目录

[1. 进入git官网(https://github.com) 1](#_Toc9973353)

[2. 注册账号并登录(Sign in) 1](#_Toc9973354)

[3. 创建本地与远程得连接(用ssh)搭桥 1](#_Toc9973355)

[3.1 下载并安装git 1](#_Toc9973356)

[3.2在本地创建ssh key(公钥) 1](#_Toc9973357)

[3.2 在远程创建ssh key 2](#_Toc9973358)

[4. 创建远程仓库 2](#_Toc9973359)

[4.1 点击“new repository” 2](#_Toc9973360)

[4.2 输入仓库名，“create repository” 2](#_Toc9973361)

[4.3可以看到对应的仓库地址 3](#_Toc9973362)

[5. 使用git下载项目仓库 3](#_Toc9973363)

[6. 在idea里面打开项目空间 3](#_Toc9973364)

[7. Git的使用 4](#_Toc9973365)

[7.1 代码的提交和更新 4](#_Toc9973366)

[7.2 新建分支 4](#_Toc9973367)

[7.3 在分支(develop)上开发代码并合并到主干(master) 5](#_Toc9973368)

[7.3.1 切换分支 5](#_Toc9973369)

[7.3.2 将分支代码合并到主干 5](#_Toc9973370)

# 进入git官网(<https://github.com>)

# 注册账号并登录(Sign in)

# 创建本地与远程得连接(用ssh)搭桥

## 下载并安装git

## 3.2在本地创建ssh key(公钥)

创建ssh key:右键在gitBash里输入：

ssh-keygen -t rsa -C [17600272806@163.com](mailto:17600272806@163.com)(替换邮箱账号)，然后一路回车，使用默认值即可

在用户主目录下(C:\Users\W\_LIURUNKAI)，看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果已经有了，可直接跳到下一步(第4步)

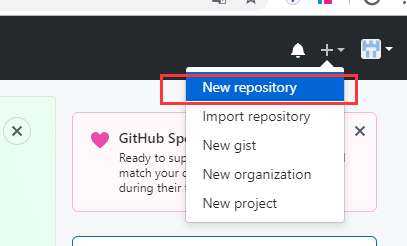
## 3.2 在远程创建ssh key

登录GitHub，打开“settings”，“ssh and gpg keys”：然后点击“new ssh key”，填上任意title，在key文本框里粘贴id\_rsa.pub文件的内容：

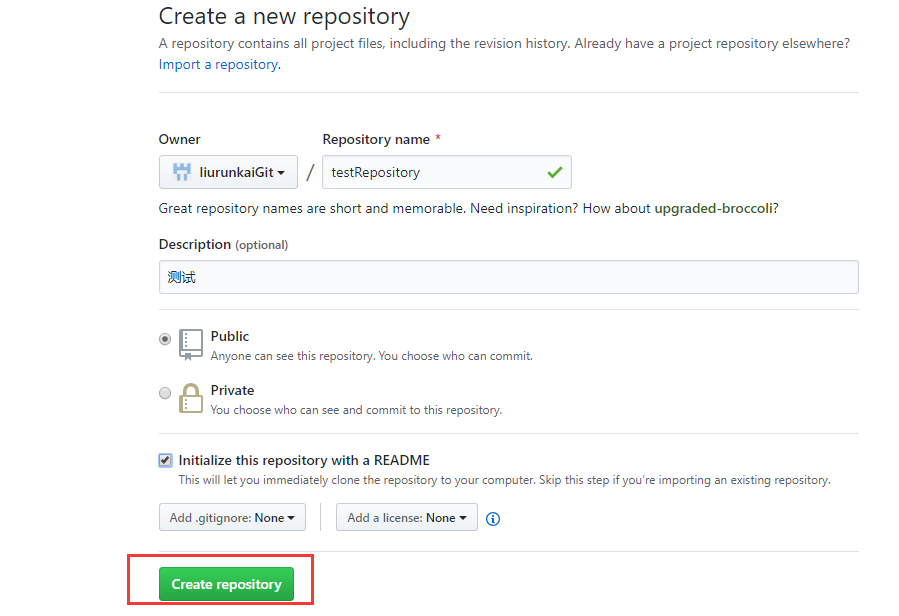
点击“add ssh key”

# 创建远程仓库

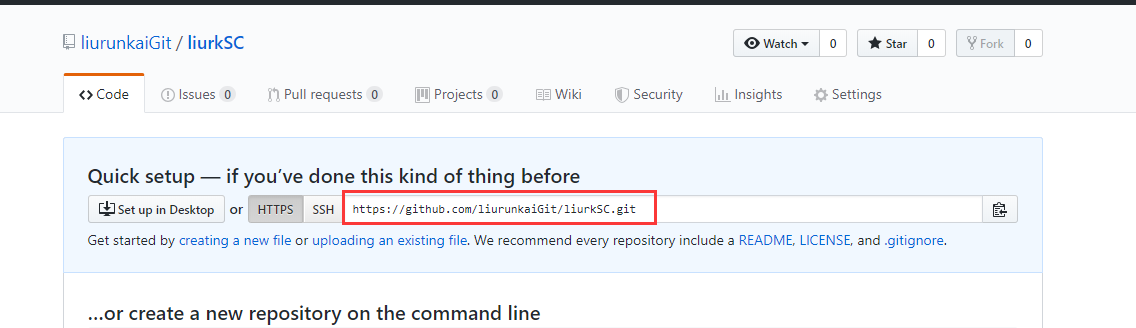
## 点击“new repository”



## 4.2 输入仓库名，“create repository”

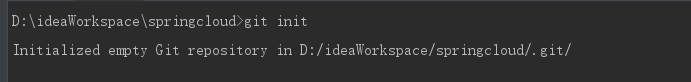


## 4.3可以看到对应的仓库地址

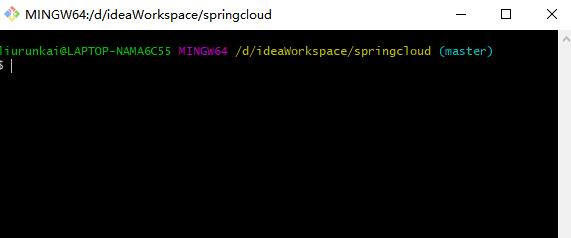


# 将本地项目上传到github

进入项目所在的目录执行git init

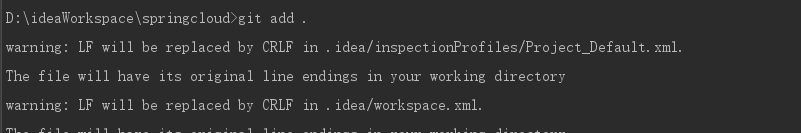


或者也可以在文件夹内右键选择Git Bash Here



使用git add .

将项目添加到git库中



使用git commit -m “项目初始化”

将项目提交到git仓库中



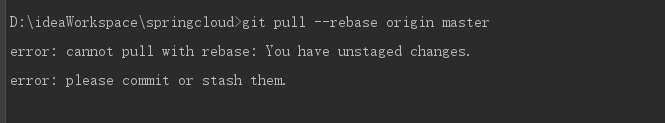
按照步骤4新建一个对应远程仓库

使用命令git remote add origin 远程地址

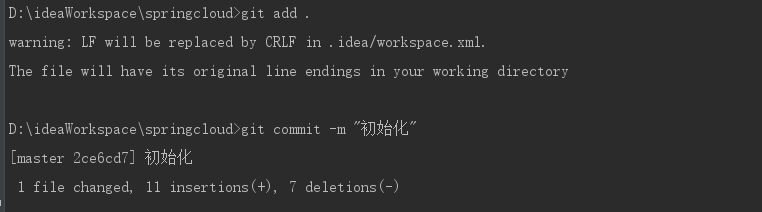
和git远程地址建立连接



使用命令git pull --rebase origin master



提示让commit代码，然后add和push

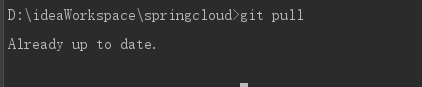


使用命令git push -u origin master

将本地的代码提交到远程上并push



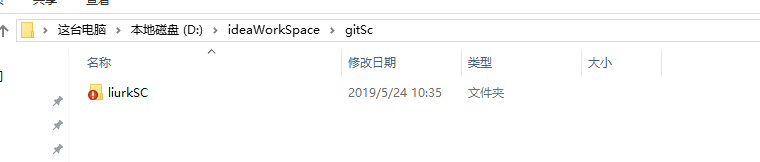
再次从github上拉代码



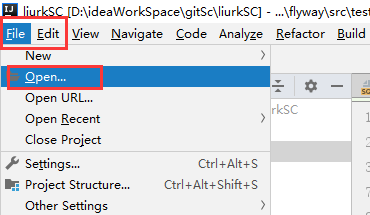
# 使用git下载项目仓库

切换到存放项目的目录(D:\ideaWorkSpace\gitSc)，右键gitBash

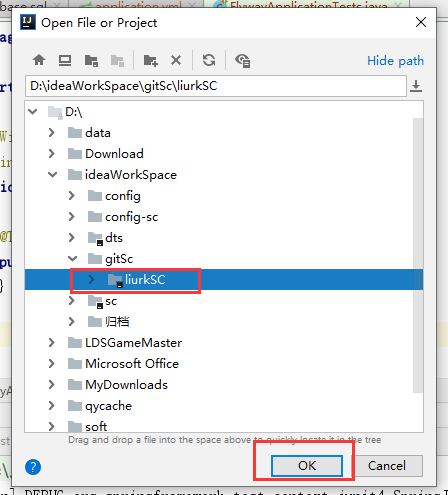
输入命令git clone <https://github.com/liurunkaiGit/liurkSC.git>进行项目的下载



# 在idea里面打开项目空间



选择刚刚下载下的目录



# Git的使用

## 克隆项目

git clone url

git clone url 别名，本地创建的仓库名称变为别名，而不是git服务器上的名字

## 代码的提交和更新

使用git界面

提交代码前先更新代码：ctrl+t

然后提交代码：ctrl+k，选择commit and push

使用git命令行工具

pull：git pull

status（查看当前文件状态）：git status

使用git status -s 可以得到另一种更紧凑的格式输出

add：git add 文件名

git add . (add所有)

git add -p(详情，要add的输入y，不add的输入n)

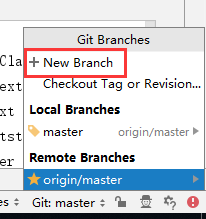
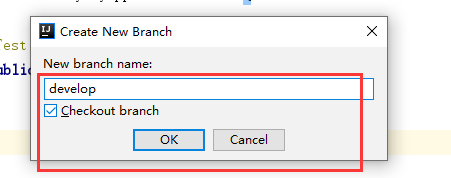
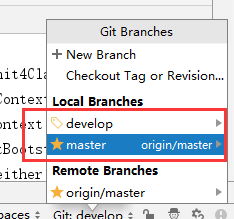
commit：git commit -m ‘修改bug’

直接使用git commit -a -m ‘提交备注’ 可以跳过git add这一步

push：git push

## 新建分支

使用git界面

使用git命令行

查看分支：git branch

新建分支：git branch 分支名

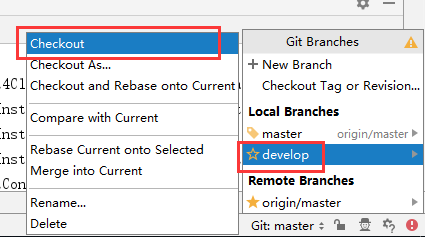
切换分支：git checkout 分支名

新建并切换分支：git checkout -b 分支名

删除分支：git branch -d 分支名

## 7.3 在分支(develop)上开发代码并合并到主干(master)

### 7.3.1 切换分支到master（git checkout master）



### 7.3.2 将分支代码合并到主干（git merge 分支名）

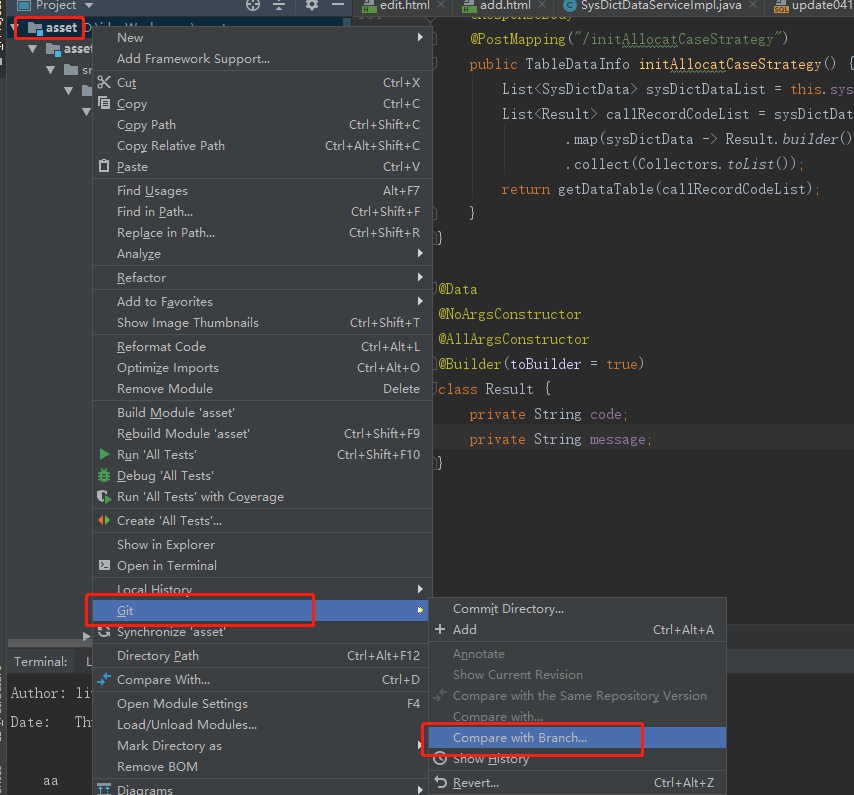
将develop的代码合并到master分支，首先切换到master，然后点击develop的merge into current（使用命令：git merge develop）

|  |
| --- |
|  |

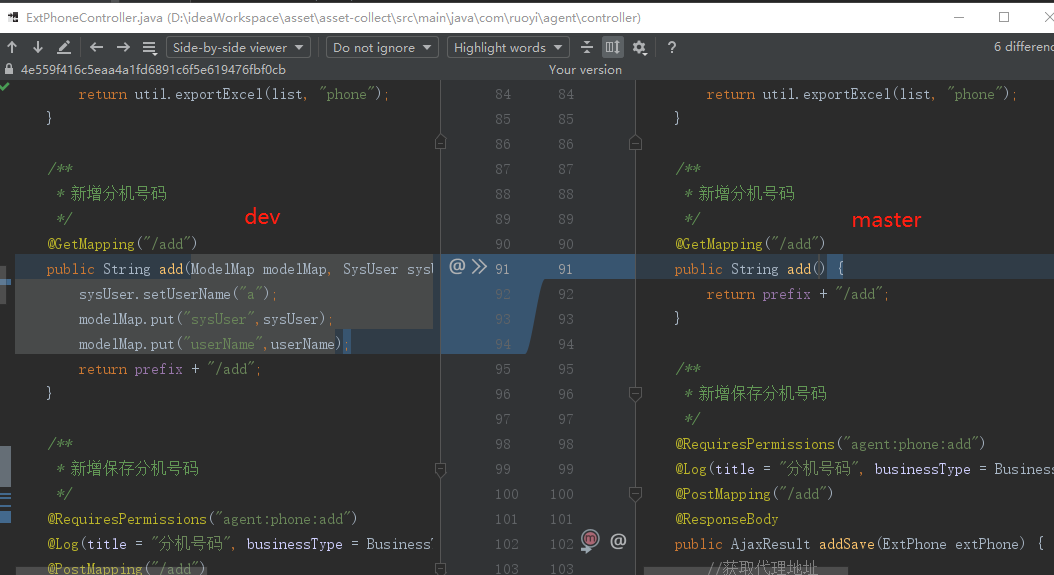
### 7.3.3 将develop代码合并到master

首先切换到master分支，拉取代码

右键项目选择git-> compare with branch..



选择dev分支，选择要合并的代码的类，左边的是dev分支，右边的是master分支，然后进行相应的合并



### 7.3.4 将develop某一次提交的代码合并到master

首先切换到master(git checkout master)，git cherry-pick commitid

合并多次提交的代码：git cherr-pick commitid commitid commitid

## 将某一个类所做的修改回退（不提交）

git checkout -- 类的全限定名

eg: git checkout -- dts-model/src/main/java/com/ksyun/kbdp/dts/model/flow/FlowSchedule.java

相当于ctrl + z

## add后撤回

git reset head 类的全限定名

git reset head asset-collect/src/main/java/com/ruoyi/caseConfig/controller/TLcAllocatCaseConfigController.java

此时再用git status查看已经是modified状态了，未add的

## 修改commit的注释

git commit --amend

此时会进入vim编辑器模式进行修改

## 撤销commit

git reset --mixed 版本号

git reset --mixed c01ea0ccbd97df933a6db0c0b27f93d958d52ec1

撤销commit的同时，也会撤销add操作，不删除修改的代码

git reset --hard 版本号

git reset --hard c01ea0ccbd97df933a6db0c0b27f93d958d52ec1

撤销commit的同时，也会撤销add操作，删除修改的代码

git reset --soft 版本号

git reset --soft c01ea0ccbd97df933a6db0c0b27f93d958d52ec1

撤销commit的同时，不会撤销add操作，不删除修改的代码

## 8.8 查看日志(版本号)

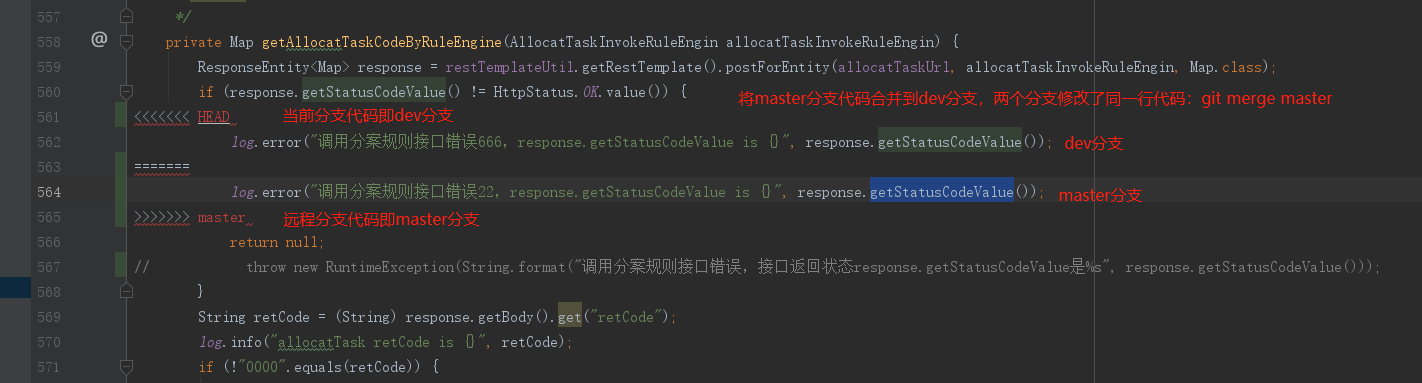
git log

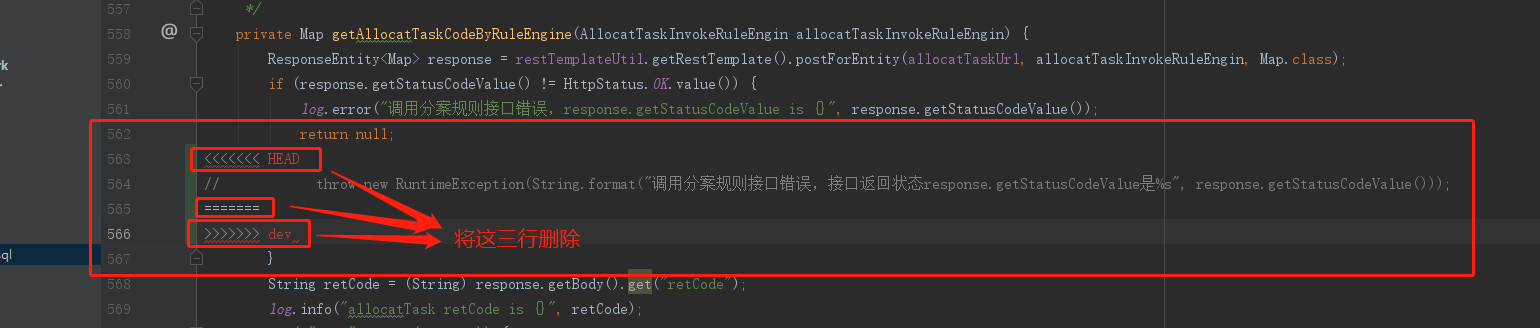
## 8.9 Git 解决冲突

git merge dev合并代码后冲突的解决办法

<<<<<<< HEAD和=======中间是自己的代码

=======和>>>>>>> master是从远程分支合并过来的别人的代码





## 8.10 git放弃merge，回到上一次commit

git reset --hard

## 8.11 git stash、git stash pop

在git stash pop前先git pull