### 安装Git

官网链接  
<https://git-scm.com/downloads>

下载后根据自己需求安装即可。

安装完成后在桌面鼠标右击可以看见Git Bash Here说明安装成功。

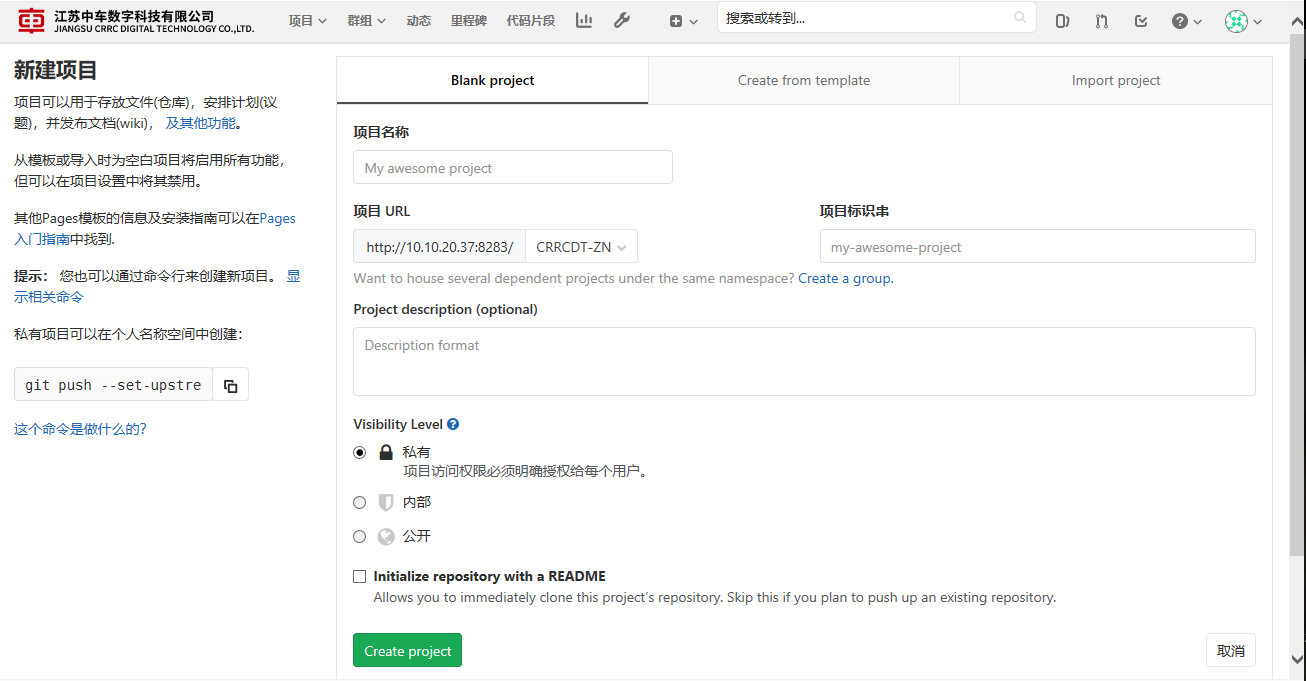
### 注册账号

账号的注册与gitlab的管理策略有关，如不能直接注册，请联系gitlab管理员并提供邮箱，姓名等信息，由管理员添加账号，添加完毕后到提供的邮箱中找到初始化邮件并按提示修改密码即可，提供开放注册的省去找管理员这一过程，直接填写相关信息并在邮件中的地址中重置密码即可。

不习惯英文环境的可以在登陆后到设置里将语言改为简体中文

### 项目的创建

找到你所在中心或部门的群组，点击新建项目，填写项目的名称，标识串，介绍信息等，点击【创建】即可，新建项目不限于新建空项目，也可以从其它你有权限的项目中引申，在创建页面的选项卡里可以选择。



### 项目的管理

创建了项目以后就可以点击项目名进入你的项目了，页面布局大体如下：



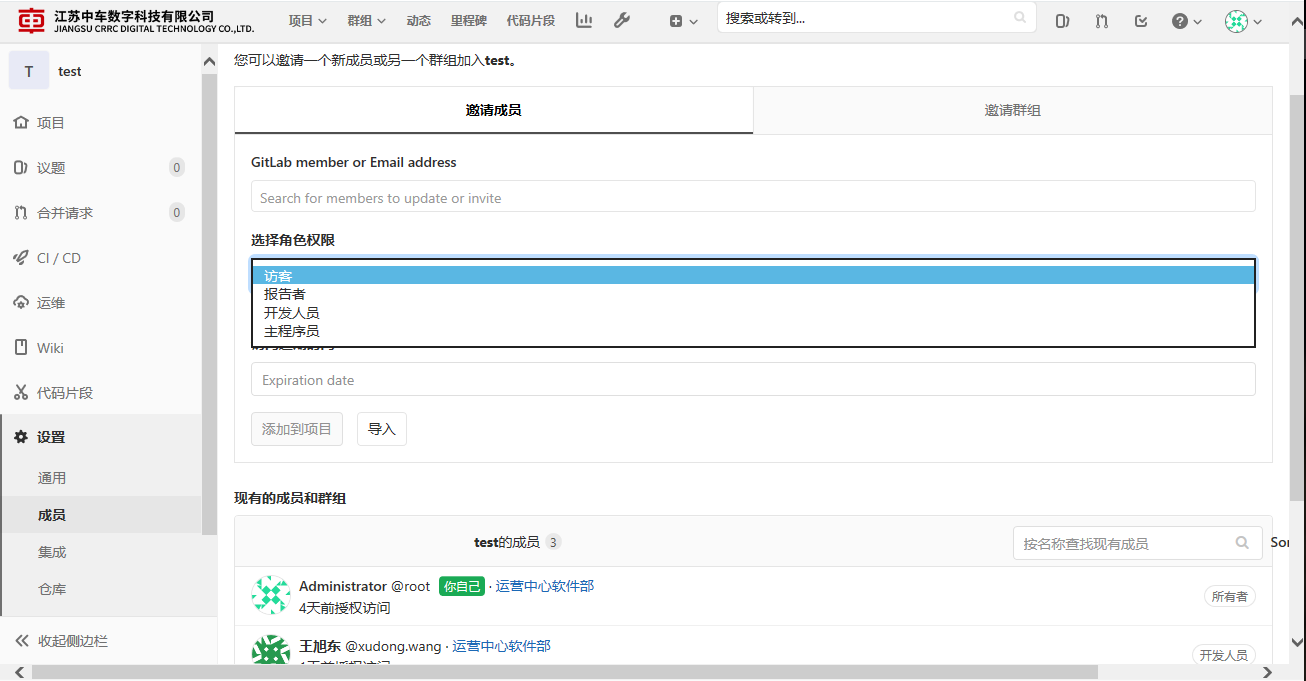
在项目页面中，可以看到项目的创建者是作为项目的所有者存在的



在左边菜单的【设置】-【通用】中，可以修改你的项目信息，展开【高级】选项，可以对项目进行整体的管理，包括导出、归档、转移、删除等。

添加成员

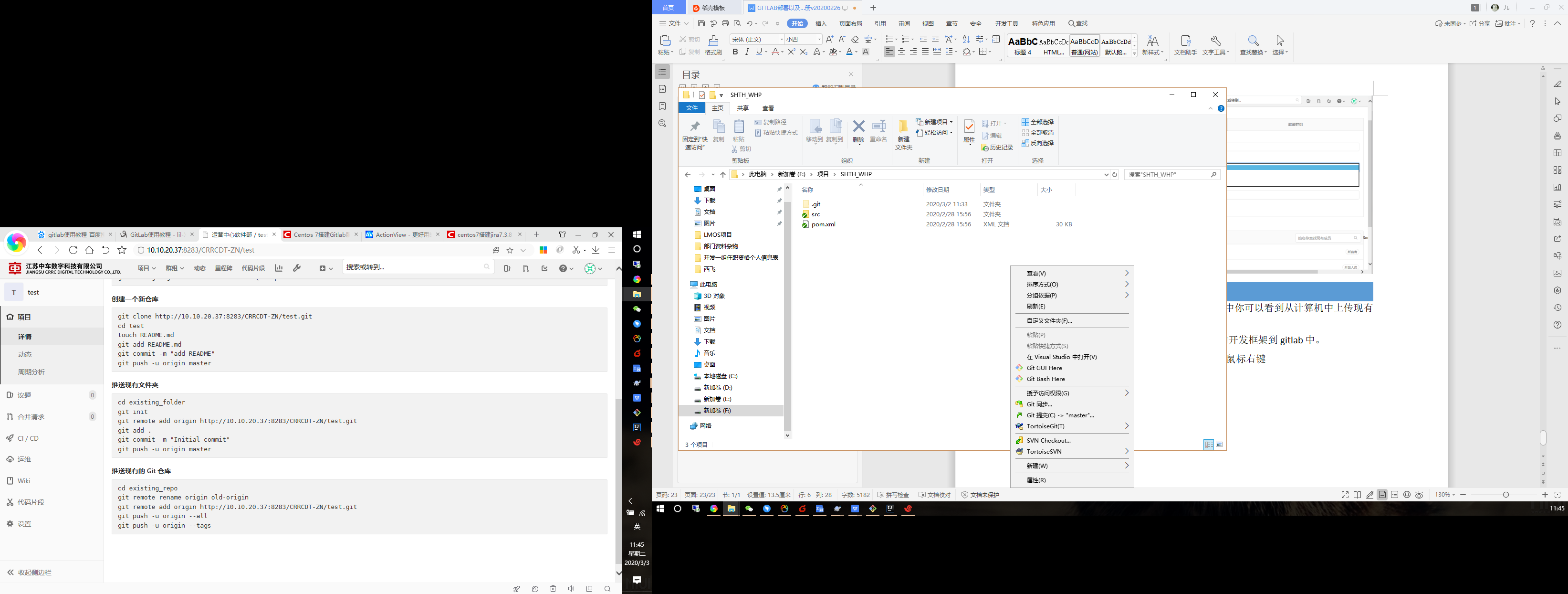
在【设置】-【成员】中，可以邀请成员与群组进行合作开发，你可以通过用户名或邮箱来邀请成员，赋予他们权限，也可以根据项目投入控制成员的访问期限。



项目初始化（一般由主程序员负责）

点击【项目】，进入项目主页，在命令行指引中你可以看到从计算机中上传现有文件的几种方式。

作为项目的拥有着，此时可以推送本地现有的开发框架到gitlab中。

首先在需要上传到gitlab的项目文件夹中点击鼠标右键

在菜单中选择Git Bash Here，弹出git的控制器窗口

git init

git remote add origin http://10.10.20.37:8283/CRRCDT-ZN/test.git

git add .

git commit -m "Initial commit"

git push -u origin master

①git init 进行git的初始化

此时在文件夹中出现了隐藏目录.git，这个文件与svn中的.svn同理，负责该文件夹中的版本控制。如果删除，则本地的文件脱离版本控制。



你可以在gitlab下载（clone）公用的空开发框架（clone方式请看之后说明），然后删除.git文件夹，再上传到你创建的gitlab空项目中作为你的项目。

②git remote add origin 项目地址.git

此命令是将本地项目与gitlab远程仓库绑定

③git add .

git add命令是将文件提交到暂存区 后面的 ’.’是将工作去所有为跟踪或者修改的文件添加到暂存区（不包括删除，git add all包括删除），如果你只想提交部分文件，有以下几种用法：

git add xxx.xml 这个是将xxx.xml文件添加到暂存区

也可以添加多个文件，中间用空格分离即可

git add a.xml b.xml

也可以使用通配符进行批量提交

git add \*.html 将所有html文件提交

在开发时，通常只需要将代码和配置文件等提交即可，不要将编译文件等加入到控制中

在git add命令执行完成后，可以输入git status查看自己添加的文件，如果添加错了文件，可输入 git reset 文件名 撤销git的add操作

④git commit -m "Initial commit"

这条命令是提交更新，将暂存区的文件（新增或更新的）提交到git仓库（本地）

-m 表示可以直接输入后面的”message”，本次提交的说明可以写在message中，如果没有-m，提交的时候会调用vim编辑器让你输入message

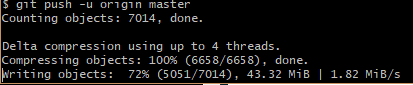
⑤git push -u origin master

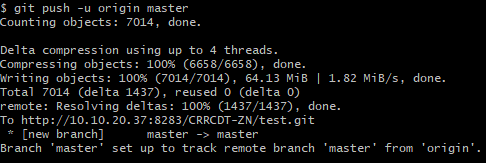
在使用git commit命令将暂存区的文件提交到本地仓库后，执行git push就可以将本地仓库的分支推送到绑定的远程服务器上（gitlab服务器）的分支了

git push的一般形式为 git push <远程主机名> <本地分支名> <远程分支名> ，例如 git push origin master：refs/for/master ，即是将本地的master分支推送到远程主机origin上的对应master分支， origin 是远程主机名。第一个master是本地分支名，第二个master是远程分支名。

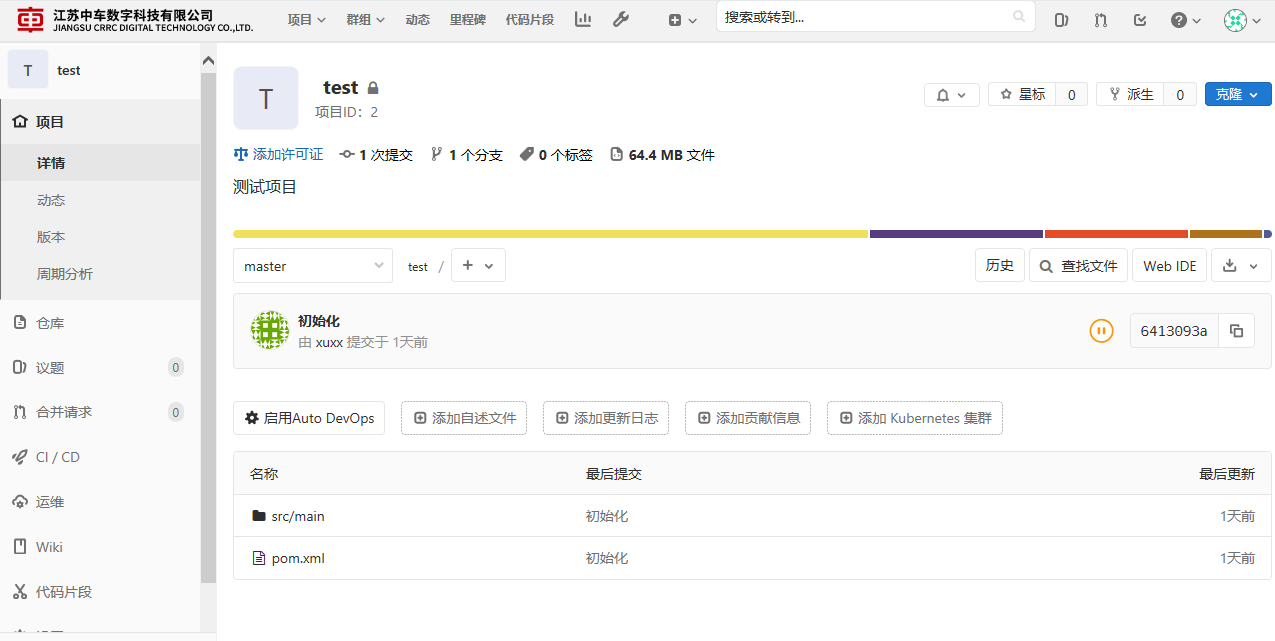
-u参数是指定一个默认的分支，指定之后使用git push命令就直接会推送到指定的分支中。

如果是第一次使用git上传到某gitlab中，push之前会有对话框弹出，要求输入gitlab的用户名与密码。





上传完成以后刷新项目网页：



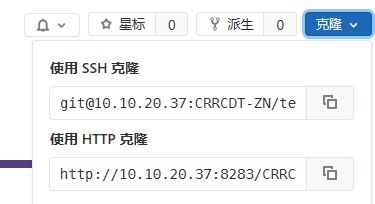
master一般作为项目的主分支，除项目的拥有者初始化上传的时候外，不建议在开发时将更新内容直接推送到master主分支（owner也不建议，developer无权限）。（详见↓分支与合并）

# 项目下载（克隆）⭐

每个项目右边都有一个Clone（克隆）选项：



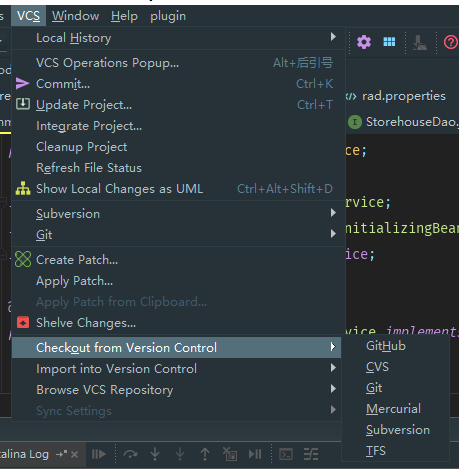
包括通过配置ssh克隆和直接使用http克隆，这里介绍第二种用HTTP连接从远程仓库下载项目到本地。



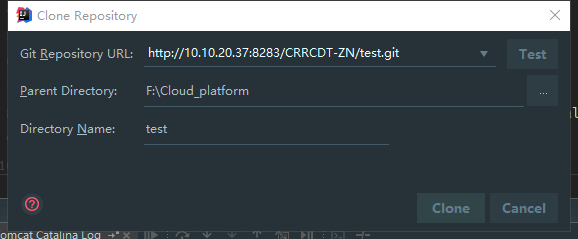
①git

复制HTTP克隆的地址，在你的为项目准备的空文件夹中右键点出git bash here，在命令行页面直接输入git clone 链接即可。

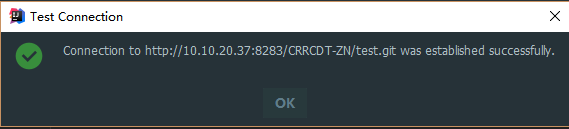
②IDEA



在IDEA的VCS菜单中选择Checkout from Version Control，选择Git（注意不是GitHub）



将URL粘贴到对话框中，点击Test测试连通性，成功则：



在选择好文件夹与命名之后点击Clone，如果是第一次使用，会要求输入gitlab的用户名与密码。

### 分支与合并

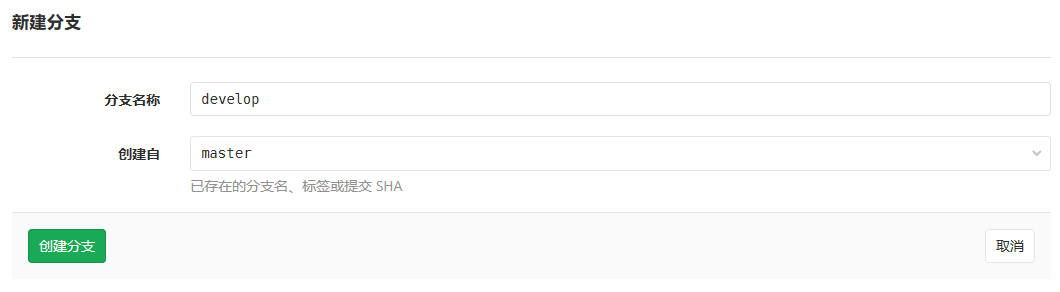
当我们需要调试某个Bug或者尝试添加或修改程序中的某个模块，而又不能影响主分支的开发时。就可以通过创建分支来满足需求。创建分支相当于是创建一个新的分支指针指向当前所在的提交。

注：有的项目将主分支当作发布版本，所以在开发分支中的程序需要经过代码review以及充分测试后才能合并到主分支；如果项目有条件，可以通过钩子将开发分支中的代码同步部署到测试服务器中进行测试。

分支的创建



如图所示，点击新建分支



可以根据开发需要来添加分支，比如多个模块同时开发，就可以建立多个分支。

如果项目分为多个版本，或者有开发内容是基于某个分支，也可以选择来源分支进行创建。

切换分支与提交

看使用习惯git与IDEA二选一（目前的主流开发工具都有支持，这里用InteliJ IDEA演示）

①git

在目录下打开git bash here

查看远程分支

git branch -r

查看本地分支

git branch

拉取远程分支并创建为同名的本地分支,从远程的develop分支更新最新的版本到本地的develop（新建）分支中

git fetch origin develop:develop

切换本地的分支（切换分支前，当前的分支不能有未提交的修改）

git checkout develop

在本地有代码新增或修改后

如果多人同时使用同一个分支：

在push之前，一般我们会先pull一下，这里不建议用git pull，而是用git fetch + git rebase

git fetch -p origin develop //更新远程commit id到本地新临时分支，以及删除远程仓库已删除的分支

git rebase //整理rebase

如果有冲突：

添加文件

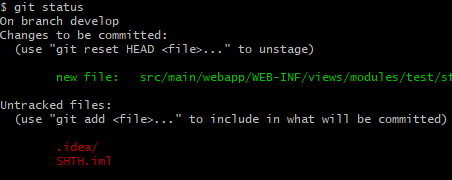
git add .

git commit -am “” //冲突解决后提交

git rebase -continue //整理commit

查看git状态：

git status



提交新增或修改的文件到本地仓库

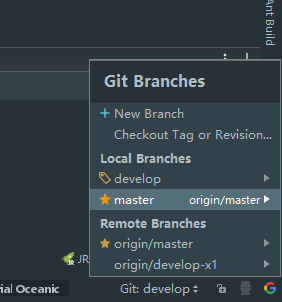
git commit -m ‘开发’

推送代码到远程分支(这里非项目的所有者无法推送到master分支)

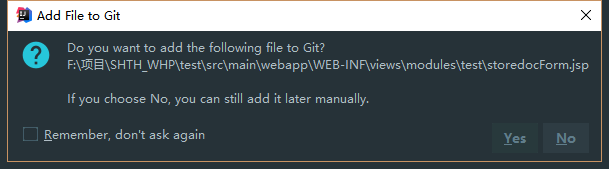
git push origin develop

# IDEA 推送与更新 ⭐

通过IDEA下载的git项目中，在右下角或VCS中的git菜单中

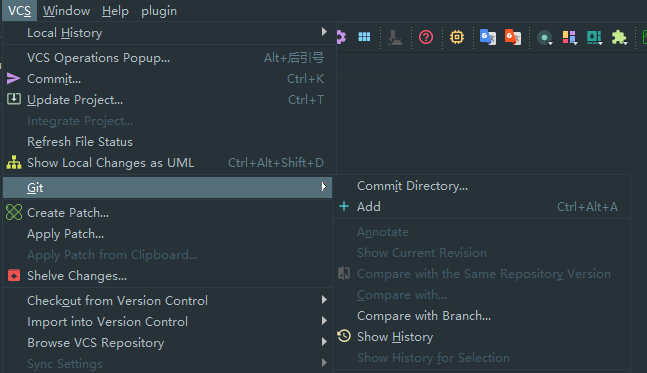
在这里你可以快速的切换仓库

IDEA在下载并打开从git下载的项目后，如果有新增的文件，会弹出对话框询问是否加入版本控制



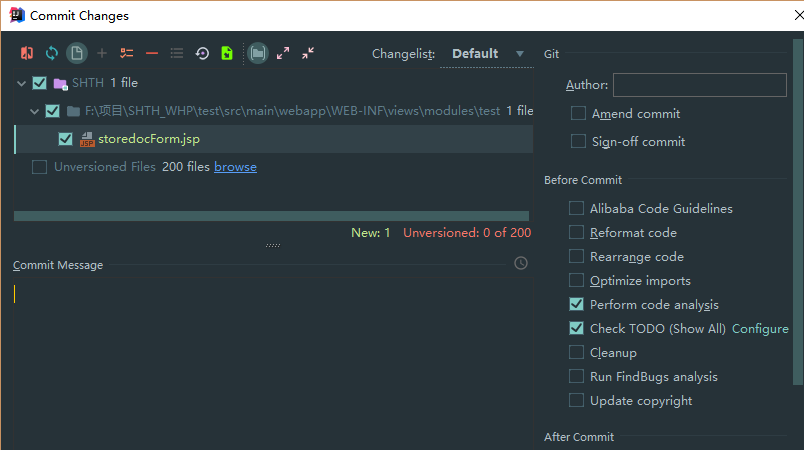
提交之前需要先update Project，如果有冲突可以在IDEA中解决后提交。

在需要提交时，点击下图所示菜单



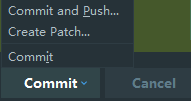
或按钮

点击commit

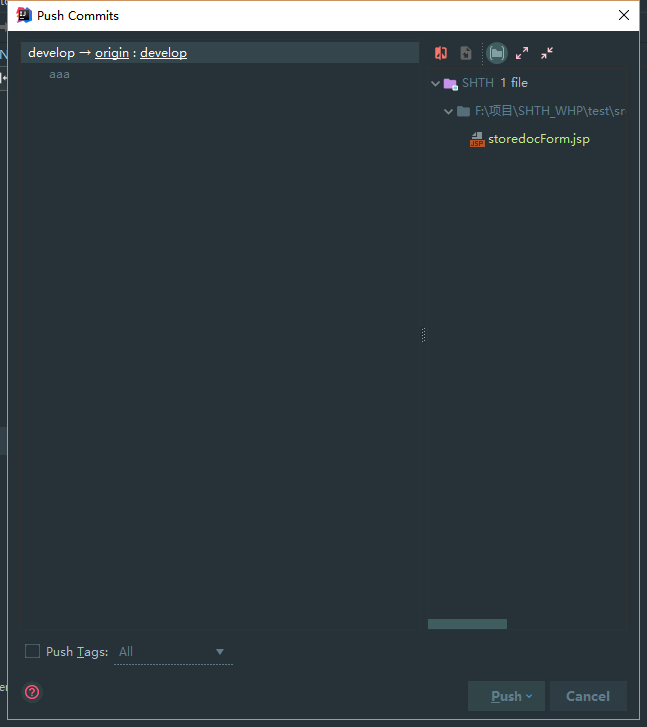


在弹出的对话框中选择你需要提交的文件，并输入Message

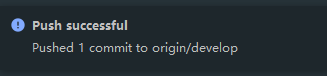
选择Commit and push



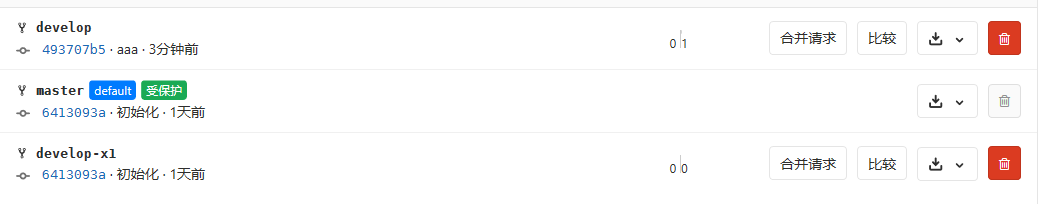
也可以选择commit先提交到本地仓库，等需要推送到远程仓库时一起推送，这时会将前几次commit但没有push的内容一并推送



push前注意填写要推送的分支，然后点击push

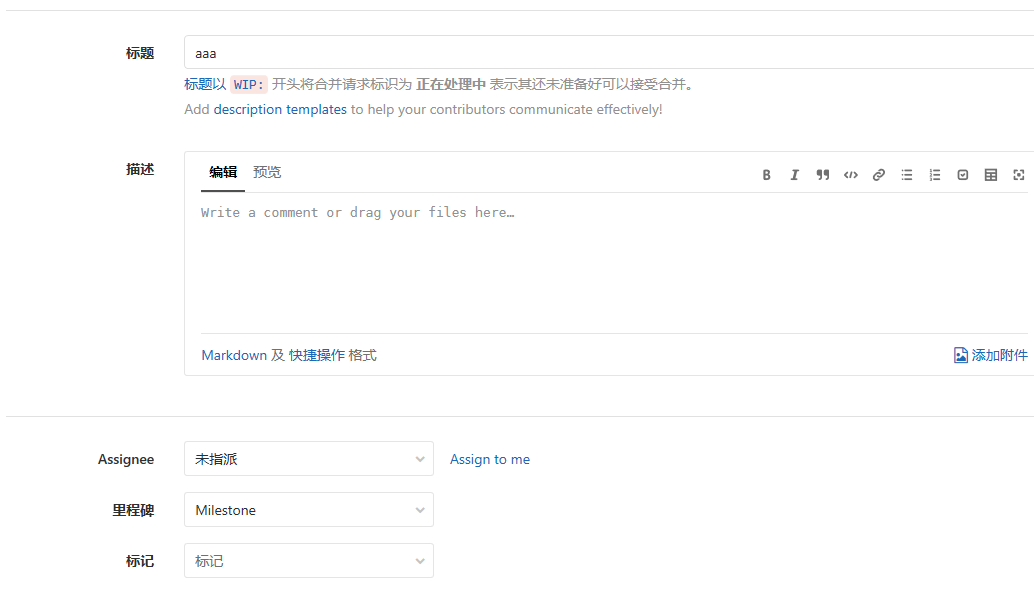


分支的合并

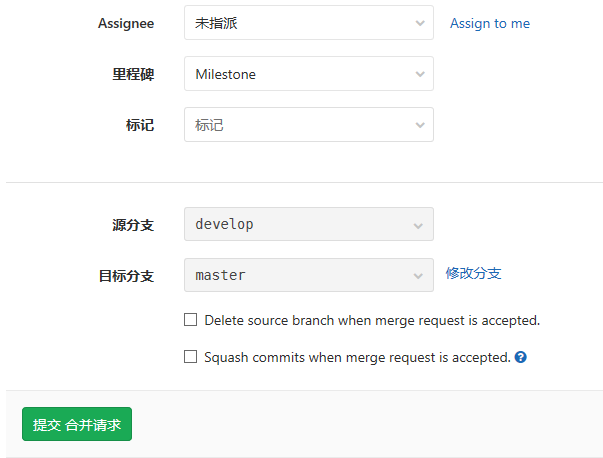


打开页面，可以看到develop分支已经有推送的信息，但此时下载master分支还不能看到更新的内容。

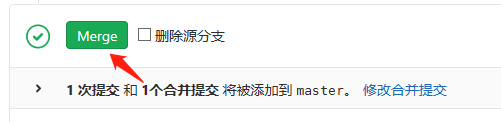
此时开发人员或者项目拥有着可以点击合并请求按钮来发起合并请求，也可以在线比较查看更新内容

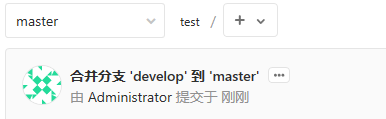


在创建请求的页面可以指定操作人（一般这一步由项目的master来操作）以及编辑请求标题与描述信息等



在确认无误后，提交合并请求，此时项目被指派的人可以看到合并请求，如果没有冲突，在确认无误后，点击Merge，即可将分支合并。





合并后可以在主分支看到合并消息，此时master中可以看到对应的新增/修改的文件

### commit撤销

在项目的主页中点击【历史】按钮，可以看到提交的记录。

点击一条记录，在操作中可以选择撤销。



撤销操作也可以作为一个合并请求，流程如前文。

### 8）使用gitlab的多人协作分支开发控制流程

①建立项目&添加协作者

②初始化项目

③建立分支

在大型项目时：

保护分支：master分支为主分支，每个项目的develop分支为开发分支

开发人员分支：从develop分支中建立开发人员个人分支，如develop\_wxd,develop\_xxx等

此时master分支为稳定发布版本分支，develop分支为开发分支，开发分支打包或同步到测试服务器进行测试，测试成功则合并到master主分支作为发布版本。

④开发

开发者将项目clone到本地进行开发，开发人员checkout develop分支的代码，运维人员checkout master主分支代码。

开发者开发结束后，推送代码，发起合并请求，经审核后合并到develop开发分支中。

测试稳定的代码合并到主干master，通常由项目owner操作

## 后话

这篇攻略仅是个人在部署与使用中的一些心得，如果有好的想法或者不准确的地方欢迎指出与补充，另外也鼓励大家多摸索功能，然后将使用新的分享。