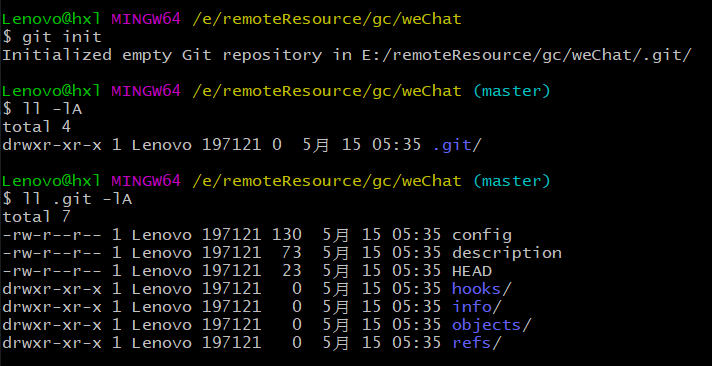
# 1．本地工作仓

初始命令：

git init



设置签名

1. 设置项目签名

git config user.name hxlNjust

git config user.email hxl\_njust@163.com

1. **设置系统签名**（目前以设定系统签名为主）

git config –global user.name hxlNjust

git config –global user.email [hxl\_njust@163.com](mailto:hxl_njust@163.com)

# 2．git基本命令操作

查看git的状态

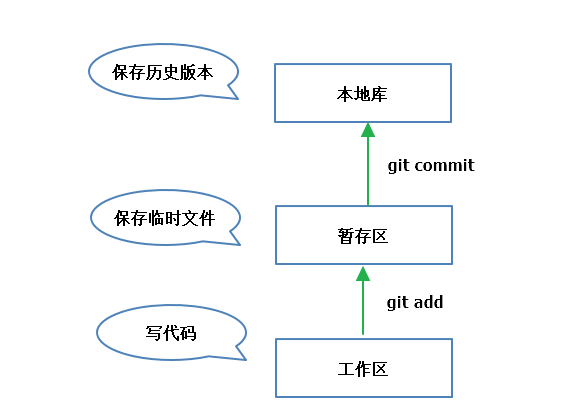
git status

添加当前目录的所有的文件到缓存区

git add ./\*

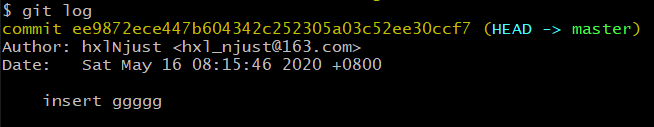
提交缓存的文件到本地库

git commit -m “xxxxxxx”

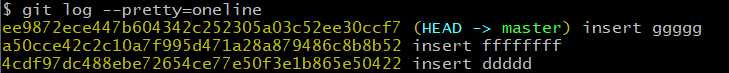


查看git的日志记录：

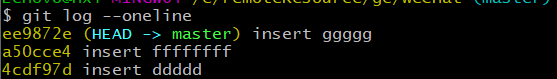
* git log 显示所有的日志信息



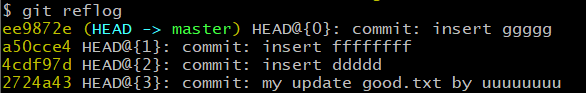
* git log –pretty=oneline 显示一行信息



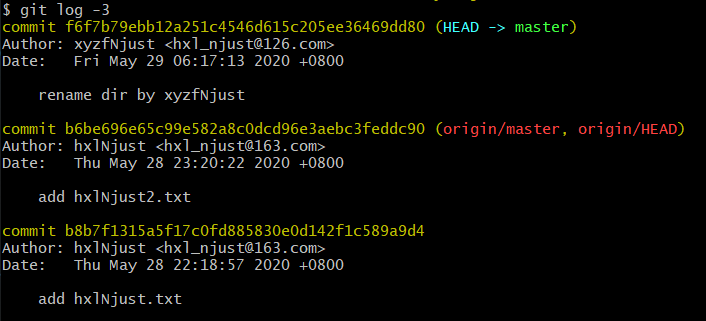
* git log –oneline 显示一行信息，但是比较简洁



* git reflog 显示的信息历史版本跳跃的信息



* git log -n 表示显示指定的n行



Git 历史版本的前进后退

* 基于索引值的移动

git reset –hard [索引编号]



* 基于 ^ 的使用，表示往后退，也只能往后退

git reset –hard HEAD^

这个地方尖括号可以有多个，一个表示后退一步，多个表示退多步



* 基于 ~ 的使用，表示后退，但是可以后退多步

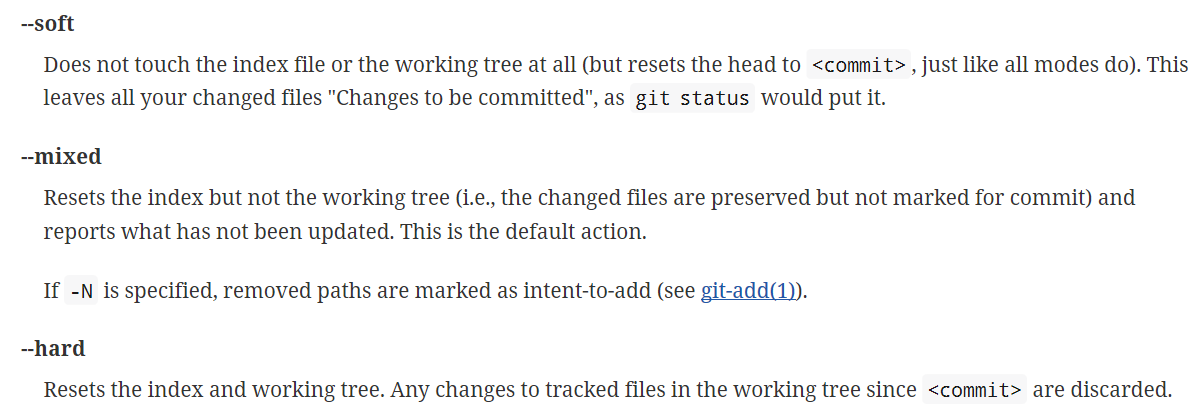
git reset –hard HEAD~n

n表示后退的步数



查看命令，命令会弹出一个本地的网页

git help reset



当删除操作已经被提交到本地库，可以根据历史版本，进行恢复

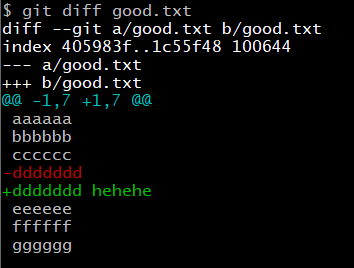
删除操作未被提交到本地库，可以直接强制命令

git reset –hard HEAD 强制恢复

比较文件

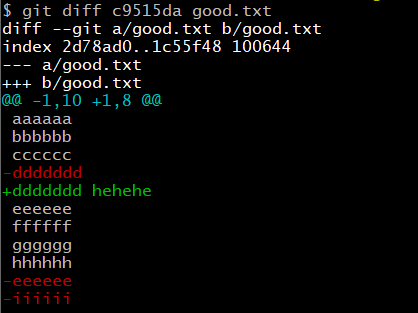
* git diff 文件名

比较工作区与暂存区的文件



* git diff 历史版本 文件名

比较工作区与本地库的文件

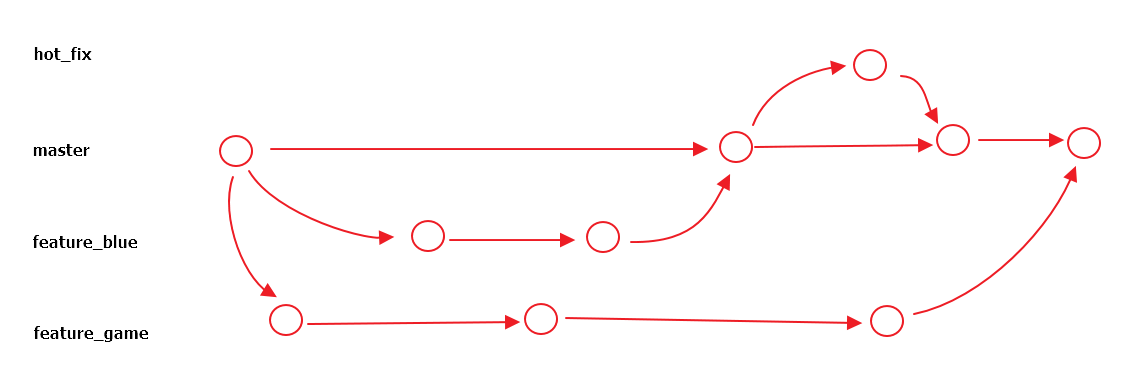


* git diff 历史版本

比较所有的工作区与本地库的文件

# Git分支命令

分支的流程



## 3.1分支操作

* 查看有多少个分支

git branch -v

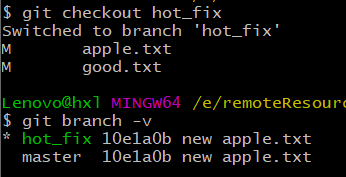


* 创建分支

git branch 分支名

* 切换分支

git checkout 分支名



* 合并分支

1：切换到分支hot\_fix，进行开发，提交，后切换到master主干

2：在主干上通过git merage hot\_fix 把hot\_fix的内容合并到主干来

冲突解决

1：git merge hot\_fix 发现冲突，无法合并，进行合并处理

2：然后重新提交，进行解决相关的问题，注意git commit -m “” （不要跟具体文件）

# Git远程

## 1.别名

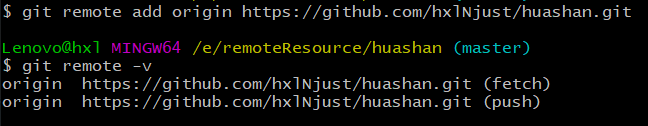
首先查看系统中是否存在别名，因为地址太长，一般会记不住

* 查看别名

git remote -v

* 添加别名

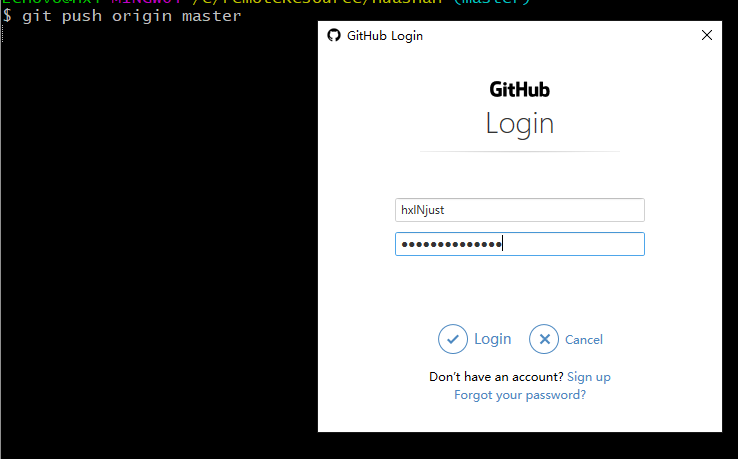
git remote add 别名 远程地址



## 2.远程推送

把master这个本地分支推送给远程

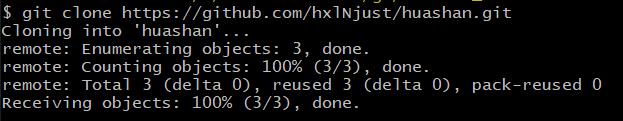
git push origin master



## 3.本地克隆远程库的信息

命令

* git clone 远程地址



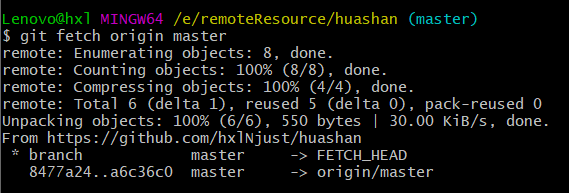
此命令效果：

* 拷贝本地库的信息
* 初始化本地库
* 拷贝远程库的别名，方便后期往本地库推送信息

## 4.git本地拉取文件

命令

* git fetch origin master



git fetch <远程主机名> //这个命令将某个远程主机的更新全部取回本地

如果只想取回特定分支的更新，可以指定分支名：

git fetch <远程主机名> <分支名> //注意之间有空格

最常见的命令如取回origin 主机的master 分支：

git fetch origin master

取回更新后，会返回一个FETCH\_HEAD ，指的是某个branch在服务器上的最新状态，我们可以在本地，通过它查看刚取回的更新信息

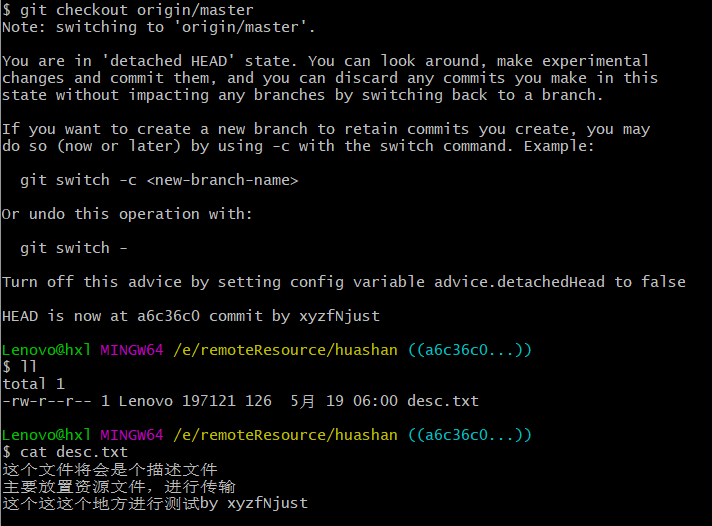
git log -p FETCH\_HEAD



可以看到返回的信息包括更新的文件名，更新的作者和时间，以及更新的代码（19行红色[删除]和绿色[新增]部分）。

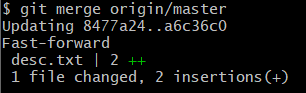
我们可以通过这些信息来判断是否产生冲突，以确定是否将更新merge到当前分支。

* git checkout origin/master



此命令查看刚才fetch的信息，然后在切换到master（git checkout master）

* git merge origin/master



此命令进行合并刚才fetch的信息到本地库

命令git pull=git fetch + git merge

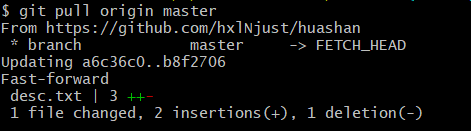
* git pull

即将远程主机的某个分支的更新取回，并与本地指定的分支合并，完整格式可表示为：

git pull <远程主机名> <远程分支名>:<本地分支名>

如果远程分支是与当前分支合并，则冒号后面的部分可以省略：

git pull origin next



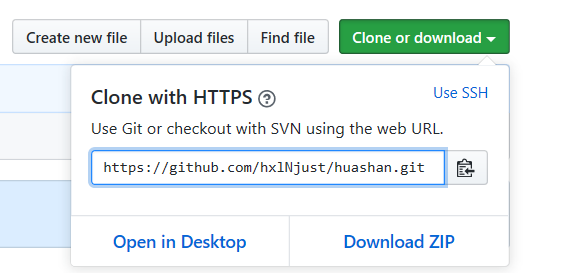
## 账号混乱处理



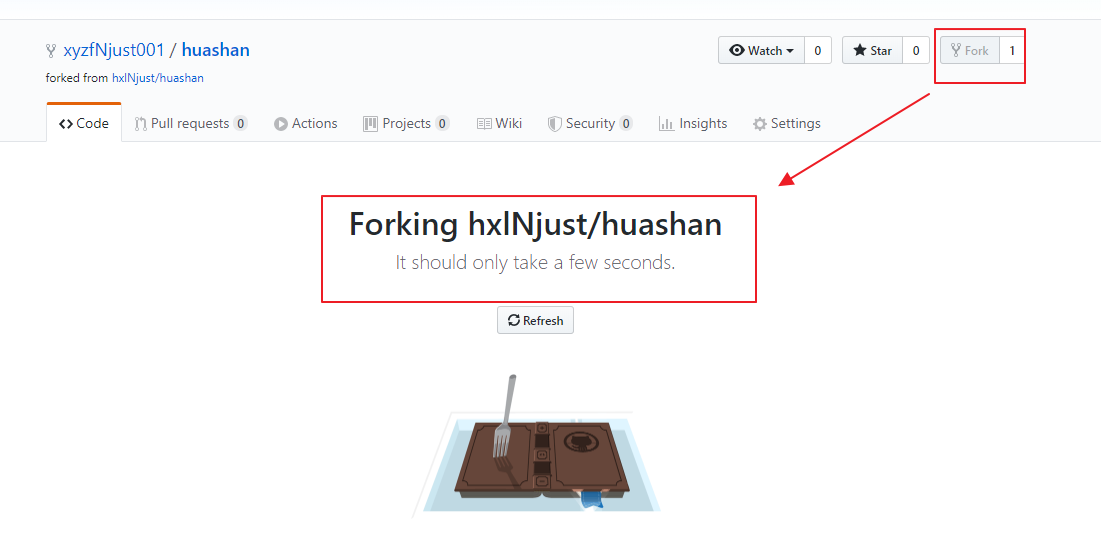
把普通凭据就删除掉即可

# 外部协同操作

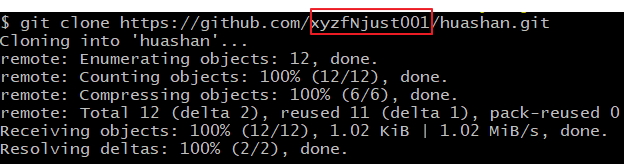
第一步：从客户hxlNjust那边获取git取出地址



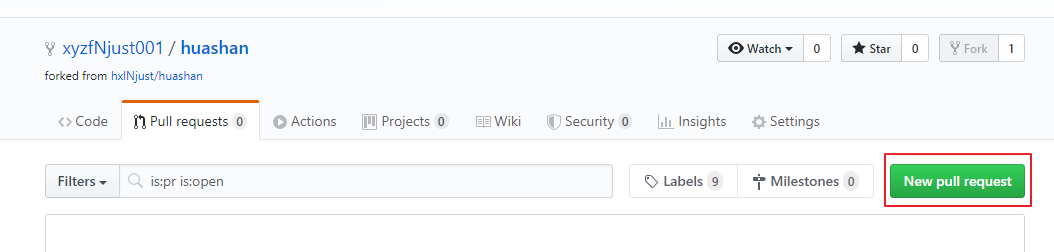
第二步：登陆协同账号xyzfNjust001，并在此账号的访问栏输入项目git地址，然后fork

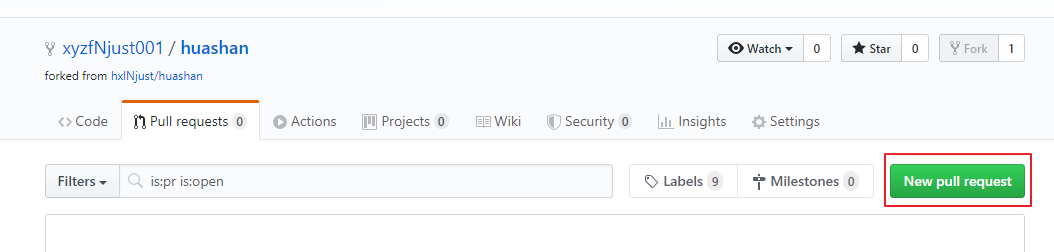


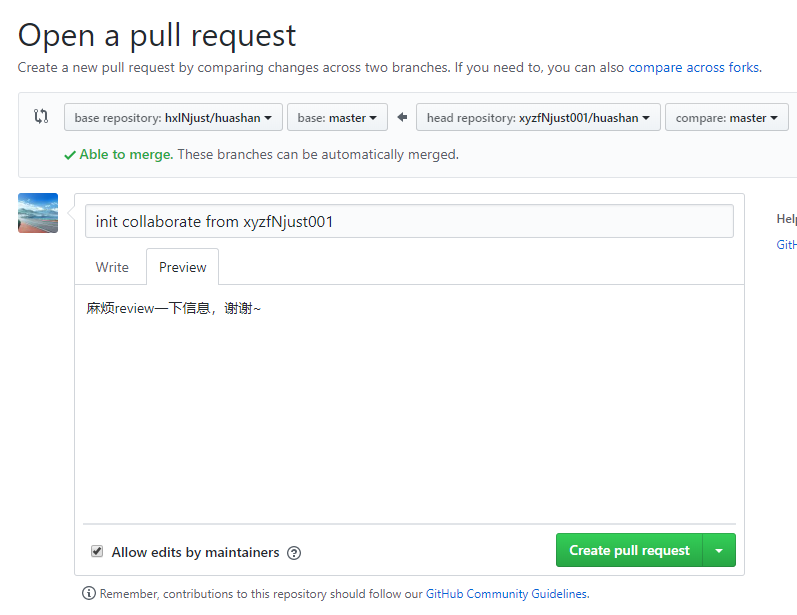
第三步：xyzfNjust001用户在自己的库克隆地址



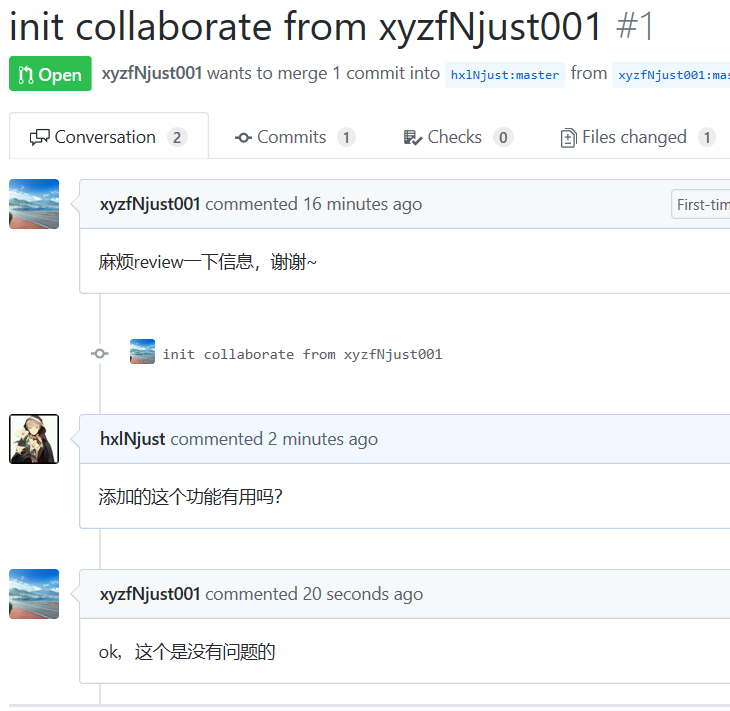
第四步：修改完毕后，进行pull request 操作，有一系列的点击



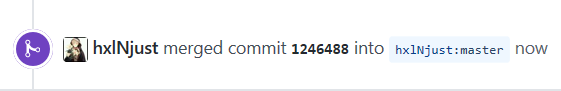




第五步：客户端（hxlNjust）就可以咨询相关的信息



第六步：然后客户hxlNjust复审没有问题后，就可以合并了



问题：git fork后如何保持与源项目的更新？

1：首先保证本地仓库的upstream是源项目的URL，若没有则添加：

git remote add upstream URL

2：然后利用fetch和merge合并upstream的master分支：

git fetch upstream

git merge upstream/master

3：此时本地的master分支就更新至upstream的master版本。然后利用push将本地分支覆盖到git远程分支上：

git push origin master:master:

核实后，发现不如意，想删除，可以用 git remote remove name

name 为远程分支的命名，如上面例子，可以删除 upstream

git remote remove upstream

或者直接删除之前fork的原始分支 origin

git remote remove origin (直接删除了原始分支后，再fork，也能达到更新了最新代码的需求，后面的步骤就不用了。)

# Ssh免密登陆

* 进入当前加目录

cd ~

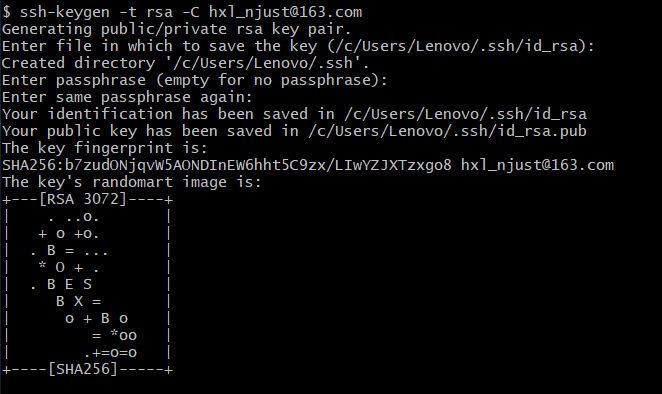
* 删除 .ssh 目录

rm -vrf .ssh

* 运行命令生成ssh秘钥

ssh-keygen -t rsa -C [hxl\_njust@163.com](mailto:hxl_njust@163.com)

【注意这个地方的-C 注意是大写】



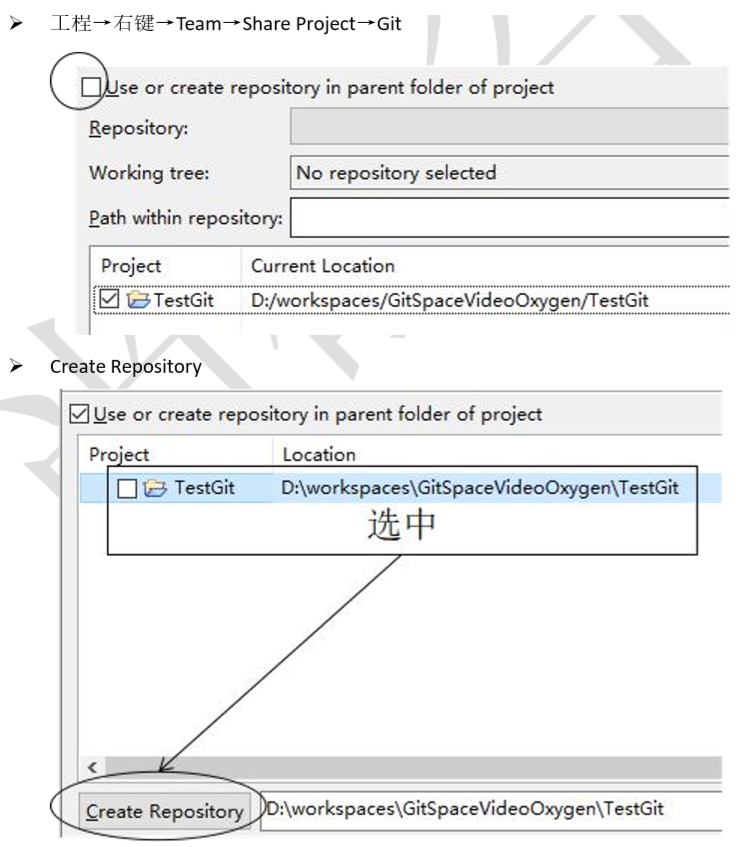
* 查看.ssh文件中的id\_rsa.pub文件
* 把秘钥中的信息添加到git账号中去

头像---settings---SSH and GPG keys---New SSH Key进行添加id\_rsa.pub的信息

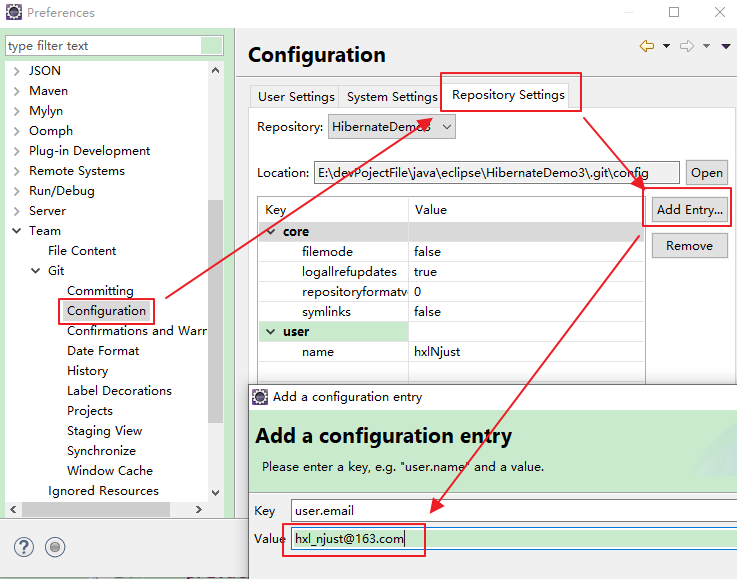
* 建立远程的地址

# Eclipse 操作

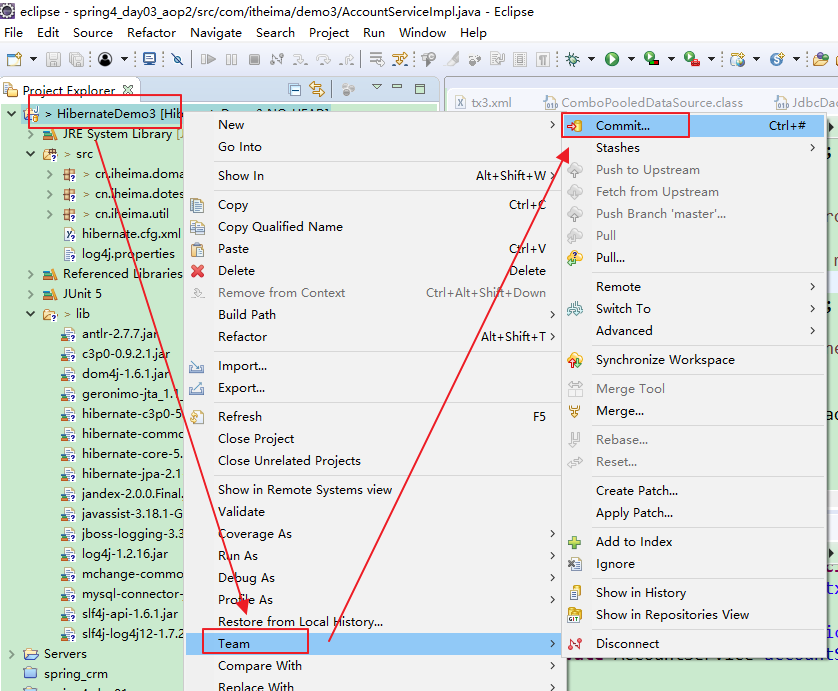
## 7.1工程初始化为本地库 



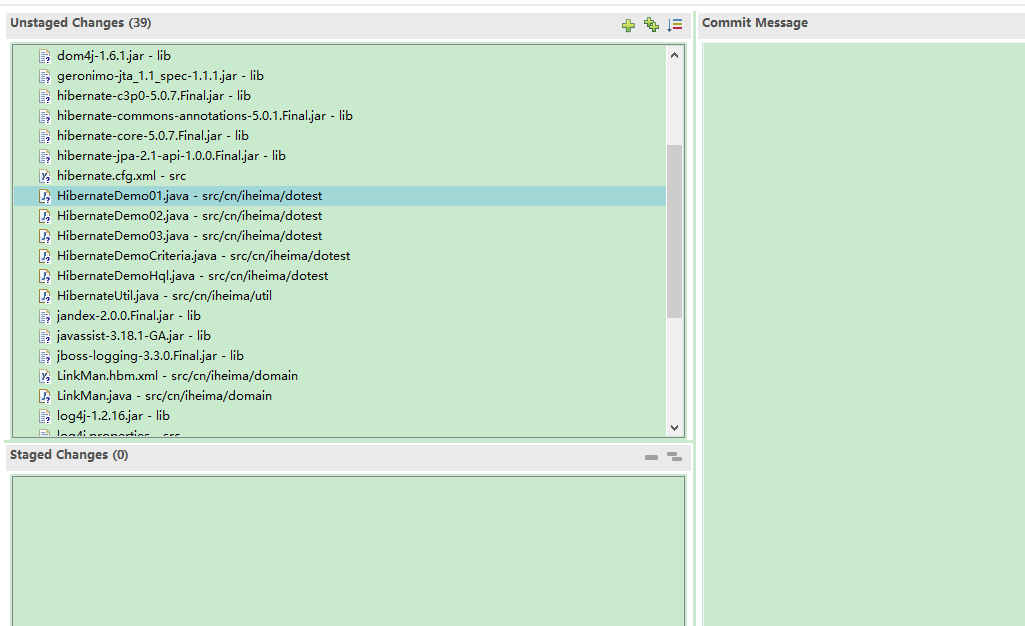
## 7.2给本地工程设定用户，邮箱进行设定



## 7.3进行提交的准备操作



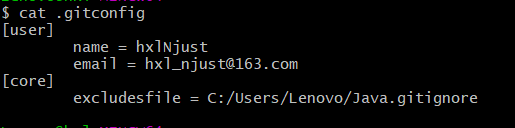
选择之后，显示三块区域。一个为为暂存，一个为暂存，一个为commit信息，如果想要在页面上看到，则搜索navigator



由于团队中成员使用的idea不一致，因此，需要忽略其中的非代码性的文件。

## 7.4忽略非业务代码配置

$ git config --global core.excludesfile C:/Users/Lenovo/Java.gitignore



# Compiled class file

\*.class

# Log file

\*.log

# BlueJ files

\*.ctxt

# Mobile Tools for Java (J2ME)

.mtj.tmp/

# Package Files #

\*.jar

\*.war

\*.nar

\*.ear

\*.zip

\*.tar.gz

\*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml

hs\_err\_pid\*

.classpath

.project

.settings

Target

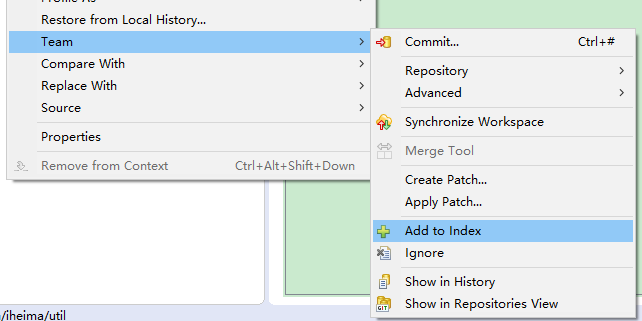
上面的配置则是忽略的文件，这个文件可以从git官网中找到

GitHub 官网样例文件

<https://github.com/github/gitignore>

<https://github.com/github/gitignore/blob/master/Java.gitignore>

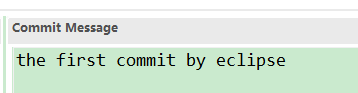
## 7.5本地库的操作—添加到暂存区（index）



这个操作直接把文件放置到了暂存区，里面的命令如下：

git add ./\*

## 7.6添加到本地库



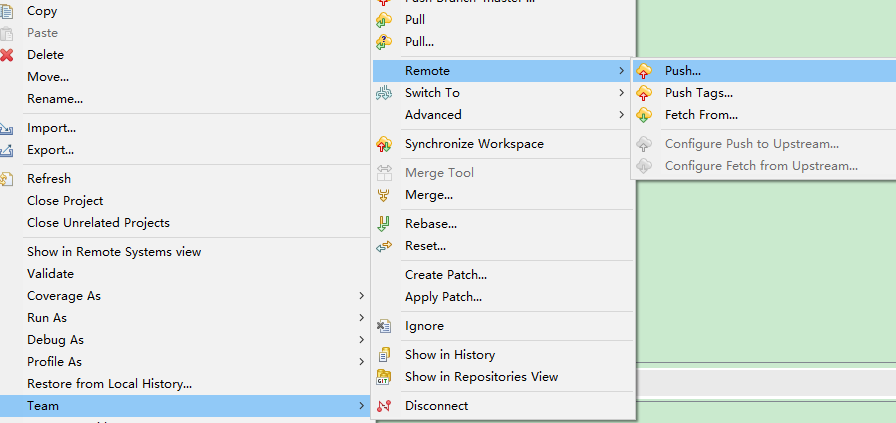
在commit message这个框中进行输入操作，然后点击下方的提交操作，等价于：

git commit -m “the first commit by eclipse

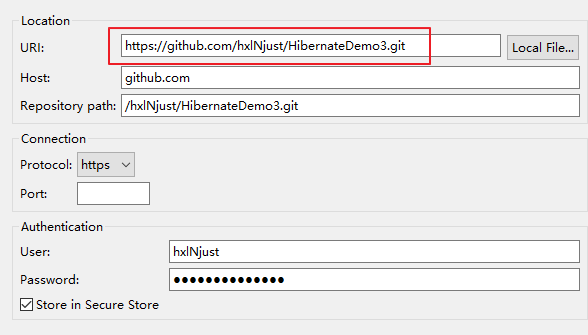
## 7.7推送到远程

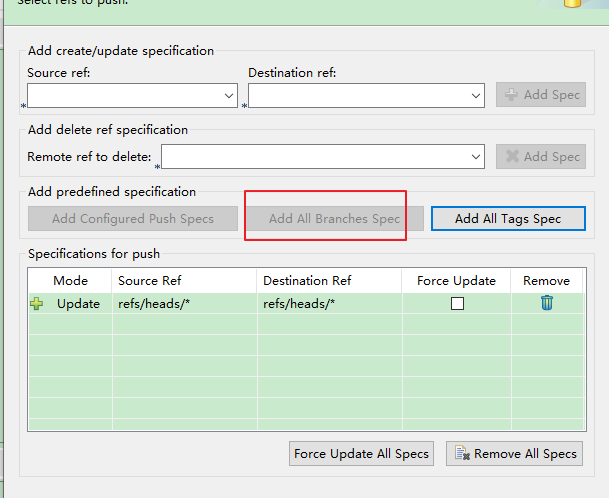
第一步在远程建立同名的仓库

第二步进行本地推送的操作：



在下面的框中粘贴git的远程地址





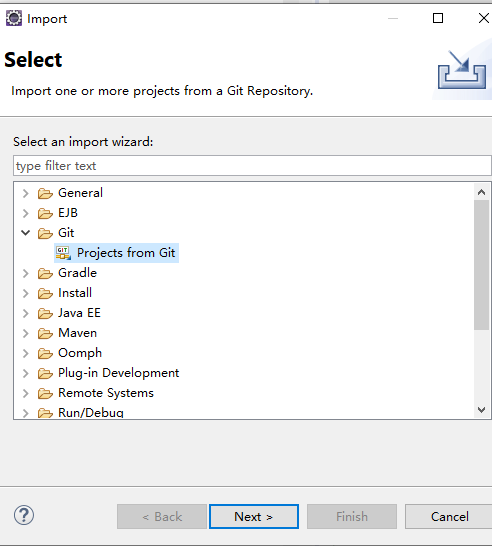


这种方式目前发现一个弊端，在git目录中看不到远程的信息

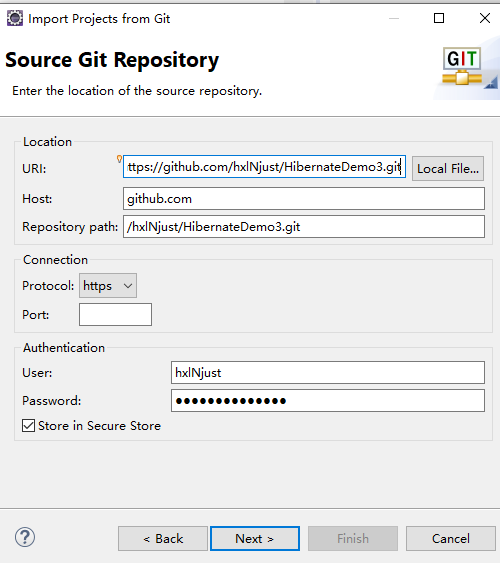
git remote -v (查询出来的信息是空的)

## 7.8 远程克隆

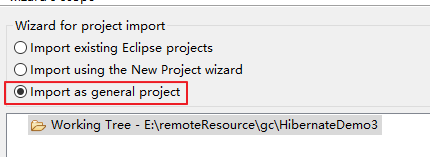
* 第一步：导入工程



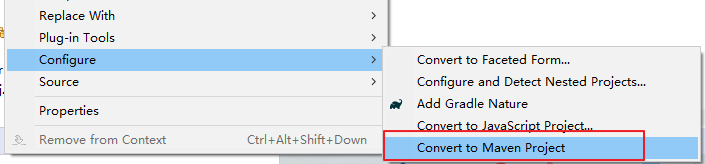
* 第二步：拷贝uri参数



* 第三步：设定wizard



* 第四步：转换为maven文件就可以了



后续分支请参考pdf文件