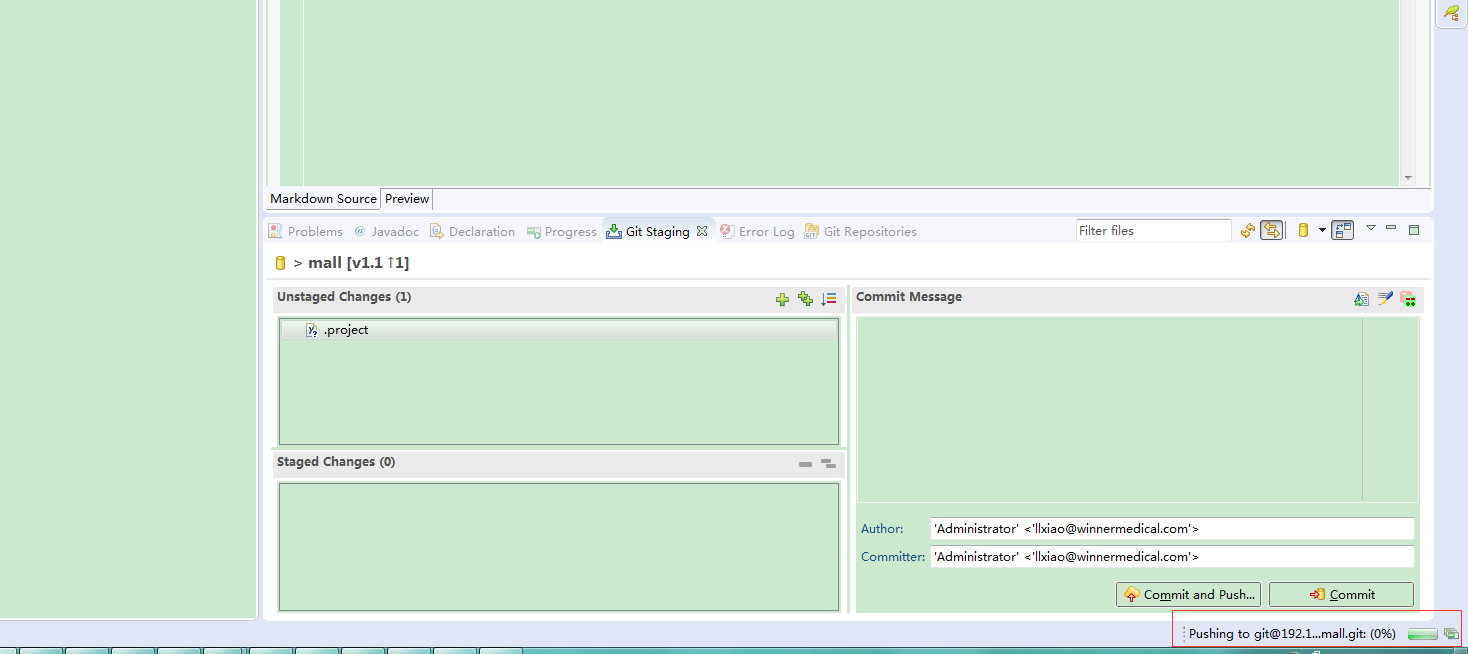
Next



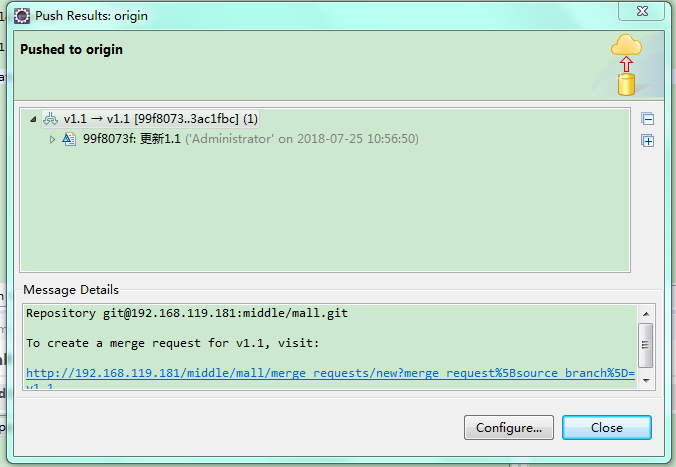
Finish



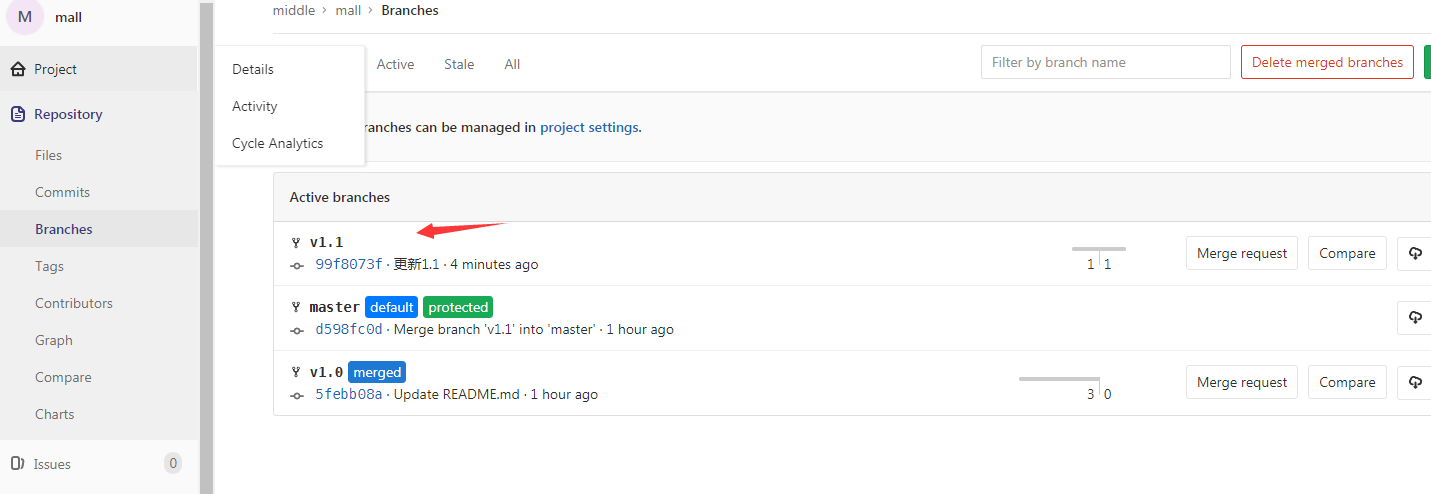
Push动作

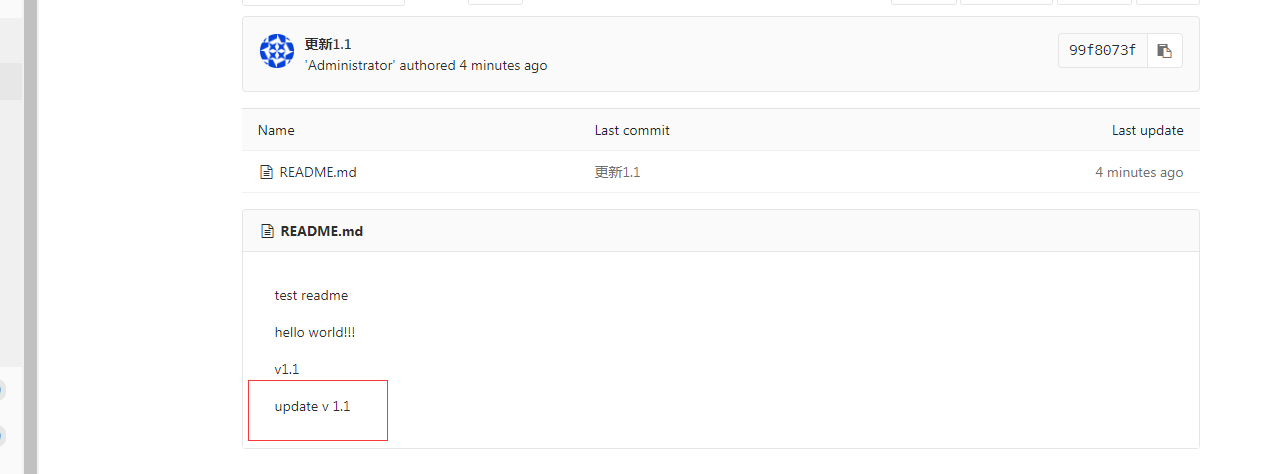


完成之后

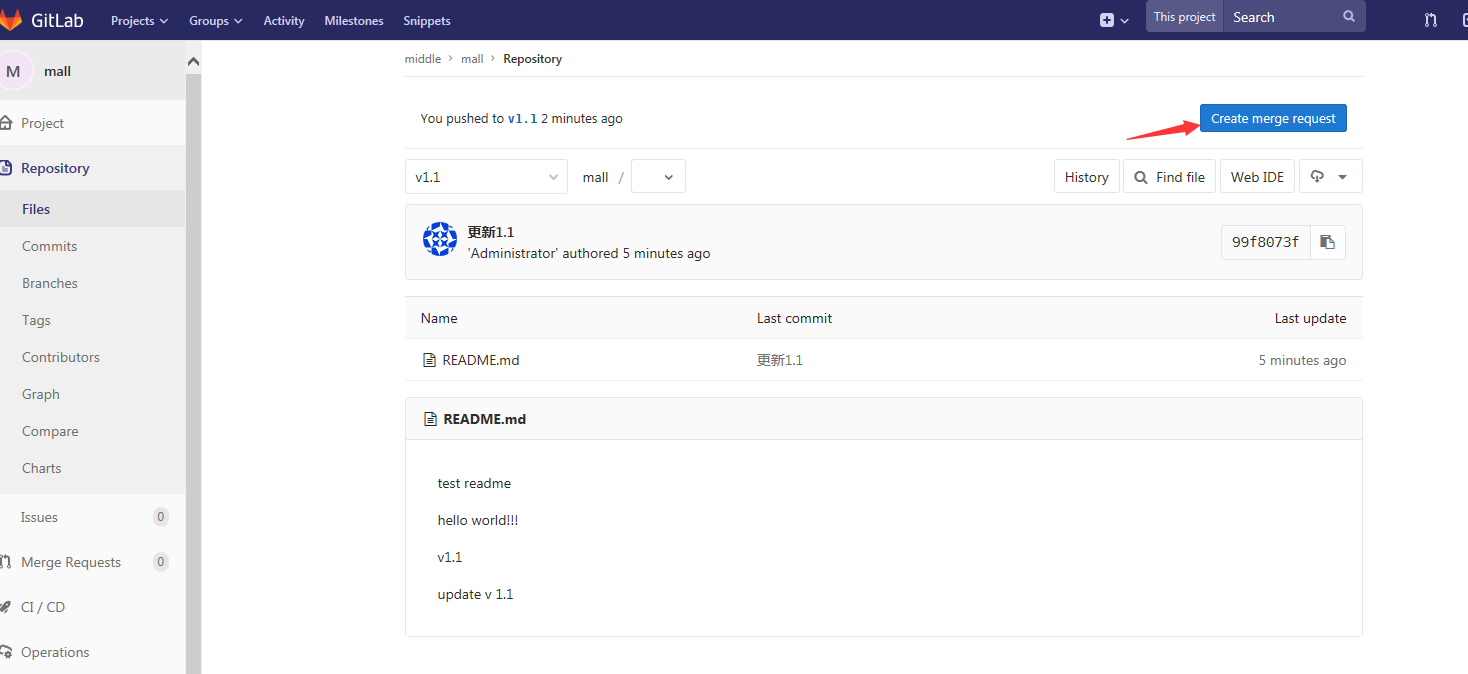


### 2.3.6 回到gitlab查看1.1的分之更新



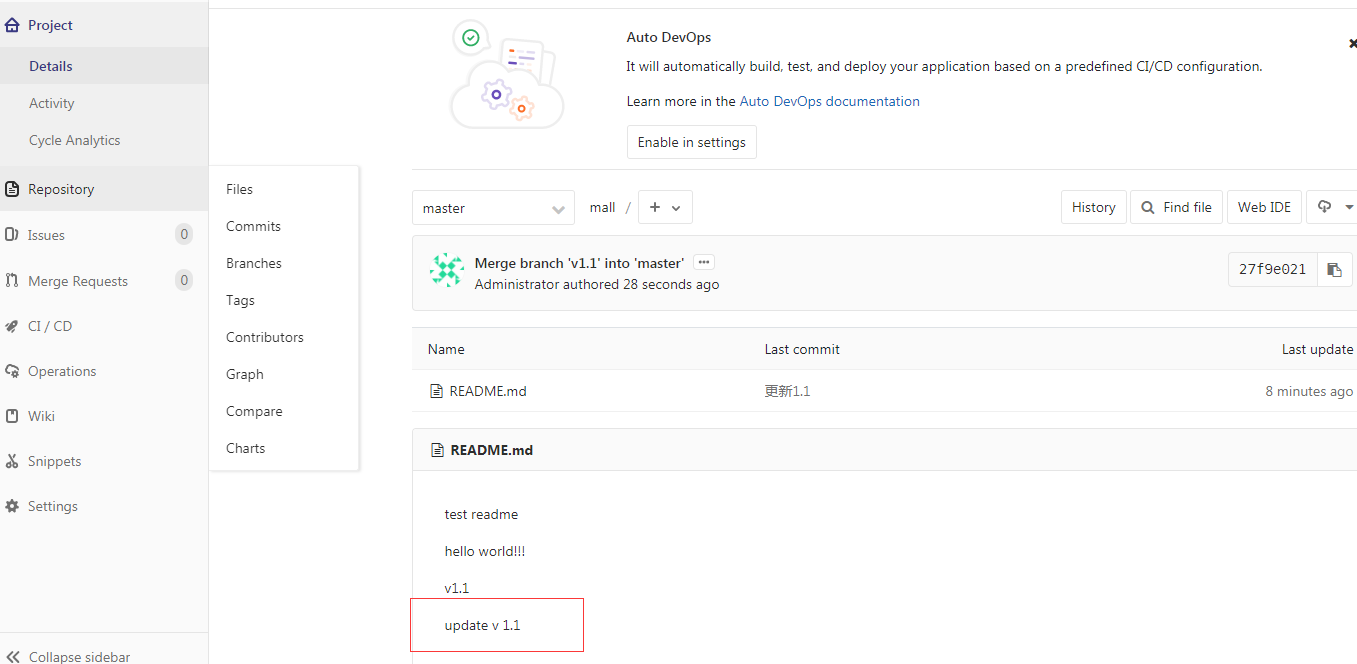


### 2.3.7 发起合并请求

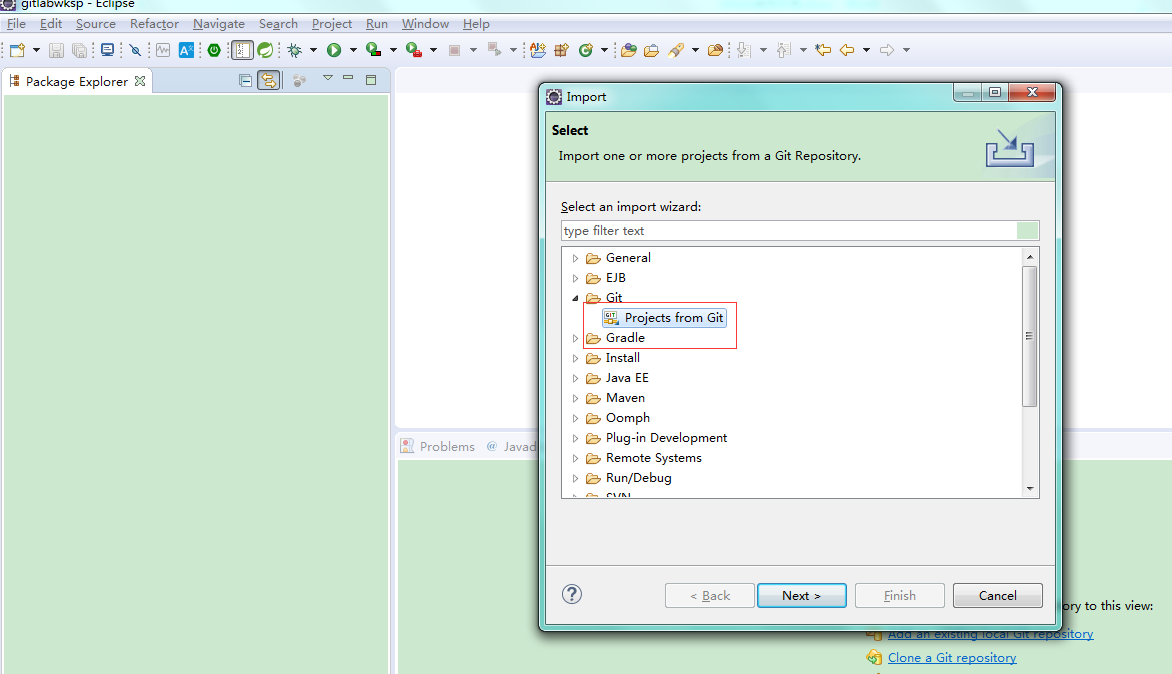


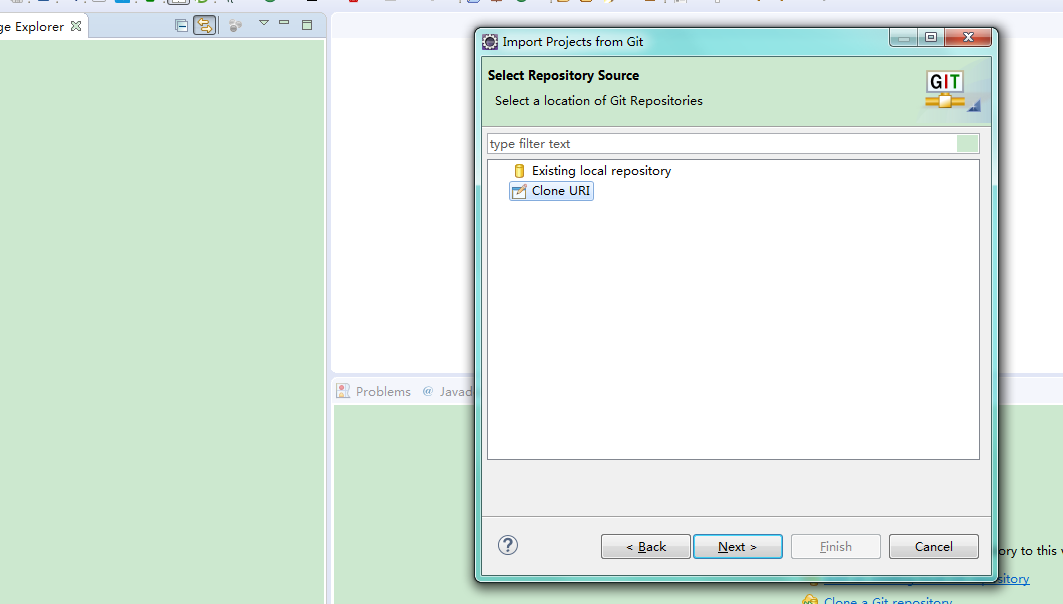
合并操作跟上一章节类似，不在累述。

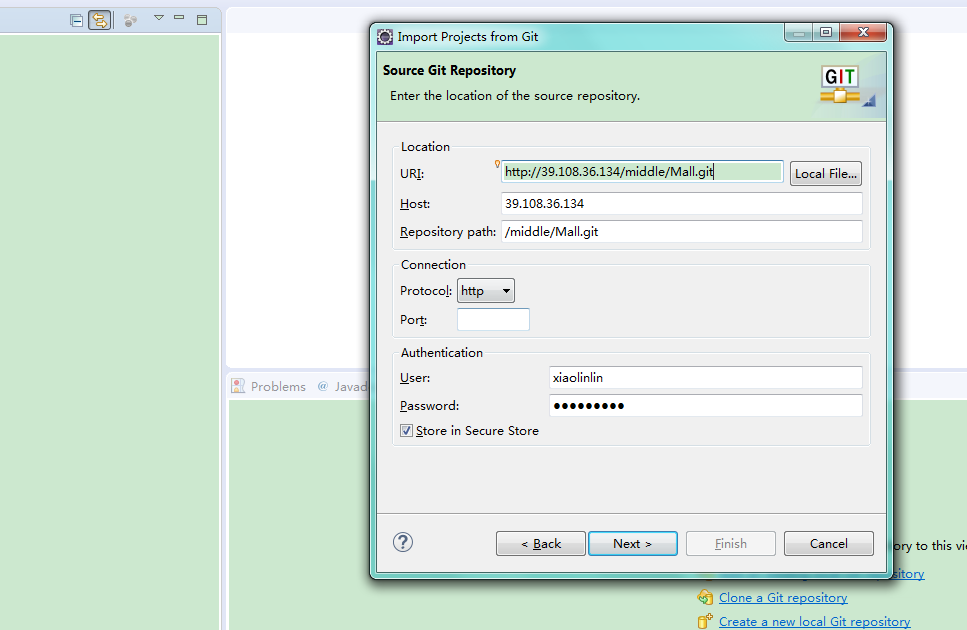
最终合并到主干如下所示：



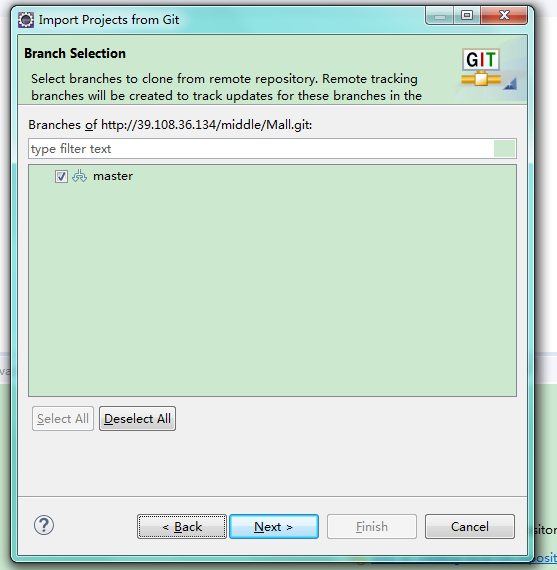
### 2.3.8 gitlab检出maven项目：

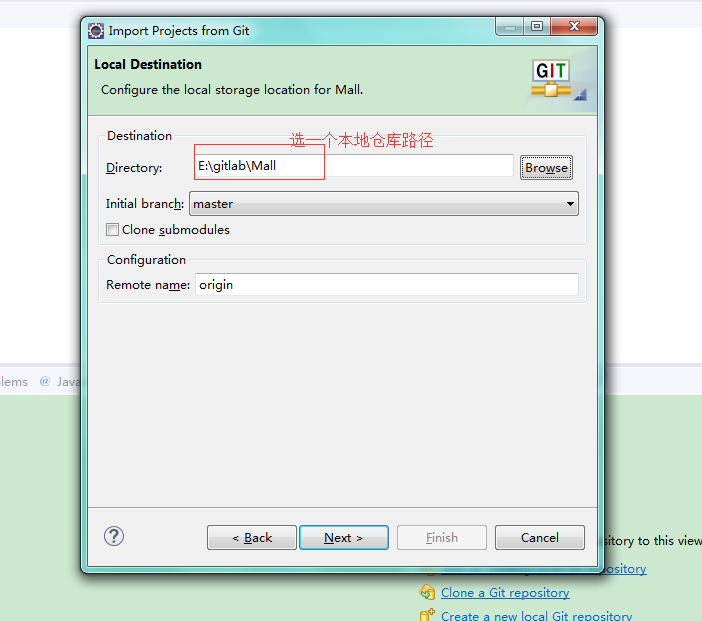


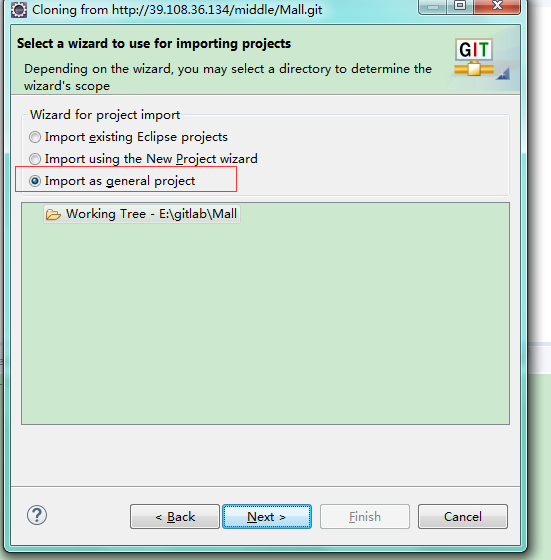


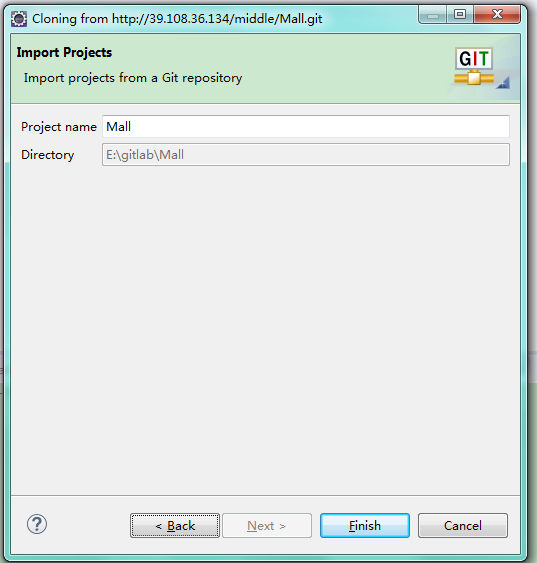


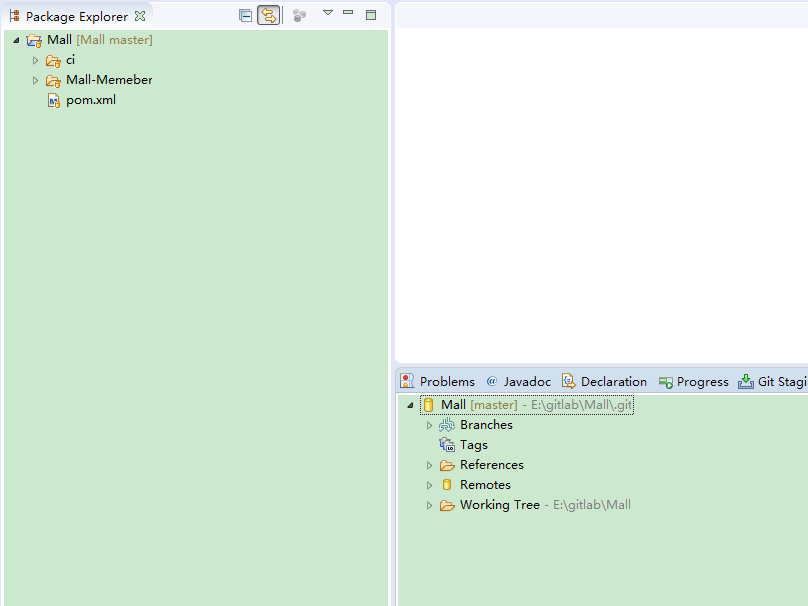
选分支



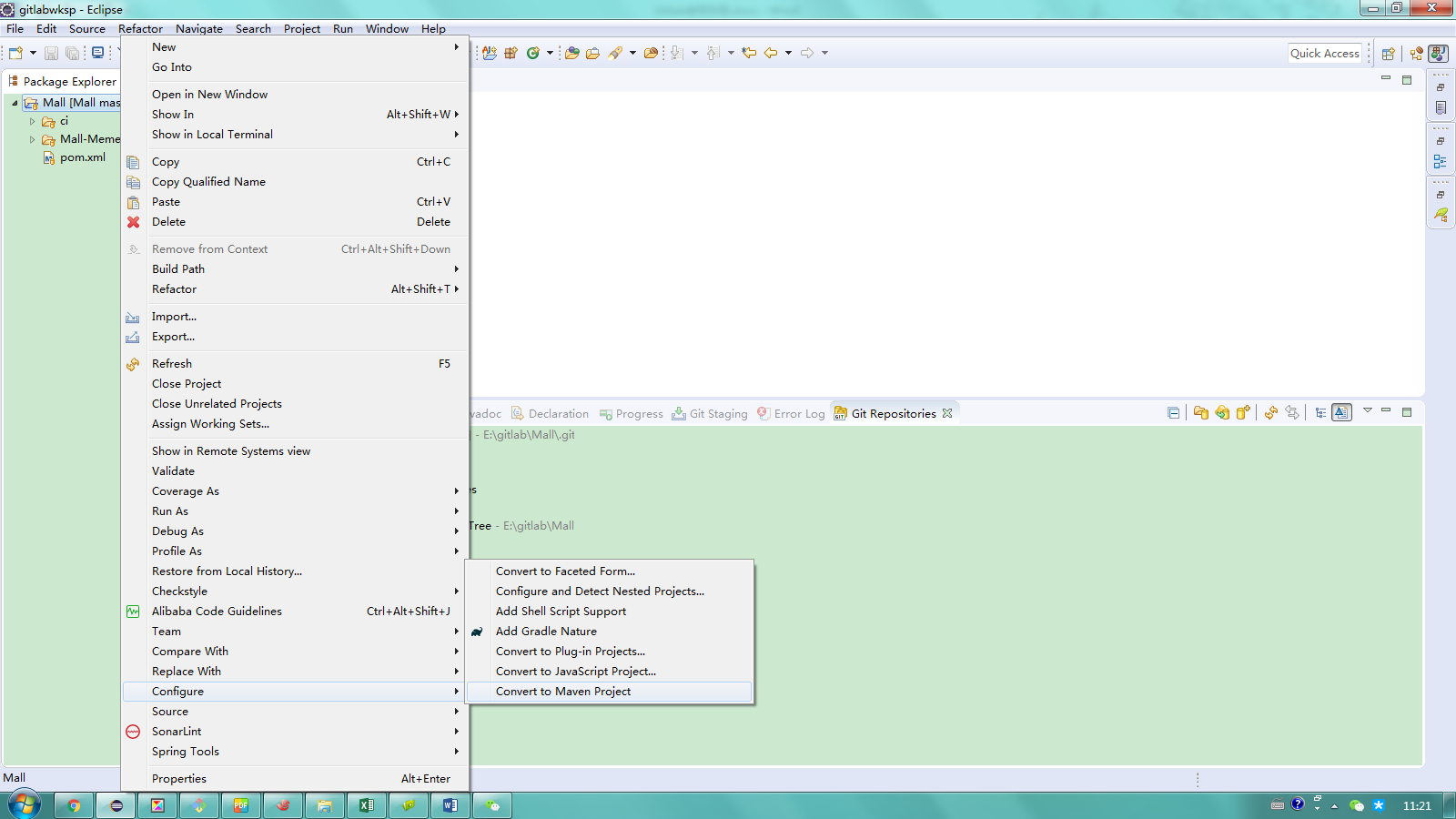






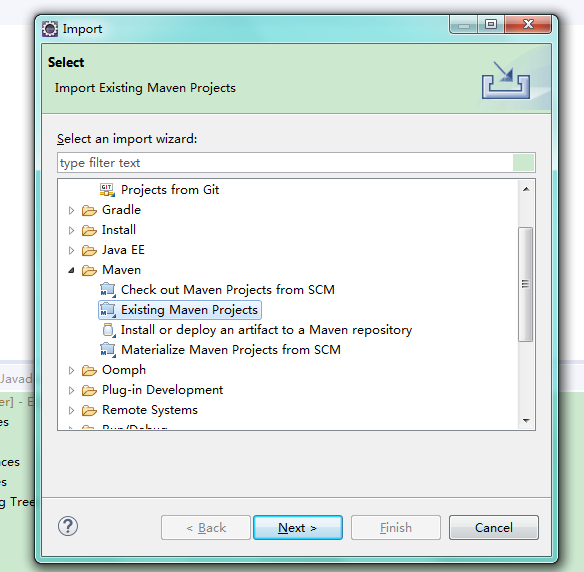


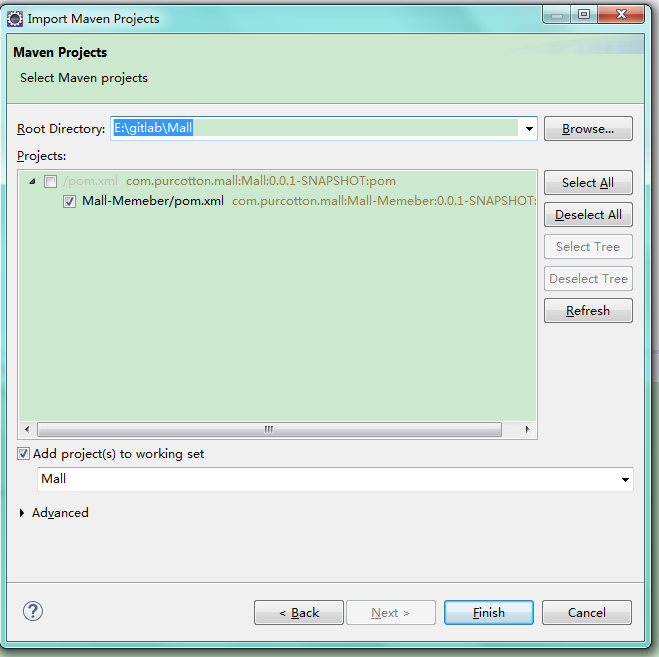
转换maven项目

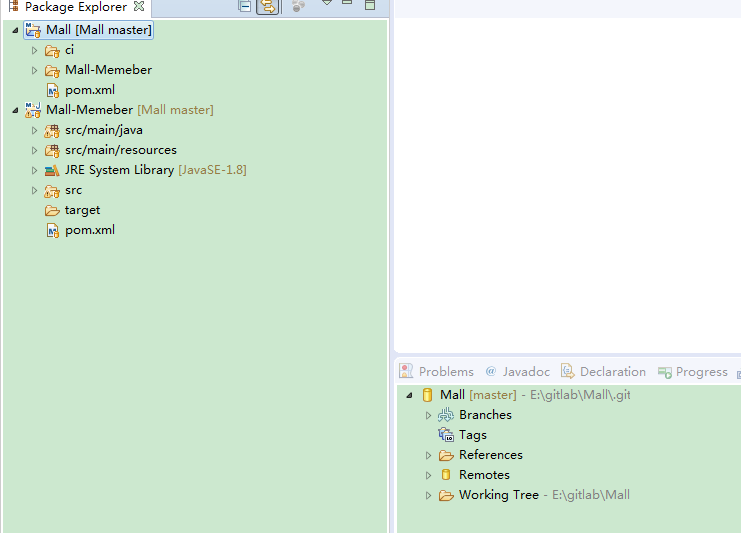


Import 子模块









更多eclipse操作参考：

<https://blog.csdn.net/Adelly/article/details/79099772>

# 三、idea+git客户端+gitlab使用

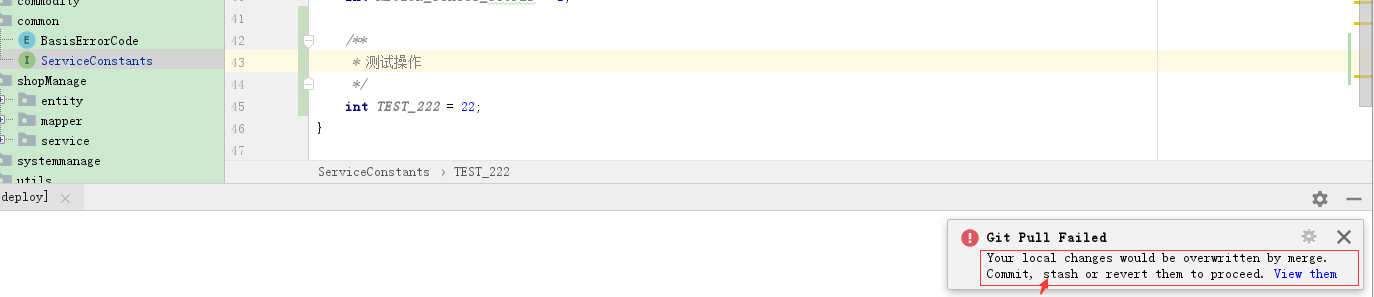
3.1

3.2

3.3

## 3.4 pull冲突之解决方法

产生原因：同一个类，本地有做修改，远程仓库有修改，进行pull的时候

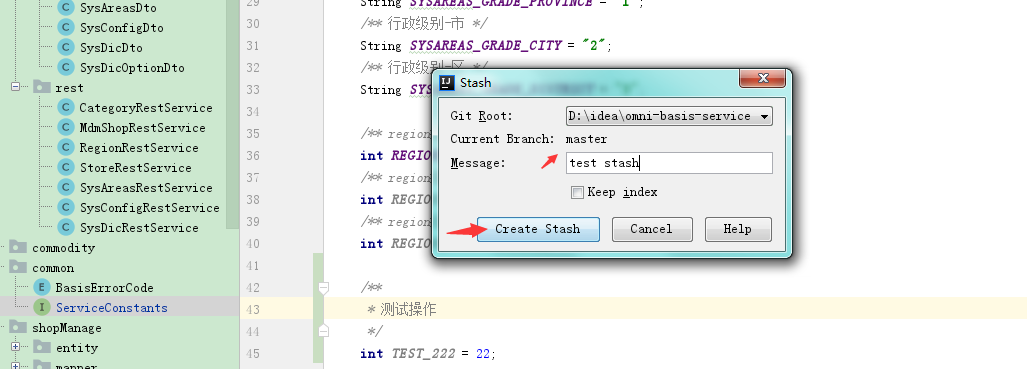


注意看提示信息！

解决方法：使用statsh解决

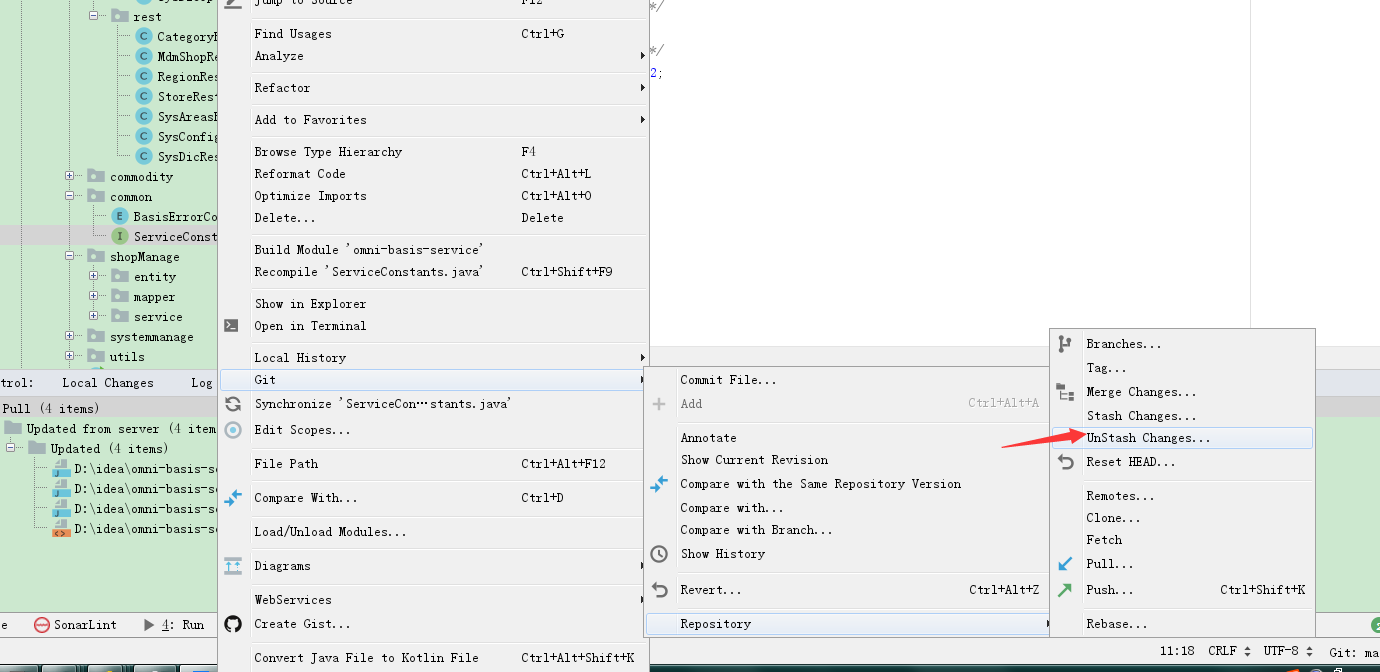
1.创建一个statsh(保存本地更改)





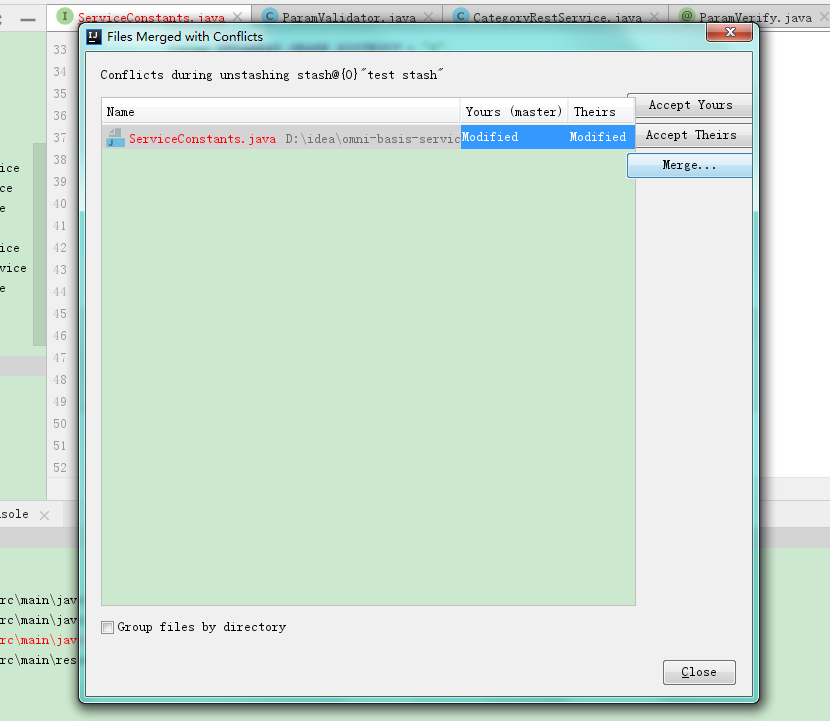
2.进行pull，更新库上的代码

3.恢复本地保存的代码

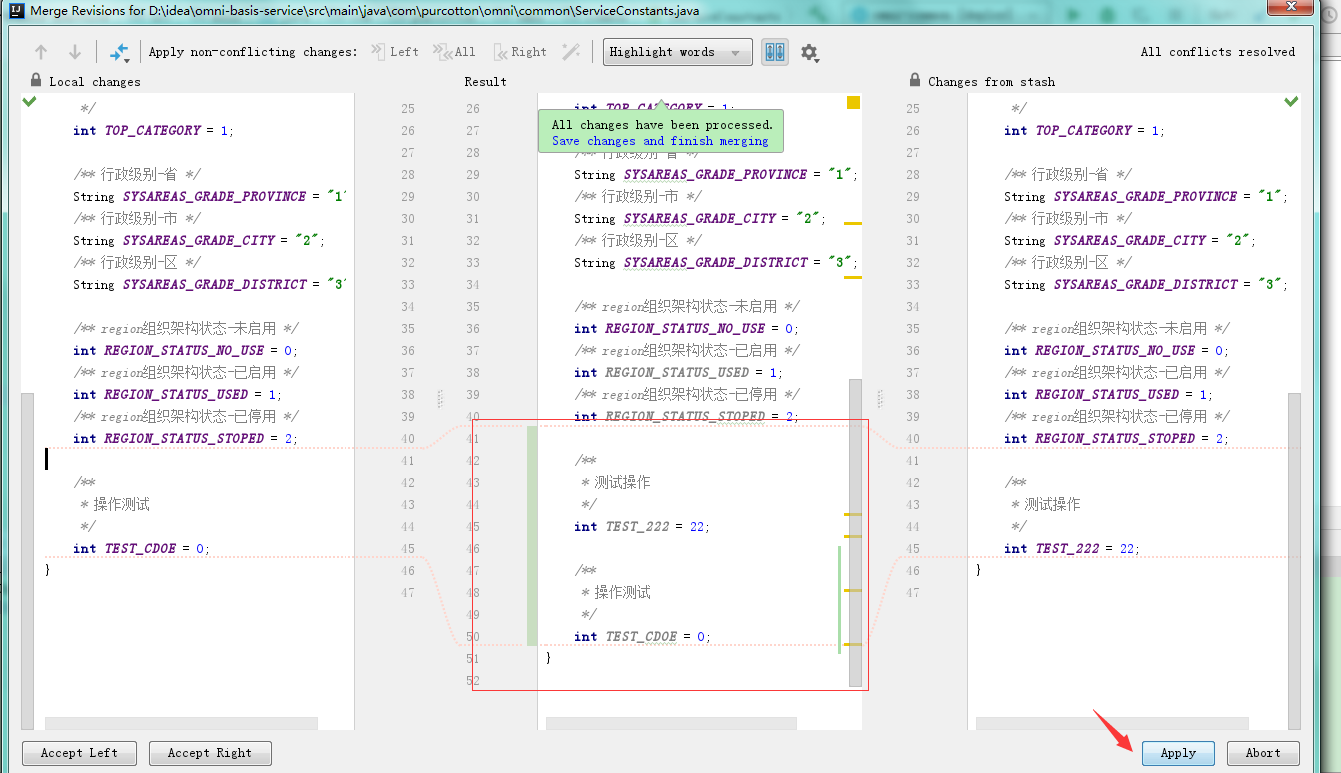




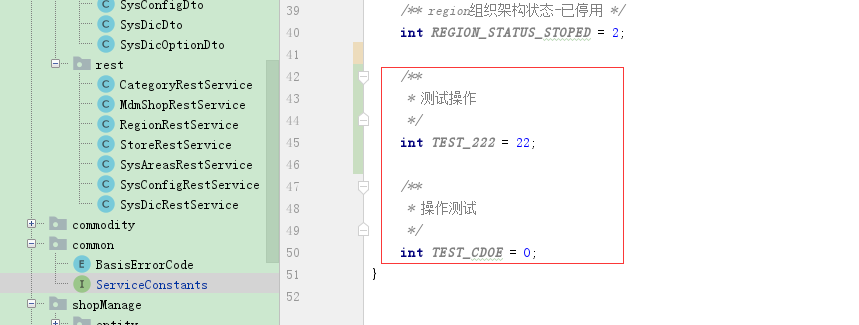
4.进行merge操作







合并后的代码



# 四、Git常用命令操作

## 4.1 配置操作

1).全局用户名、邮箱和默认commit提交配置，默认覆盖原用户名和邮箱，

取消操作 --unset

添加操作 --add

$ git config --global --add user.name "test"

$ git config --global --add user.email "test@example.com"

$ git config --global push.default simple

2).单个项目配置

$ git config user.name "test"

$ git config user.email "test@example.com"

3).文本编辑器

$ git config --global core.editor emacs

4).差异分析工具

$ git config --global merge.tool vimdiff

5)查看配置信息

$ git config --list

6)查看帮助

$ git config --help

7)查看自带的版本

$ git –version

## 4.2 基本操作

1)克隆仓库

$ git clone url

2)将文件添加到缓存,git add是添加所以文件

$ git add [filename]

3)查看项目的当前状态

$ git status -s

4)显示已写入缓存与已修改但尚未写入缓存的改动的区别

$ git diff [--cached/HEAD/--stat]

5)将快照的内容写入缓存区,a直接提交，m是添加注释

$ git commit [-a/m]

6)撤销已缓存的内容

$ git reset HEAD

7)将条目从缓存区中移除,默认是将文件从缓存区和你的硬盘中删除

$ git rm [--cached]

## 4.3 分之管理

1)创建分支命令

$ git branch (branchname)

2)切换分支命令

$ git checkout (branchname)

3)合并分支命令

$ git merge

4)列出分支基本命令

$ git branch

5)删除分支命令

$ git branch -d (branchname)

## 4.4 stash命令

$ git stash save "feature" 　　 储藏修改内容

$ git stash apply 　　　　　　 应用第一个储藏

$ git stash list [--stat] 　　　　 查看保存的储藏

$ git stash show stashname 　　 显示一个stash内容

$ git stash show --patch 　　　　 显示文件差异

$ git stash drop 　　　　　　　　 删除最近的一个

$ git stash clear　　　　　　　　 删除所有的