

# Git历史

同生活中的许多伟大事件一样，Git 诞生于一个极富纷争大举创新的年代。Linux 内核开

源项目有着为数众广的参与者。绝大多数的 Linux 内核维护工作都花在了提交补丁和保存

归档的繁琐事务上（1991－2002年间）。到 2002 年，整个项目组开始启用分布式版本控制

系统 BitKeeper 来管理和维护代码。

到 2005 年的时候，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结

束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux

的缔造者 Linus Torvalds ）不得不吸取教训，只有开发一套属于自己的版本控制系统才不

至于重蹈覆辙。他们对新的系统订了若干目标：

• 速度

• 简单的设计

• 对非线性开发模式的强力支持（允许上千个并行开发的分支）

• 完全分布式

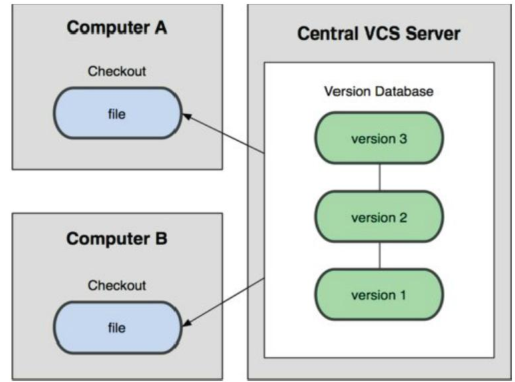
• 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）

# Git与svn对比

## Svn

svn属于集中式版本管理控制系统，系统中保存了所有文件的修订版本，而协同工作人员通过连接svn服务器，提取出最新的文件，获取提交更新

下图就是标准的集中式版本控制工具管理方式：



集中管理方式在一定程度上看到其他开发人员在干什么，而管理员也可以很轻松掌握每个人的开发权限。

但是相较于其优点而言，集中式版本控制工具缺点很明显：

1. 服务器单点故障
2. 容错性差

## Git

在 Git 内都有三种状态：已提交（committed），已修改（modified）和已暂存（staged）。已提交表示该文件已经被安全地保存在本地数据库中了；已修改表示修改了某个文件，但还没有提交保存；已暂存表示把已修改的文件放在下次提交时要保存的清单中。

由此我们看到 Git 管理项目时，文件流转的三个工作区域：Git 的本地数据目录，工作

目录以及暂存区域。





1. 如果是 git 目录中保存着的特定版本文件，就属于已提交状态；
2. 如果作了修改并已放入暂存区域，就属于已暂存状态；
3. 如果自上次取出后，作了修改但还没有放到暂存区域，就是已修改状态。

# 安装git服务端

1、安装git服务环境准备

yum -y install curl curl-devel zlib-devel openssl-devel perl cpio expat-devel gettext-devel gcc cc

2、下载git-2.5.0.tar.gz

1）解压缩

2）cd git-2.5.0

3）autoconf

4）./configure

5）make

6）make install

3、添加用户

adduser -r -s /bin/sh -c 'git version control' -d /home/git git

4、设置/home/git文件夹操作权限

mkdir /home/git

chown git:git /home/git

5、设置密码

passwd git

输入两次密码

6、切换到git用户

su git

7、创建git仓库

git --bare init /home/git/first

注意：如果不使用“--bare”参数，初始化仓库后，提交master分支时报错。这是由于git默认拒绝了push操作，需要.git/config添加如下代码：

[receive]

denyCurrentBranch = ignore

推荐使用：git --bare init初始化仓库。

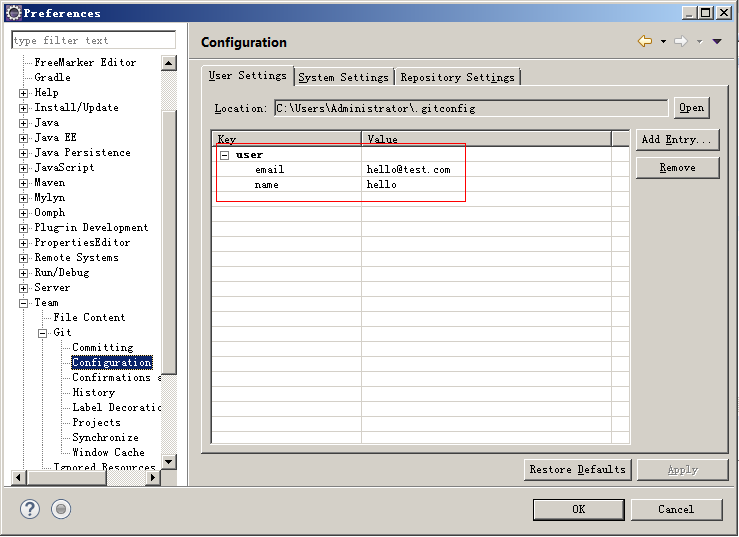
# Eclipse客户端egit

## 安装egit插件

EclipseMars版本自带此插件

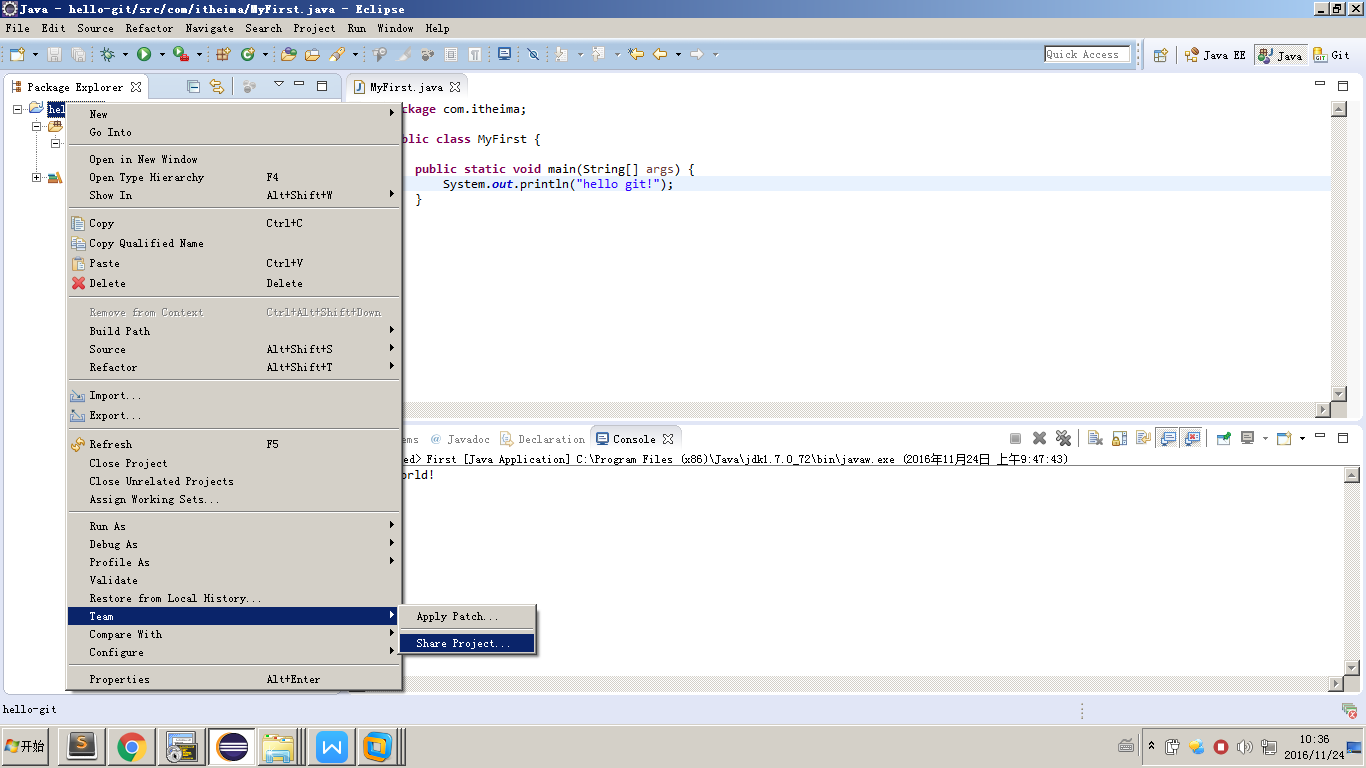
## Egit使用

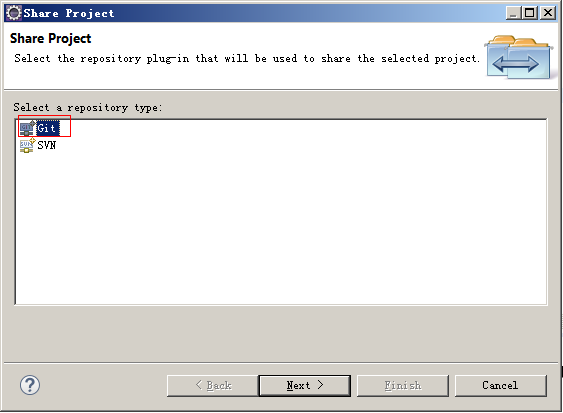
### 用户信息

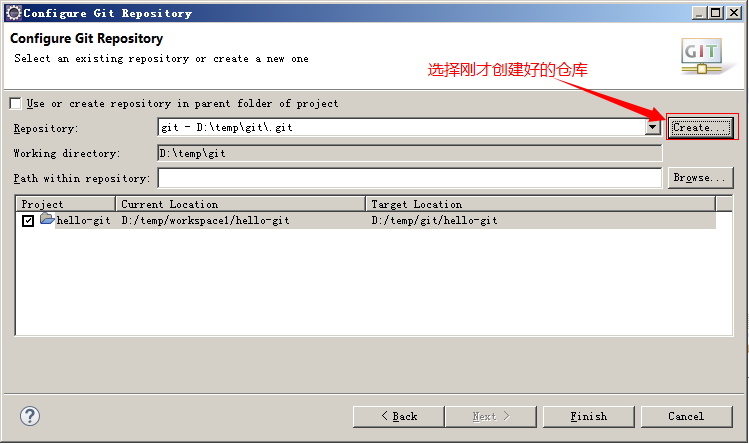


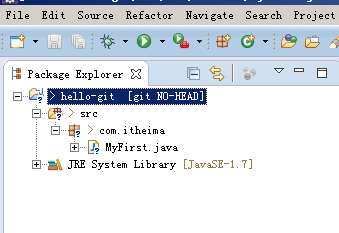
配置用户信息，此信息为提交代码时，提交者的身份认证信息。

### 将工程添加到本地仓库

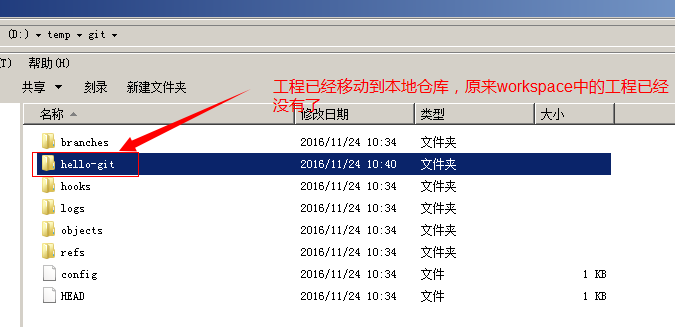








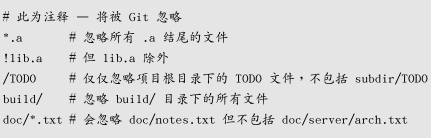
当工程添加到本地仓库后，工程的位置以及移动到本地仓库。



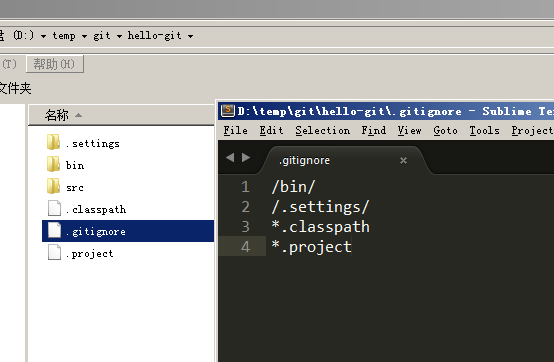
### 添加暂存区

#### 忽略文件

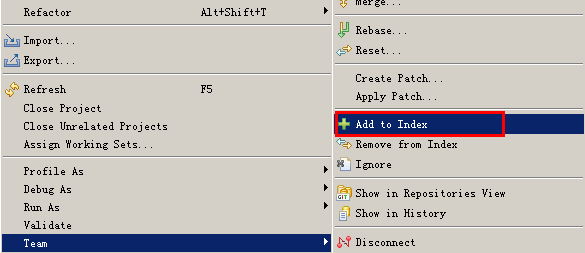
在添加暂存区之前，项目中有些文件不需要上传到仓库中，可以使用git排除配置文件.gitignore来忽略上传文件。



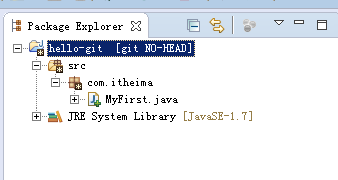




#### 添加到暂存区

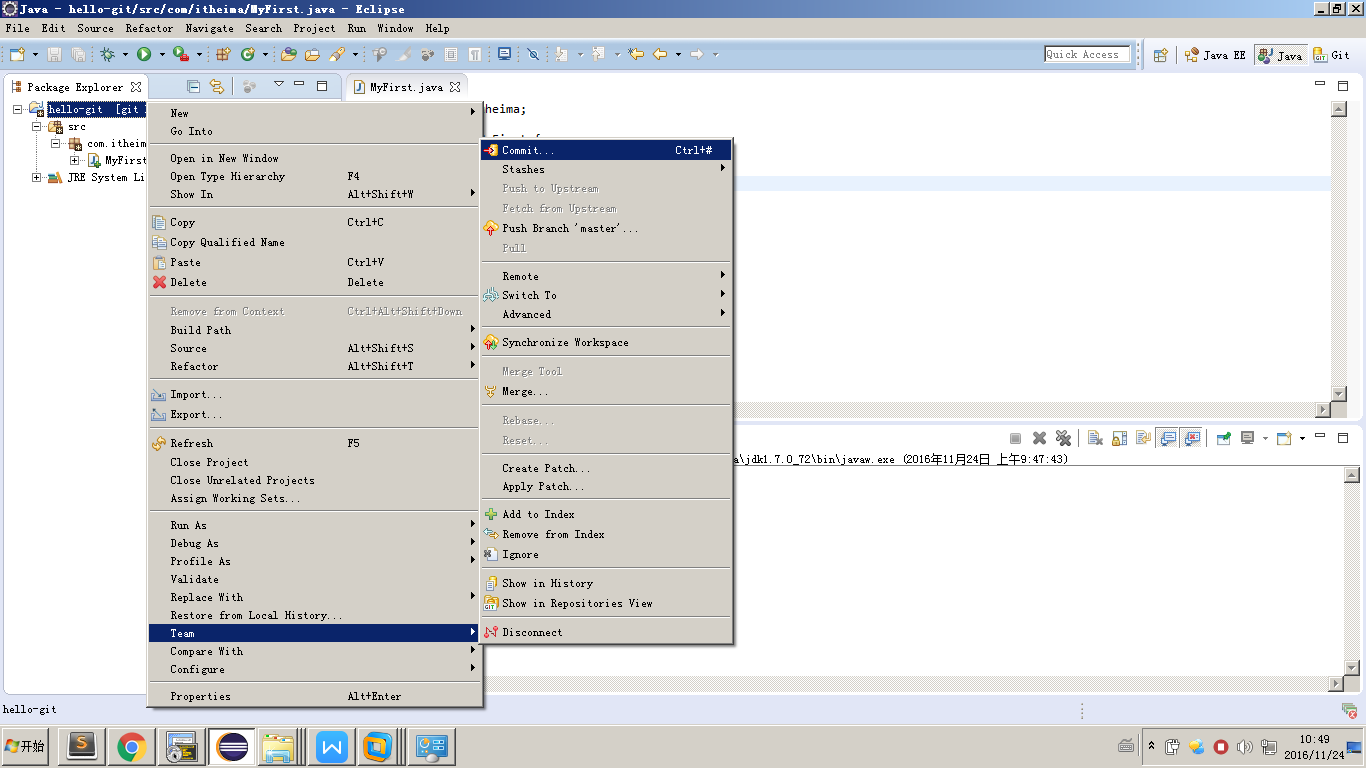


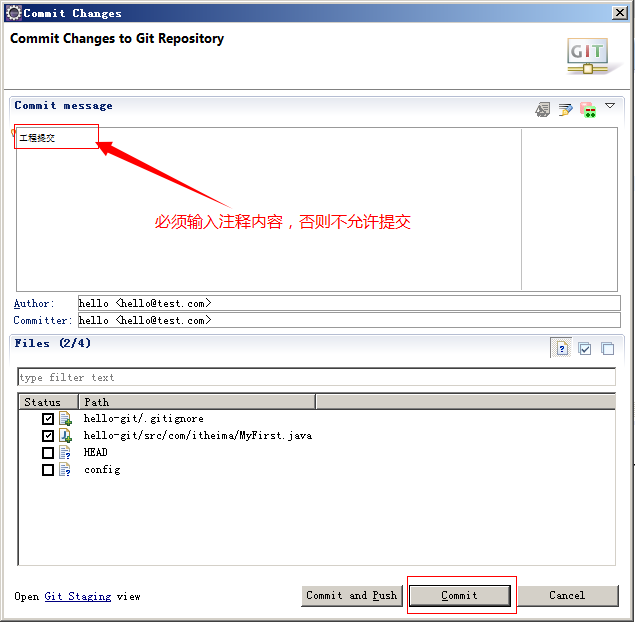
把本地文件添加本地版本库暂存区域。



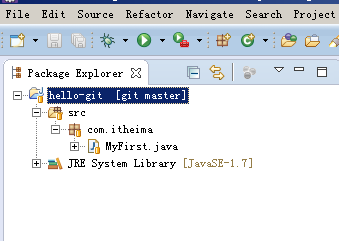
### 提交（本地仓库）

必须先添加到暂存区域后才能提交代码





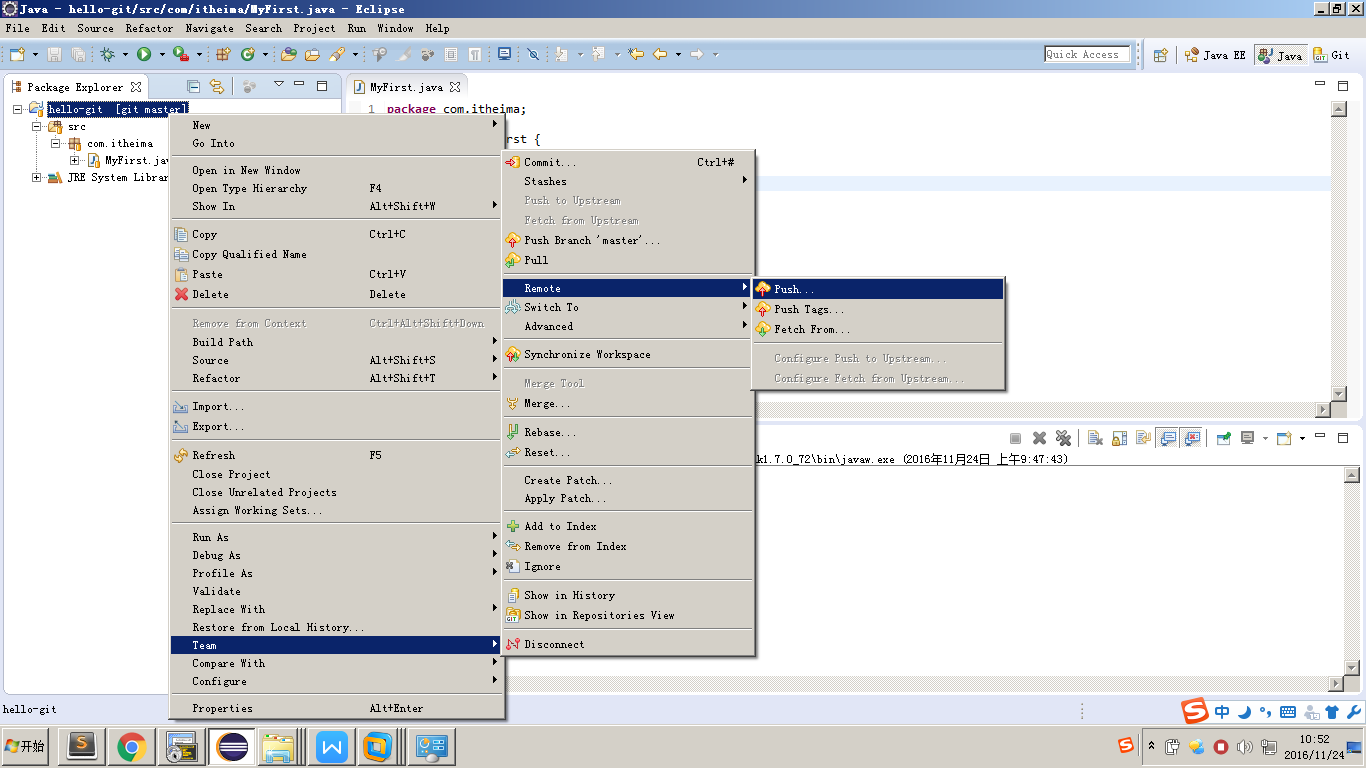
注意：提交时必须编写提交信息，不能添加空格。



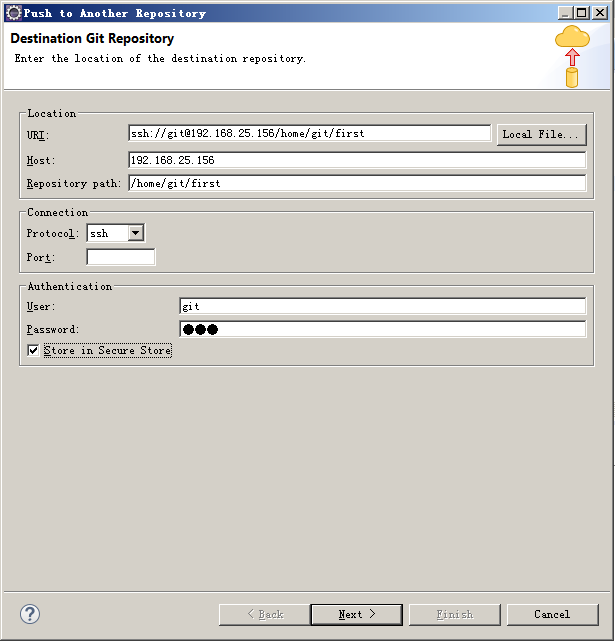
### 共享版本库（push到远程仓库）

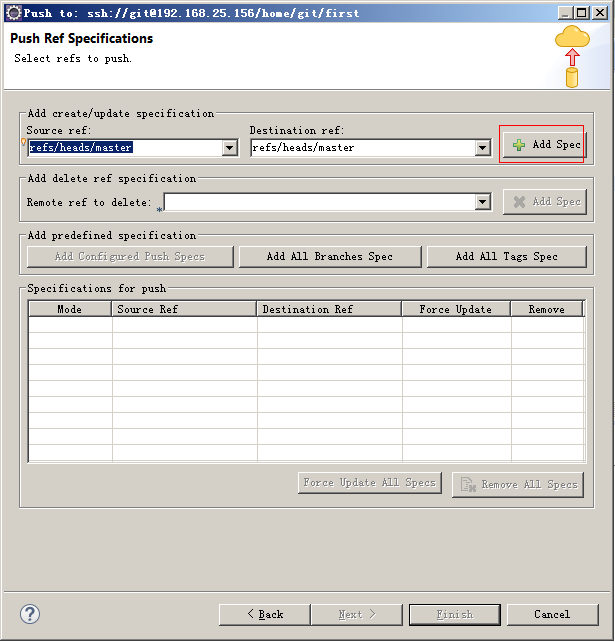
本地用户提交先提交到本地版本库，如果需要提交到共享版本库，那么就需要push到远程共享版本库。

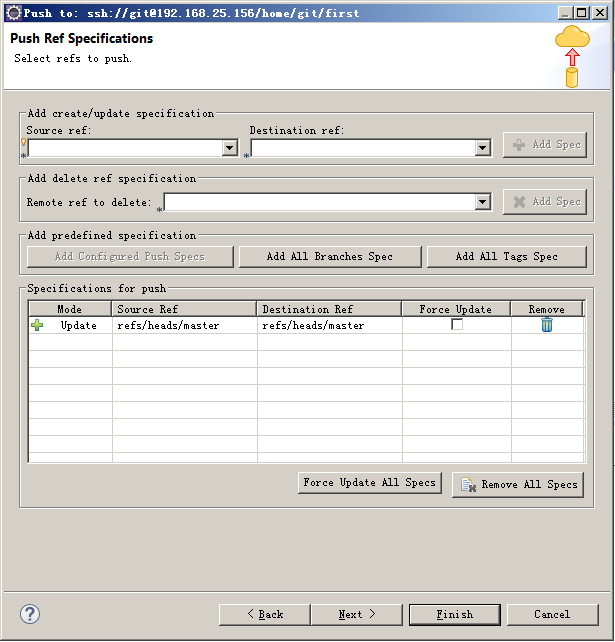
选择push把当前项目代码push到共享版本库中。



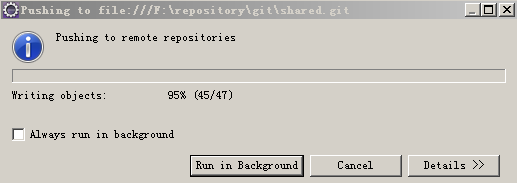
填写远程仓库地址：



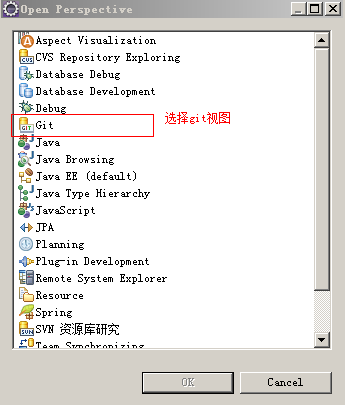




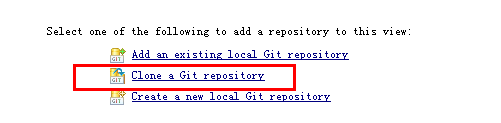
点击finish提交成功。



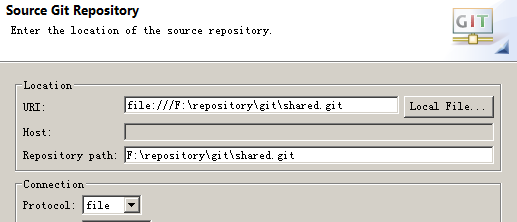
### 克隆（从远程仓库下载代码）



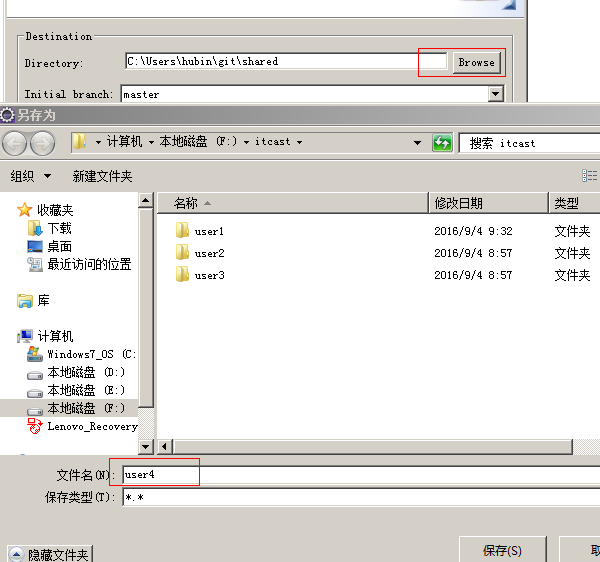
选择git视图，创建共享版本库：



指定共享版本库位置，连接共享版本库。更新代码到本地仓库

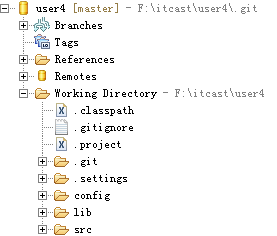


创建新的本地仓库：

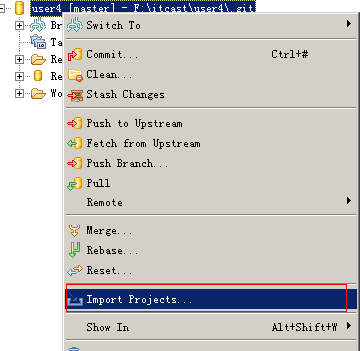


本地仓库创建完毕：

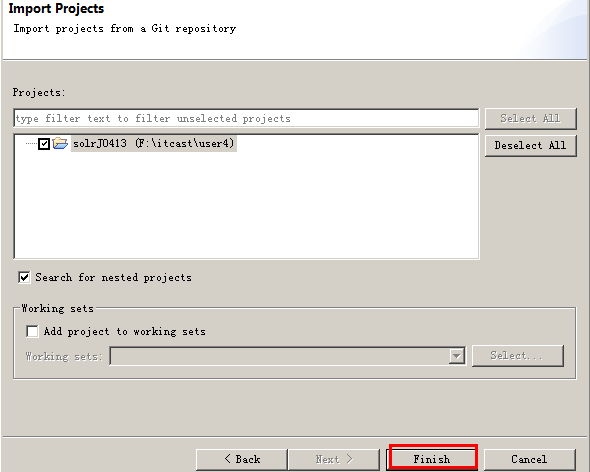
此操作是把远程仓库克隆到本地仓库。



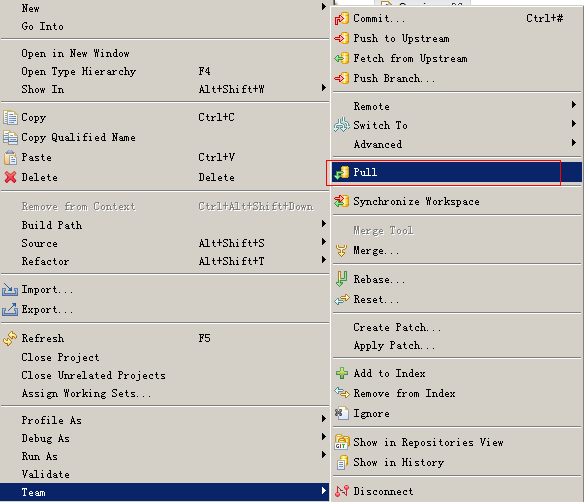
接下来需要把本地仓库项目导入到本地eclipse中进行开发：



直接导入项目：

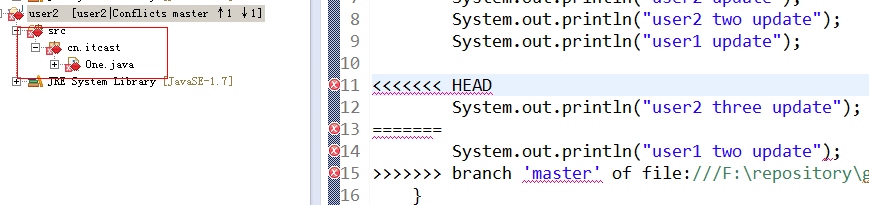


### 更新

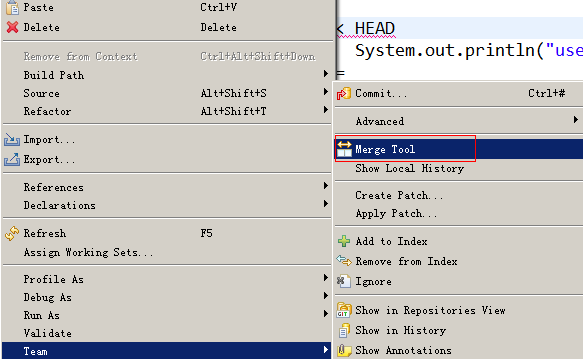


直接从共享版本库更新版本内容。

### 解决冲突



使用合并工具merge tool解决冲突：



合并有冲突代码

