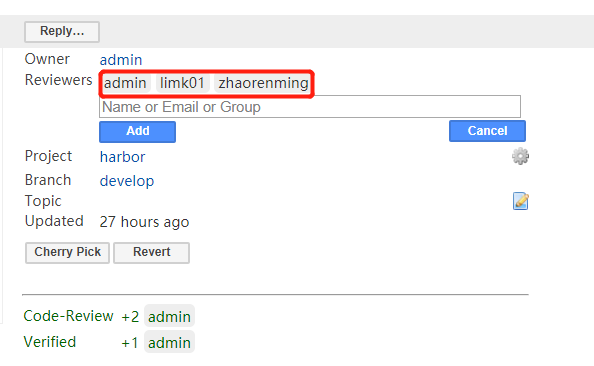
## Gerrit\_review使用手册

## Gerrit简介

Gerrit一种免费、开放源代码的代码审查软件，使用网页界面。利用网页浏览器，同一个团队的软件程序员，可以相互审阅彼此修改后的程序代码，决定是否能够提交，退回或者继续修改。

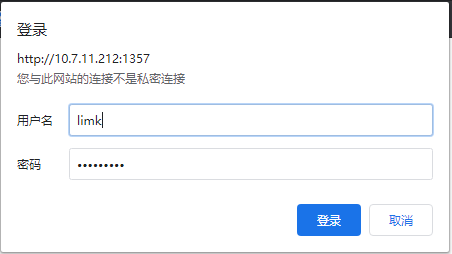
相对于gitlab添加了**多人review的机制**；可以通过质量管理部提供的评审记录导出插件，**导出评审记录**。另外可以与CICD集成（调研中）。



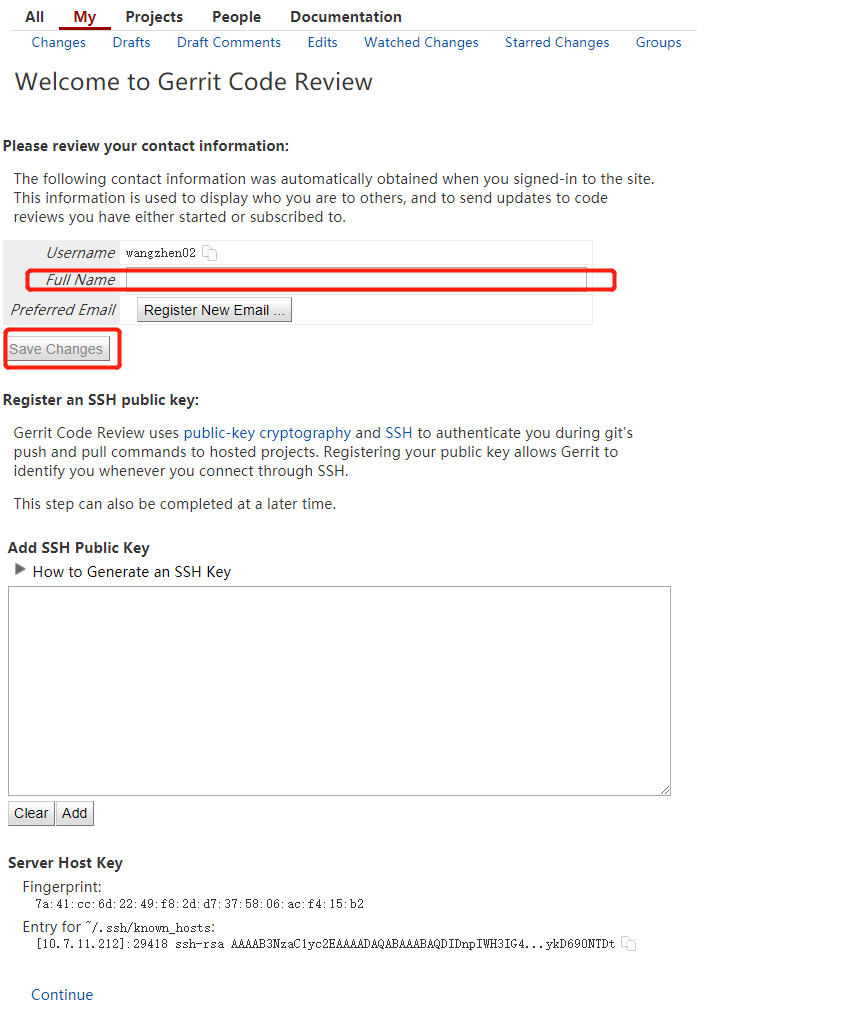


## 普通用户使用方法

登录gerrit：<http://10.7.11.212:1357/>

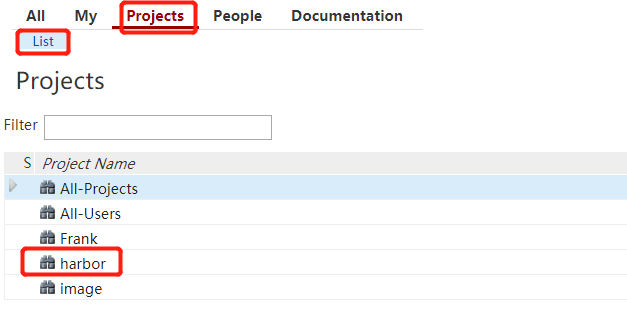


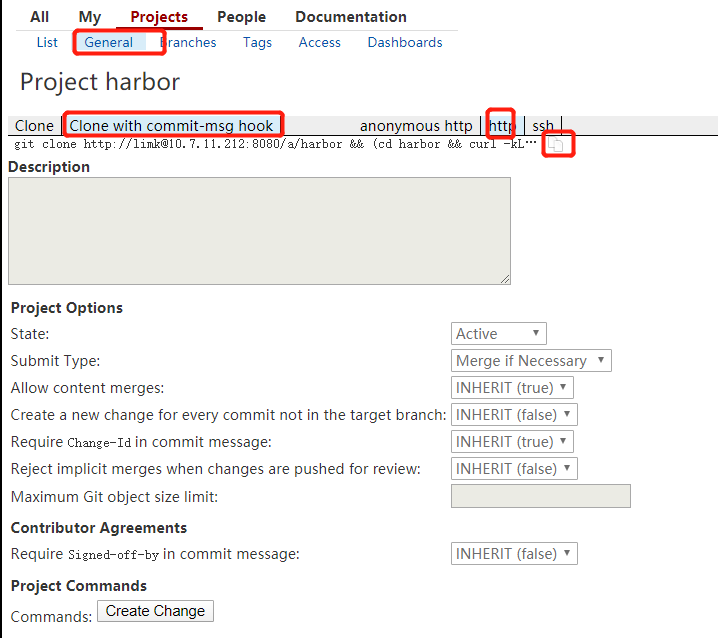
如果是初次登陆，会出现设置界面。



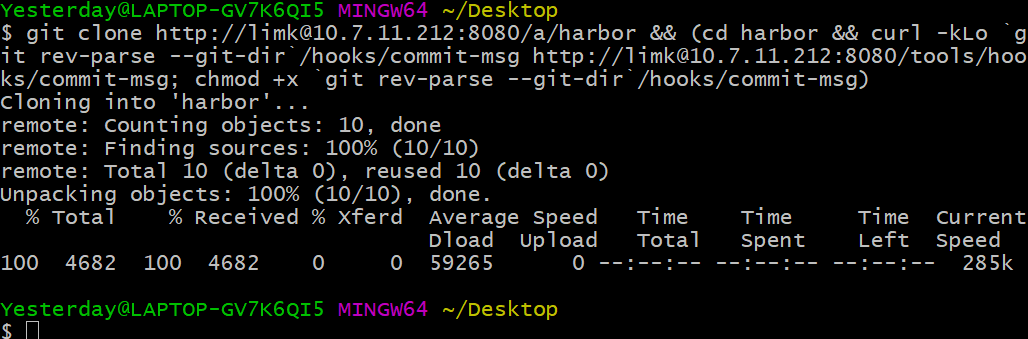
Fullname和username设置为一样即可。**下方的邮箱不需要注册**。点击保存。

找到harbor项目，点击进入





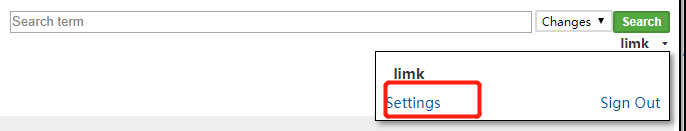
点击clone with commit-msg hook 和 http，点击复制，将复制好的粘贴到命令行中。



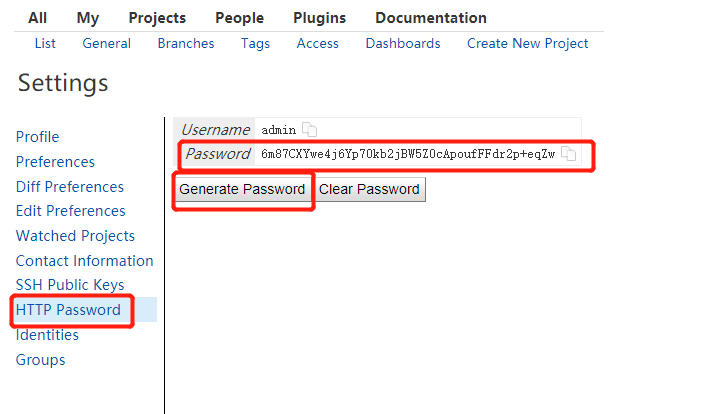
**注意：下载不同的分支 需要在git 命令加入-b 分支名。如下所示。**

git clone http://admin@10.7.11.212:8080/a/harbor **-b develop** && (cd harbor && curl -kLo `git rev-parse --git-dir`/hooks/commit-msg http://admin@10.7.11.212:8080/tools/hooks/commit-msg; chmod +x `git rev-parse --git-dir`/hooks/commit-msg)

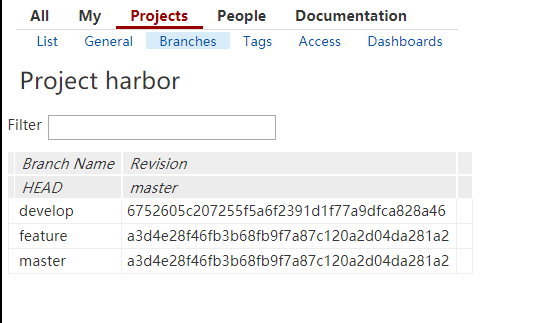
如果出现提示需要输入密码,界面右上角找到自己的用户名，点击箭头，找到settings，点击进入



找到http password，点击粘贴按钮。如果password中不显示密码，点击下方的Generate password。复制密码，就可以下载了



自由发挥，在项目中创建一次commit。然后提交。可提交的分支如下（自行查看），可自行选择分支，目前分支没有做限制。



但是提交的方式有所改变,将原来的git push改为如下：

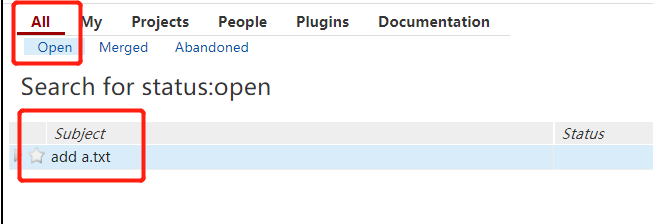
**git push -u origin HEAD:refs/for/develop**

**其中develop可以修改为master feature（根据实际情况自行填写）**



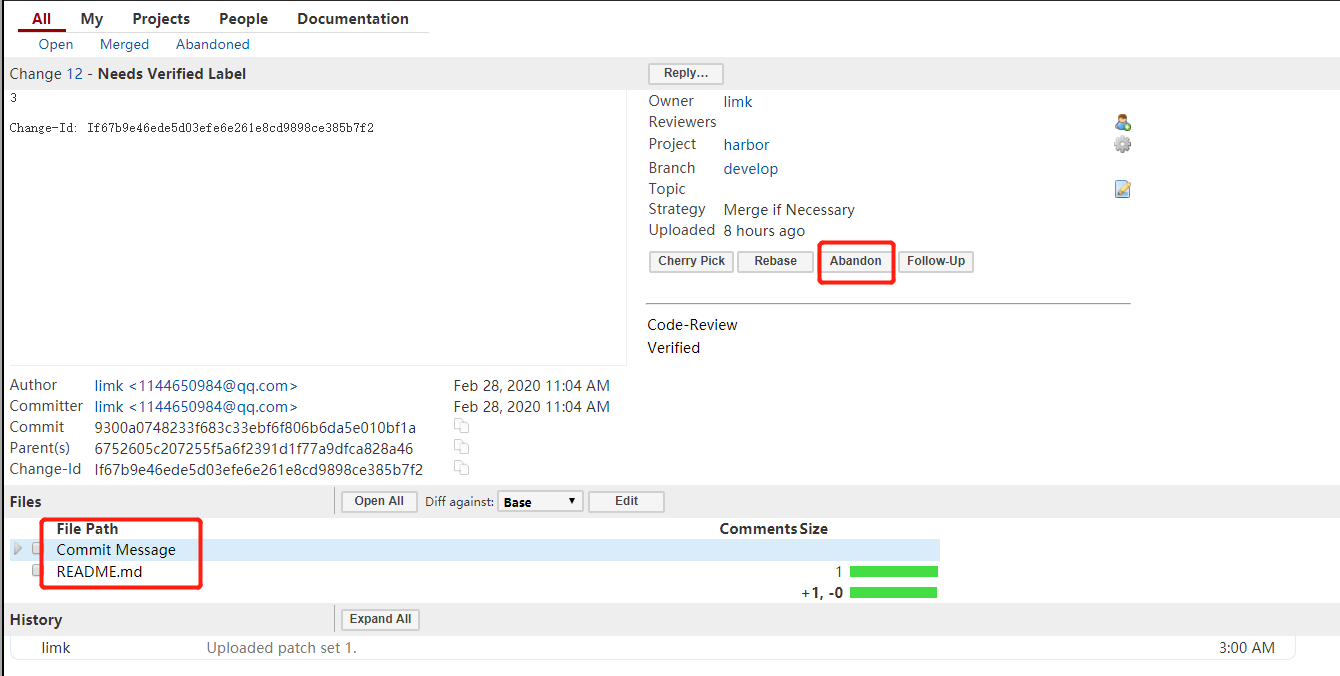
可能还需要输入密码，密码获取参考之前的方法。

查看gerrit界面。



点击进入。

普通用户的操作如下。

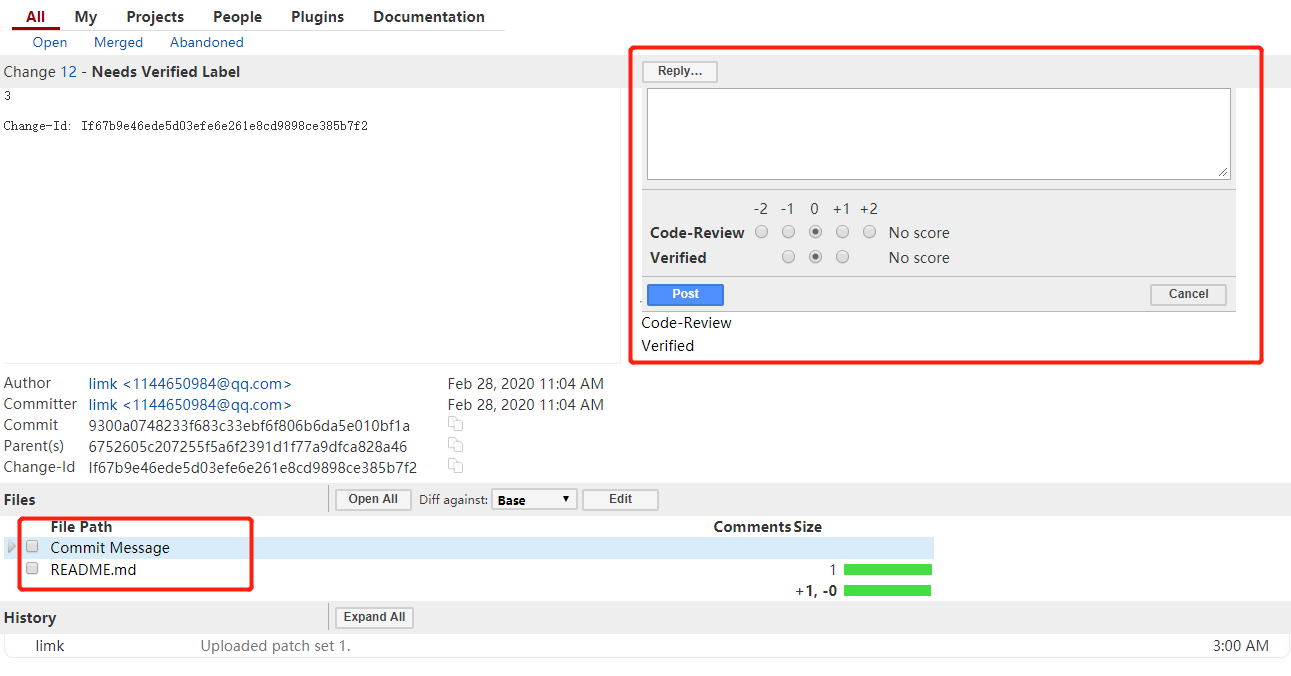


左下可以查看本次的修改是否正确。Abandon可以放弃本次提交。后续等待reviewer审批。

审批通过可以在ALL->Merged看到本次记录已经合并了。

## Reviewer审批

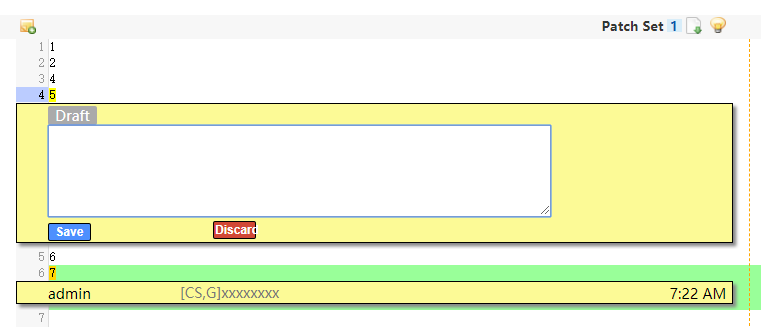
Reviewer在ALL->opened找到提交,点击进入。



同样的左下查看改动。

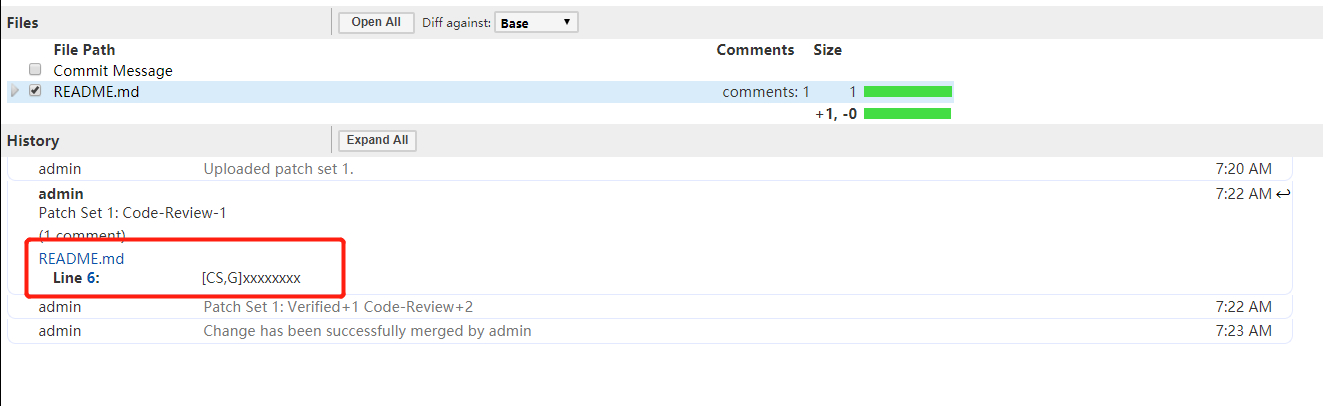
右上角的reply点击出现打分，如果决定可以合并code-review点击+2，如果需要其他的reviewer审阅就点击+1，负数的表示不通过。Verified用于自动化验证，目前手动+1。打分之后点击post。

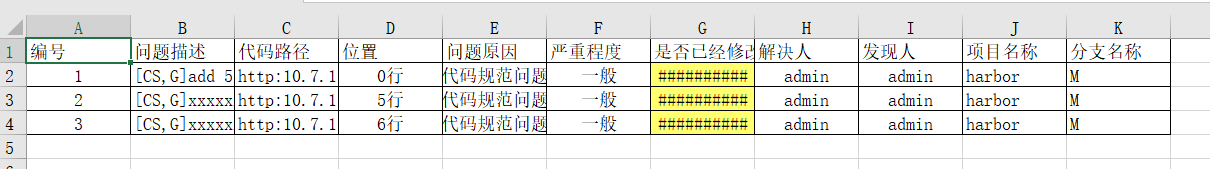
对于不能够合并的代码可提出修改意见。具体的使用方法为：点击相应的文件，双击所在行，会出现评论框。填写内容，点击保存。



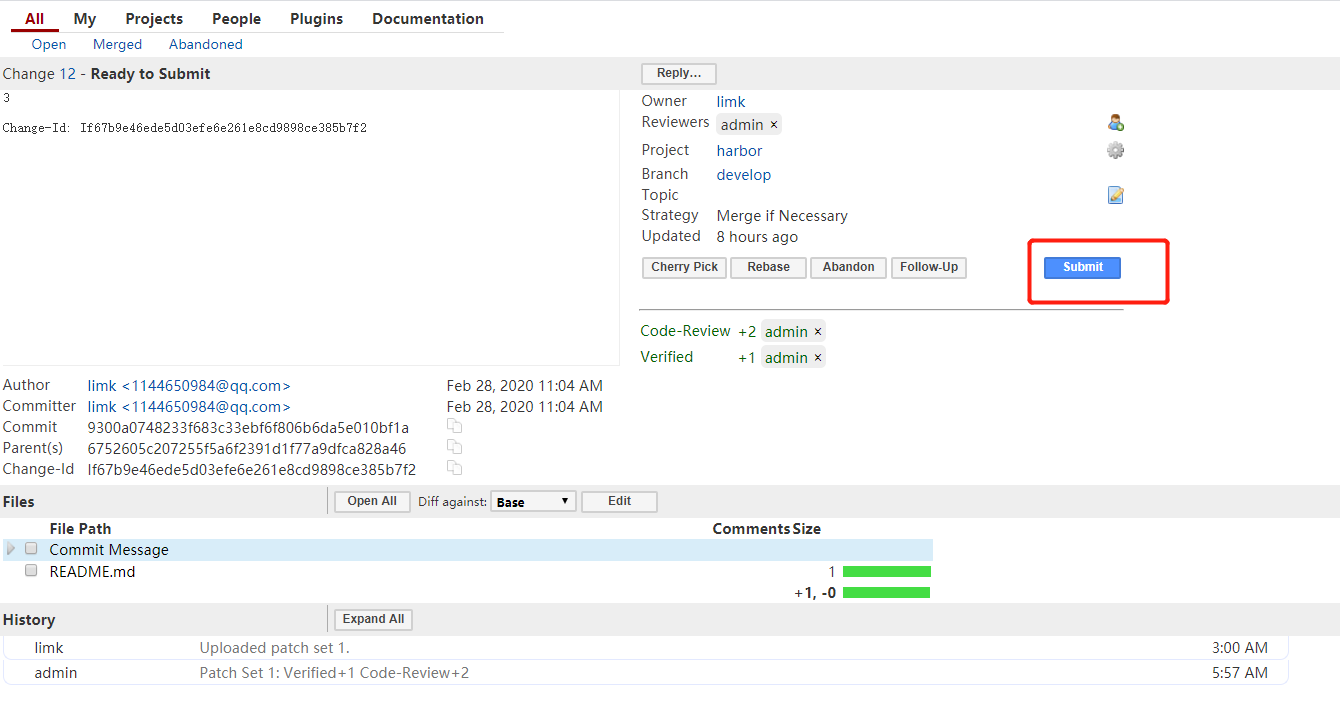
书写的格式参考质量管理部的使用文档。

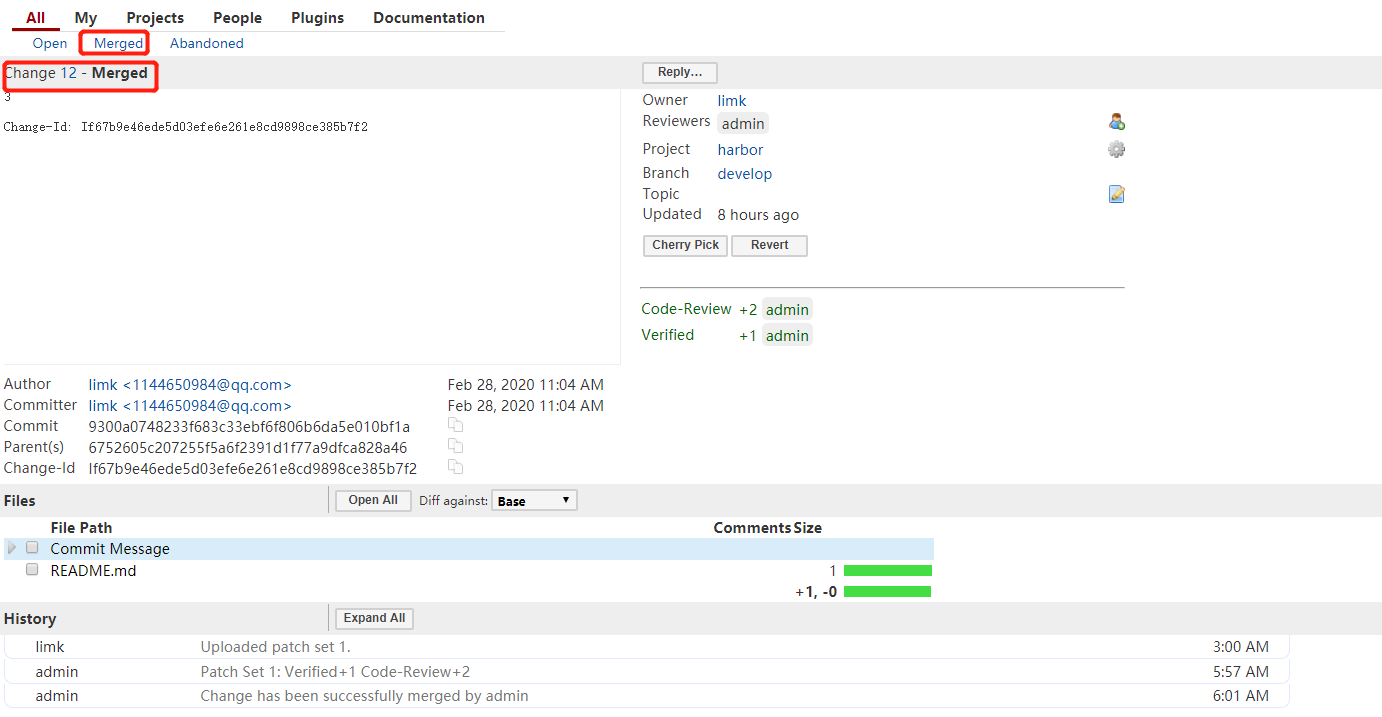
相应的评论会在history显示。后续可以使用相关的软件导出这些评论。





点击submit。这次提交将合并到库中。





## 代码review记录导出

使用质量管理部的exe包。质量管理部来配置生成exe包。

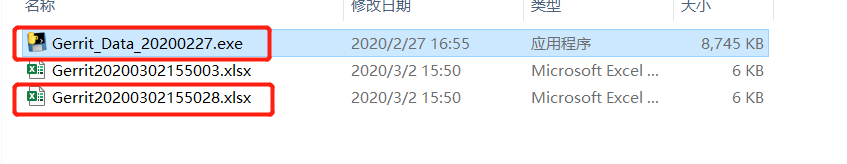
需要：项目名称、Gerrit的IP和能访问Gerrit数据库的账号密码；

数据库采用mysql。

主要是评审时需要按照《评审规范》要求填写评审意见，请参考：<http://10.151.12.58:8090/pages/viewpage.action?pageId=1245406>

<http://10.151.12.58:8090/pages/viewpage.action?pageId=1245242>

实际运行下图exe，根据提示输入时间，即可生成相应的excel表格。



表格内容如下。



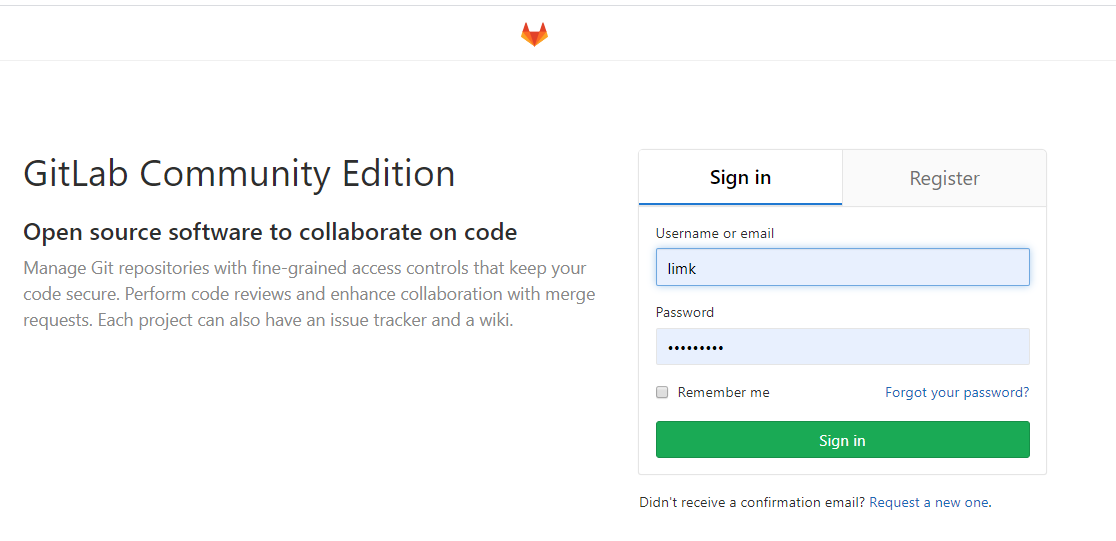
## Gitlab查看

提交后的代码可以在gitlab中查看。

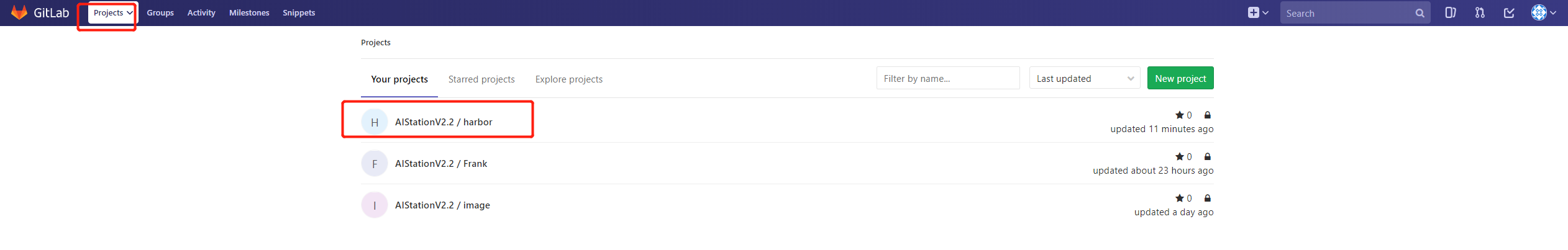
**Gitlab的用户权限会被限制为reporter，不能直接在gitlab上提交代码，所有的代码改动都需要通过gerrit。**

Gitlab地址: <http://10.7.11.211/>

登录。



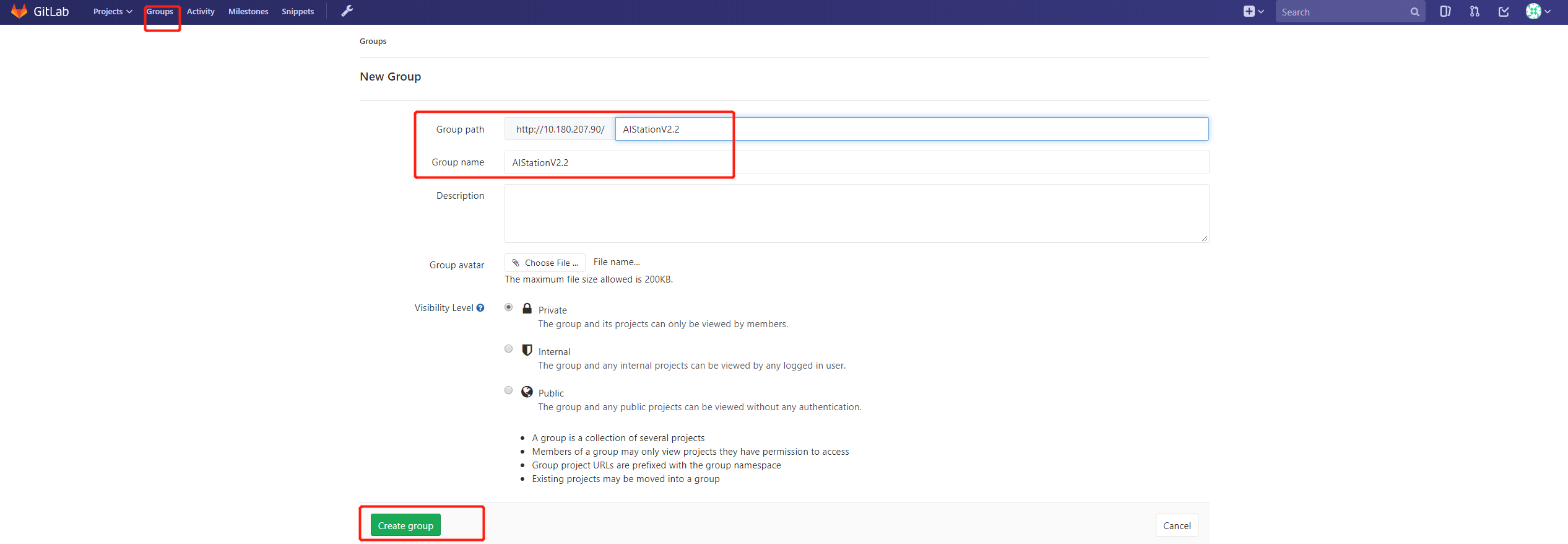
选择项目。



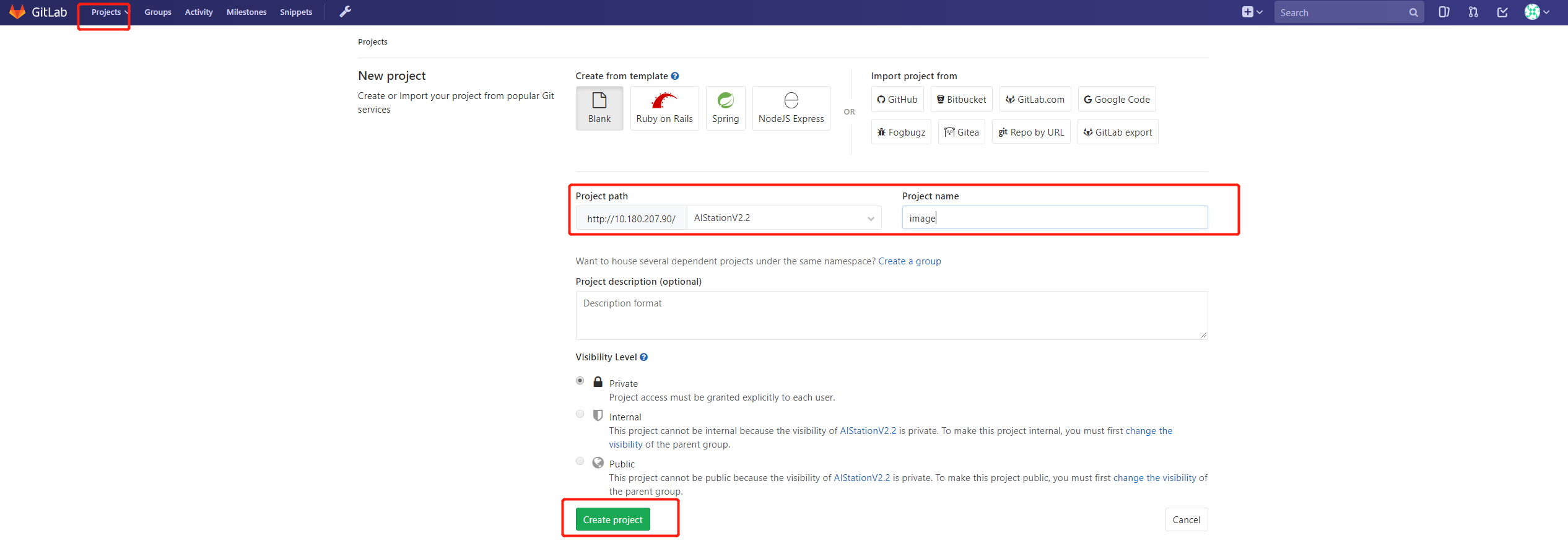
选择分支，查看代码。

## Gitlab创建项目

先创建一个group。



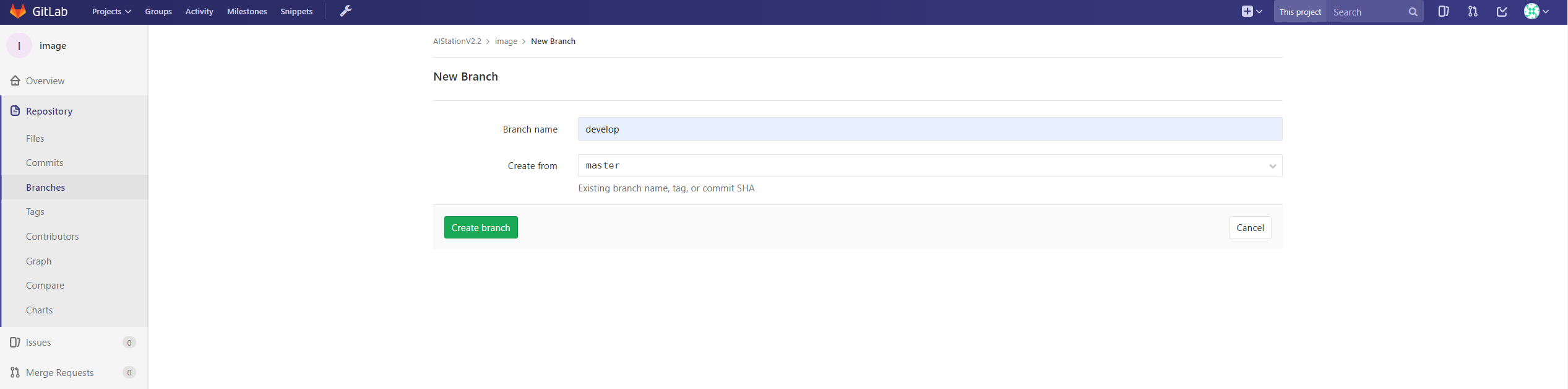
再创建一个项目。



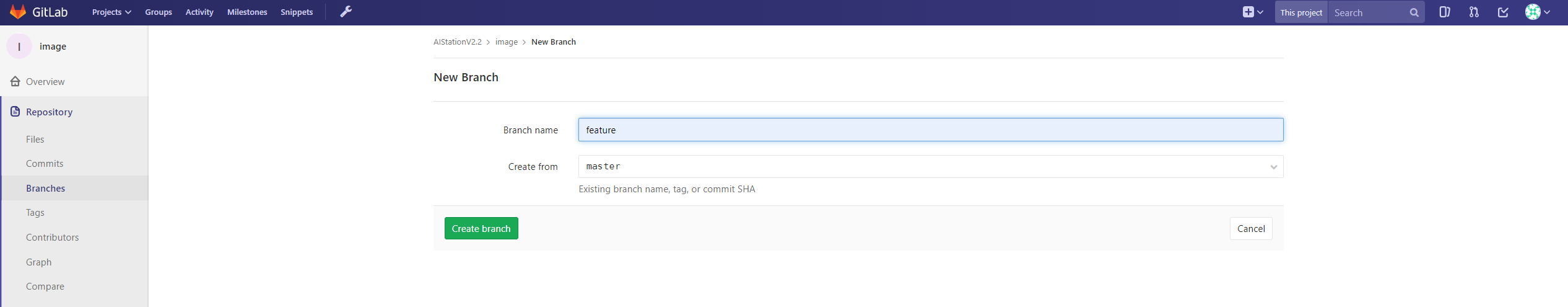
先gitclone仓库做一次初始的提交。

然后创建新的分支：

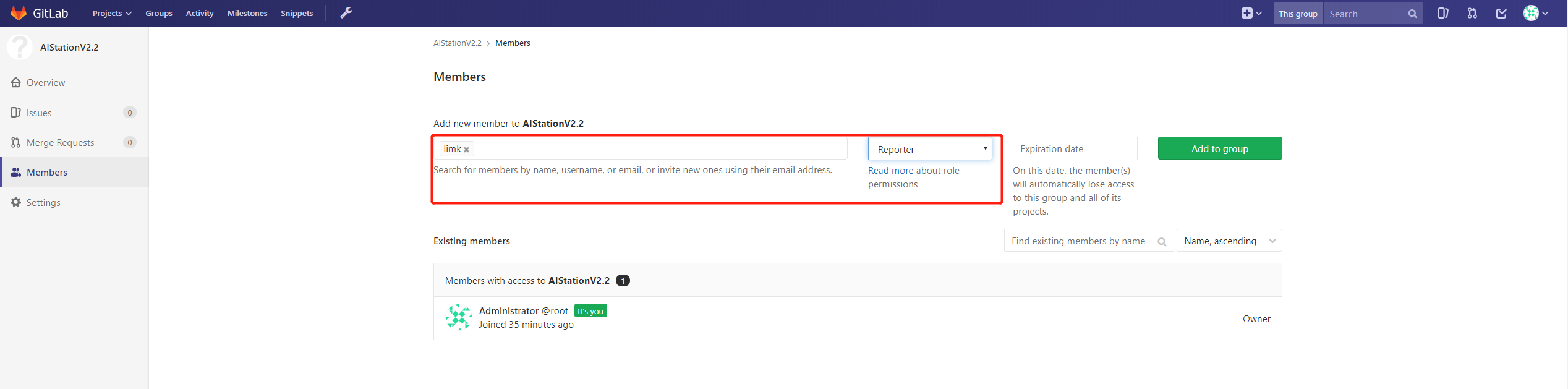
Develop



Feature

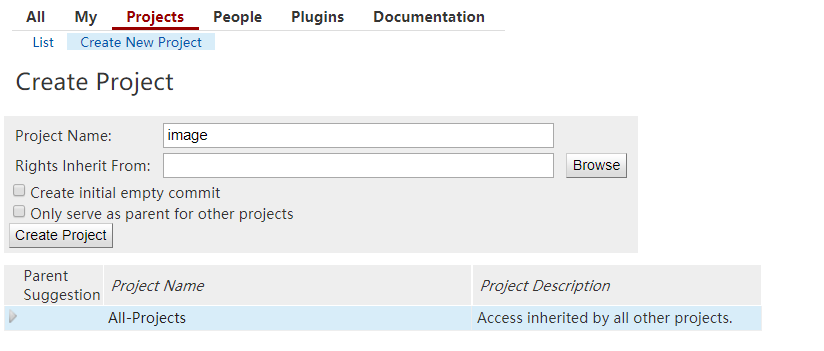


给group分组添加一个用户。权限设置为reporter。



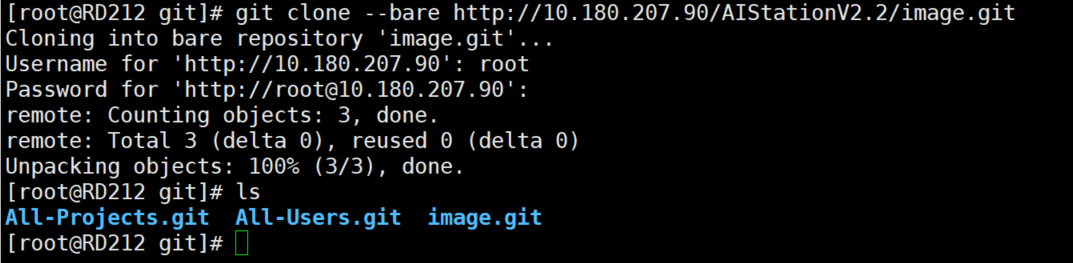
## Gerrit创建项目

在gerrit创建一个同名的项目



在gerrit中/path/to/review\_site/git删除test.git文件夹。

重新从gitlab克隆项目。git clone --bare your\_prject\_http（**此处选择git的url，不要选择http的url，下图为错误示范**）



配置Gerrit和gitlab的代码同步：

`vim /usr/local/gerrit/etc/replication.config`

[remote "10.180.207.90"] //gitlab地址

projects = test //项目名称

url = [git@10.180.207.90:dev-group/test.git //](mailto:git@10.180.207.90:dev-group/test.git%20//)需要配置git路径不要使用http路径。

push = +refs/heads/\*:refs/heads/\*

push = +refs/tags/\*:refs/tags/\*

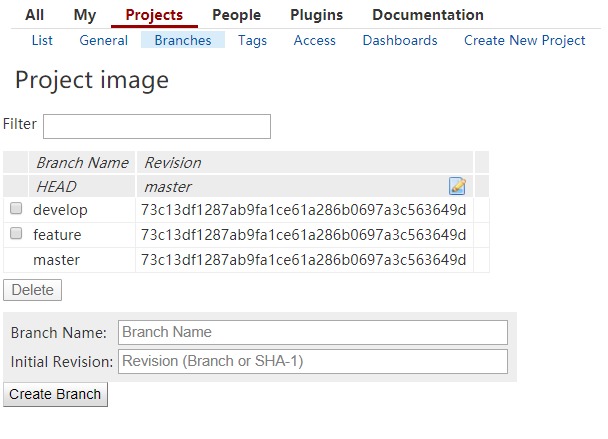
push = +refs/changes/\*:refs/changes/\*

threads = 3

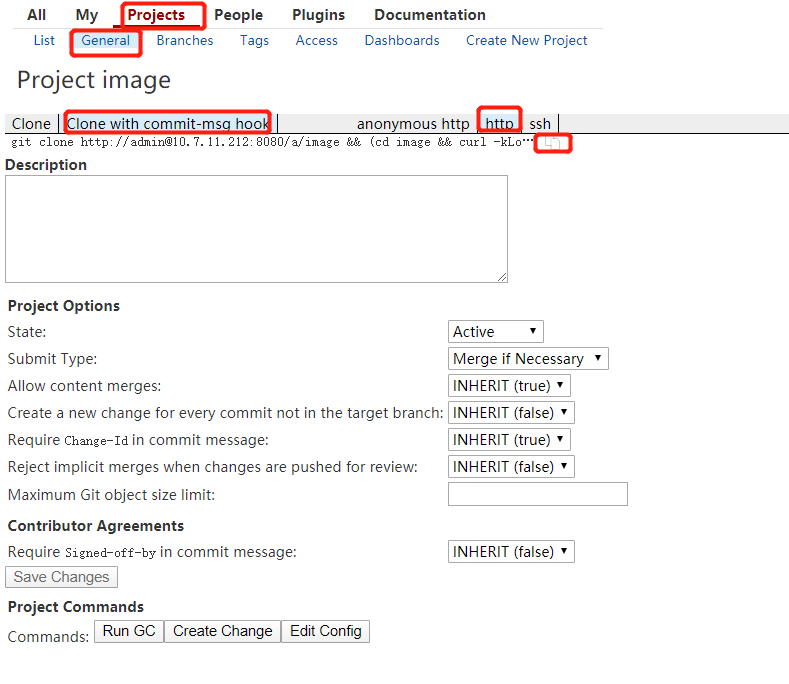
```

如果有新的项目添加，继续在此文件添加相应的项目即可

现在gerrit的项目显示为如图，有了三个分支。

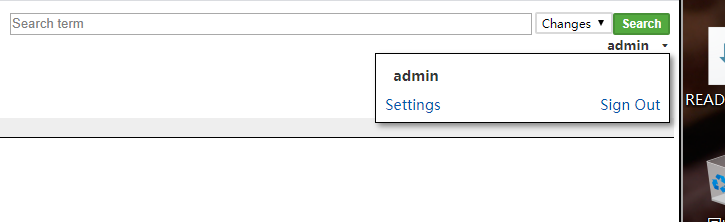


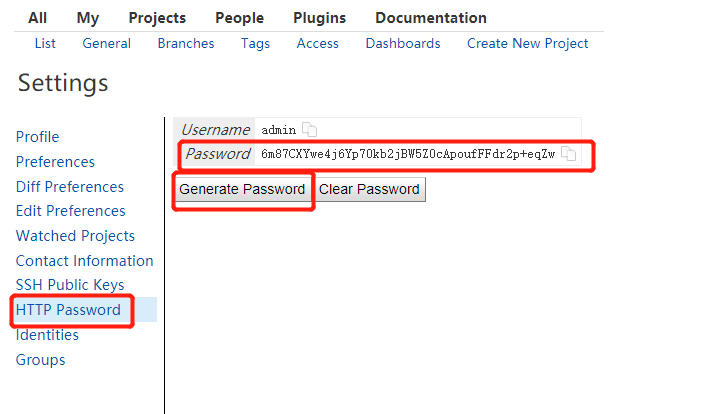
从gerrit下载代码



选择这几个选项，粘贴到项目下就可以了

需要输入密码。密码在账号的setting中。





如果显示没有密码点击Generate password

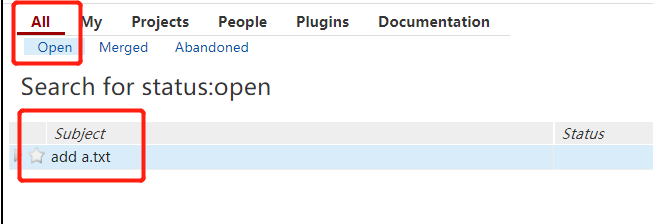
随意创建一次提交。然后push。格式如下。



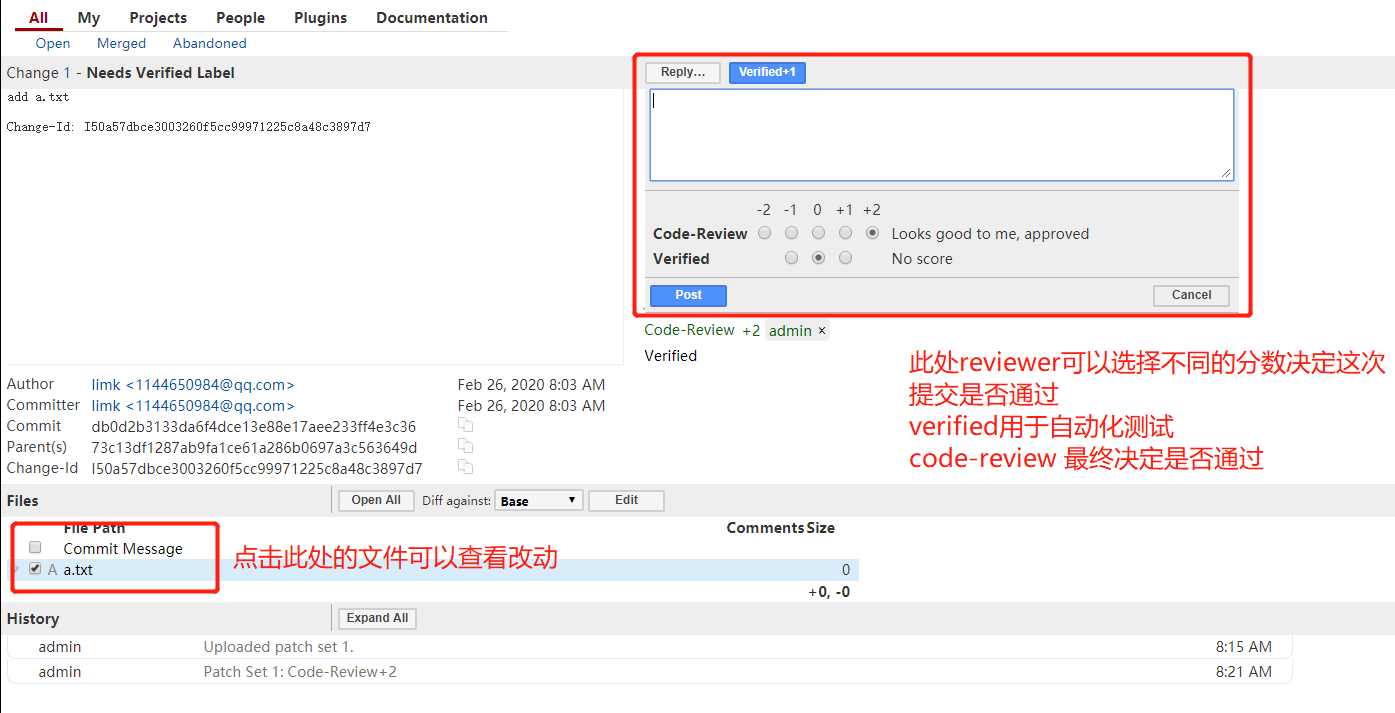
我们把这次commit提交提交到develop分支上。命令修改为如下。

**git push -u origin HEAD:refs/for/develop**

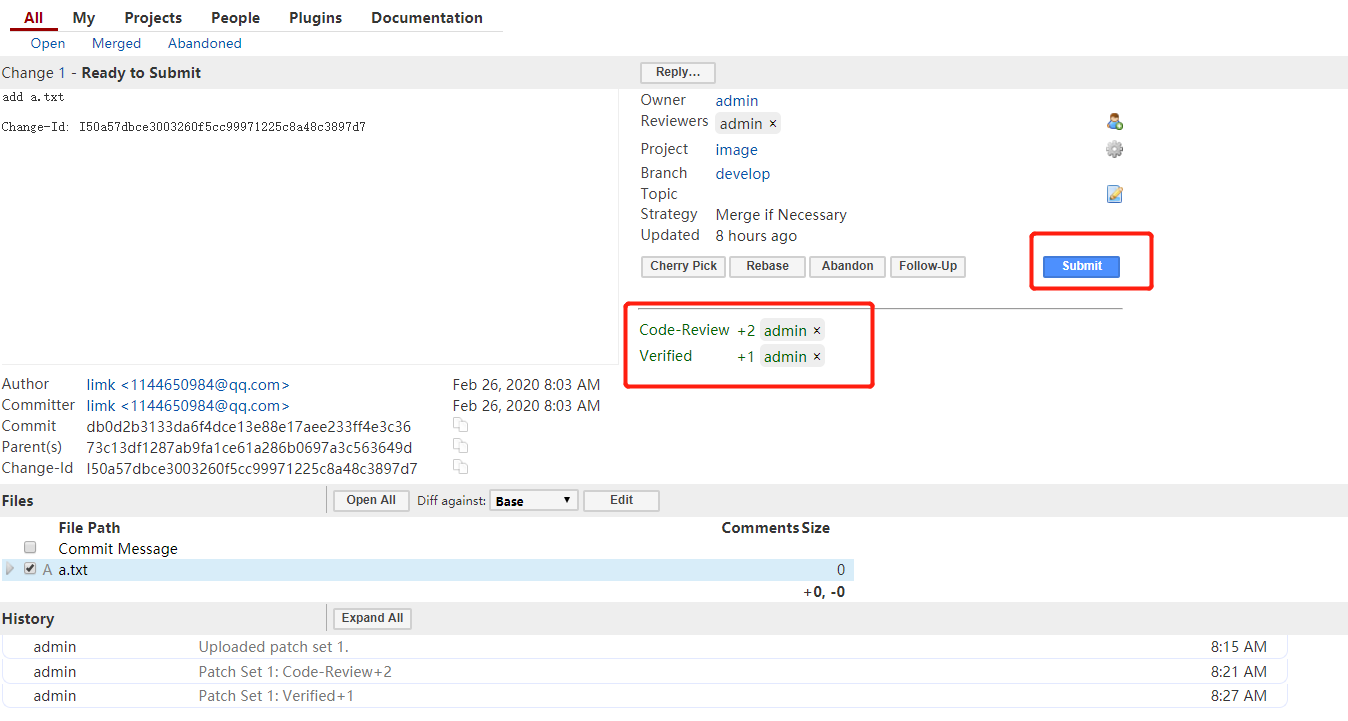
查看gerrit界面。



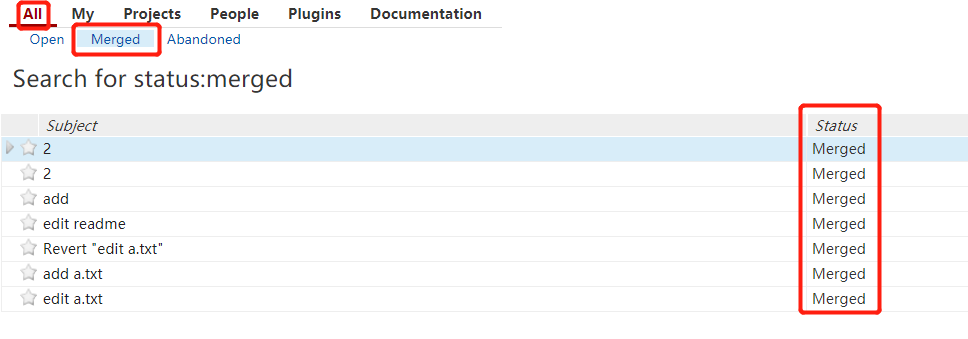
**点击进入**



**给出得分之后，点击submit通过这次提交。**



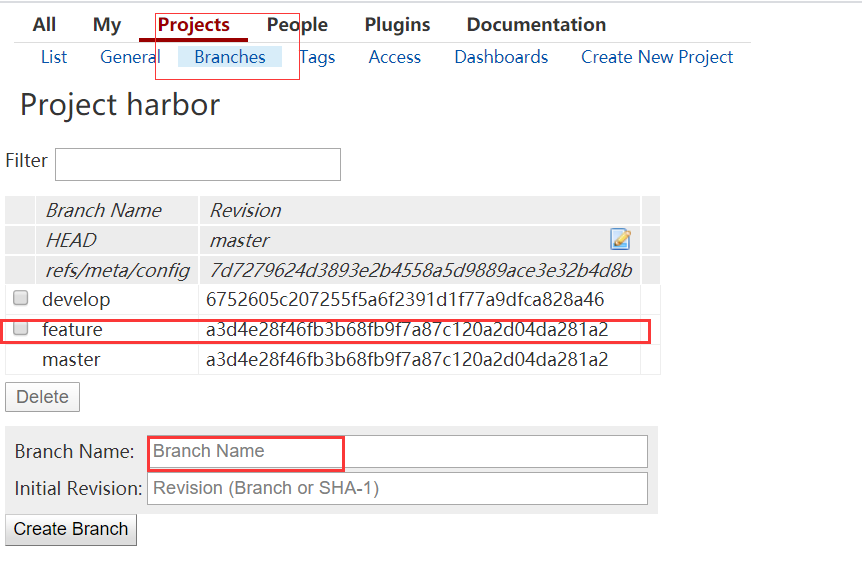
点击界面可以发现这个commit已经合并了。



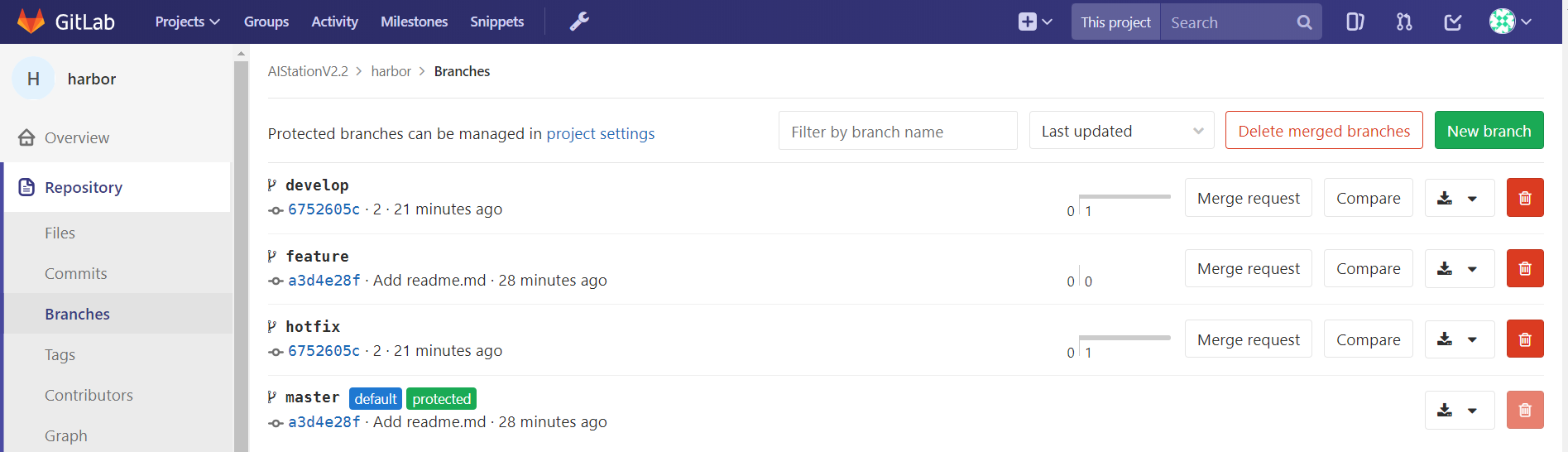
随后可以查看gitlab，gerrit的提交会自动合并到gitlab上。

### 添加分支

点击branch，输入新的分支名，点击创建。Initial Revision可以选择当前创建的分支基于某个分支创建。

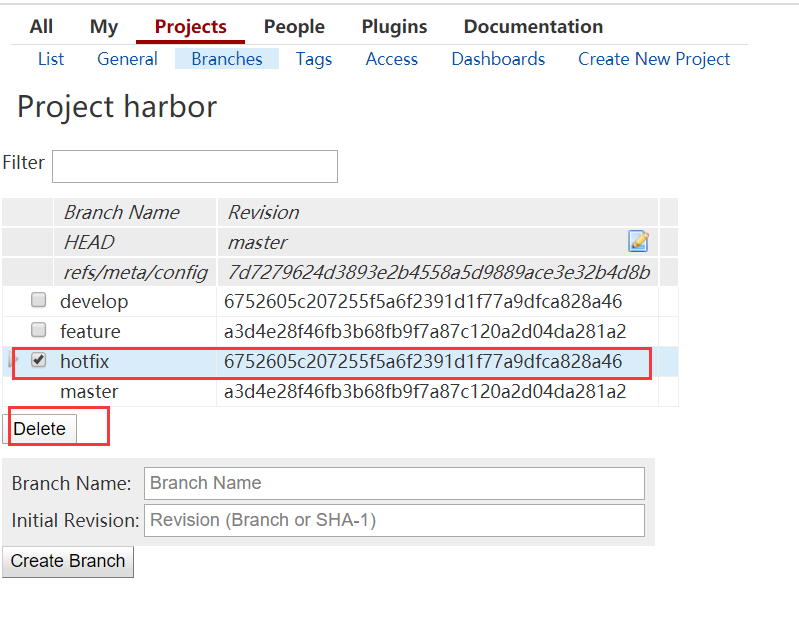


gitlab中会显示新创建的分支。如图的feature、hotfix分支都是这样创建出来的。

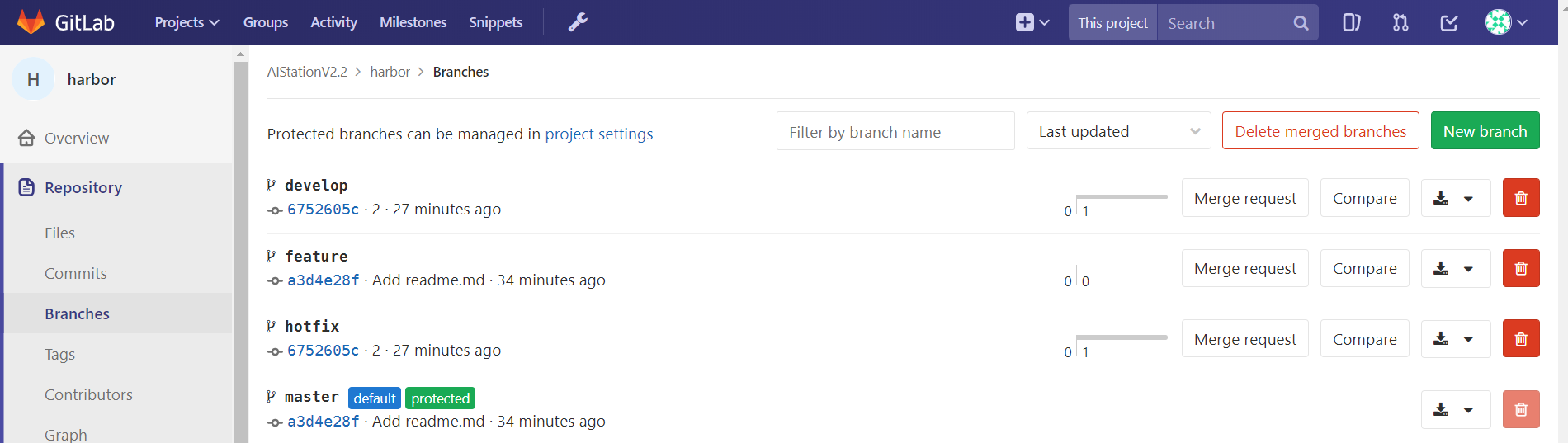


### 删除分支

选中某个分支点击delete。

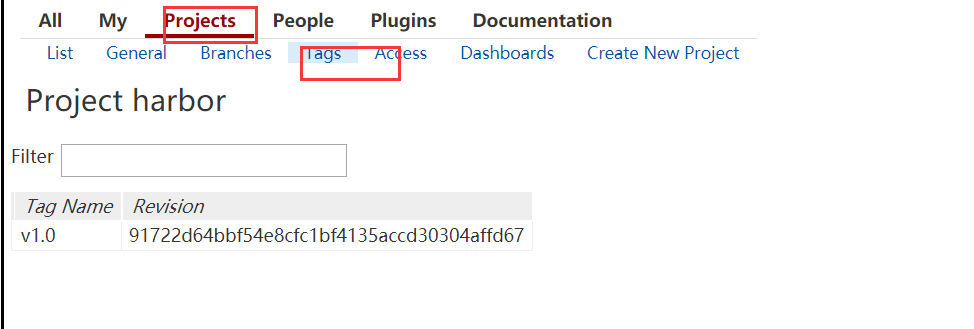


但是git没有删除当前分支（待解决）。



### 添加标签

此处显示的是打好的tag。



Git命令：

git tag -a v1.0 -m "version 1.0"

a参数表示标签，m参数表示相关的注释。

git tag 查看标签

git show v1.0 查看v1.0标签的详细信息

git push origin -tags 提交所有的tag

git push -u origin HEAD:refs/tags/v1.0 提交指定的标签

权限设置：

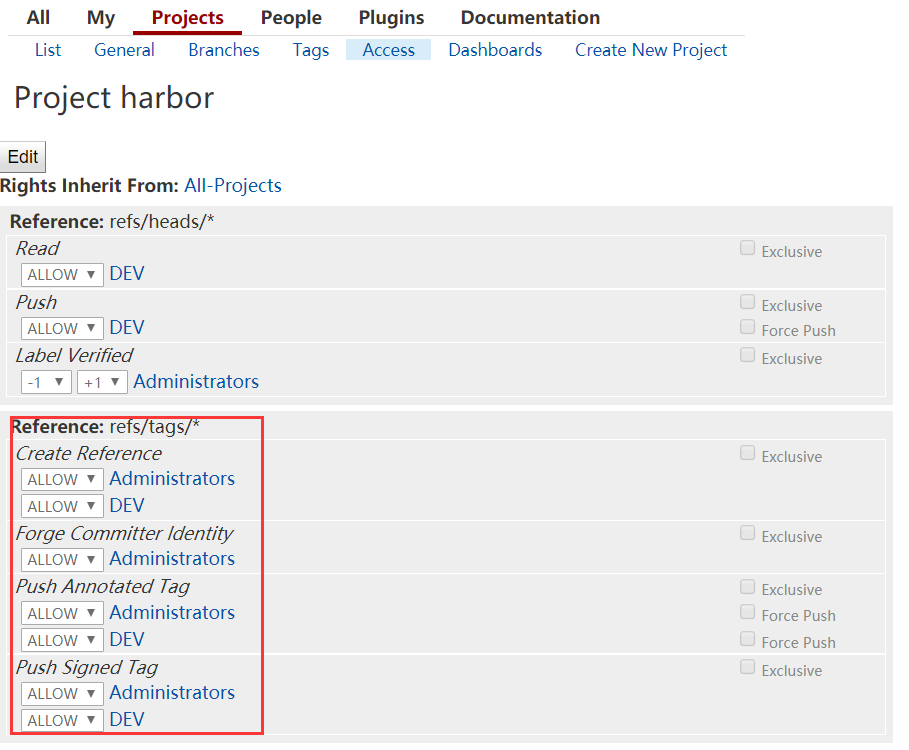
**Create references**:此权限管理用户是有可以创建references，branches，tags。此权限一般与普通的push权限一起被分配。

**Push Annotated Tag**:此类权限允许用户向工程仓库提交一个annotated tag。

**Push Signed Tag**:此类权限允许用户向工程仓库提交一个PGP签名的 tag。

**Forge Committer Identity**:如果要提交不是自己打的Tag，则必须同时分配Forge Committer Identity权限。

更多权限参考：<https://www.jianshu.com/p/d95faa9f8b05>



## Gerrit账号相关操作

生成管理员账号

touch /gerrit.password

htpasswd -m /gerrit.password admin 123456a?

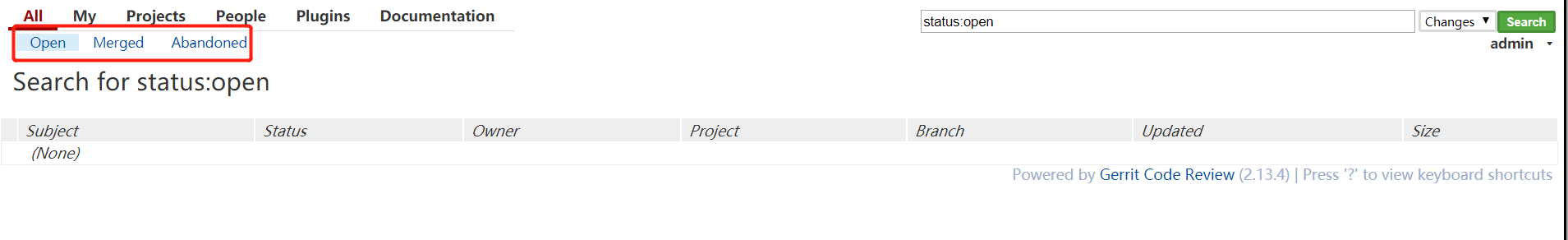
添加其他账号

htpasswd -b /gerrit.password Jack 123456a?

Gerrit的安装相关操作参考：

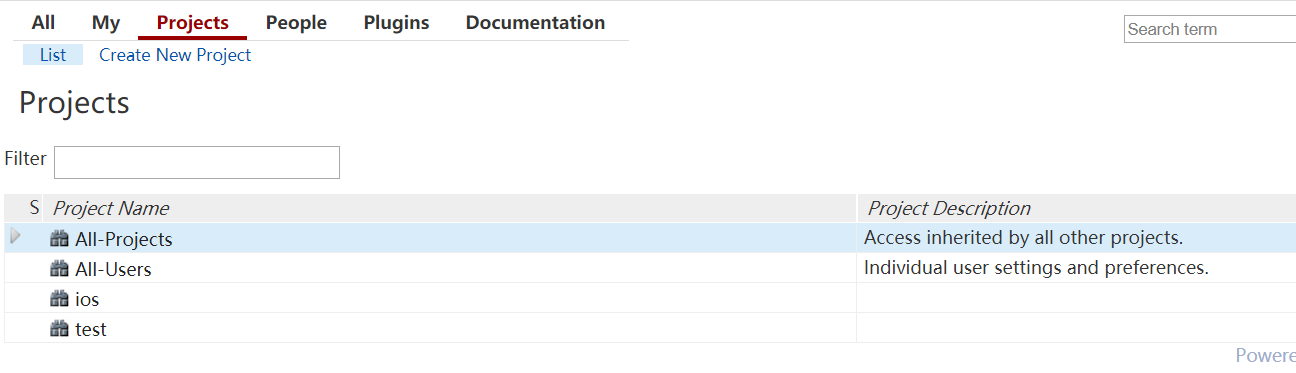
## Gerrit界面介绍

### 提交页面

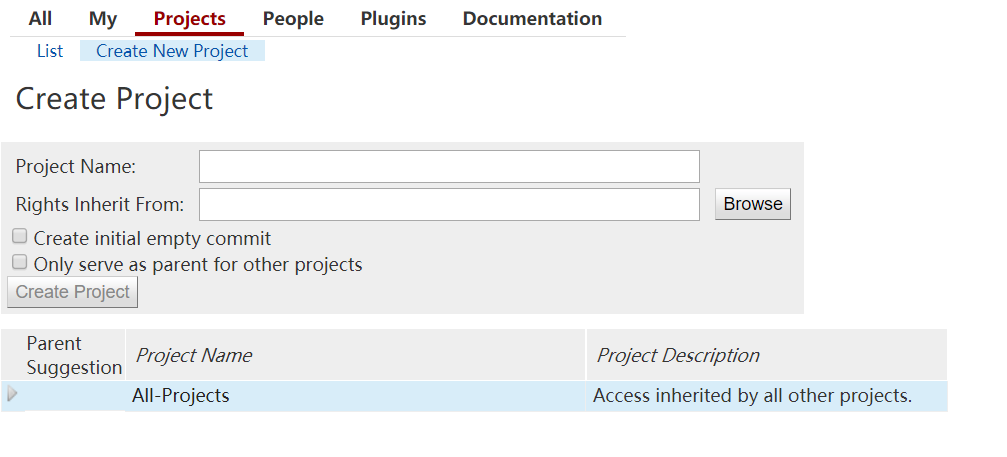


这个open merged abandoned分别代表未被审批过的提交，已经合并成功的提交，被放弃的提交。

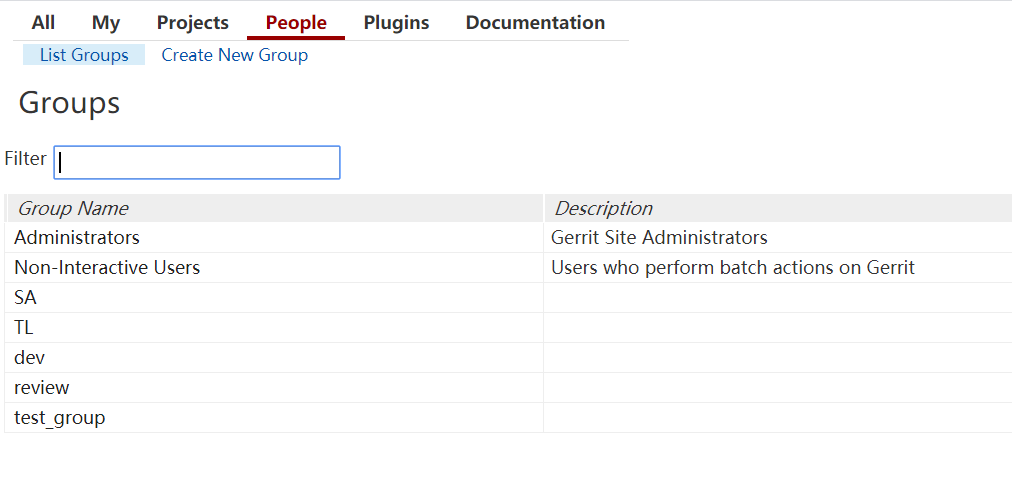
### 项目页面



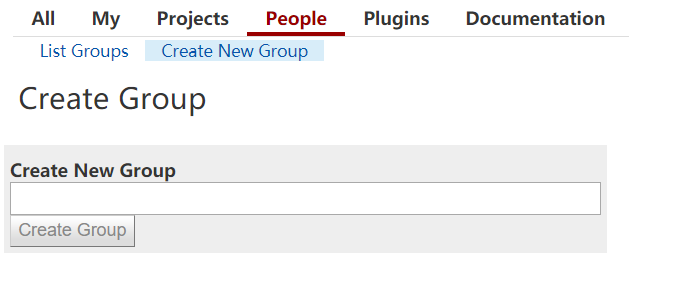
这个项目页面可以看到所有的项目。也可以创建项目。



### 用户组页面



这个用户组页面可以看到所有的用户组。也可以创建用户组。



## 项目设置

考虑到项目需要和gitlab集成。项目的创建参考文档：



## 项目权限设置

参考：<https://www.jianshu.com/p/d95faa9f8b05>

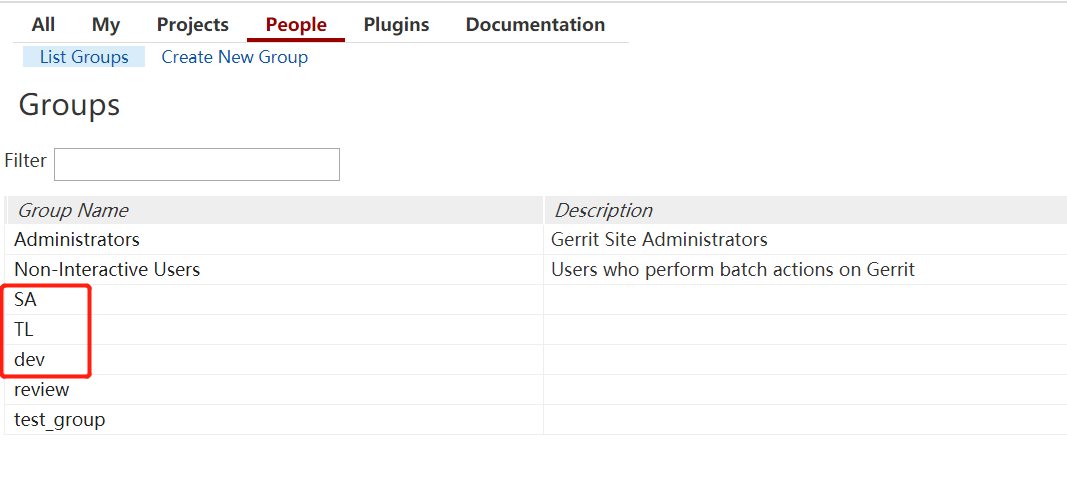


Forge Author， Forge Committer两个权限用于绕过git中的user和email验证。（后续可加入）

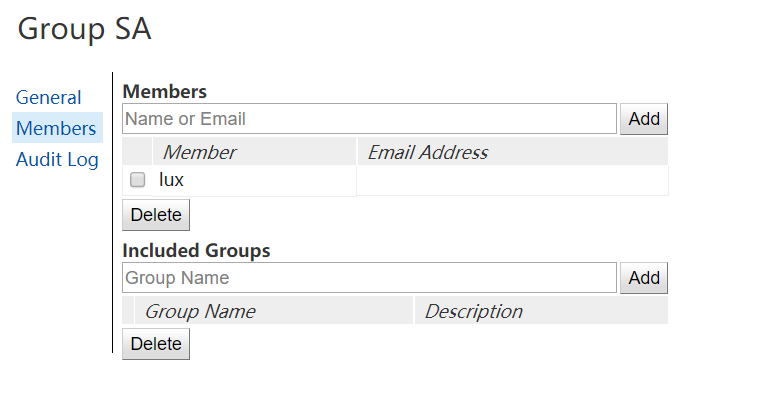
## 用户组设置

成员组设置

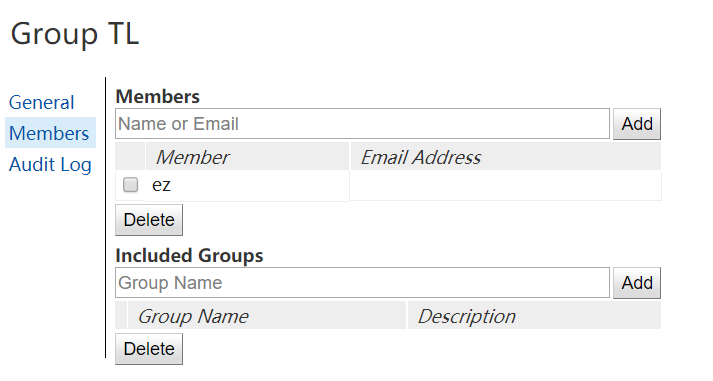
分为三组：SA TL DEV



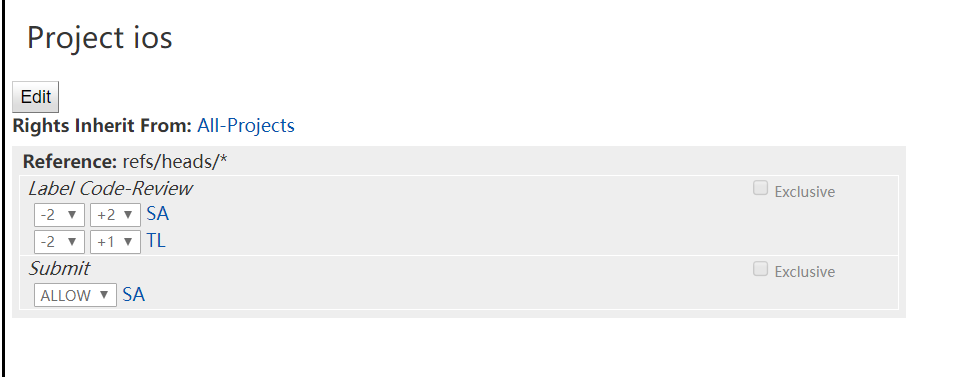
SA 成员



TL 成员



项目设置



重点设置Label Code-Review

SA为-2~+2

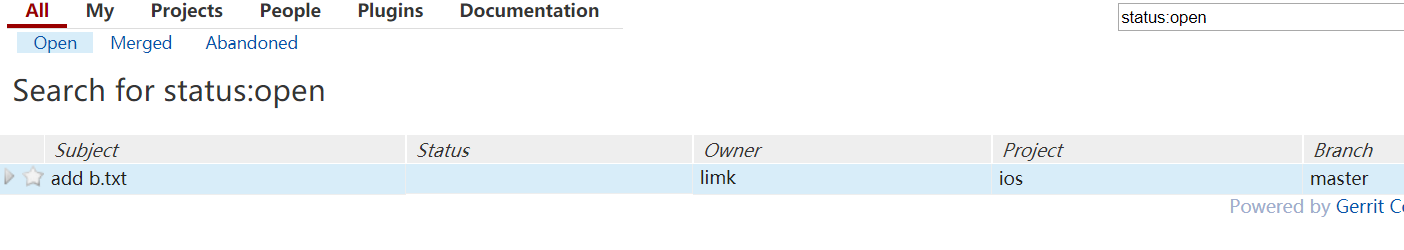
TL为-2~+1

重点设置 Submit添加SA组

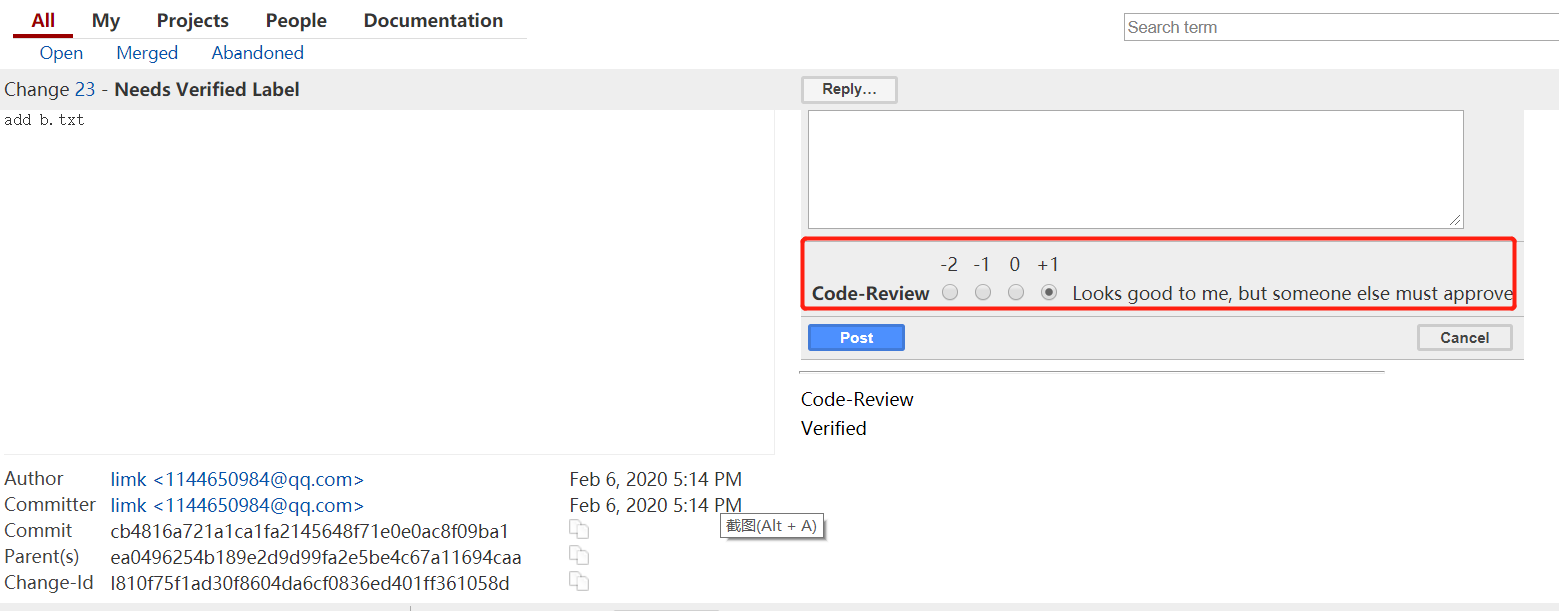
## 代码评审

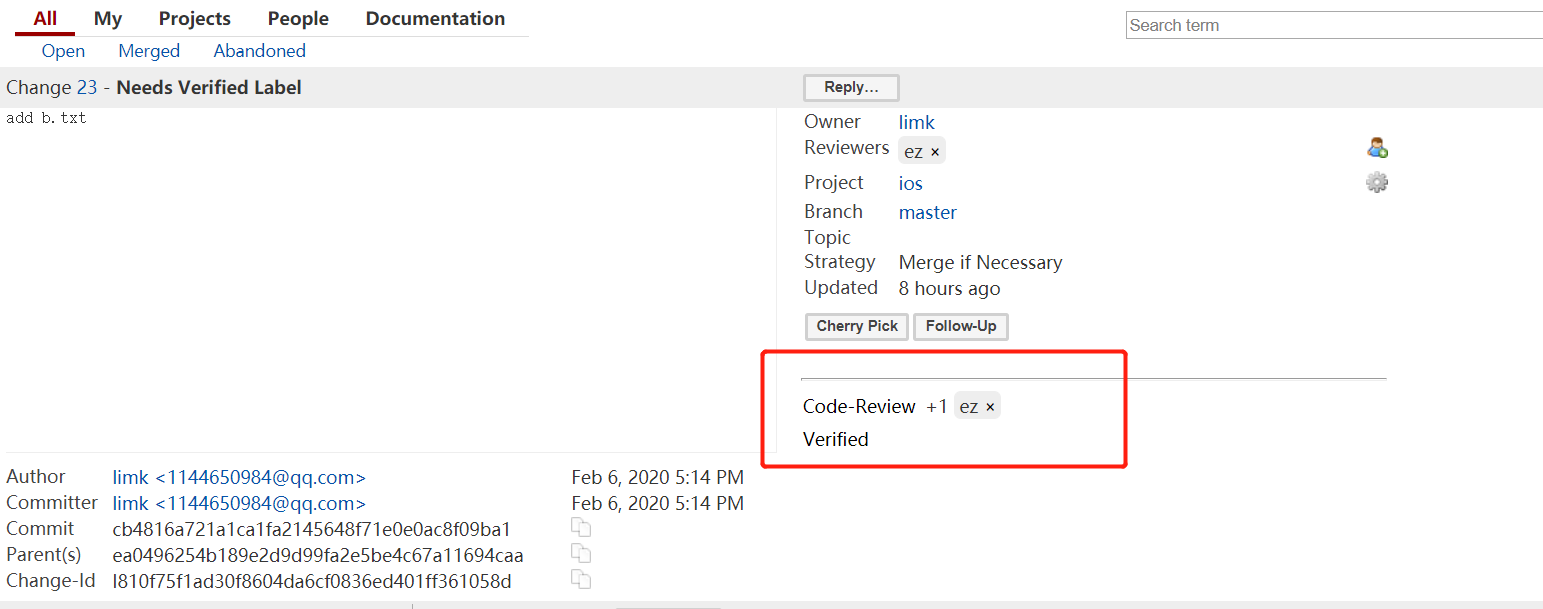
每当有dev的成员提交代码，TL和SA的成员可以进行代码review，打分。

此时有一个提交。

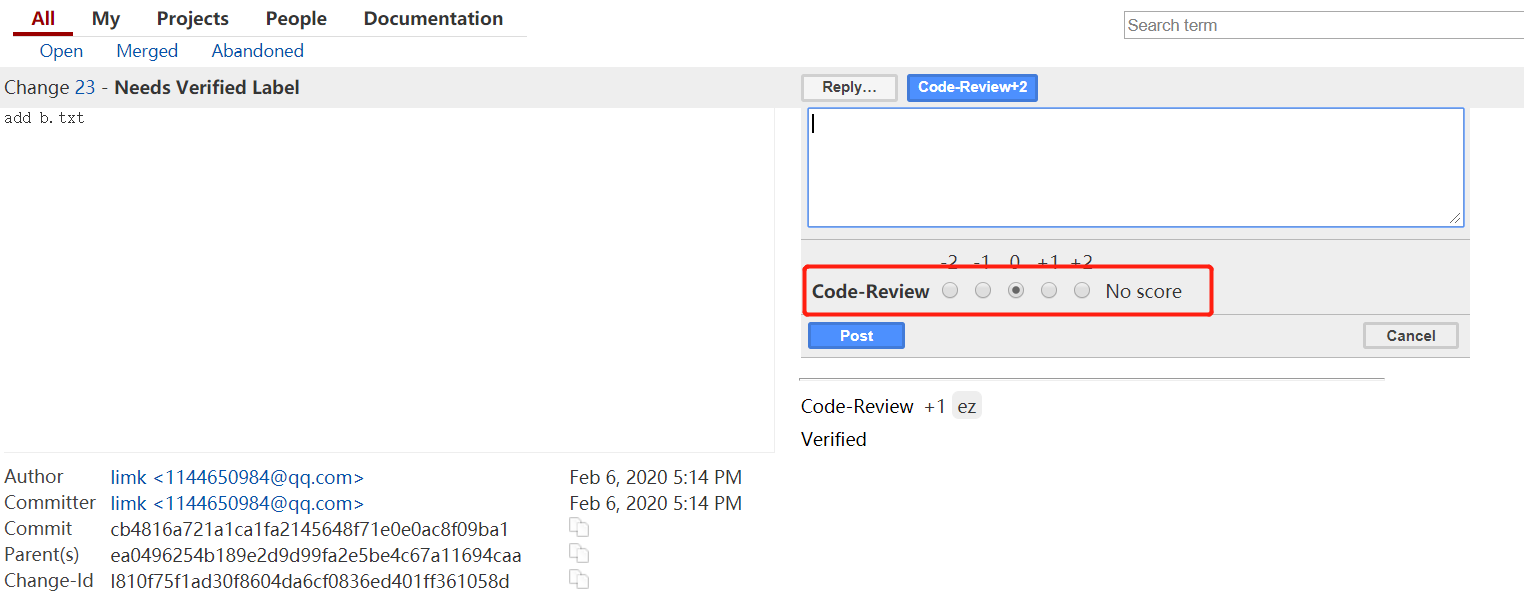


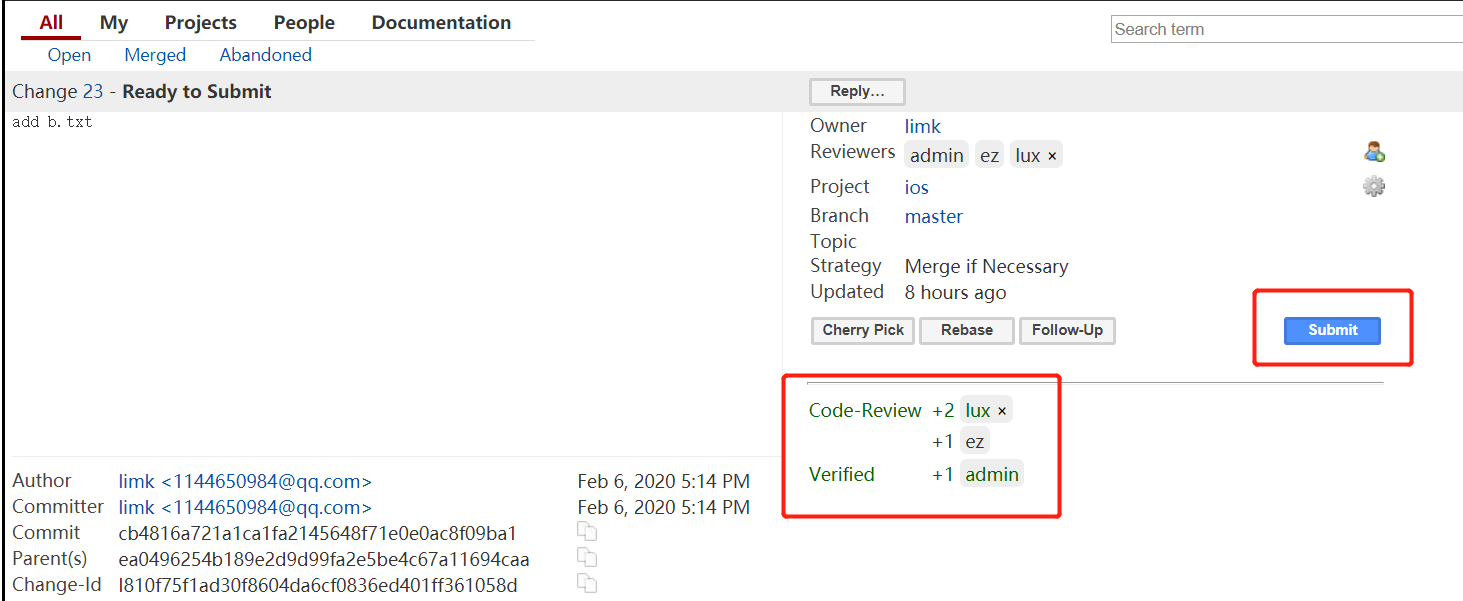
TL的成员可以看到如下，他可以+1通过。



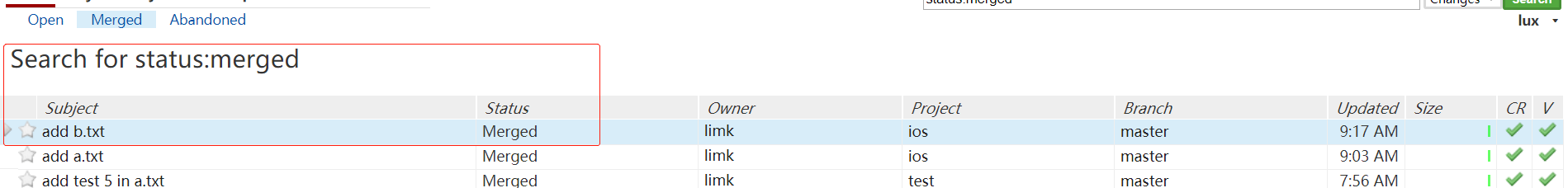


SA的成员可以看到如下，他可以+2通过。



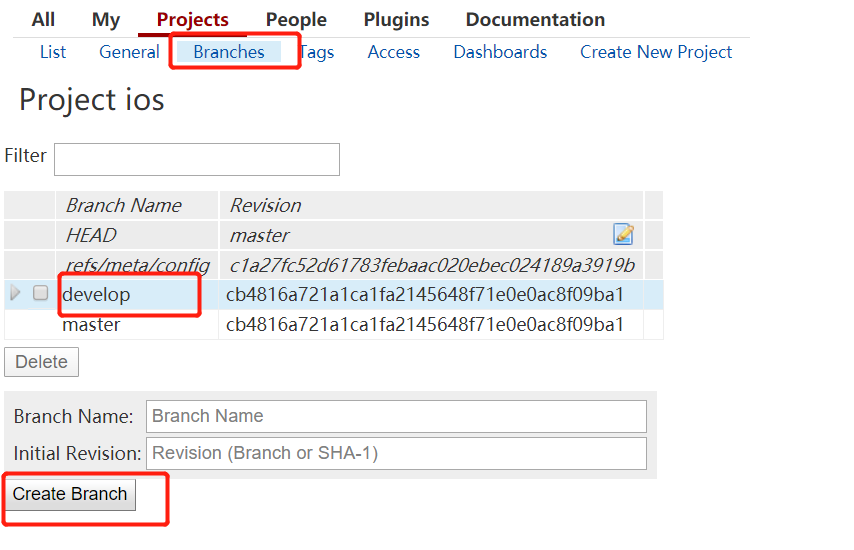


SA点击submit。（关于verified的选项在后续补充）



### 多分支代码合并

首先创建分支



此时是创建的develop分支，其他分支同理（hotfix，feature）。

## 分支的规则设置

总体原则：谁有最终决定权，谁的Label Code-Review就设置为+2、并且有submit的权限。

以下执行基于develop分支，其他分支同理（分支情况根据实况决定，下面只是参考）。

Master分支可以设置为SA组（+2 submit）。

Hotfix分支可以设置为SA组（+2 submit）、TL组（+1）。

此时的develop分支的规定为SA和DL组都可以review和submit。

那么设置如下：

此时的TL和SA都可以+2，并且可以Submit

