**本地库和GitHub中的库不同步**

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

git pull --rebase origin master合并 GitHub 和 本地 的库，本地会多出之前不同步的文件

# 多次commit一次push导致ahead

git reset --mixed （之前的commit id）

把commit版本回退到与远程库同一commit版本,这个指令只清空Staging Area(Index)和Local Repository，但会保留Working Directory

所以工作区并不会被改变

# 忽略已提交到远端仓库的文件

# 第一步：为避免冲突需先同步远程仓库

git pull

# 第二步：编辑.gitignore文件，添加忽略条件配置

vim .gitignore

# 第三步：在本地项目目录下删除缓存

git rm -r --cached .

# 第四步：再次add所有文件，添加到本地仓库缓存中

git add .

# 第五步：commit，push

git commit -m "filter new files"

git push

# 文件管理-添加文件

git status 查看工作区的状态

git add . 文件1 文件2 把文件提交到暂存区

git add \* 会把所有文件提交(无视.gitignore文件的匹配) 不要用!!!

补救方法: git rm -r --cached .  清除缓存,之后. gitignore文件就会生效了

git commit -m ‘批注信息’ 把当前分支的暂存区所有数据提交到当前分支

修改1->git add->修改2->git commit 提交的是修改1

每次修改必须先add再commit

git diff 文件名 查看当前文件和当前分支的文件内容的区别 如果没有区别就不显示任何东西

# 04【掌握】文件管理-撤销及版本回退

## git checkout 文件名 文件在工作区或暂存区[撤销]

git log 文件名 查看版本号

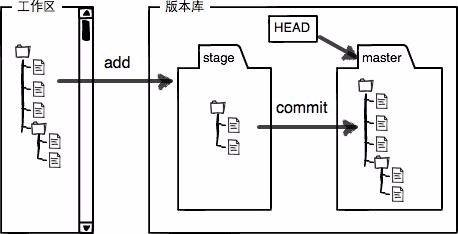
Svn集中式,版本号为简单的阿拉伯数字.所有人提交到同一个中央服务器,版本号由这个中央服务器统一管理.所以版本号可以为简单的阿拉伯数字

git是分布式.版本号为字符串.代码提交到本地版本库.在需要与其他人进行版本合并时,如果用数字版本号,就会造成冲突

## Git reset --hard 版本号 文件到分支里面[回退]

Git reset --hard HEAD^ 回退上一个版本

# 05【掌握】工作区和暂存区

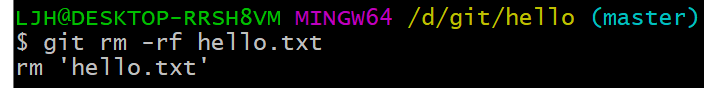


如果在仓库添加一个文件，默认在工作区

如果使用add 之到暂存区

如果使用commit 之后 是提交到当前分支

# 06【掌握】文件管理-删除文件



先使用git rm -rf 文件名

再提交一次

# 08【掌握】分支管理

git branch 查看分支

git branch dev创建dev分支

git branch -d dev 删除分支(先离开这个分支)

git checkout dev 切换分支

git checkout -b dev创建并切换

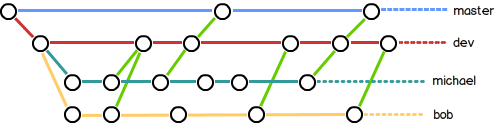
git merge 分支名 合并(先回到要合并的分支)

冲突问题？(master和dev都推进了一个版本,导致版本冲突,只有一方推进不冲突)

先合并 表示合并中

再手动解决文件中的冲突 再add commit

# 09【熟悉】分支管理=实际开发中的做法



# 10【了解】分支管理-Bug分支

当出现bug时,假设还在编辑一个文件.此时需要放下手头的工作去改bug

但是正在编辑的文件还不能进行提交,在未提交的状态下切换分支,其他分支上也会有这个文件

此时需要存储当前工作环境,等解决bug之后再返回工作环境

git stash 存储当前工作环境

git stash list 存储的所有工作环境

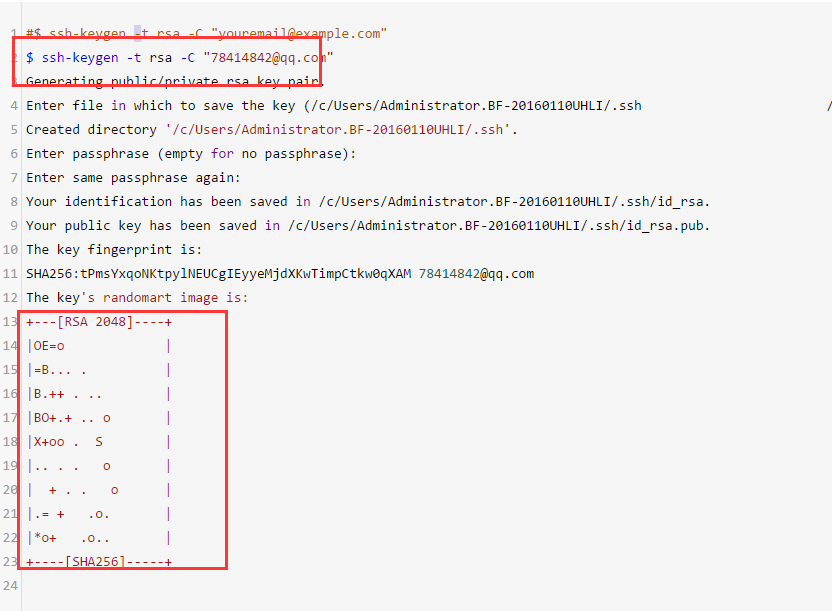
git stash pop 恢复工作环境

# 11【掌握】使用GitHub远程仓库

### 第1步：创建SSH Key。

        在用户主目录下，看看有没有.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。如果没有，打开Shell（Windows下打开Git Bash），创建SSH Key：

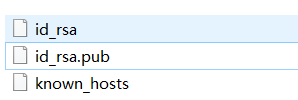
ssh-keygen -t rsa -C “1169732264@qq.com”



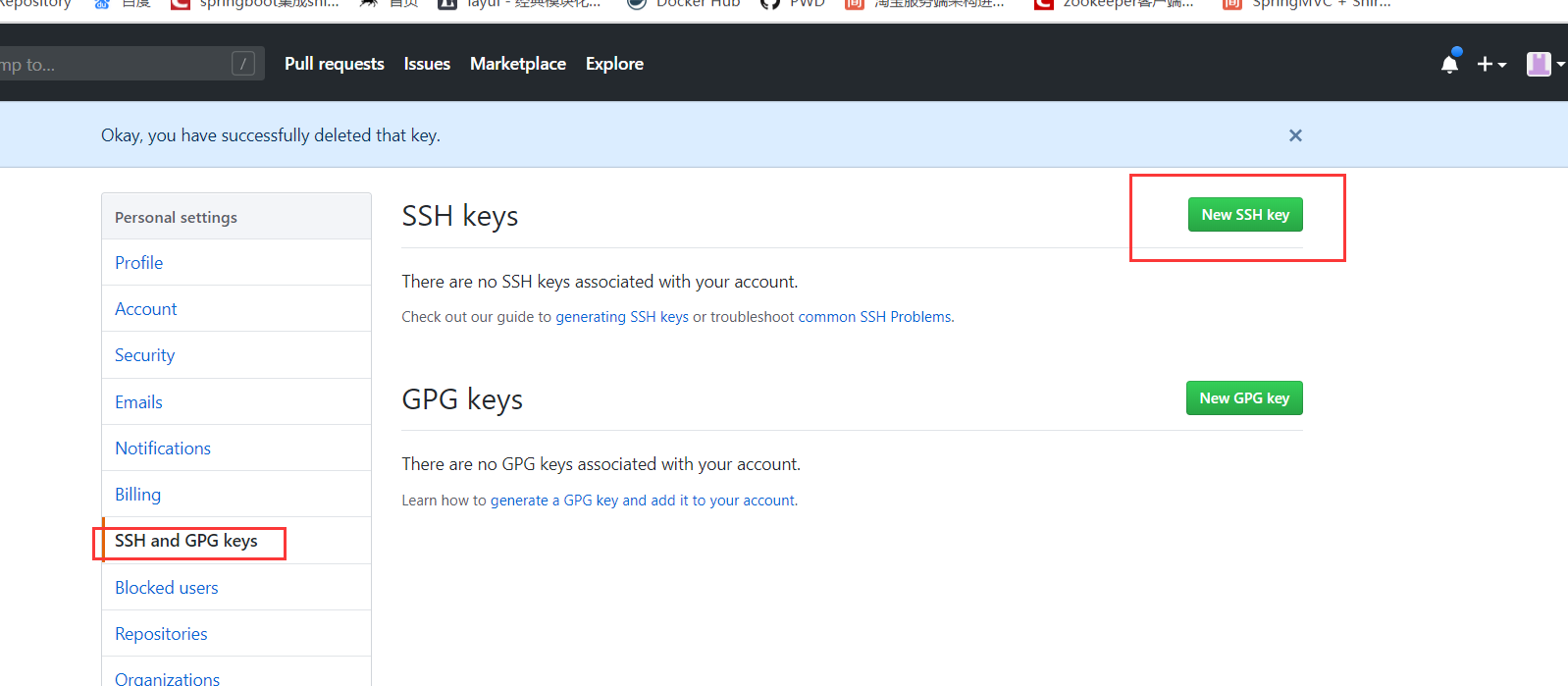
可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件

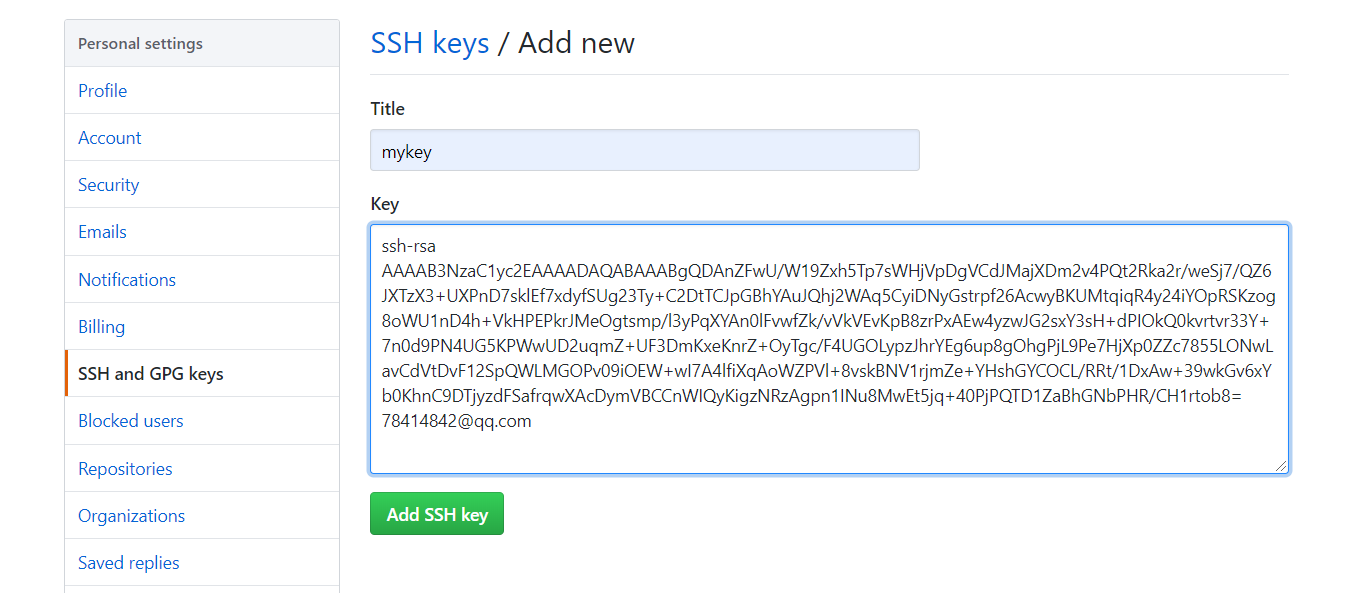
id\_rsa是私钥，不能泄露

id\_rsa.pub是公钥，可以放心地告诉任何人



### 第2步：登陆GitHub，打开“Account🡪settings”，“SSH Keys”页面：





如果想要别人修改你的代码,需要配置上别人的ssh key公钥

# 设置代理：

git config --global http.proxy 'socks5://127.0.0.1:1080'

git config --global https.proxy 'socks5://127.0.0.1:1080'

查看代理：

git config --global --get http.proxy

git config --global --get https.proxy

取消代理：

git config --global --unset http.proxy

git config --global --unset https.proxy

## 2，添加远程仓库

### git remote add origin 链接 关联远程仓库

### git push –u origin master 推送

## 3，从自己远程仓库克隆代码

git clone 链接 clone会下载所有分支,不过本地只会有master,其余分支需要手动关联

如何把其它分支关联起来

git checkout -b dev origin/dev 创建分支并关联远程仓库的分支

## 4，从其它远程仓库克隆代码

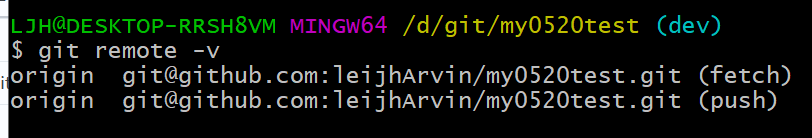
只在推送才需要ssh 克隆不用

git clone 链接

# 13【掌握】分支管理-多人协作

## git remote 查看远程库的信息

## git remote –v 详细信息



Fetch 拉取文件的权限

Push 推送文件的权限

也就是说,从别人git仓库里clone下来的代码,是没有push权限的

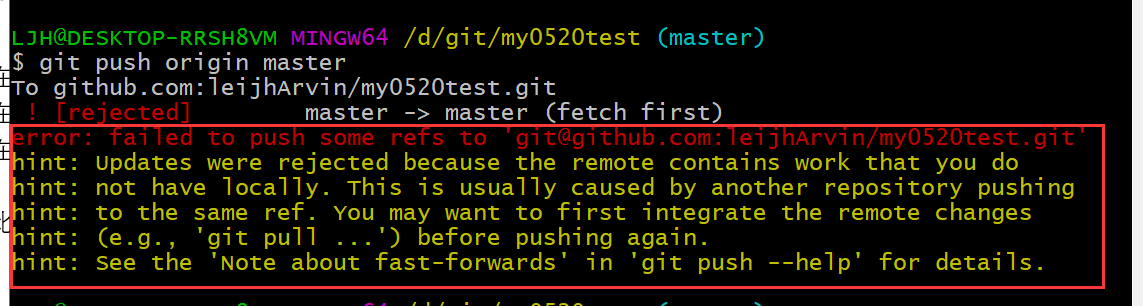
## 推送分支

git push origin master

git push origin dev

## 冲突问题

在clone远程库到本地后,远程库和本地都进行了改动->版本冲突-.>无法推送



## git pull origin master 拉取分支

## 之后手动修改冲突,再add,commit

# 14【掌握】使用码云

## 1，概述

    使用GitHub时，国内的用户经常遇到的问题是访问速度太慢，有时候还会出现无法连接的情况（原因你懂的）。 如果我们希望体验Git飞一般的速度，可以使用国内的Git托管服务——码云（<https://gitee.com>）。

        和GitHub相比，码云也提供免费的Git仓库。此外，还集成了代码质量检测、项目演示等功能。对于团队协作开发，码云还提供了项目管理、代码托管、文档管理的服务，5人以下小团队免费。

**码云的免费版本也提供私有库功能，只是有5人的成员上限。**

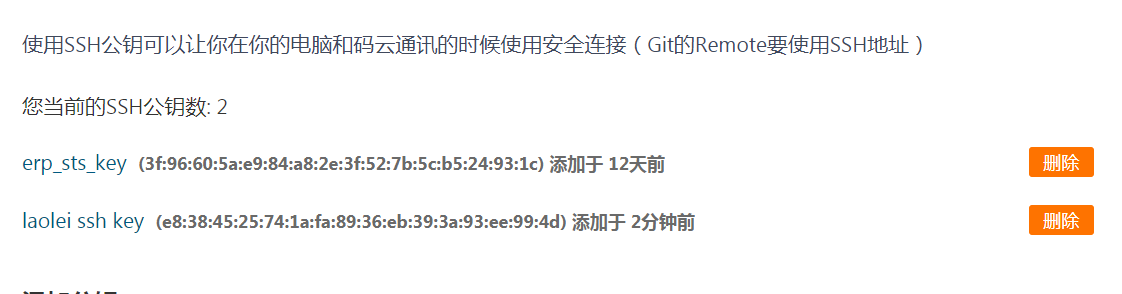
## 2，设置公匙

使用码云和使用GitHub类似，我们在码云上注册账号并登录后，需要先上传自己的SSH公钥。选择右上角用户头像 -> 菜单“修改资料”，然后选择“SSH公钥”，填写一个便于识别的标题，然后把用户主目录下的.ssh/id\_rsa.pub文件的内容粘贴进去：



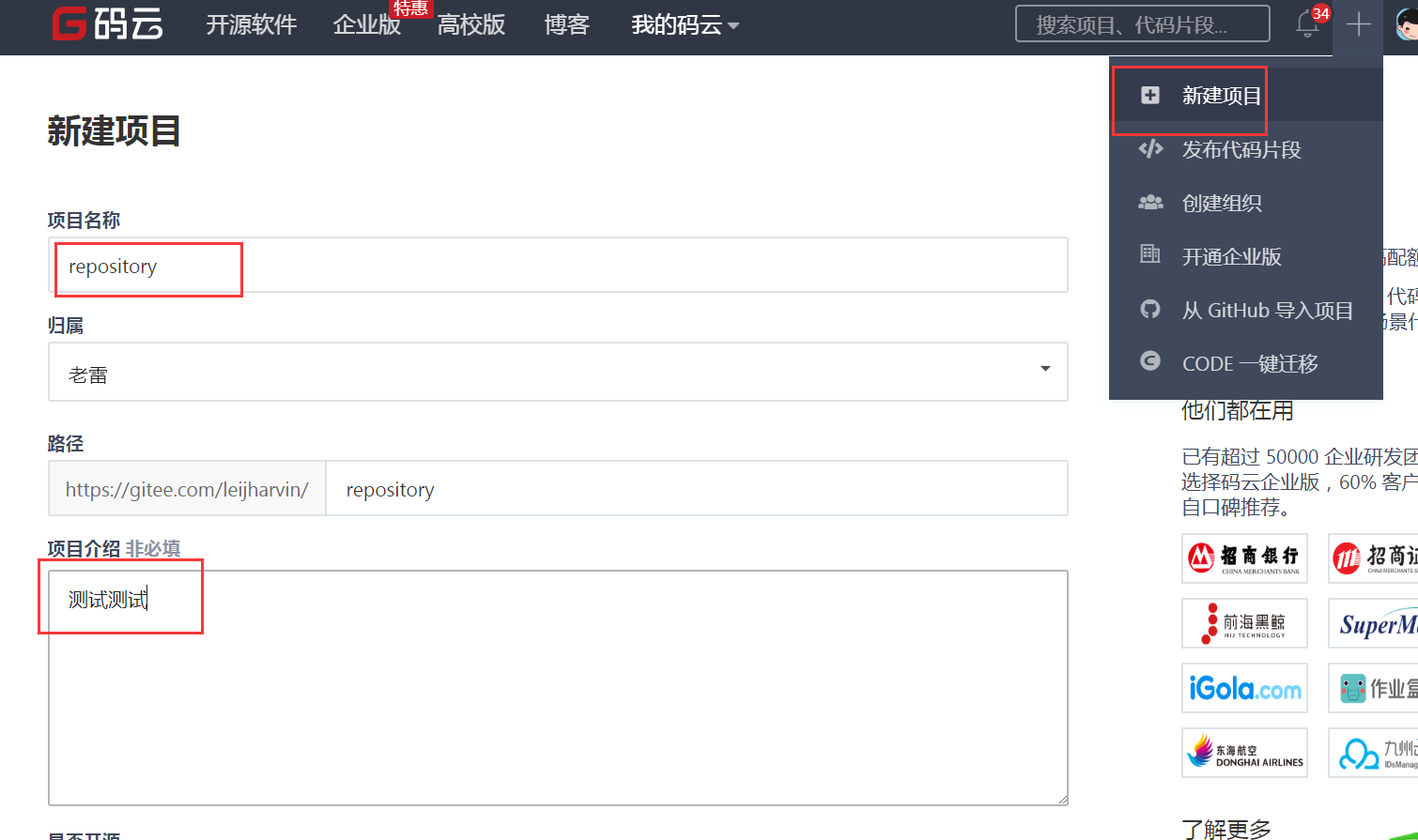
点击“确定”即可完成并看到刚才添加的Key：





## 3，创建项目

在码云上创建一个新的项目，选择右上角用户头像 -> 菜单“控制面板”，然后点击“创建项目”：



项目名称最好与本地库保持一致：

然后，我们在本地库上使用命令git remote add把它和码云的远程库关联：

git remote add origin https://gitee.com/leijharvin/repository.git

## 4，拉取远程代码库

由于在创建远程仓库时会初始化一个README.md文件，而本地仓库里没有，所以需要先执行pull操作将远程仓库拉取合并到本地仓库，否则会出错。执行代码：

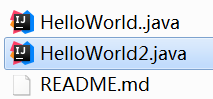
git pull origin master

此时可以看到在本地代码仓库中多了一个README.md文件。



## 5，创建本地文件并提交

5.1创建两个文件



5.2添加并提交 到本地仓库

Arvin@Arvin-pc MINGW64 /d/git/repository (master)

$ git add .

​

Arvin@Arvin-pc MINGW64 /d/git/repository (master)

$ git commit -m '添加两个文件'

[master 4b9ab65] 添加两个文件

2 files changed, 12 insertions(+)

create mode 100644 HelloWorld..java

create mode 100644 HelloWorld2.java

​

5.3  推送到gitee

$ git push --set-upstream origin master

提交完成之后看到



## 6，删除文件

#1，使用git命令删除本地的文件

$ git rm HelloWorld2.java

rm 'HelloWorld2.java'

#2，再提交到本地的版本库

Arvin@Arvin-pc MINGW64 /d/git/repository (master)

$ git commit -m "删除了一个文件"

[master d43e6c2] 删除了一个文件

1 file changed, 6 deletions(-)

delete mode 100644 HelloWorld2.java

#3，再推送到gitee

$ git push --set-upstream origin master

Counting objects: 2, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (2/2), 245 bytes | 245.00 KiB/s, done.

Total 2 (delta 1), reused 0 (delta 0)

remote: Powered by Gitee.com

To https://gitee.com/leijharvin/repository.git

4b9ab65..d43e6c2 master -> master

Branch 'master' set up to track remote branch 'master' from 'origin'.

​

​

再看gitee上面就只有一个文件了



# 15【掌握】eclipse里面使用gitee

之前使用bash连接github时做了哪些事

1. 生成ras.pub 公钥
2. 配置sshkey
3. 创建仓库
4. 关联远程库
5. 向仓库推送数据
6. 拉取数据

## 一，把项目分享到gitee

**1、准备工作**

1、一个注册好的码云帐号 https://gitee.com/

2、安装好的Eclipse （本示例用的版本 sts ，已经集成Git）

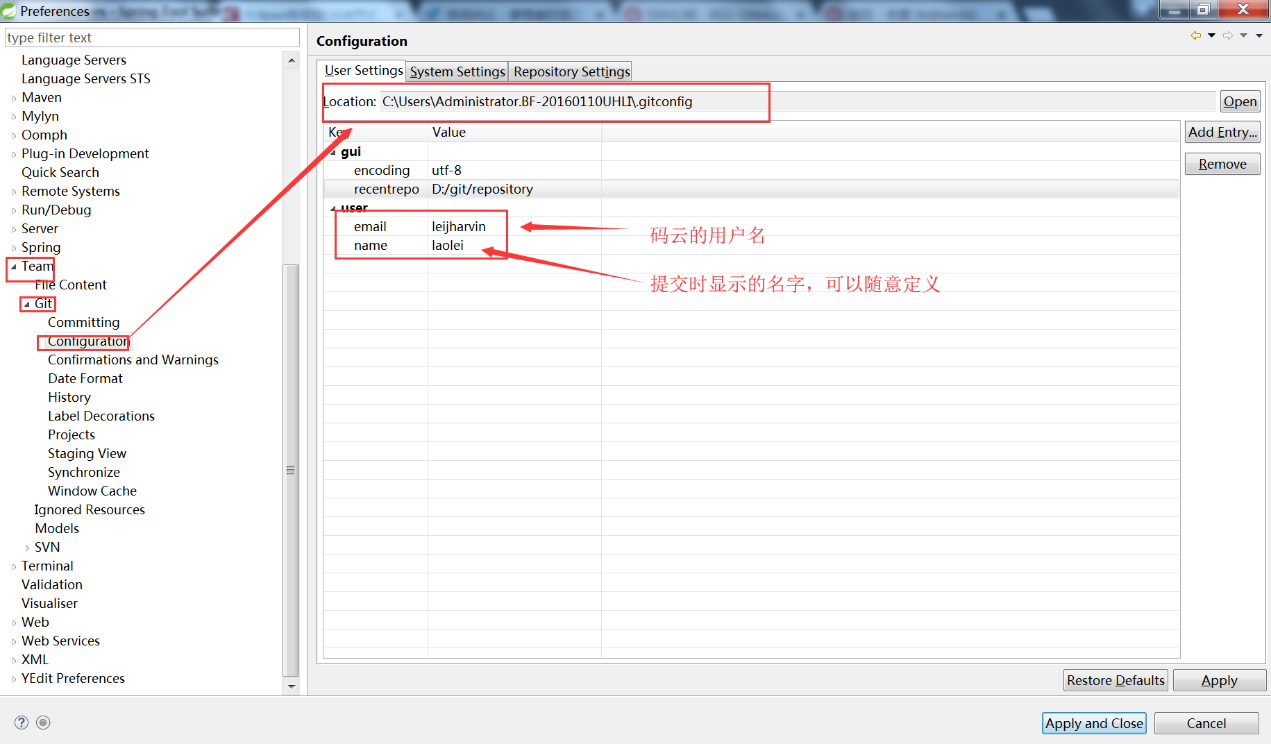
**2、Eclipse 配置Git 相关帐号资料**

1、路径： Window --- Preferences --- Team--- Git --- Configuration --- User Settings --- Add Entry --- 如下图：

a. user.email ： “ 您的码云帐号”

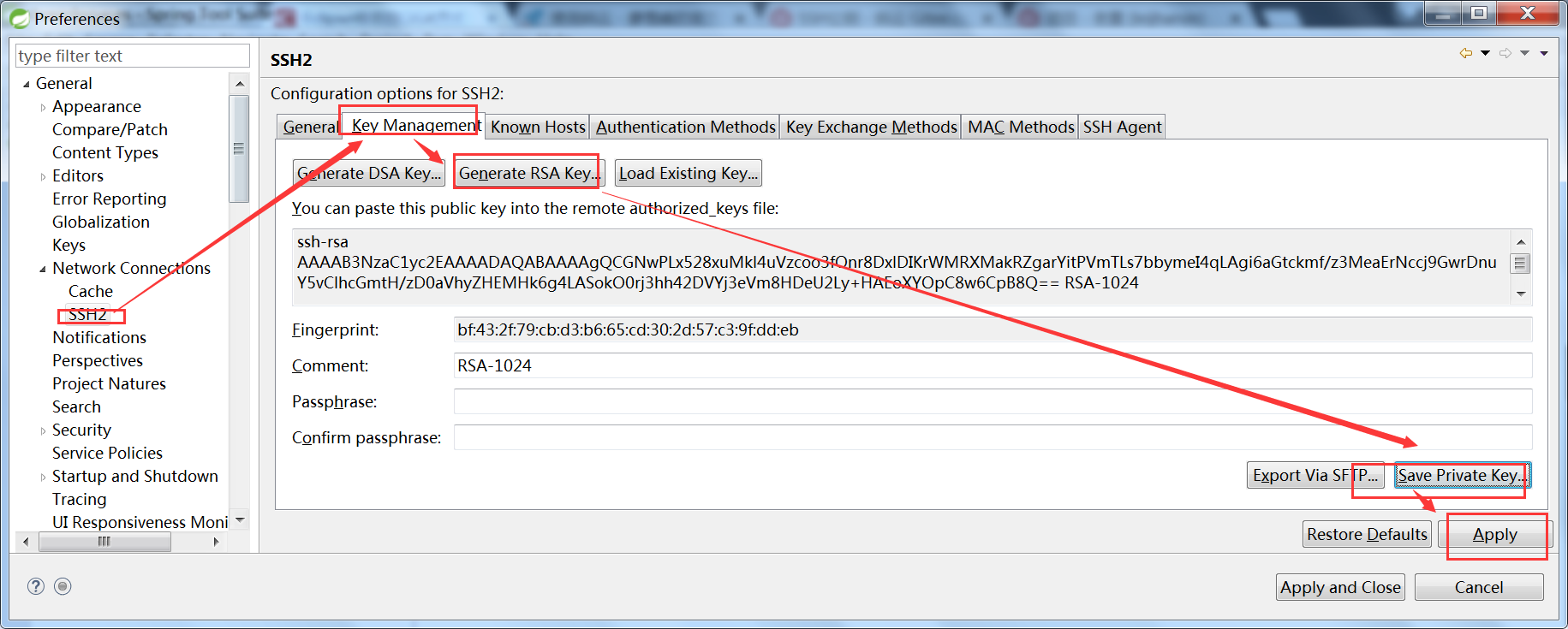
b. user.name： "提交时显示的名字，自定义。"

c. 等价于git命令： git config --global user.name/user.email "xxxx"

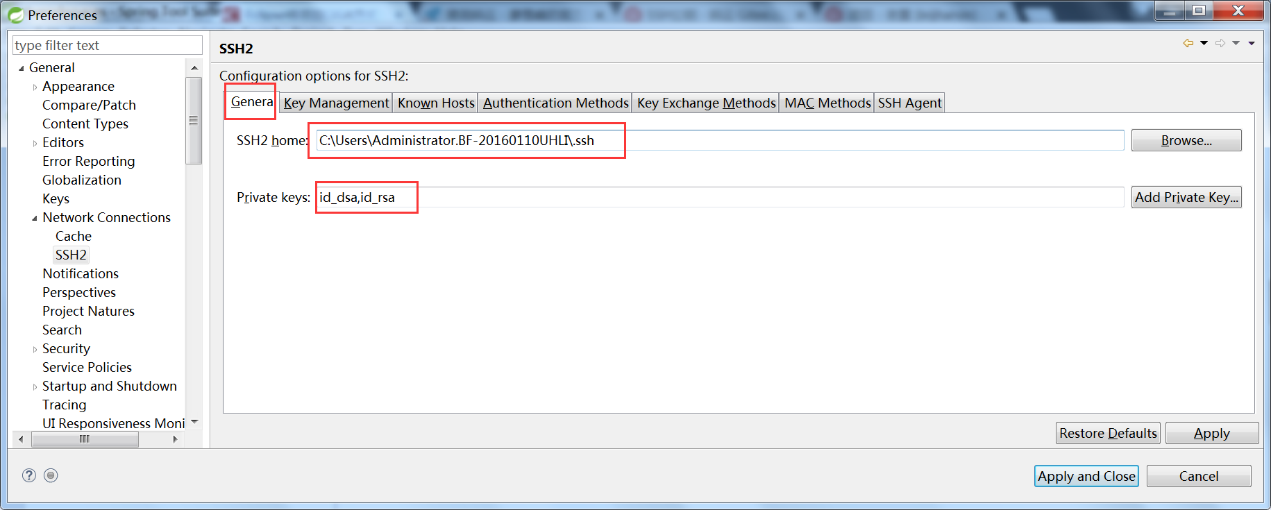


**3、Eclipse 配置 publickKey**

1、生成 key 路径： Window --- Preferences --- General --- Network Connections --- SSH2 --- Key Management --- Generate RSA Key ---



        2、获取 key 路径： Window --- Preferences --- General --- Network Connections --- SSH2 --- General



**4、在码云中配置 用户 ssh key**

1、步骤： 登录码云 --- 进入 个人设置中心 --- SSH公钥 （即： 用户 ssh key）



    2、补充：为什么要配置用户 ssh key 而非 项目 ssh key？

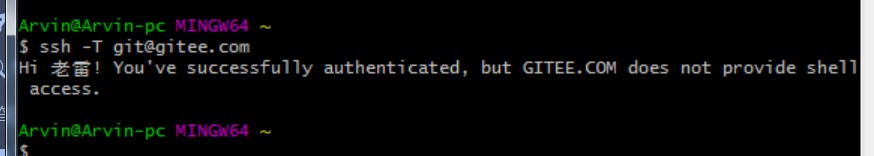
项目的 sshkey 只针对项目，且我们仅对项目提供了部署公钥，即项目下的公钥仅能拉取项目，这通常用于生产服务器拉取仓库的代码。

用户的 key 则是针对用户的，用户添加了 key 就对用户名下的项目和用户参加了的项目具有权限，一般而言，用户的key具有推送和拉取的权限，而项目的 key 则只具有拉取权限。

3、补充：验证添加的 key 是否生效。

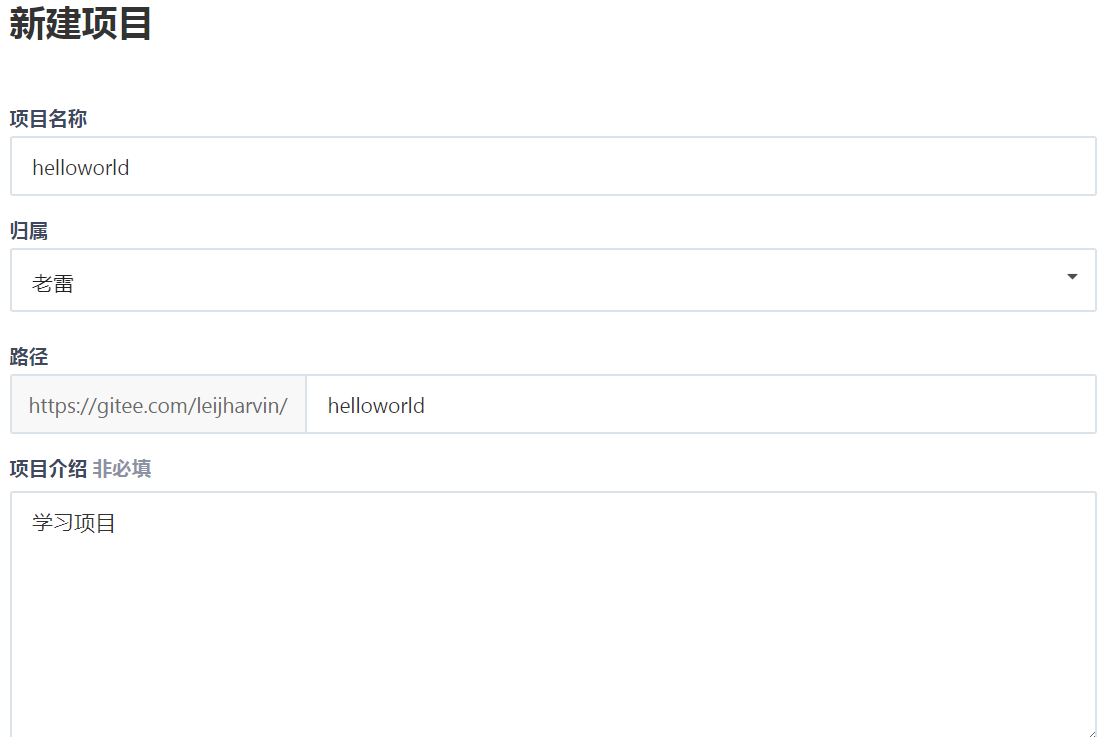
在 Git Bash 中输入命令： ssh -T git@gitee.com

返回： Welcome to Gitee.com, xxx ! ---- 则表示添加key成功！如下

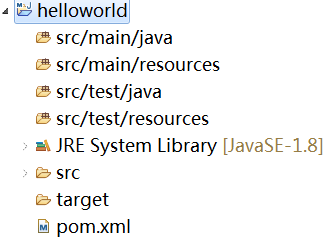


**5、Eclipse项目分享到码云**

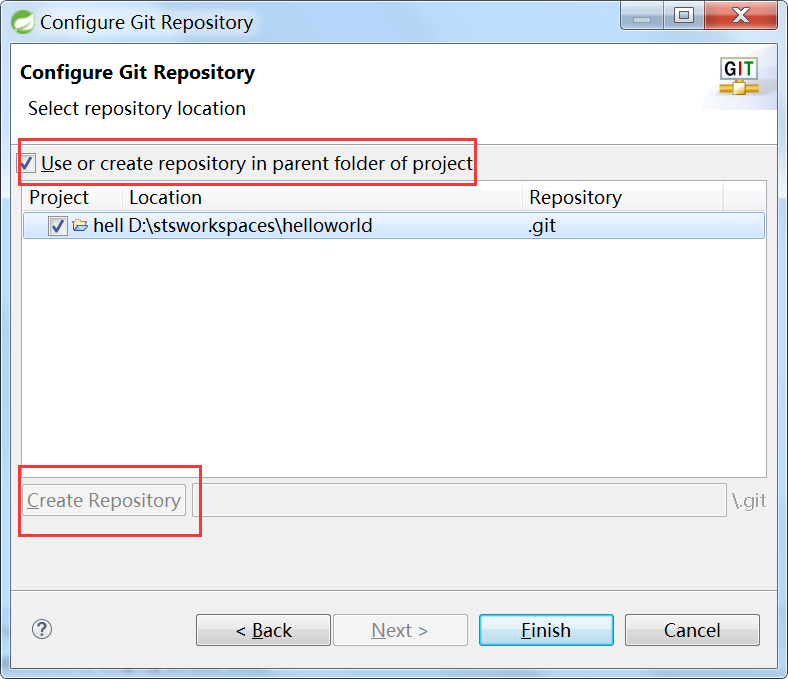
1、码云上创建一个项目，名字：helloworld 注意不要创建README.md这个文件



2、Eclipse上创建一个项目，名字：helloworld

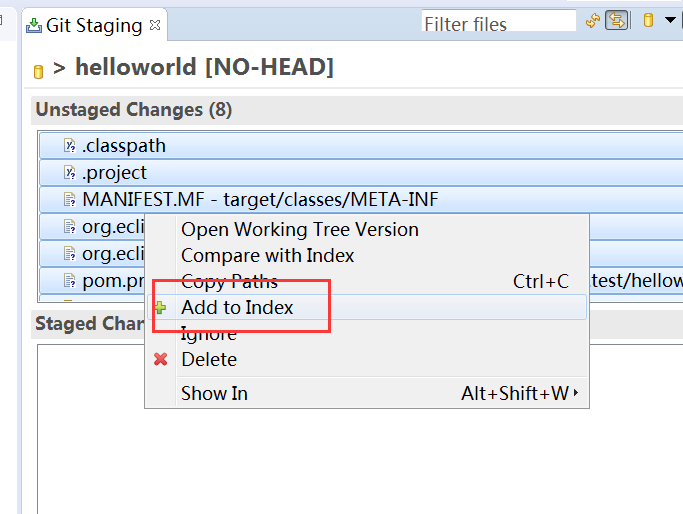


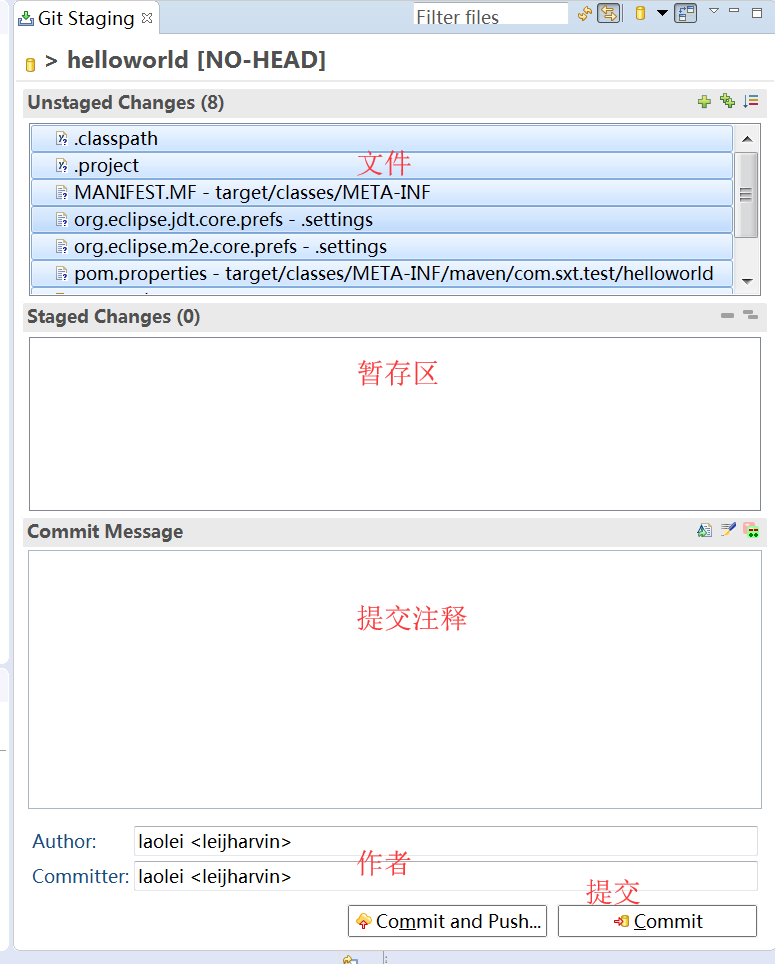
3、Eclipse项目分享到码云： 选中项目gitee --- 右键 --- Team --- Share Project --- Git --- 如下

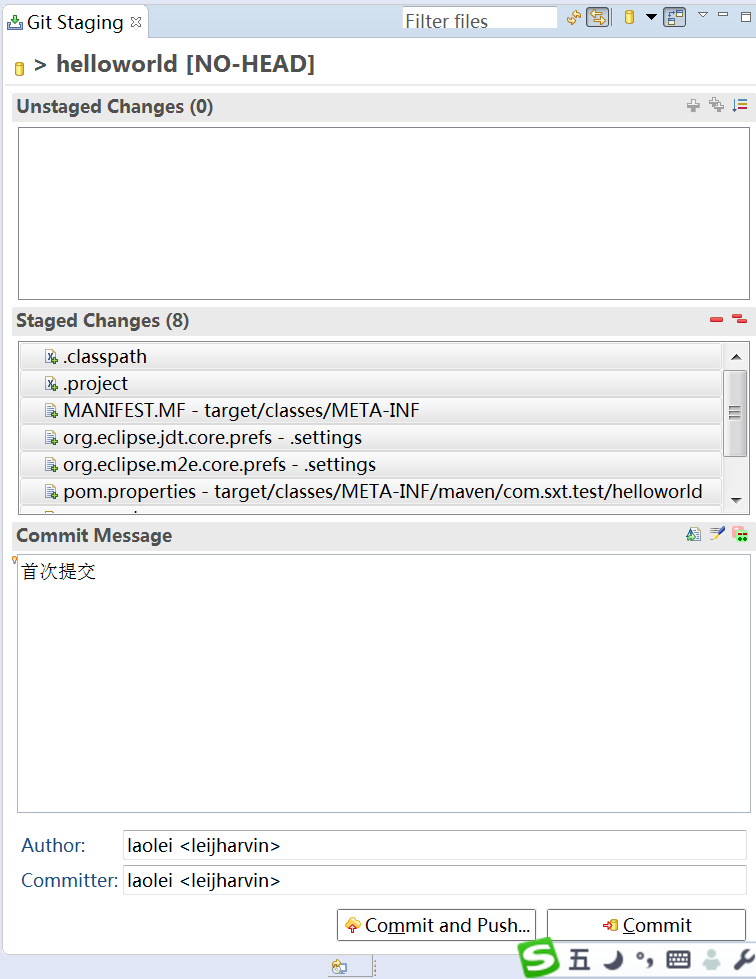


4、提交项目： 选中项目helloworld--- 右键 --- Team --- Commit ---

选择中要提交的文件添加到暂存区





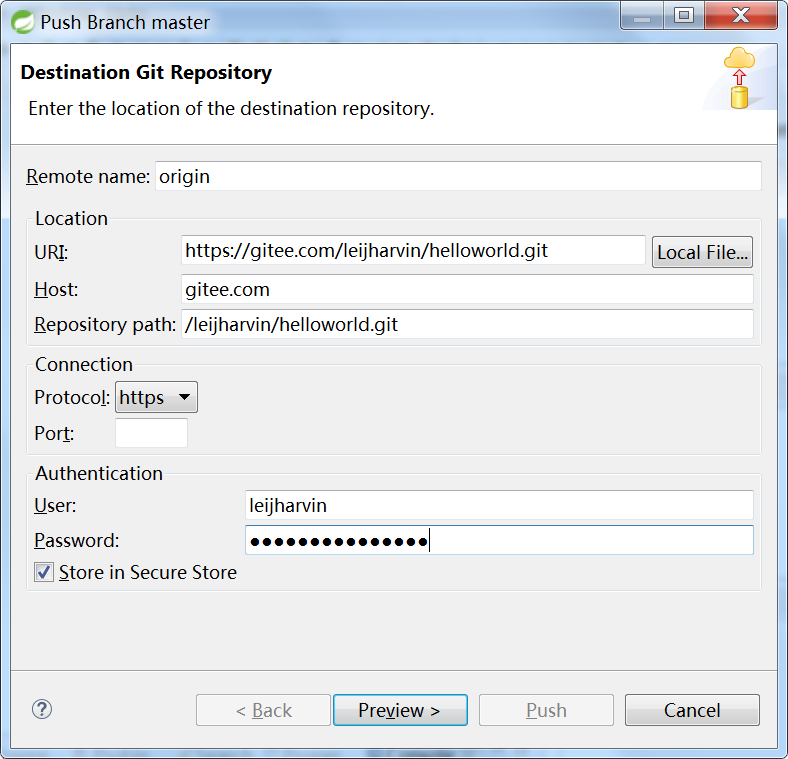


以上的两个按钮，第一个是提交到本地仓库并push到gitee

第二个按钮是提交到本地仓库

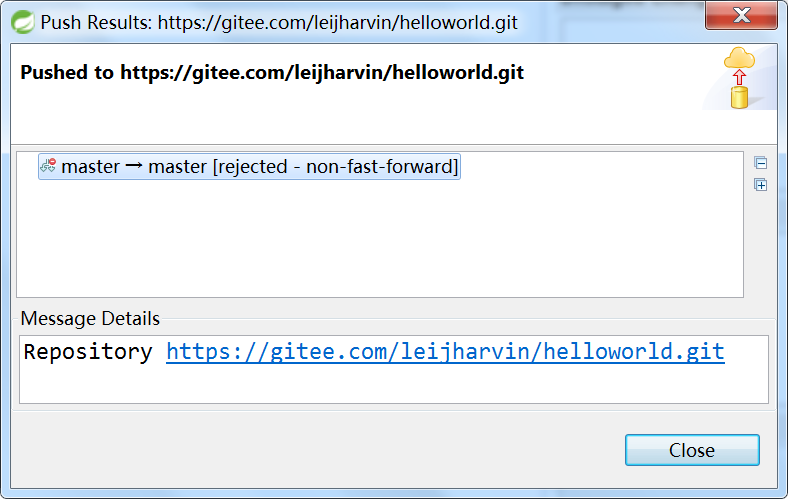
push到gitee

右键项目--team-->push Branch master



输入密码找回的问题





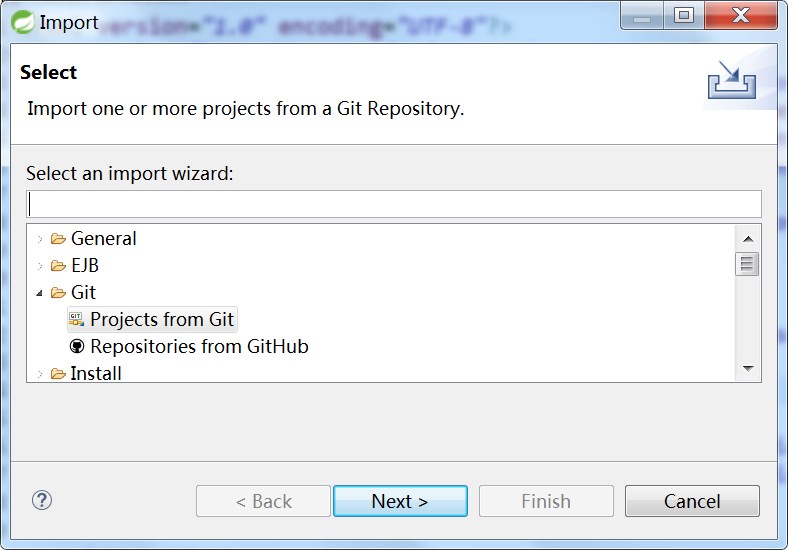
之后再看gitee里的代码就成功了

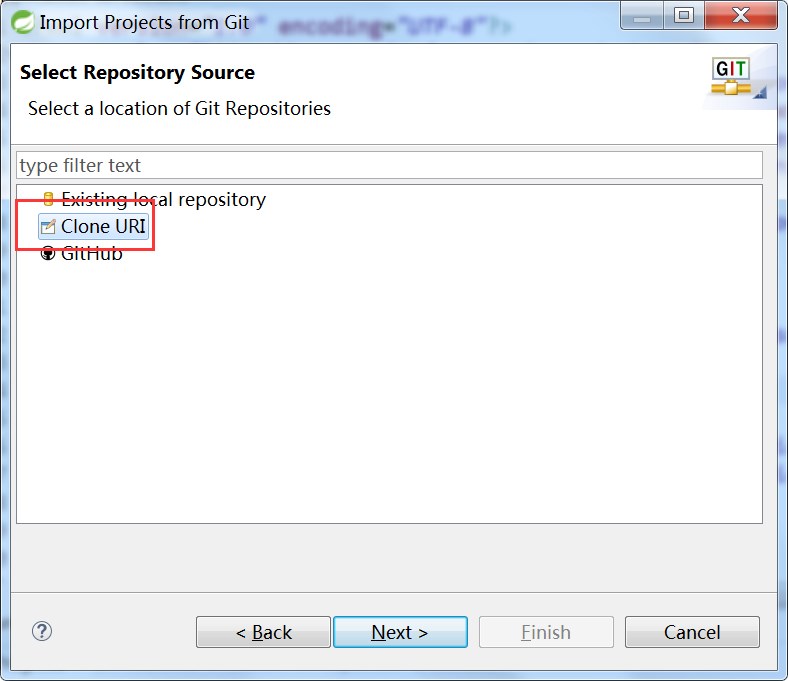
## 二，从gitee把项目拉取下来

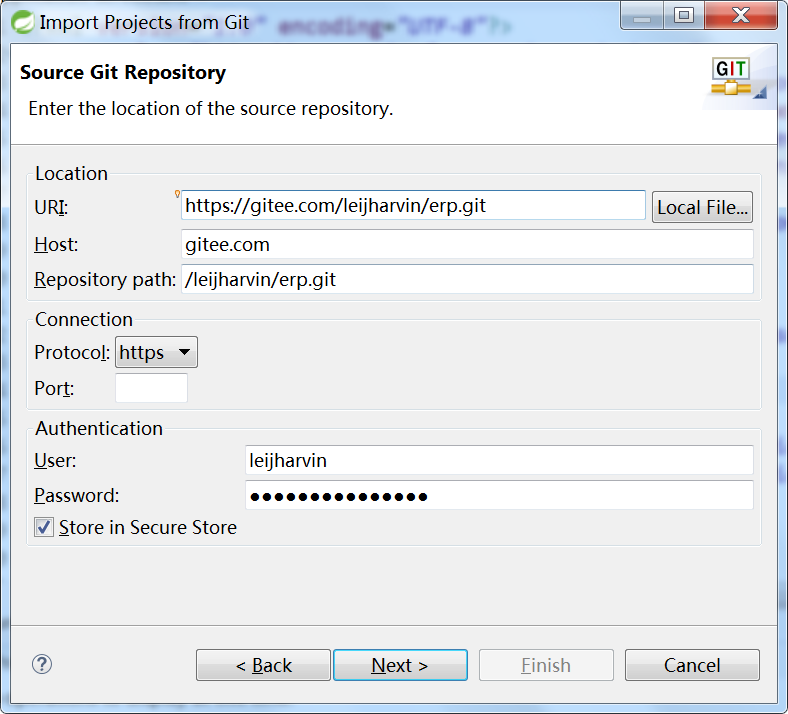
1，找到项目地址



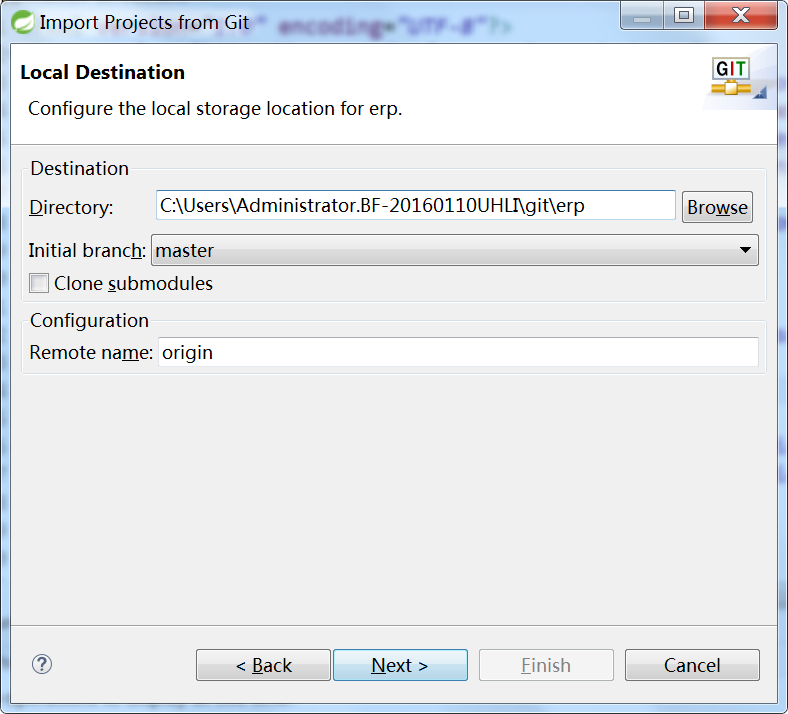
2，项目区域右键import  选择git







选择保存路径完事



# 01【熟悉】git简介

SVN ==版本控制工具 [集中式的版本控制工具]

|——所有的文件由中央服务器管理

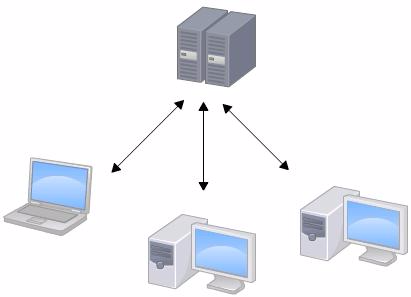
|----使用者必须能连接到中央服务器

GIT==版本控制工具[分布式版本控制工具]

## 3，集中式VS分布式

        Linus一直痛恨的CVS及SVN都是集中式的版本控制系统，而Git是分布式版本控制系统，集中式和 分布式版本控制系统有什么区别呢？

        先说集中式版本控制系统，版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。中央服务器就好比是一个图书馆，你要改一本书，必须先从图书馆借出来，然后回到家自己改，改完了，再放回图书馆。

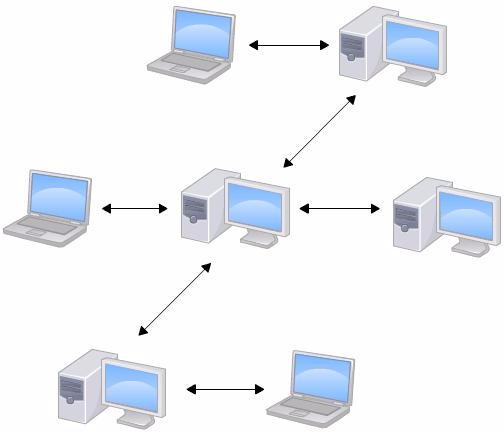


        集中式版本控制系统最大的毛病就是必须联网才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

        那分布式版本控制系统与集中式版本控制系统有何不同呢？首先，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。既然每个人电脑上都有一个完整的版本库，那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

        和集中式版本控制系统相比，分布式版本控制系统的安全性要高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。而集中式版本控制系统的中央服务器要是出了问题，所有人都没法干活了。

        在实际使用分布式版本控制系统的时候，其实很少在两人之间的电脑上推送版本库的修改，因为可能你们俩不在一个局域网内，两台电脑互相访问不了，也可能今天你的同事病了，他的电脑压根没有开机。因此，分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已。



        当然，Git的优势不单是不必联网这么简单，后面我们还会看到Git极其强大的分支管理，把SVN等远远抛在了后面。

        CVS作为最早的开源而且免费的集中式版本控制系统，直到现在还有不少人在用。由于CVS自身设计的问题，会造成提交文件不完整，版本库莫名其妙损坏的情况。同样是开源而且免费的SVN修正了CVS的一些稳定性问题，是目前用得最多的集中式版本库控制系统。

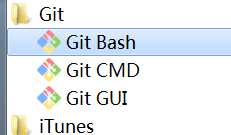
        除了免费的外，还有收费的集中式版本控制系统，比如IBM的ClearCase（以前是Rational公司的，被IBM收购了），特点是安装比Windows还大，运行比蜗牛还慢，能用ClearCase的一般是世界500强，他们有个共同的特点是财大气粗，或者人傻钱多。

        微软自己也有一个集中式版本控制系统叫VSS，集成在Visual Studio中。由于其反人类的设计，连微软自己都不好意思用了。

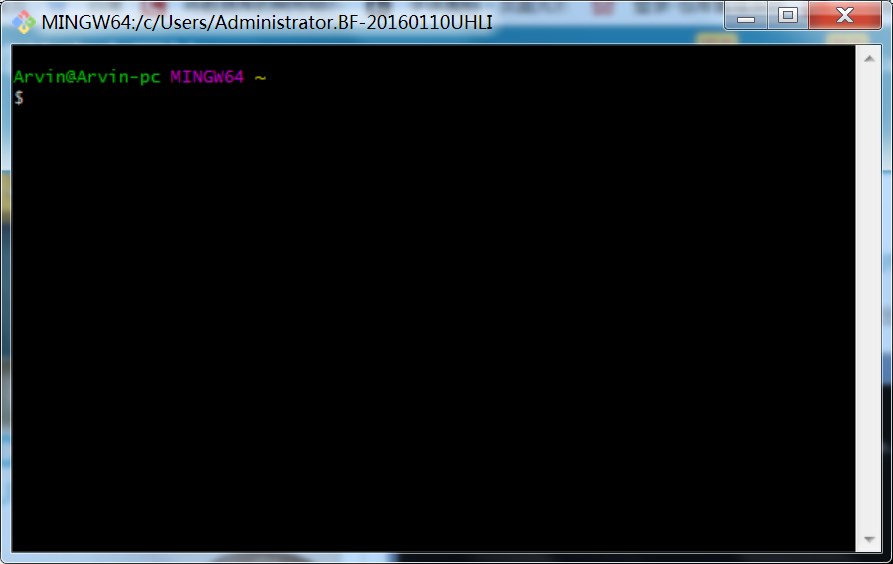
        分布式版本控制系统除了Git以及促使Git诞生的BitKeeper外，还有类似Git的Mercurial和Bazaar等。这些分布式版本控制系统各有特点，但最快、最简单也最流行的依然是Git！

## 3，设置全局身份

1，打开git Bash



如果出现以下界面说明安装成功了（除非脸黑，否则是没有任何问题的）



因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和Email地址。

$ git config --global user.name "Your Name"

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

