
# Git与Github

讲师：夏磊

# 为什么要学Git

1、必点天赋

2、必然趋势

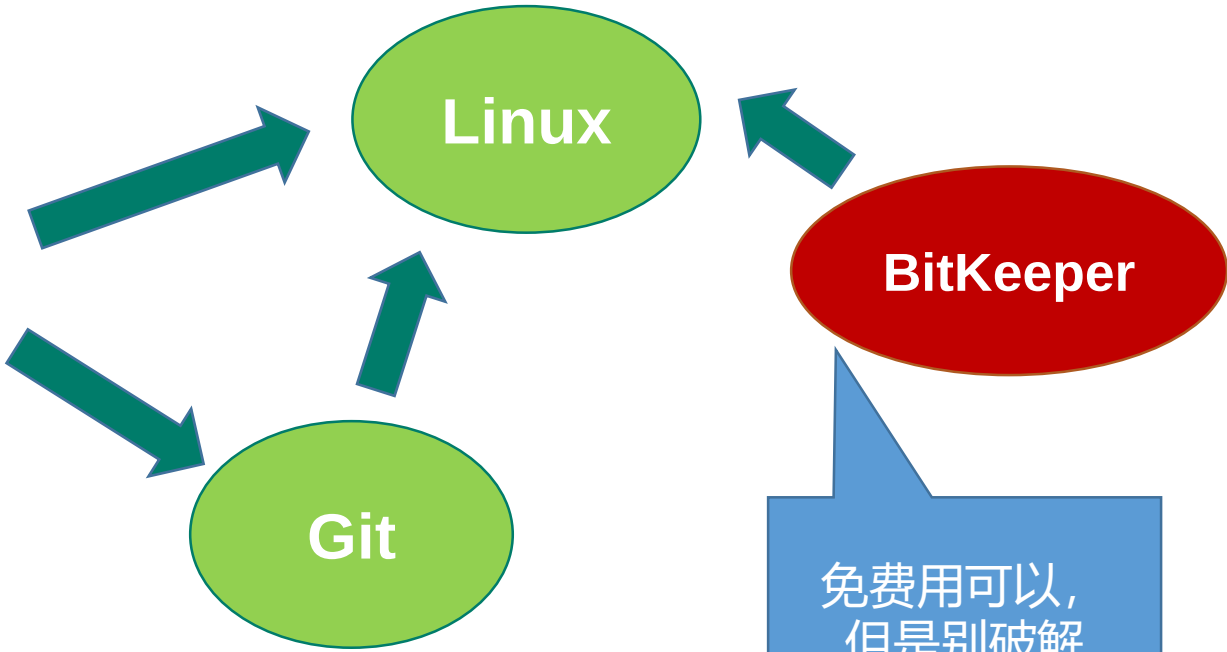
# 目录



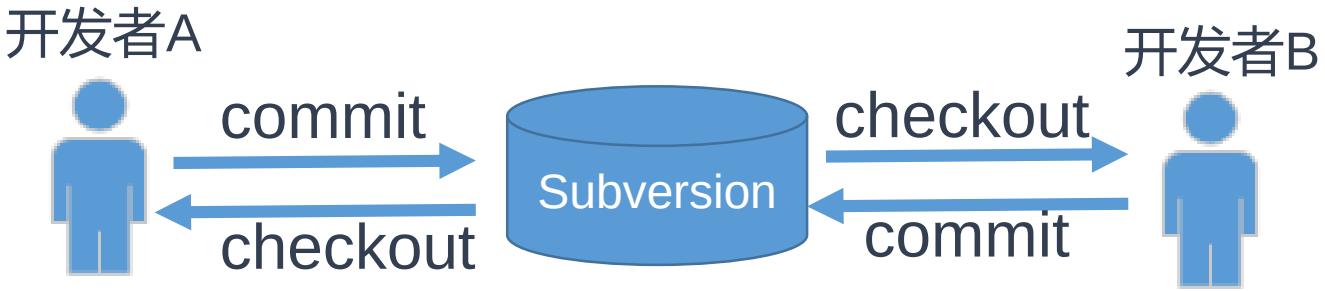
尚硅谷

# Git是目前世界上最先进的分布式版本控制系统。

Linus Torvalds
林纳斯·托瓦兹



# 版本管理系统能干什么




经典的集中管理型（CVS、VSS、SVN）

特点：

实现了大部分开发中对版本管理的需求

结构简单，上手容易。

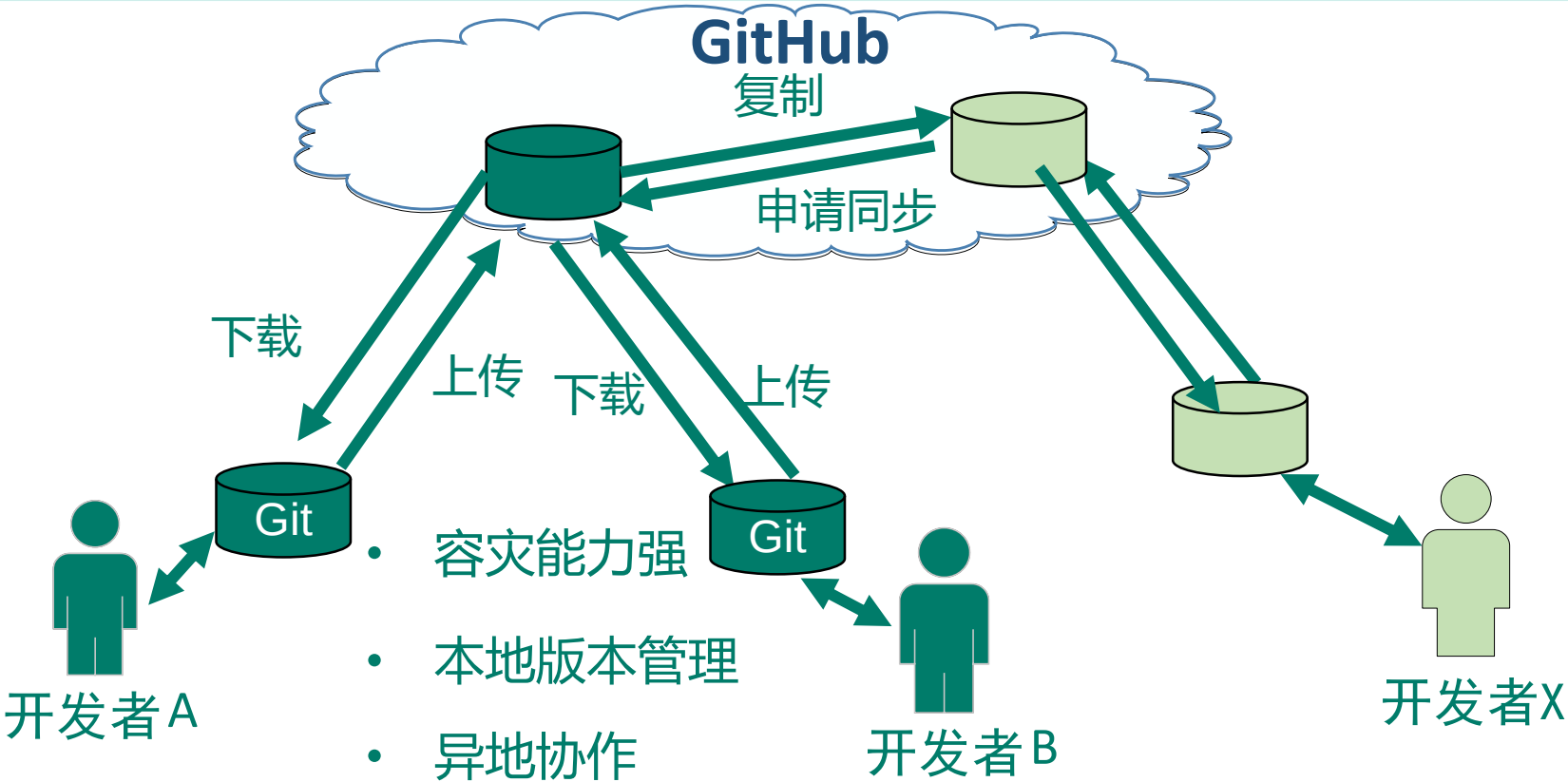
1、版本管理的服务器一旦崩溃，硬盘损坏，代码如何恢复?

2、程序员上传到服务器的代码要求是完整版本，但是程序员开发过程中想做小版本的管理，以便追溯查询，怎么破?

3、系统正在上线运行，时不时还要修改bug，要增加好几个功能要几个月，如何管理几个版本?

4、如何管理一个分布在世界各地、互不相识的大型开发团队?

# Git简介及安装—分布式




- 容灾能力强
- 本地版本管理
- 异地协作
- 灵活分支

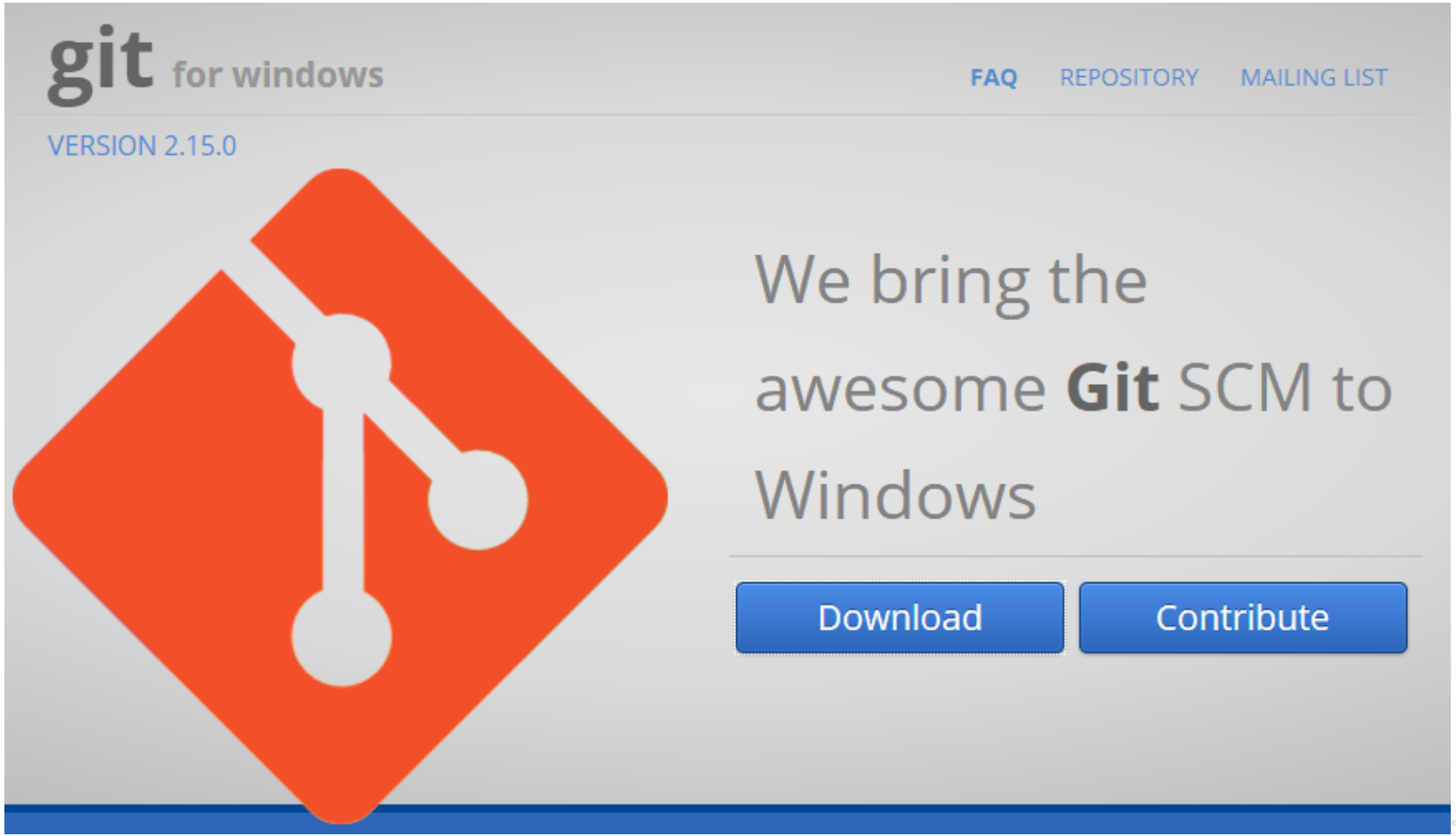
1、命令行工具：Git for windows
下载地址：https://git-for-windows.github.io/

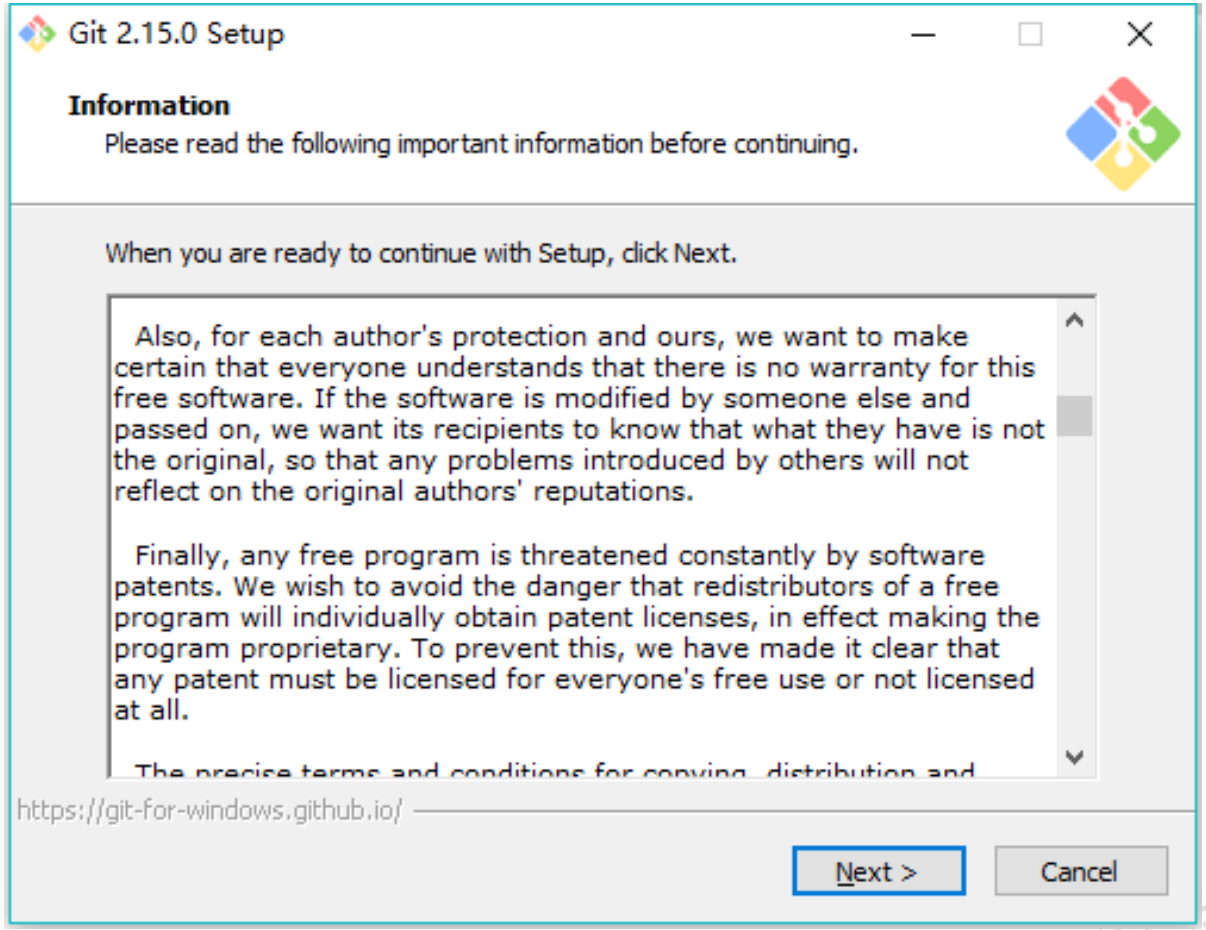
2、 操作系统中可视化工具：TortoiseGit
下载地址： https://tortoisegit.org/

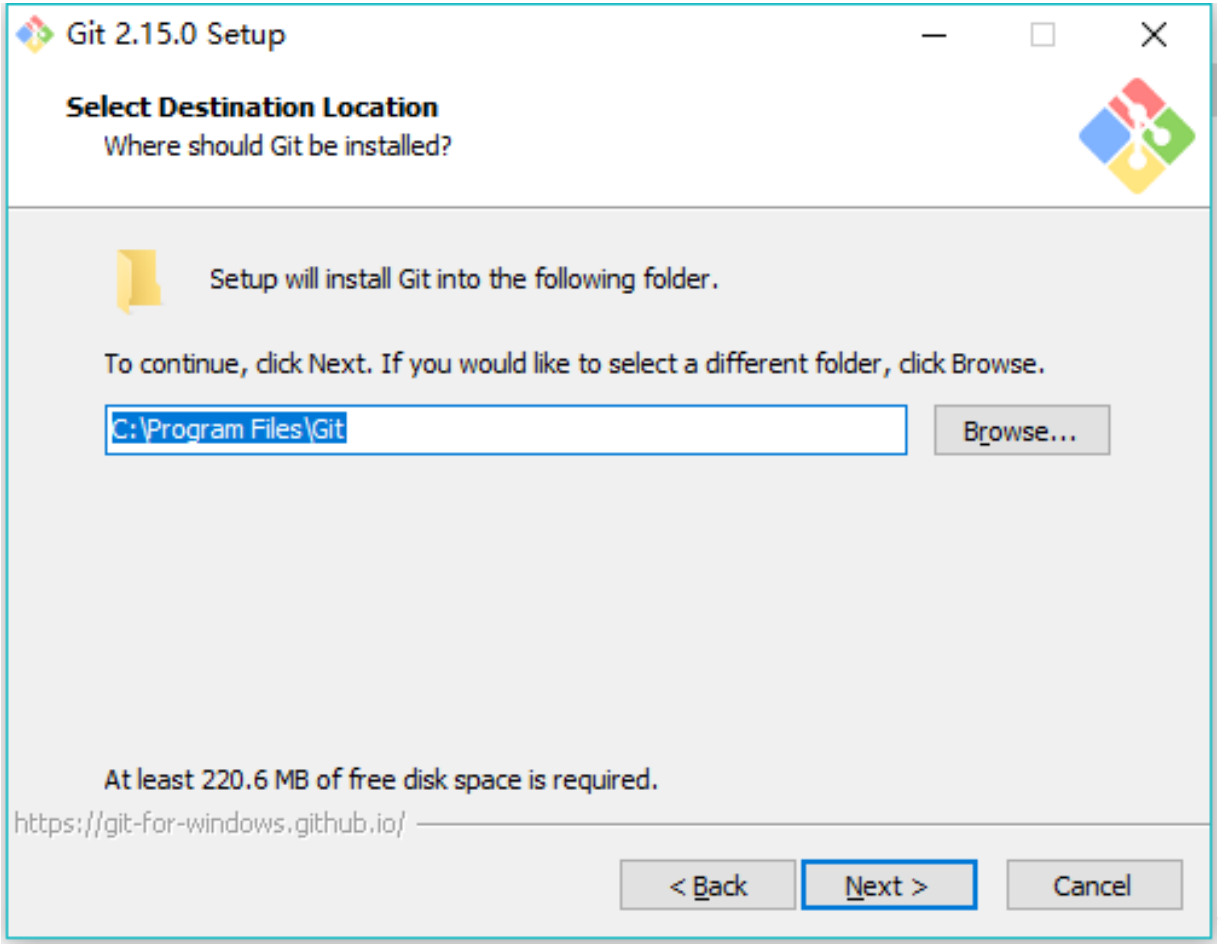
3、 Eclipse插件： Egit
Eclipse自带，插件市场搜索最新版

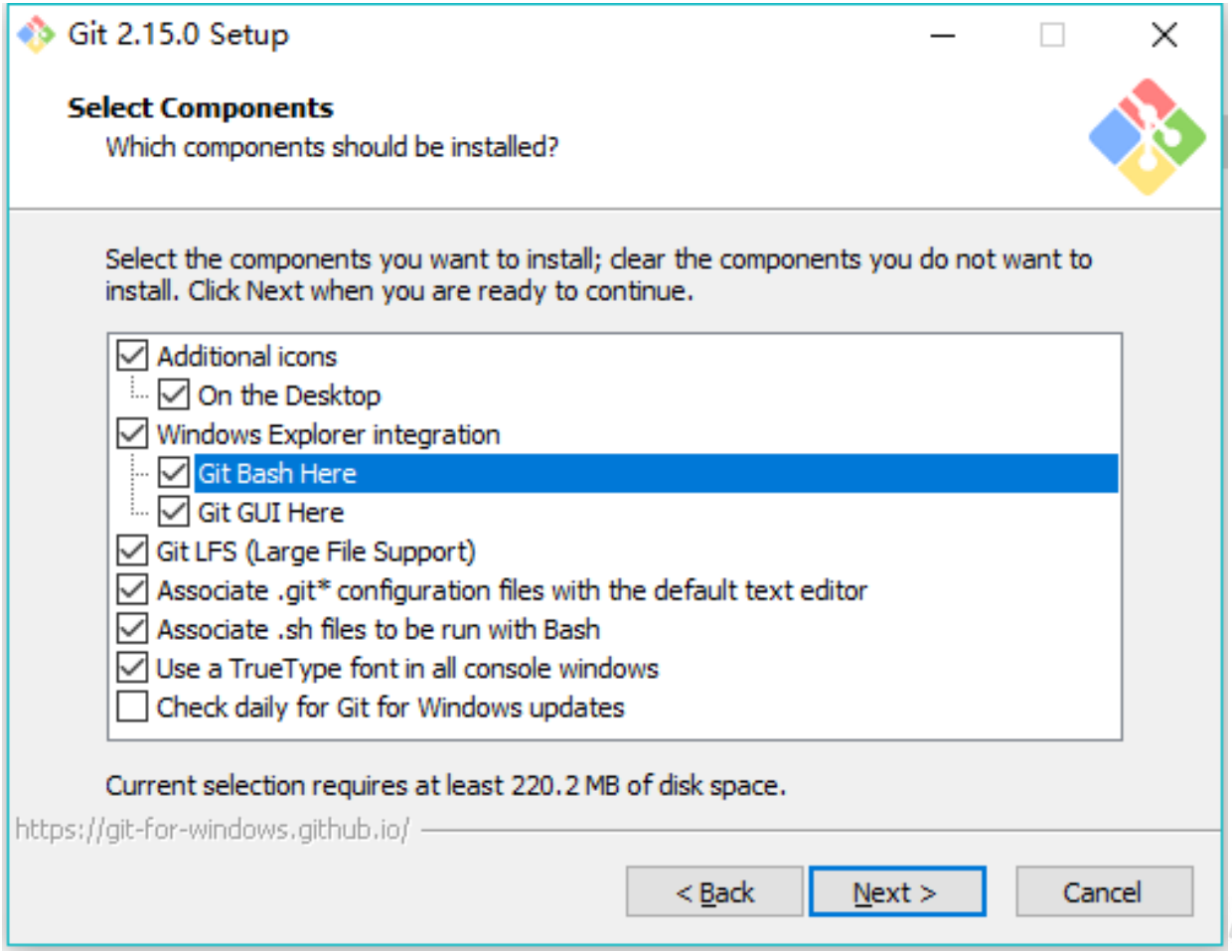
4、 GitHub网站
http://www.github.com

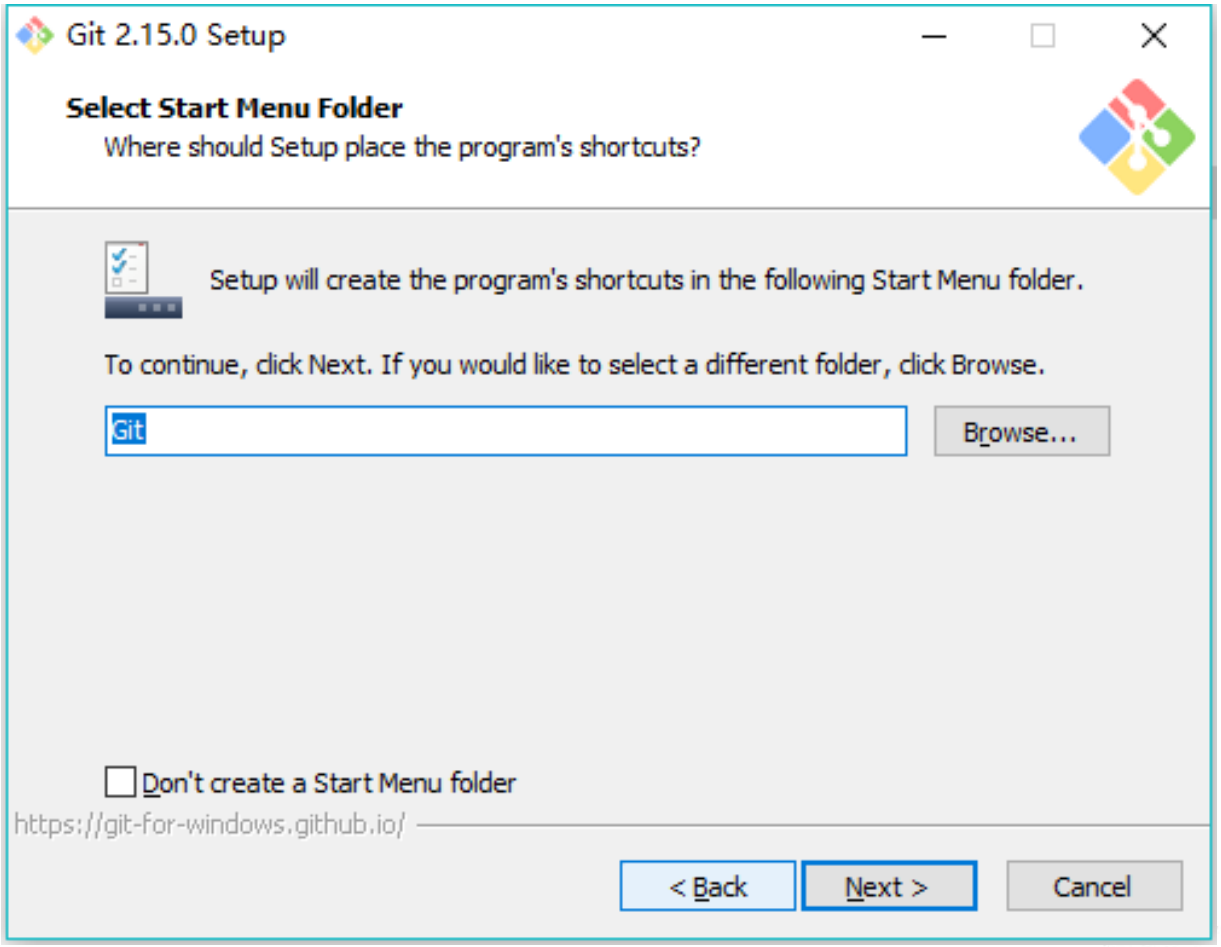
# Git简介及安装—安装


git for windows
FAQ
REPOSITORY
MAILING LIST
VERSION 2.15.0
We bring the
awesome Git SCM to
Windows
Download
Contribute

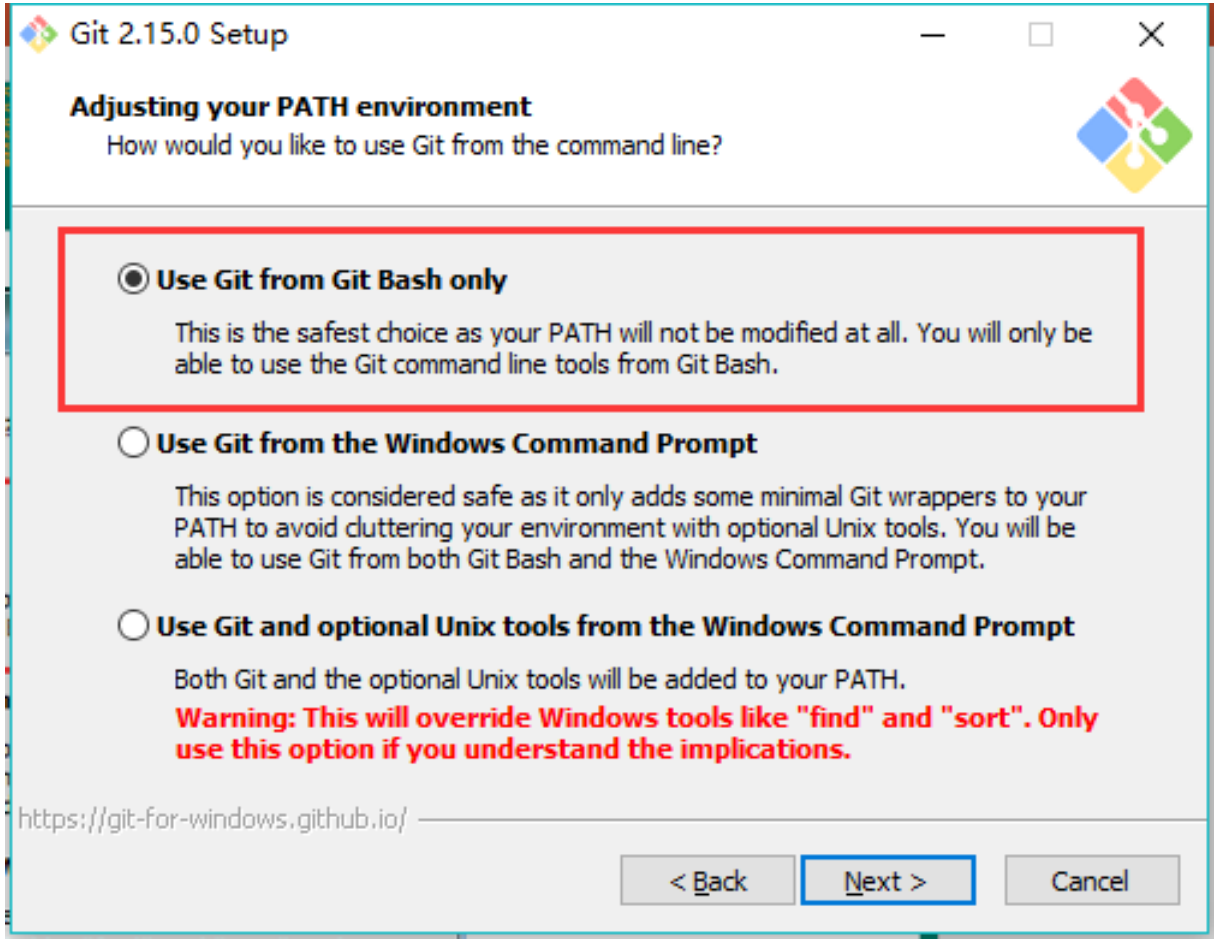

Git 2.15.0 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.
https://git-for-windows.github.io/
Next >
Cancel


Git 2.15.0 Setup
Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Git
Browse...
At least 220.6 MB of free disk space is required.
https://git-for-windows.github.io/
< Back
Next >
Cancel


Git 2.15.0 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Check daily for Git for Windows updates
Current selection requires at least 220.2 MB of disk space.
https://git-for-windows.github.io/
< Back
Next >
Cancel

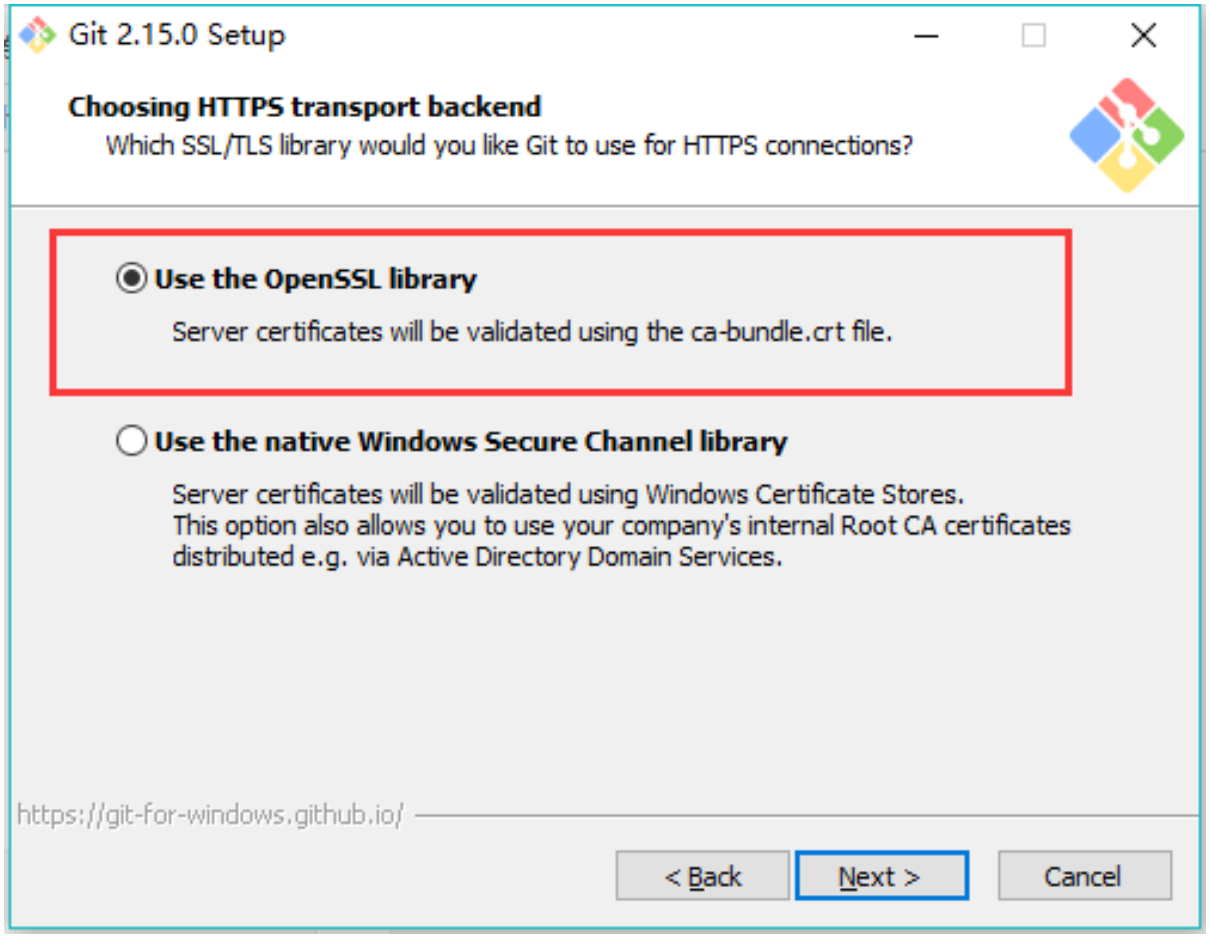

Git 2.15.0 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://git-for-windows.github.io/
< Back
Next >
Cancel

# Git简介及安装—安装

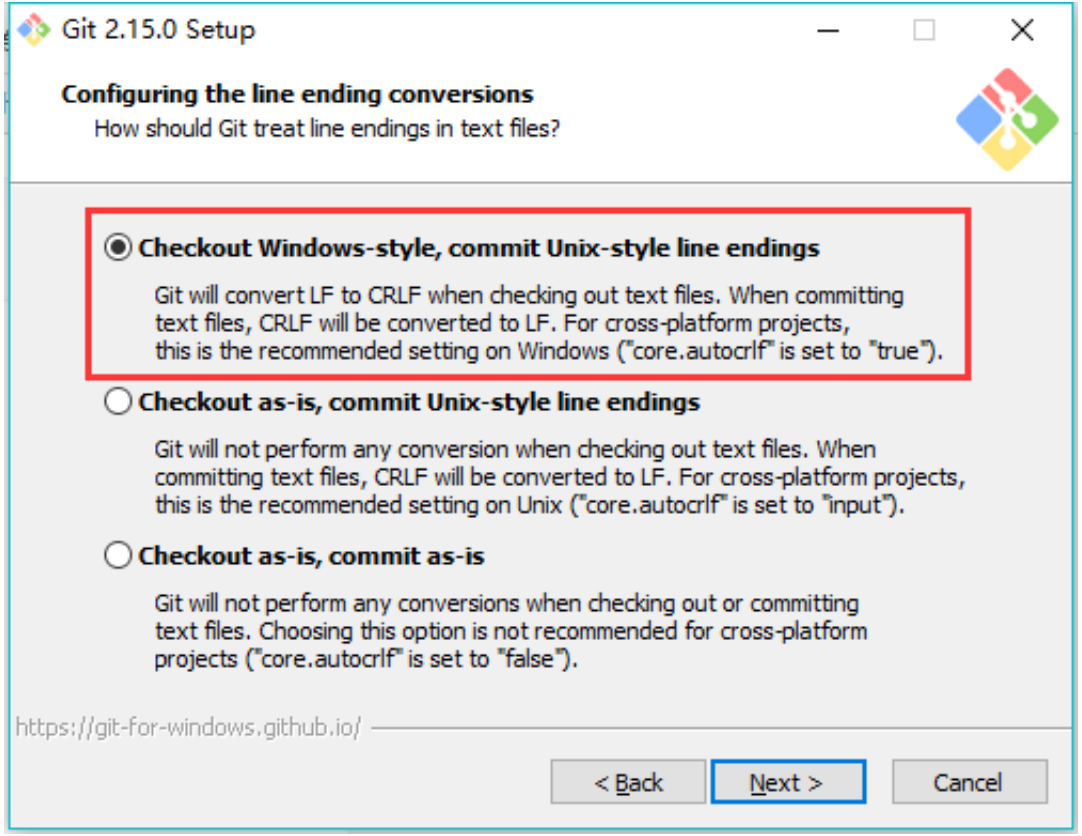



选择Git命令的执行环境，这里推荐选择第一个，就是单独用户Git自己的命令行窗口。不推荐和windows的命令行窗口混用。

# Git简介及安装—安装

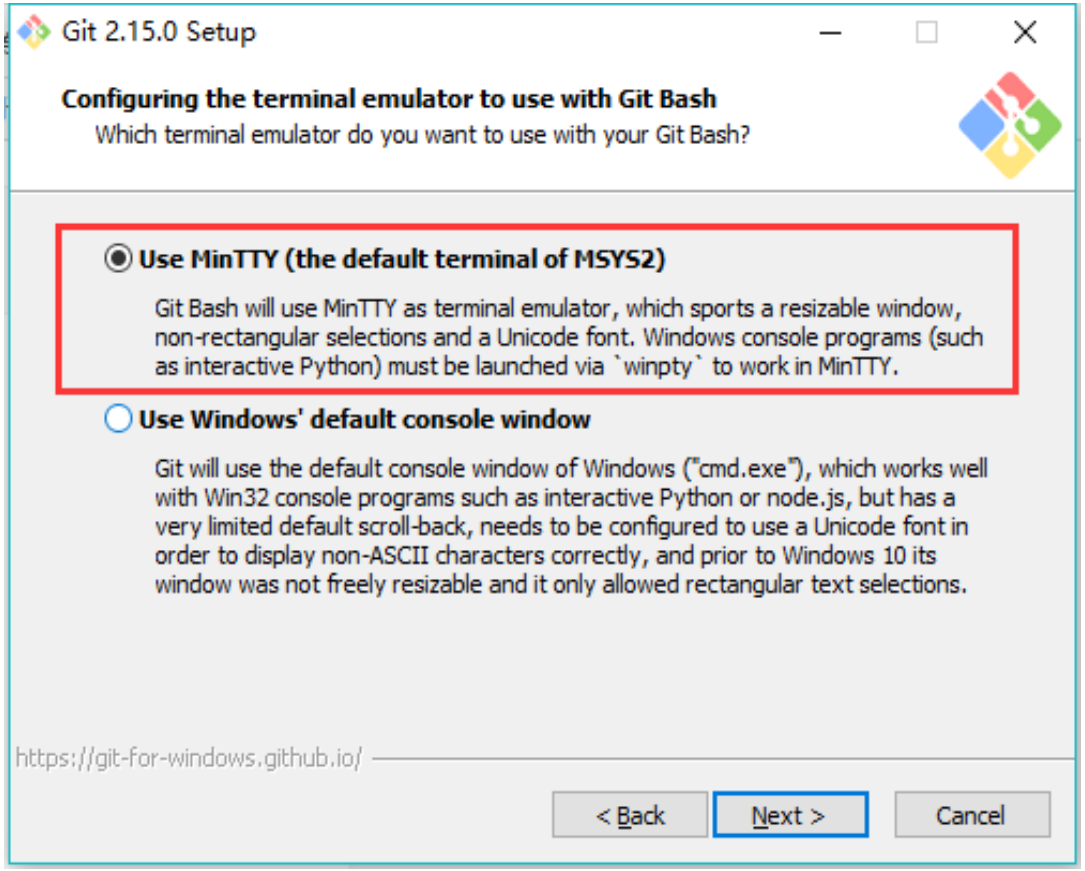



HTTPS传输：
使用SSL传输协议



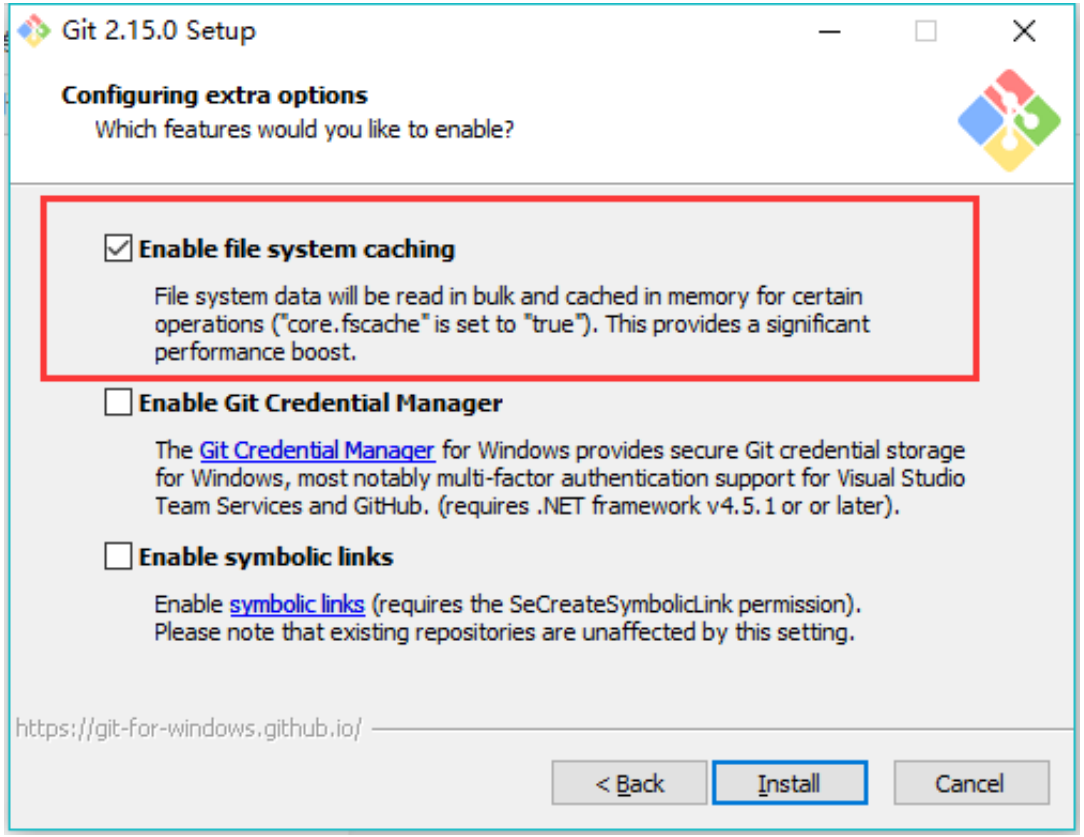

在“Configuring the line ending conversions”选项中，

- 第一个选项：如果是跨平台项目，在windows系统安装，选择；
- 第二个选项：如果是跨平台项目，在Unix系统安装，选择；
- 第三个选项：非跨平台项目，选择。



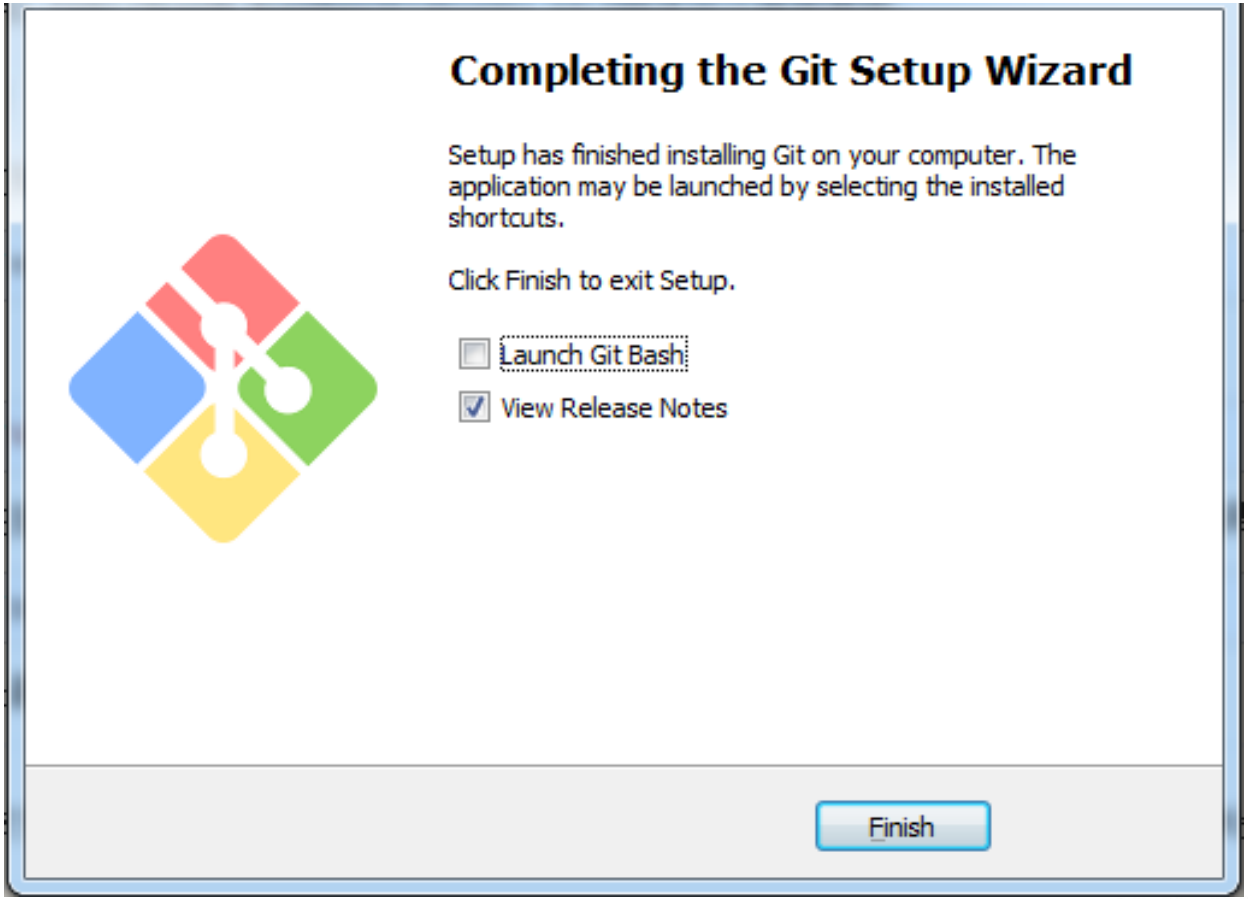

在“terminal emulator”选项中，

- 第一个选项：使用专用的Git窗口（推荐）
- 第二个选项：使用windows的cmd命令行窗口。

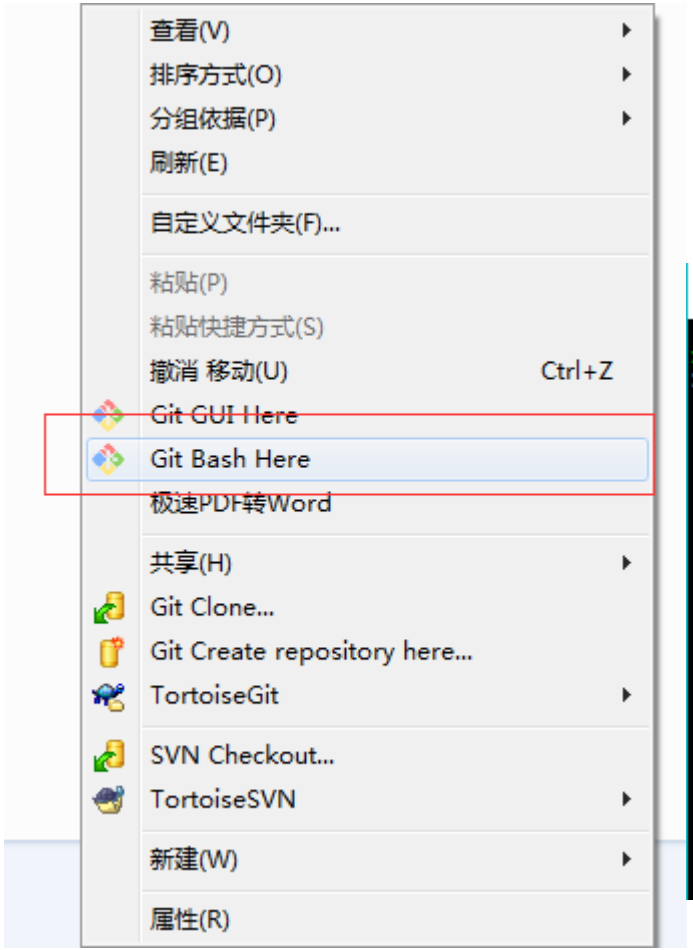



在“Configuring extra”选项中，

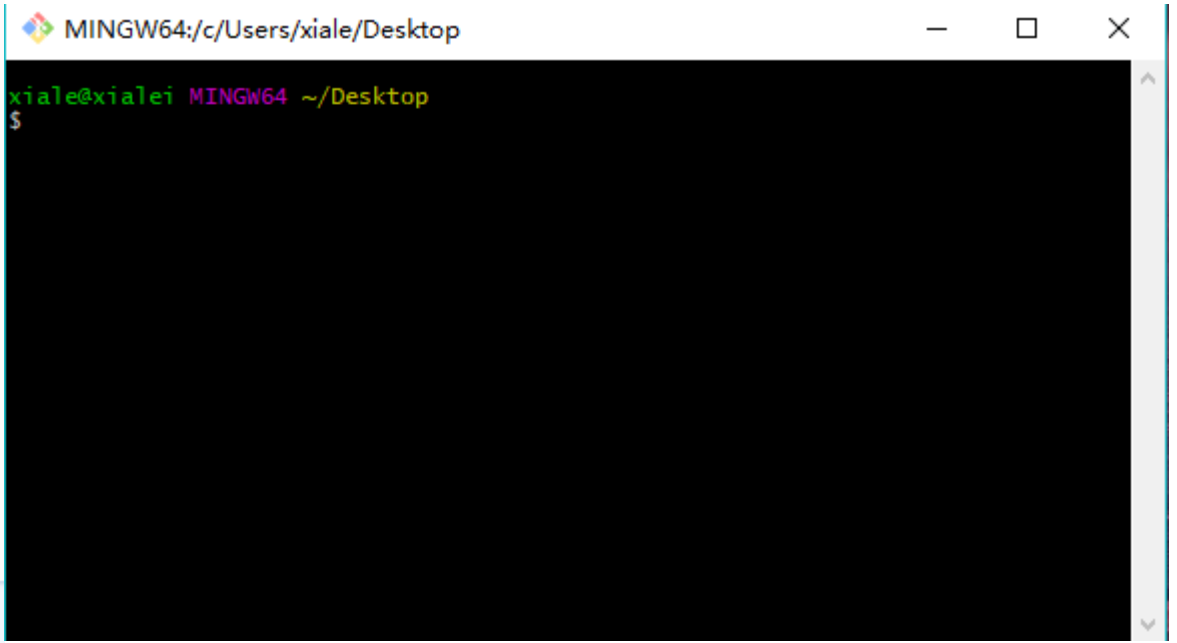
- 默认开启文件缓存即可（推荐）


Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Finish

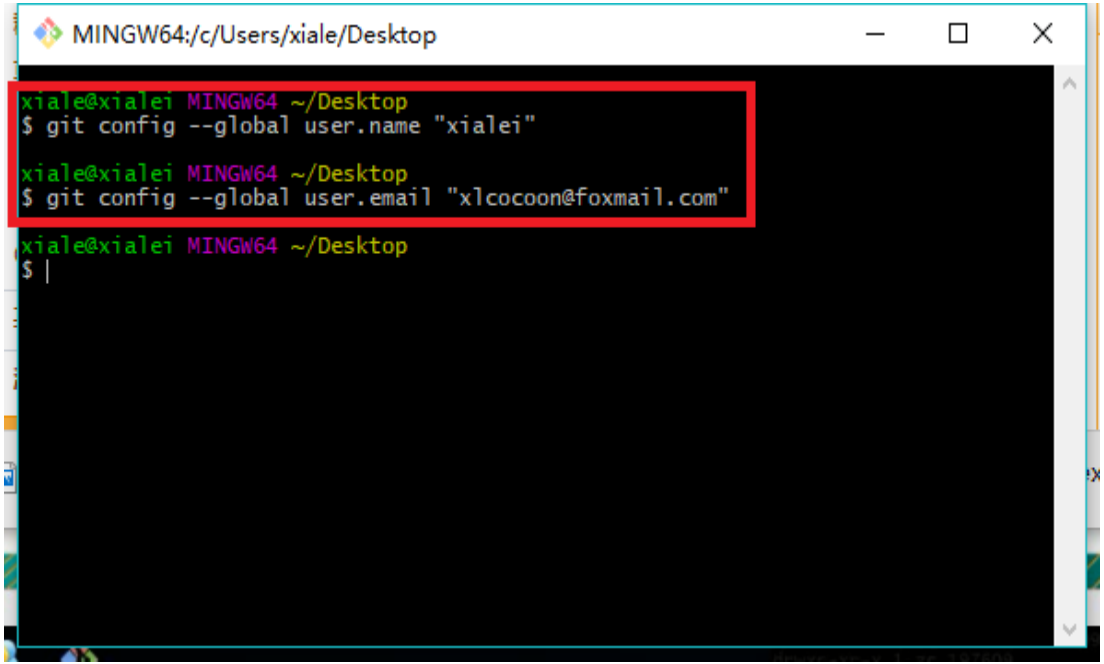
# Git简介及安装—安装




安装完成后，在任意的文件目录下，
右键都可以开打Git的命令行窗口。



安装完成后，还需要最后一步设置，在命令行输入如下：




Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识。

C:\Users\admin路径下的.gitconfig文件里面可以看到

--global 表示全局属性，所有的git项目都会共用属性

# 目录



尚硅谷

## 1.创建版本库

- 在项目文件夹内，执行: git init

## 2.提交文件

- 新建文件后，通过git status 进行查看文件状态
- 将文件添加到暂存区 git add 文件名
- 提交文件到本地库 git commit
- 编写注释，完成提交
- 或者也可以git commit –m “注释内容”, 直接带注释提交

## 3.查看文件提交记录

- 执行 git log 文件名 进行查看历史记录
- git log --pretty=oneline 文件名 简易信息查看

## 4.回退历史

- git reset --hard HEAD^ 回退到上一次提交
- git reset --hard HEAD~n 回退n次操作

## 5.版本穿越

- 进行查看历史记录的版本号，**执行 git reflog 文件名**
- 执行 **git reset --hard 版本号**

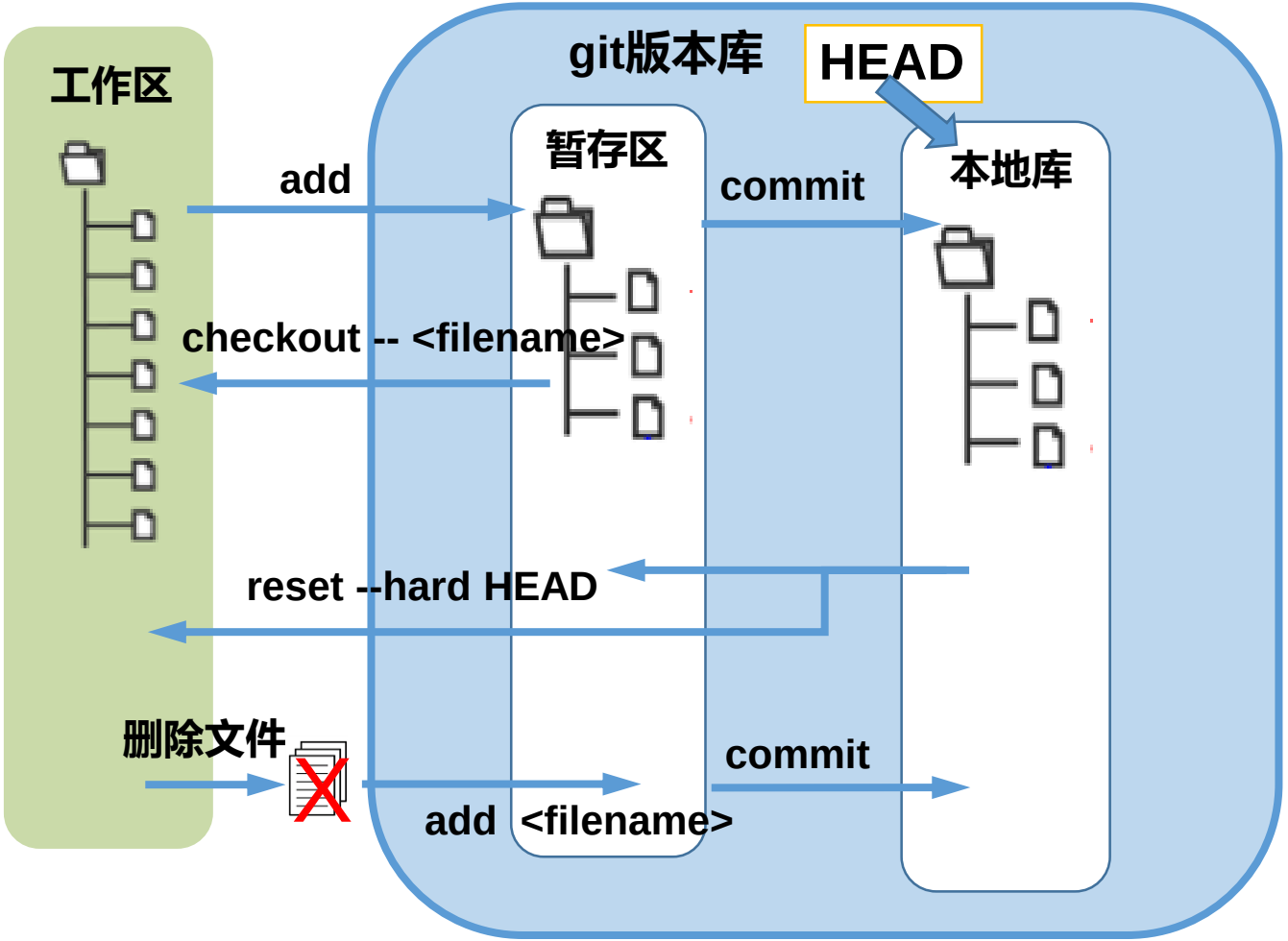
## 6.还原文件

- git checkout -- 文件名

## 7.删除某个文件

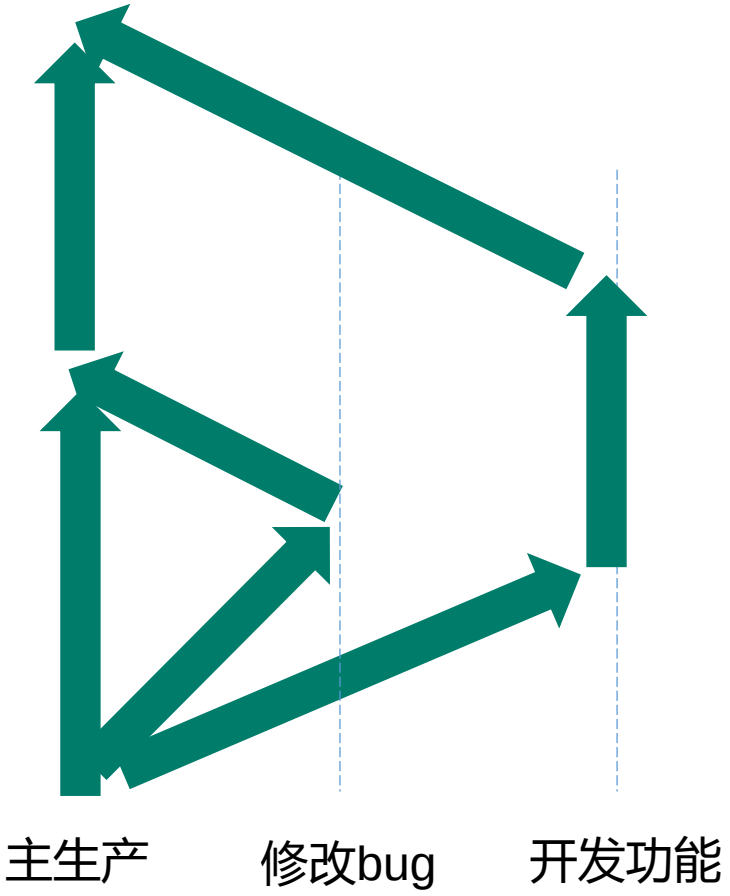
- 先删除文件
- 再git add 再提交

# Gits实战操练—工作区+暂存区+本地库




- **工作区**(Working Directory):就是你电脑本地硬盘目录
- **本地库**(Repository):工作区有个隐藏目录.git,它就是Git的本地版本库
- **暂存区**(stage):一般存放在"git目录"下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

系统上线了，但是产品经理又提了新的需求，评估一下工期要两个月，但是同时系统正在上线运行，时不时还要修改bug，如何管理几个版本？



## 创建分支

- git branch <分支名>
- git branch –v 查看分支

## 切换分支

- git checkout <分支名>
- 一步完成： git checkout –b <分支名>

## 合并分支

- 先切换到主干 git checkout master
- git merge <分支名>

## 删除分支

- 先切换到主干 git checkout master
- git branch -D <分支名>

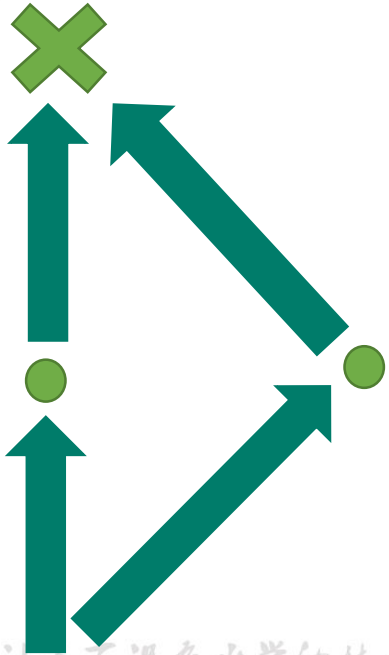
# 冲突

- 冲突一般指同一个文件同一位置的代码，在两种版本合并时版本管理软件无法判断到底应该保留哪个版本，因此会提示该文件发生冲突，需要程序员来手工判断解决冲突。

# 合并时冲突

- 程序合并时发生冲突系统会提示CONFLICT关键字，命令行后缀会进入MERGING状态，表示此时是解决冲突的状态。

```
xiale@xialei MINGW64 /c/ProgramCode/git/gittest (master)
$ git merge dev
Auto-merging gittest.java
CONFLICT (content): Merge conflict in gittest.java
Automatic merge failed; fix conflicts and then commit the result.

xiale@xialei MINGW64 /c/ProgramCode/git/gittest (master|MERGING)
```



# 解决冲突

- 此时通过git diff 可以找到发生冲突的文件及冲突的内容。

```
xiale@xialei MINGW64 /c/ProgramCode/git/gittest (master|MERGING)
$ git diff
diff --cc gittest.java
index 33e47c0,6ac8ac0..0000000
--- a/gittest.java
+++ b/gittest.java
@@@ -4,4 -4,4 +4,8 @@@
  44444444444444444444444
  AAAAAAAAAAAAAAAAAAAAAAA
  BBBBBBBBBBBBBBBBBBBBBBB
++<<<<<<< HEAD
 +YYYYYYYYYYYYYYYYYYYYYY
++=======
+ XXXXXXXXXXXXXXXXXXXXXX
++>>>>>>> dev
```

- 然后修改冲突文件的内容，再次git add <file> 和git commit 提交后，后缀MERGING消失，说明冲突解决完成。

```
xiale@xialei MINGW64 /c/ProgramCode/git/gittest (master|MERGING)
$ git commit -m"merge done"
[master 9163bc3] merge done

xiale@xialei MINGW64 /c/ProgramCode/git/gittest (master)
$
```

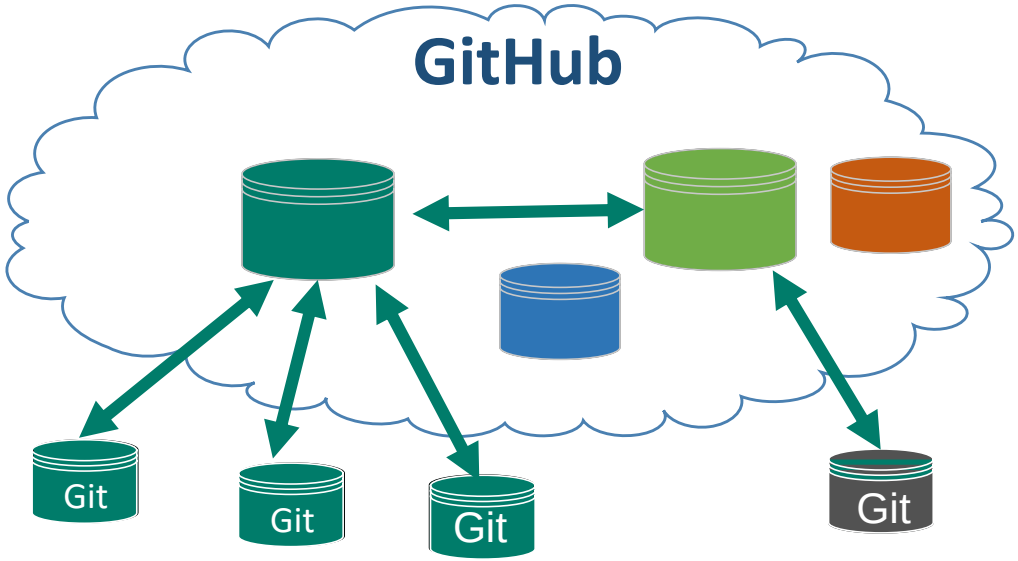
# 目录



尚硅谷

# 是什么

- GitHub是一个Git项目托管网站,主要提供基于Git的版本托管服务



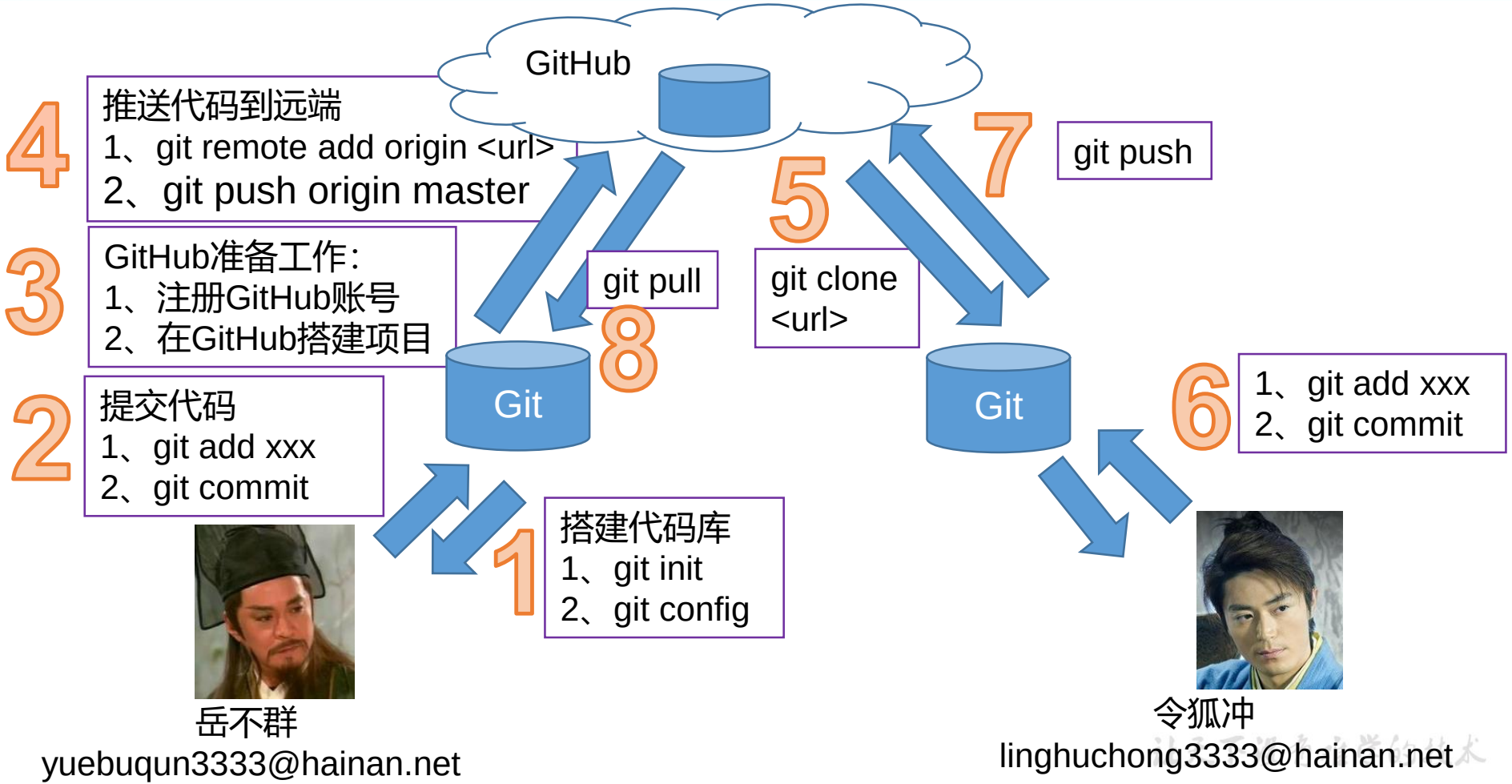
## 网址

https://github.com/

## 注册账号的注意事项

- 不要使用163的邮箱，有可能收不到验证邮件。
- 较长时间不使用有可能被Github冻结账号。请登录其客服页面https://github.com/contact，填写账号恢复申请。

# Github简介与实操—实操



## 增加远程地址

- **git remote add <远端代号> <远端地址>。**
- **<远端代号>**是指远程链接的代号，一般直接用**origin**作代号，也可以自定义。
- **<远端地址>**默认远程链接的url
- 例：**git remote add origin https://github.com/user111/Helloworld.git**

## 推送到远程库

- **git push <远端代号> <本地分支名称>。**
- **<远端代号>**是指远程链接的代号。
- **<分支名称>**是指要提交的分支名字，比如master。
- 例：**git push origin master**

## 从GitHub上克隆一个项目

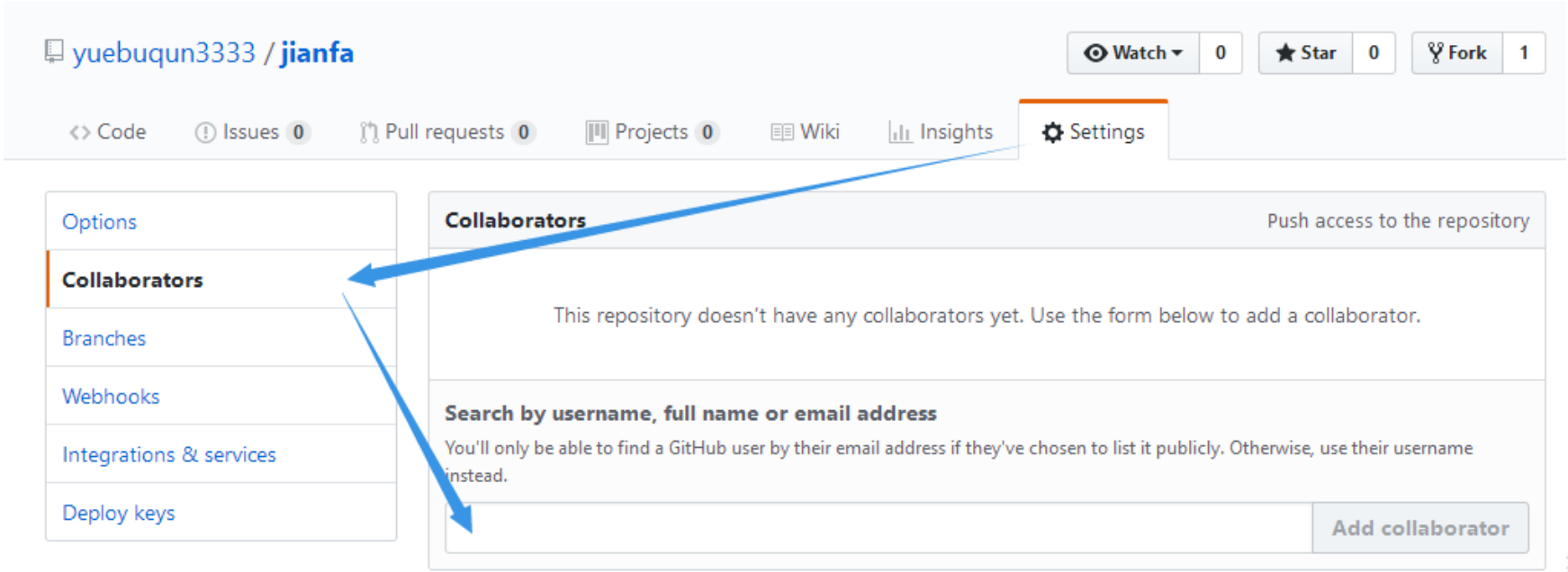
- **git clone <远端地址> <新项目目录名>。**
- **<远端地址>** 是指远程链接的地址。
- **<项目目录名>** 是指为克隆的项目在本地新建的目录名称，可以不填，默认是GitHub的项目名。
- 命令执行完后，会自动为这个远端地址建一个名为**origin**的代号。
- 例 **git clone https://github.com/user111/Helloworld.git hello_world**

## 从GitHub更新项目

- **git pull <远端代号> <远端分支名>。**
- **<远端代号>** 是指远程链接的代号。
- **<远端分支名>**是指远端的分支名称，如master。
- 例 **git pull origin master**

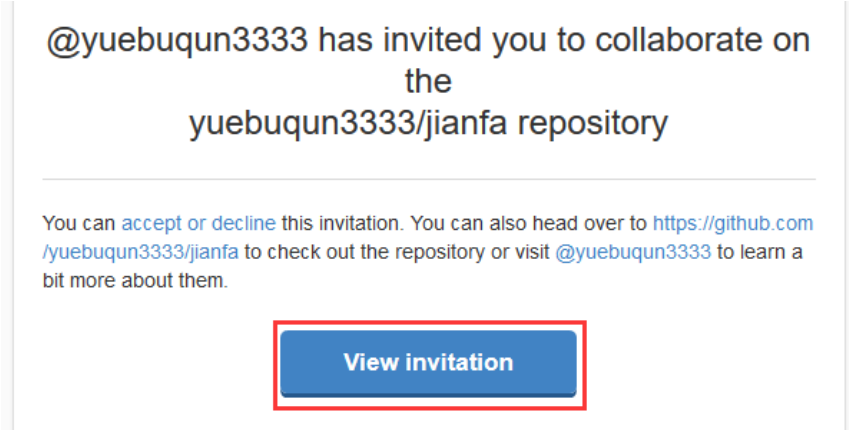
# ➢ 以上对项目的操作方式，必须是项目的创建者或者合作伙伴。

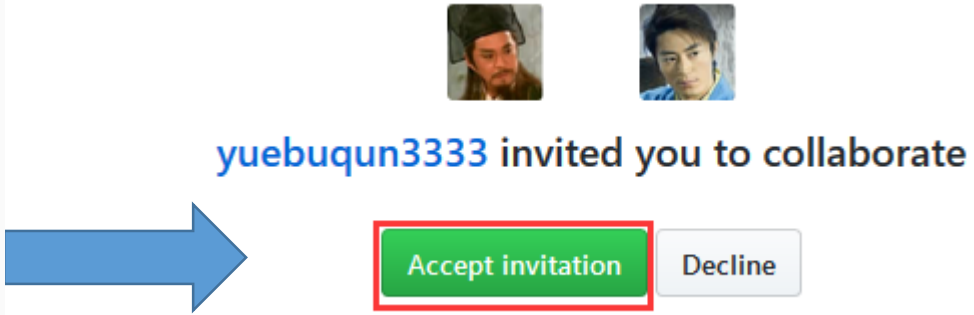
- 合作伙伴添加方式如下图: 在项目中点击settings页签，然后点击Collaborators,然后在文本框中搜索合作伙伴的邮箱或者账号。点击添加。
- 添加后GitHub会给合作伙伴对应的邮箱发一封，邀请邮件。



## ➢ 邀请你的同事成为合作伙伴

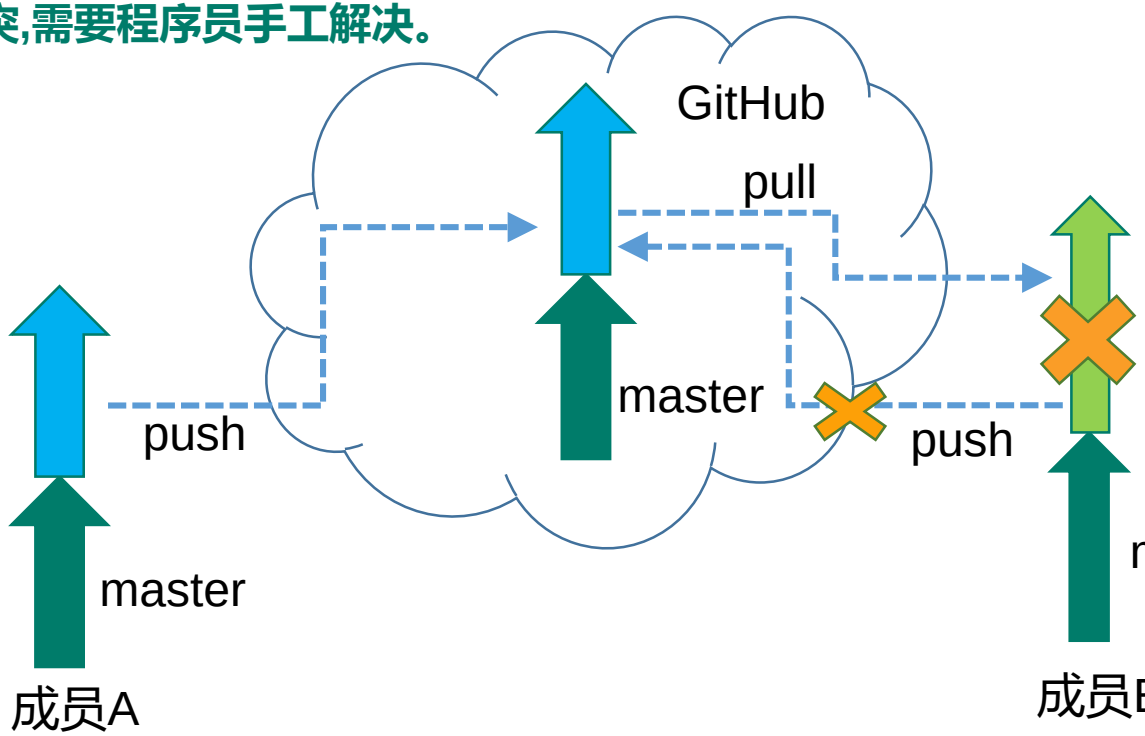
- 合作伙伴会收到邀请邮件。点击View invitation 按钮后会跳转至GitHub页面，让合作伙伴选择，是否接受邀请。
- 点击接受后，则合伙伙伴正式加入项目，获得直接提交代码的权限。



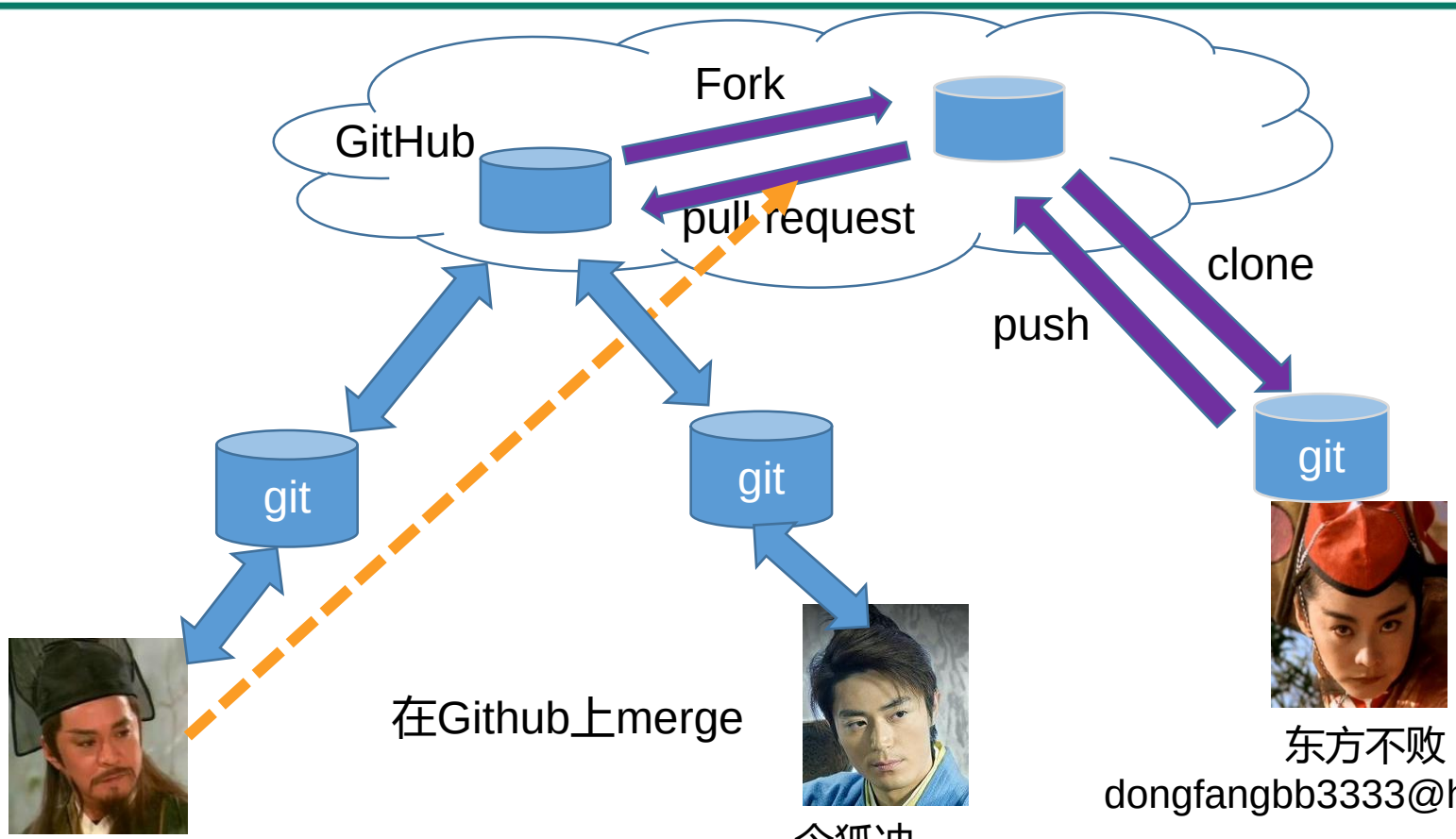
# 协作冲突

- 在上传或同步代码时，由于你和他人都改了同一文件的同一位置的代码，版本管理软件无法判断究竟以谁为准，就会报告冲突,需要程序员手工解决。




**解决冲突**
**三板斧：**
**1、修改合并**
**2、git add**
**3、git commit**

Conflict!




岳不群
yuebuqun3333@hainan.net

令狐冲
linghuchong3333@hainan.net

东方不败
dongfangbb3333@hainan.net

# 番外篇:每次输密码很烦篇

# 两种模式：https VS ssh

- ssh模式比https模式的一个重要好处就是，每次push、pull、fetch等操作时，不用重复填写遍用户名密码。
- 前提是你必须是这个项目的拥有者或者合作者，且配好了ssh key。

# 如何配置SSH key

- 步骤1：检查你的电脑上是否已经生成了SSH Key 在git bash下执行如下命令

```
xiale@xialei MINGW64 ~/.ssh
$ cd ~

xiale@xialei MINGW64 ~
$ cd .ssh
bash: cd: .ssh: No such file or directory
```

如果已经有这个文件包 删除就行了

- 步骤2：创建SSH Key： ssh-keygen -t rsa -C XXXXXX@hainan.net

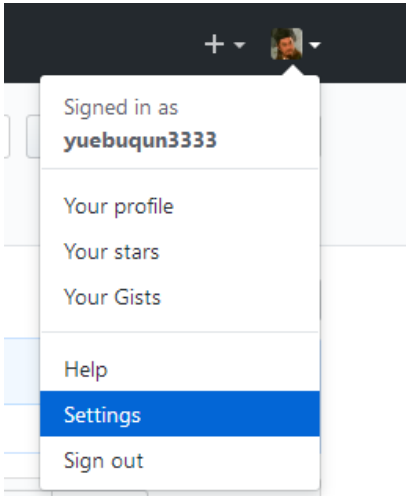
成功的话会在~/下生成.ssh文件夹，进去，打开id_rsa.pub，复制里面的key。

```
xiale@xialei MINGW64 ~
$ ssh-keygen -t rsa -C linghuchong3333@hainan.net
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/xiale/.ssh/id_rsa): 回车
Created directory '/c/Users/xiale/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:                                    回车x2
Your identification has been saved in /c/Users/xiale/.ssh/id_rsa.
Your public key has been saved in /c/Users/xiale/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LHuyJQ6VBCfg1Gs3b7rAGllUvGBicYrlqS7eI6IDNGg linghuchong3333@hainan.net
The key's randomart image is:
+---[RSA 2048]----+
|   =++.o         |
|  *o++=.         |
|o.=o.o..         |
|.E .o.+o         |
|+ ....+oS        |
|o  + . oo        |
|o.o + +oo        |
|=..+ +.*         |
|+oo.. +.         |
+----[SHA256]-----+
```

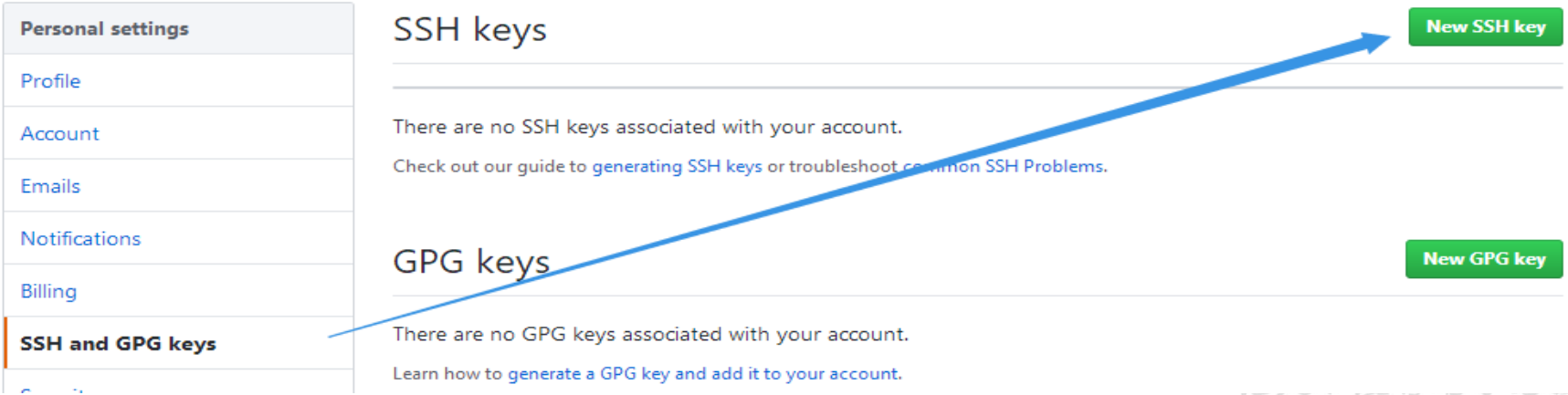
- **步骤3：进入.ssh文件包，打印id_rsa.pub的内容，复制全部内容**

```
xiale@xialei MINGW64 ~
$ cd .ssh

xiale@xialei MINGW64 ~/.ssh
$ ll
total 5
-rw-r--r-- 1 xiale 197609 1679 11月 25 10:32 id_rsa
-rw-r--r-- 1 xiale 197609  408 11月 25 10:32 id_rsa.pub

xiale@xialei MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDbeRxctb6AWN6u0EkWQ6iSGL5bxj5YpapPg7Oxshv+
ZLM8Vm6nwZlY5hQDSZmD4ivq3Y2mI5FFbCcAV4sb4rJF8NCRkh5HsoyKcKSIR8gjMQISsLtx9SJLXQoj
XBj/LcNH9JaoJDrCFJVmXS61SOarnwk+JOtVZhbTHe+rkNJ7PNpmCT1TxeFBMlgbPIeUUKzkX81YkPIB
N7U54rWGIH1ZfwnENexuEavDKV8F7qn2JUr1Cz/jr/ZreNPjBxsBYiYvnoUPyeVXOBwYaPPI/wNdzndF
dO7rfMvhqfawJHOxNp8tpxz1FkxhlxfBqJhMV21ZrG7R6O78CTWDVTJZf/+T linghuchong3333@hai
nan.net
```

## 步骤4.登录Github后，右上角点击setting




## 步骤5.在左侧菜单中选择SSH and GPG keys，在右边点击New SSH key



## 步骤6 :Title随便写，Key 把之前id_rsa.pub的内容复制进去，点击Add SSH key，设置ssh key 完成。

There are no SSH keys associated with your account.

**Title**

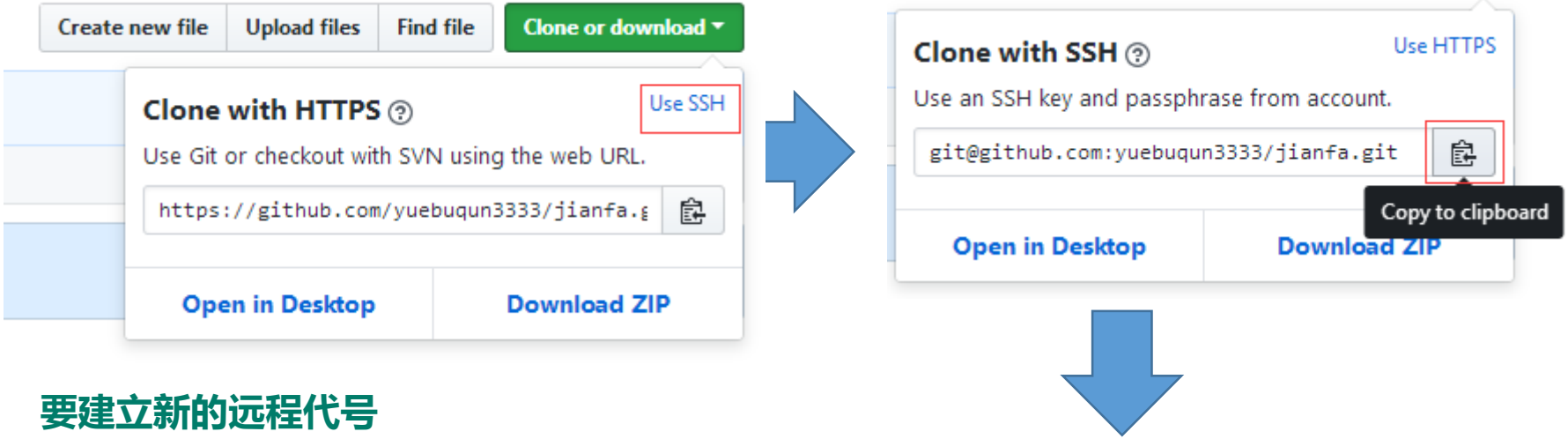
keys

**Key**

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDAknsuP0dMBJNeHP2dTlUo1Gjp5nTiAD6HdJ4qCEYmxL1ja4UB/6ww
dKmezr1615/9YbngGQb9AgcUx3uEDj0LHWUWS0Map5Ya3WiLQ+nU03u+/AOx/PSEz6bwF1+Kp1n50Bs4QMs
QStXFoAlDYR1oJAV+tG56z0qDOf394snvExjmGON1g+svlm0vOgf2pInIOBdXDNiCijaD+nVkkGCNBHJ6pb/DWx
qCiYB/JdrVe2ITaGAFNUyvFqqq1UTGBhhTD6lfgDOzmgvfhdO/HCPe9zuchuZC6bKlZTrqVMBvbyYuD64CWS8NU
5OPssigw6/ICaE5t/ier+tWSZ9cHm05 yuebuqun3333@hainan.net
```

**Add SSH key**

## 测试连通性：要改用ssh连接




## 要建立新的远程代号

git remote add originssh git@github.com:yuebuqun3333/jianfa.git

以后再提交代码的时候就不用输入密码了（第一次使用会要求输入个 yes）

git push originssh master

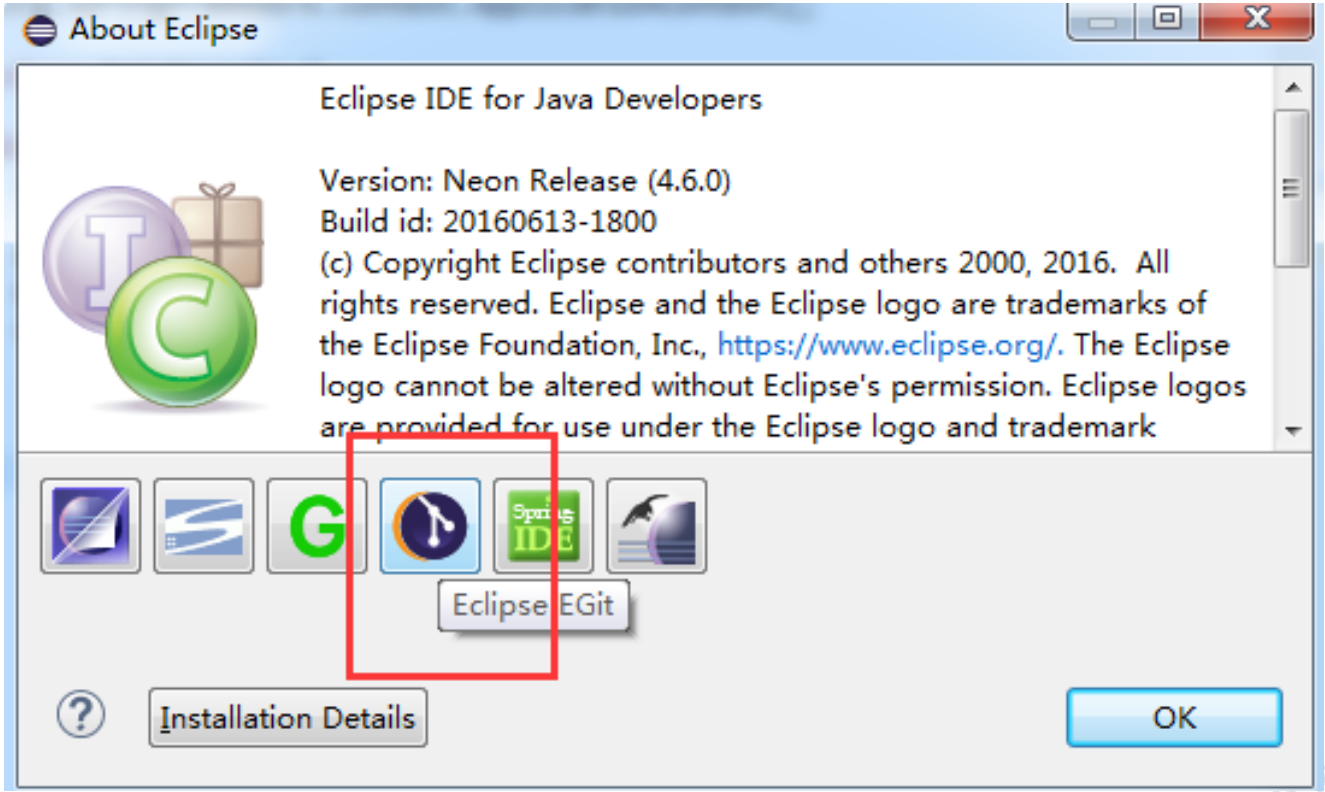
# 目录



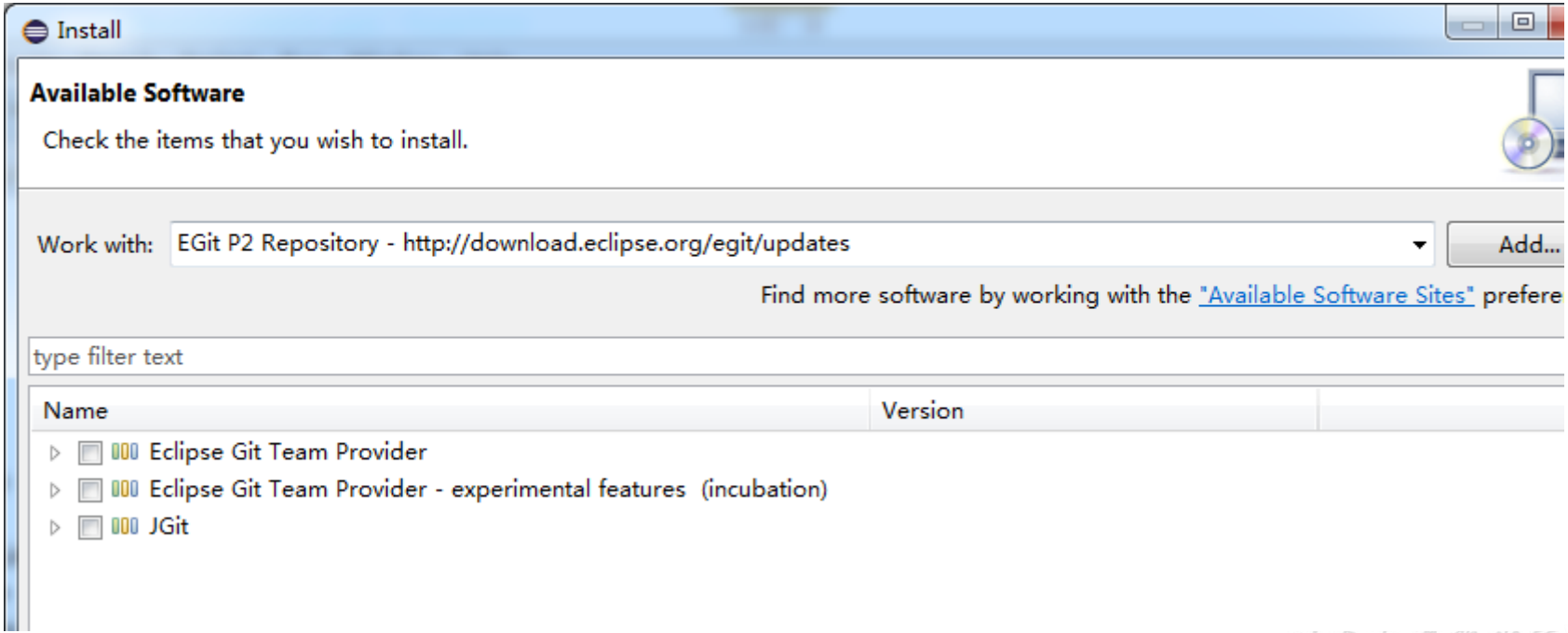
尚硅谷

现在的Eclipse下载后一般都提供Git插件了。

在Eclipse的Help中，点击About Eclipse,查看是否有该插件

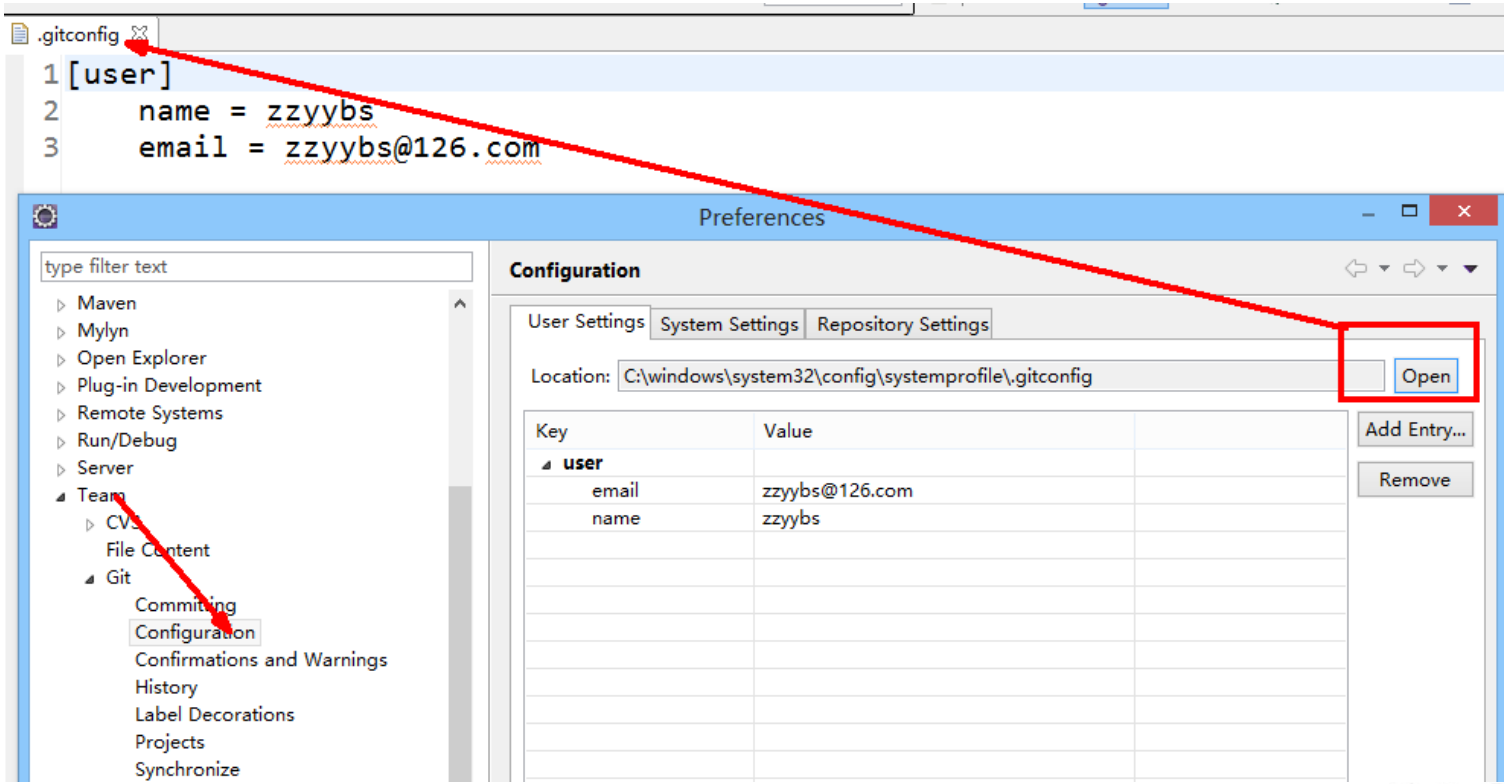


如果没有，菜单栏Help -> Install New Software...，在Work with中输入http://download.eclipse.org/egit/updates，勾选Eclipse Git Team Provider和JGit，点击Next，进入安装，重启Eclipse，安装完成。

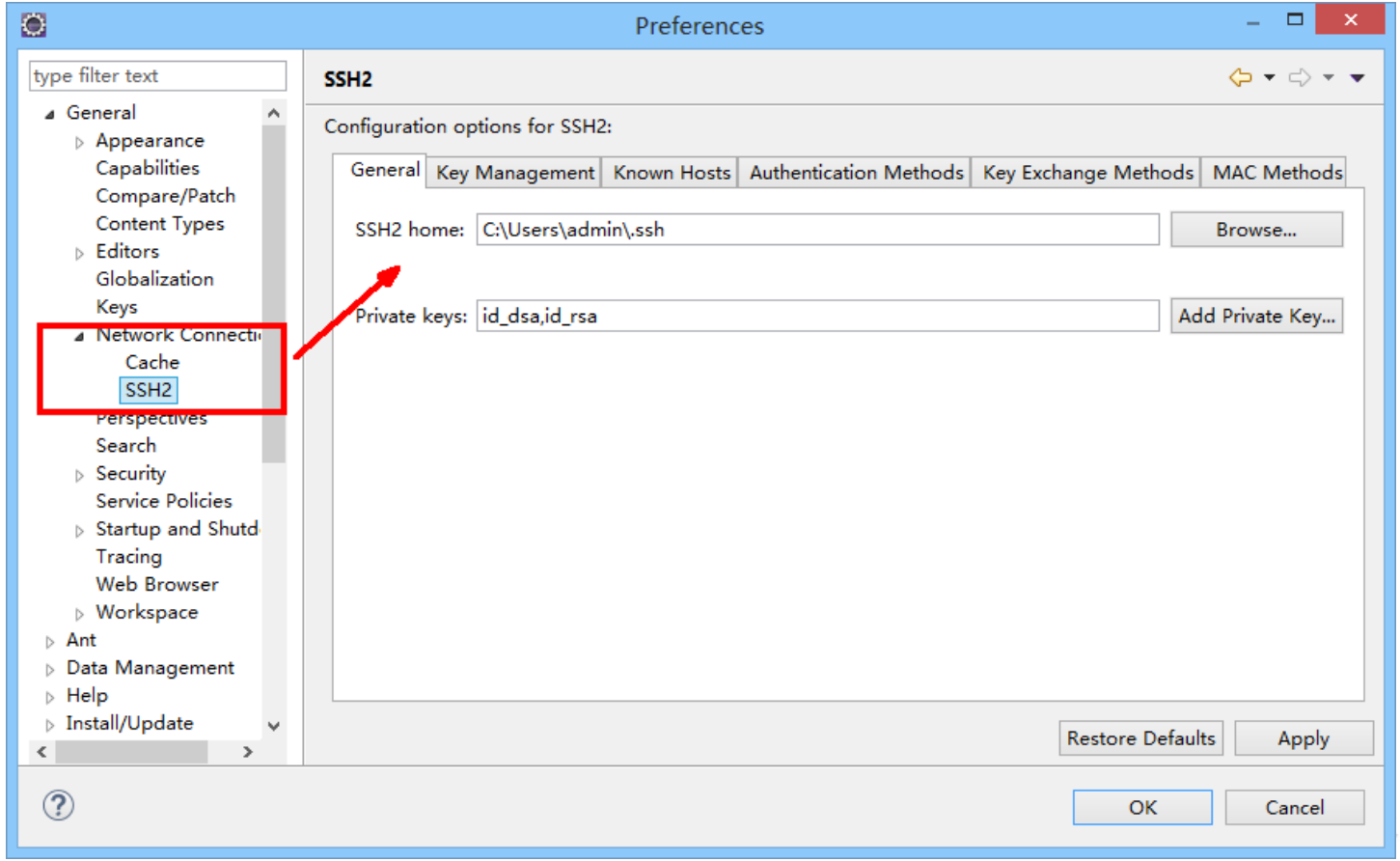


## 配置用户名、email

### 1 Windows-->Preferences-->Team-->Git-->Configuration



## 检查SSH key



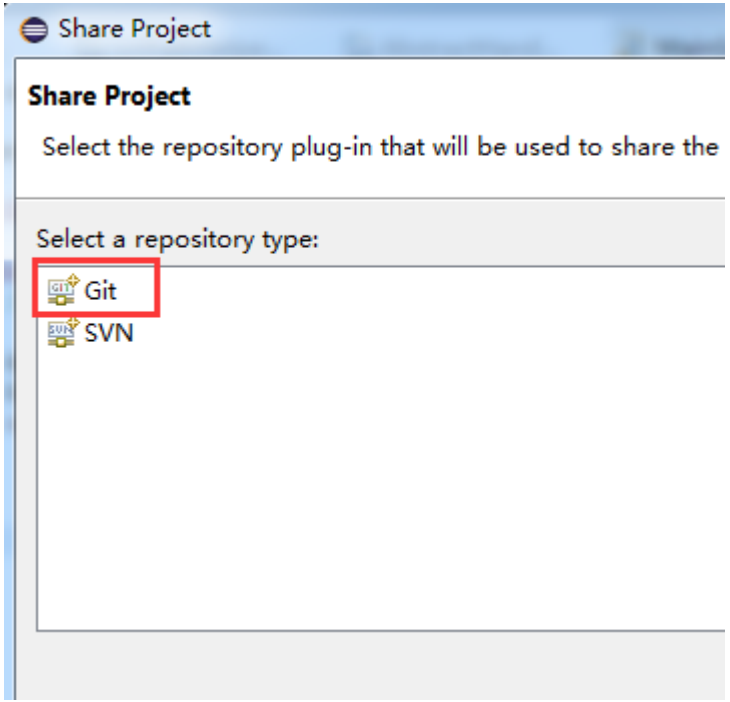
**新建一个web项目，此时它只是一个普通的javaWeb项目，未纳入Git管理**

**变为Git管理的项目**

1 选中工程鼠标右键

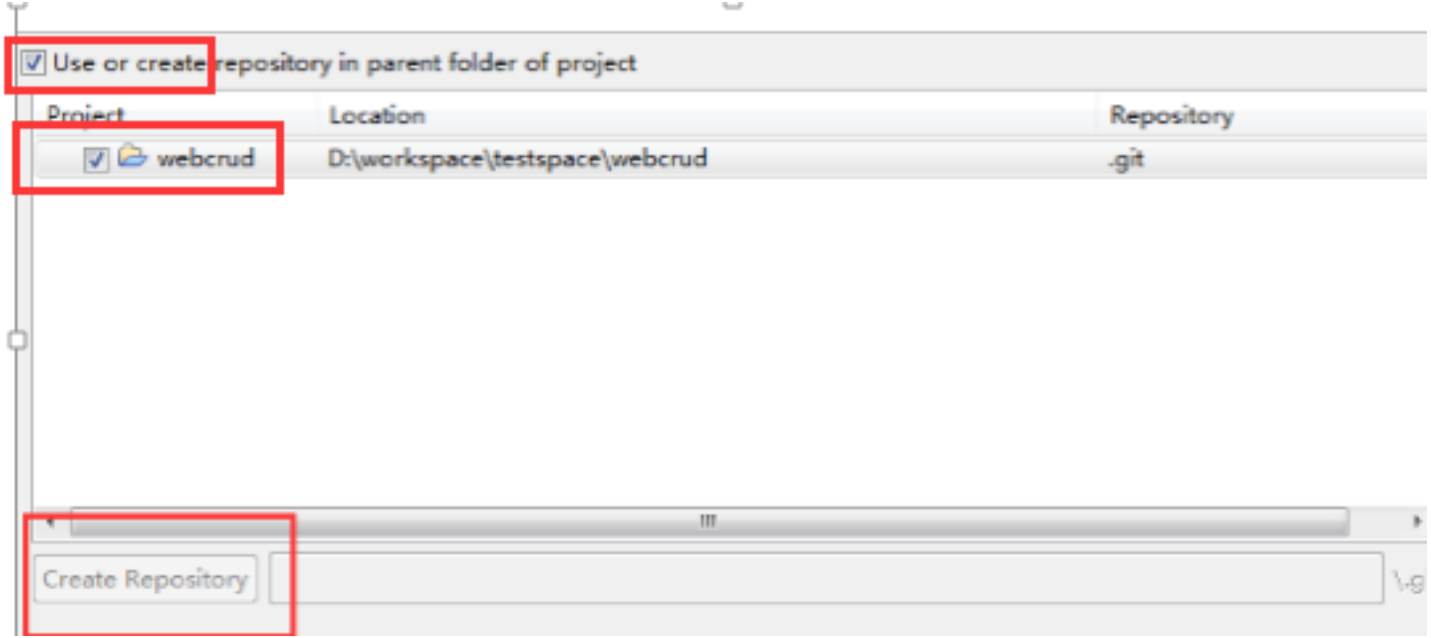
2 Team

3 Share Project……

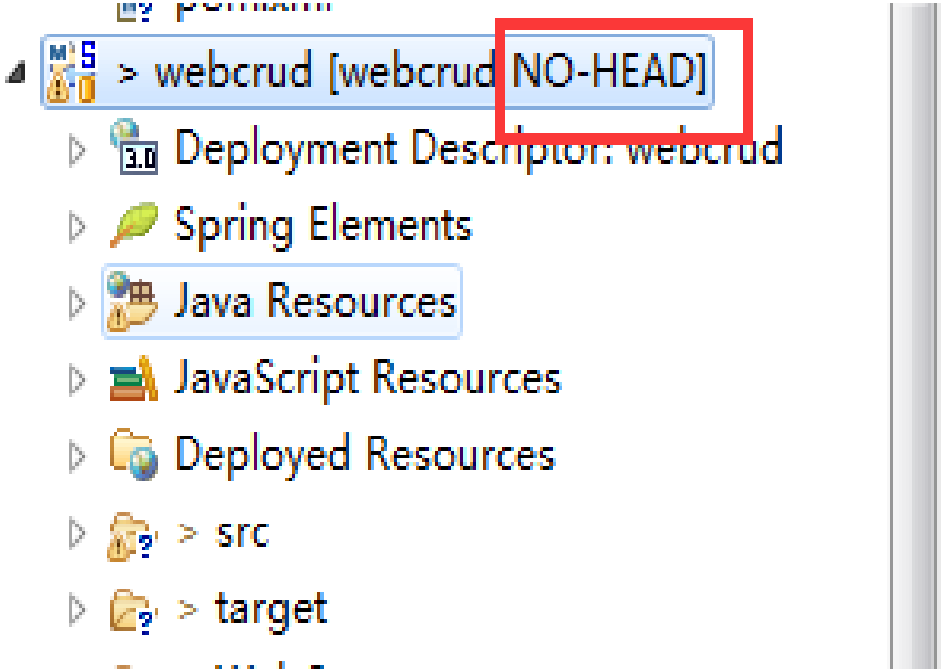


勾选上方Use or create repository ....

勾中项目，再点击下方create Repository，再点击Finish

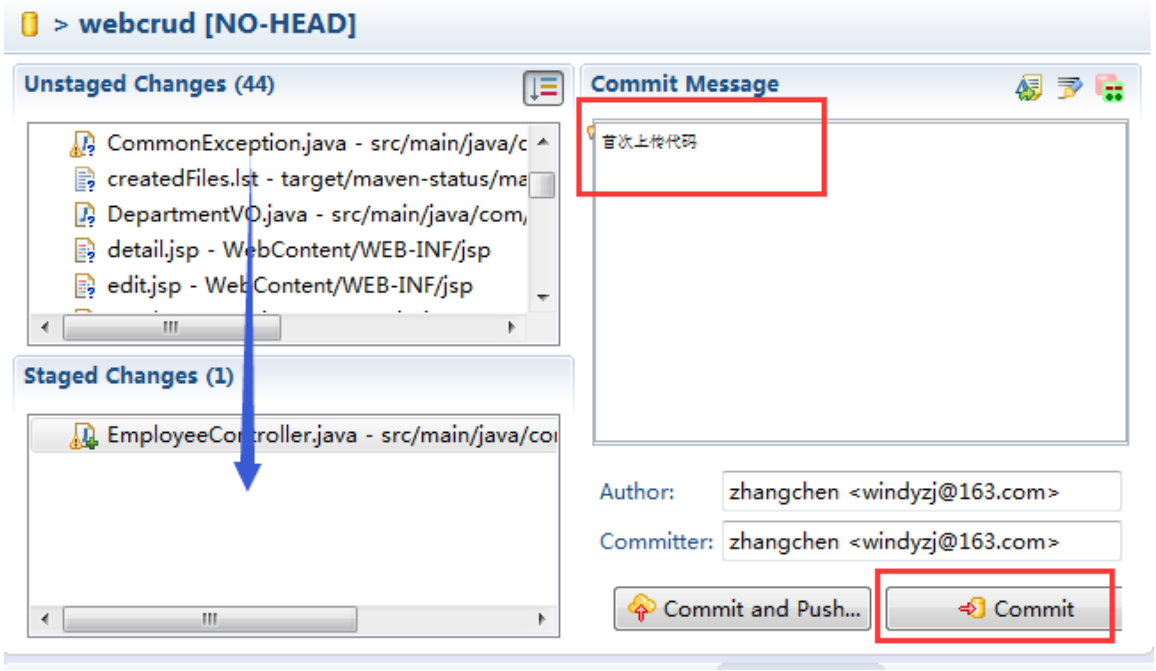


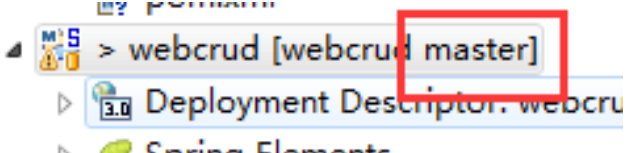
初始化完成后，项目后缀会显示NO-HEAD，表示版本库已建立，但是还没有任何提交文件，因此没有主干分支。



# Egit操作—操作

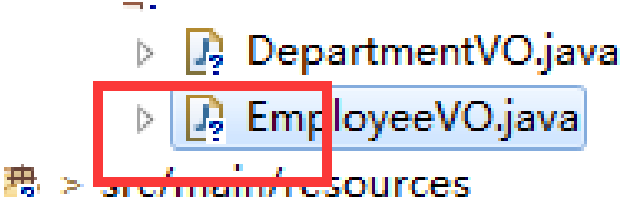
在项目上右键Team>>commit,出现如下对话框，将左上列出的文件列表，拖入至左下方，实现git add 功能。右边填写提交备注，则可点击右下角的Commit按钮。完成一次本地库的提交，可以看到左边的项目名称后缀多了个master。



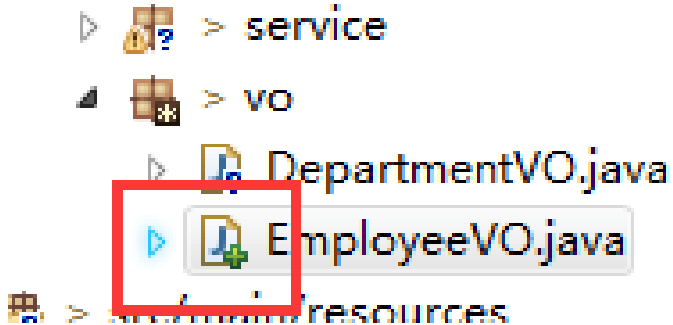



## 新建文件、Add、Commit……

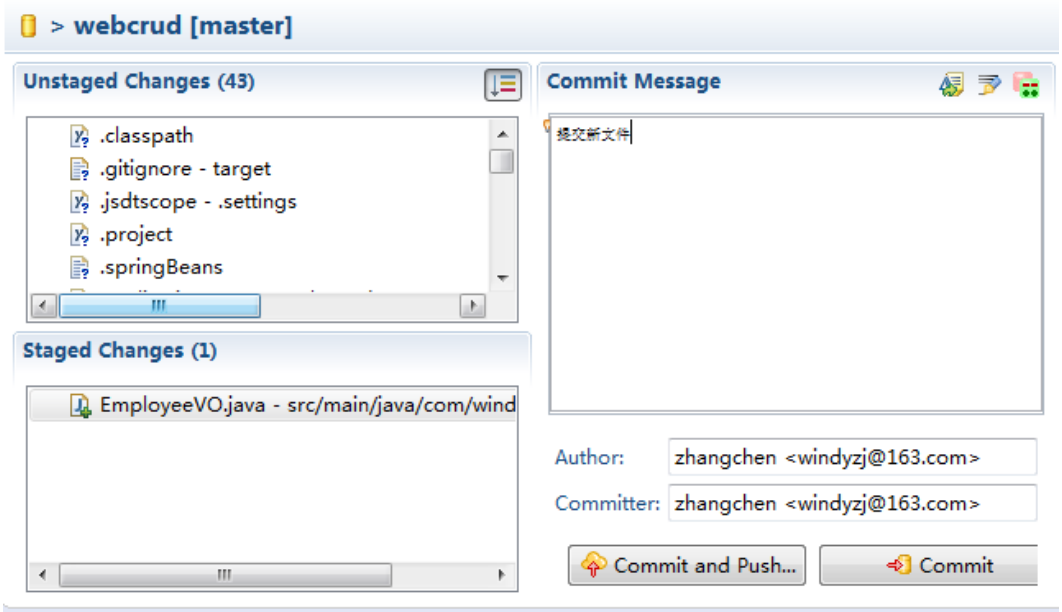
1、新建一个文件，可以看到图标依然是问号，处于untracked状态，即git没有对此文件进行监控



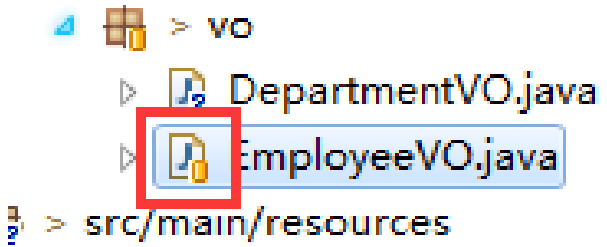

2 、通过Team -> Add to index可以将文件加入git索引，进行版本监控;可以看到图标显示也有了变化（EGIT中只要Commit就可以默认将untracked的文件添加到索引再提交更新，不需要分开操作）



## 3、commit…提交到本地库

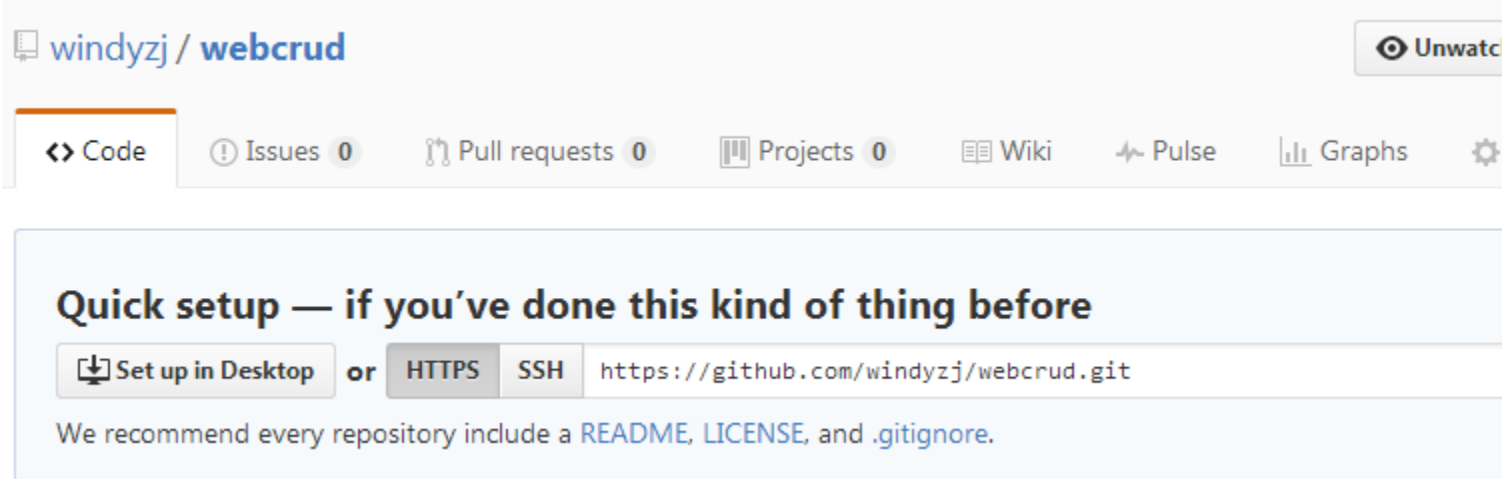



## 4、commit…提交完成后，图标发生变化。

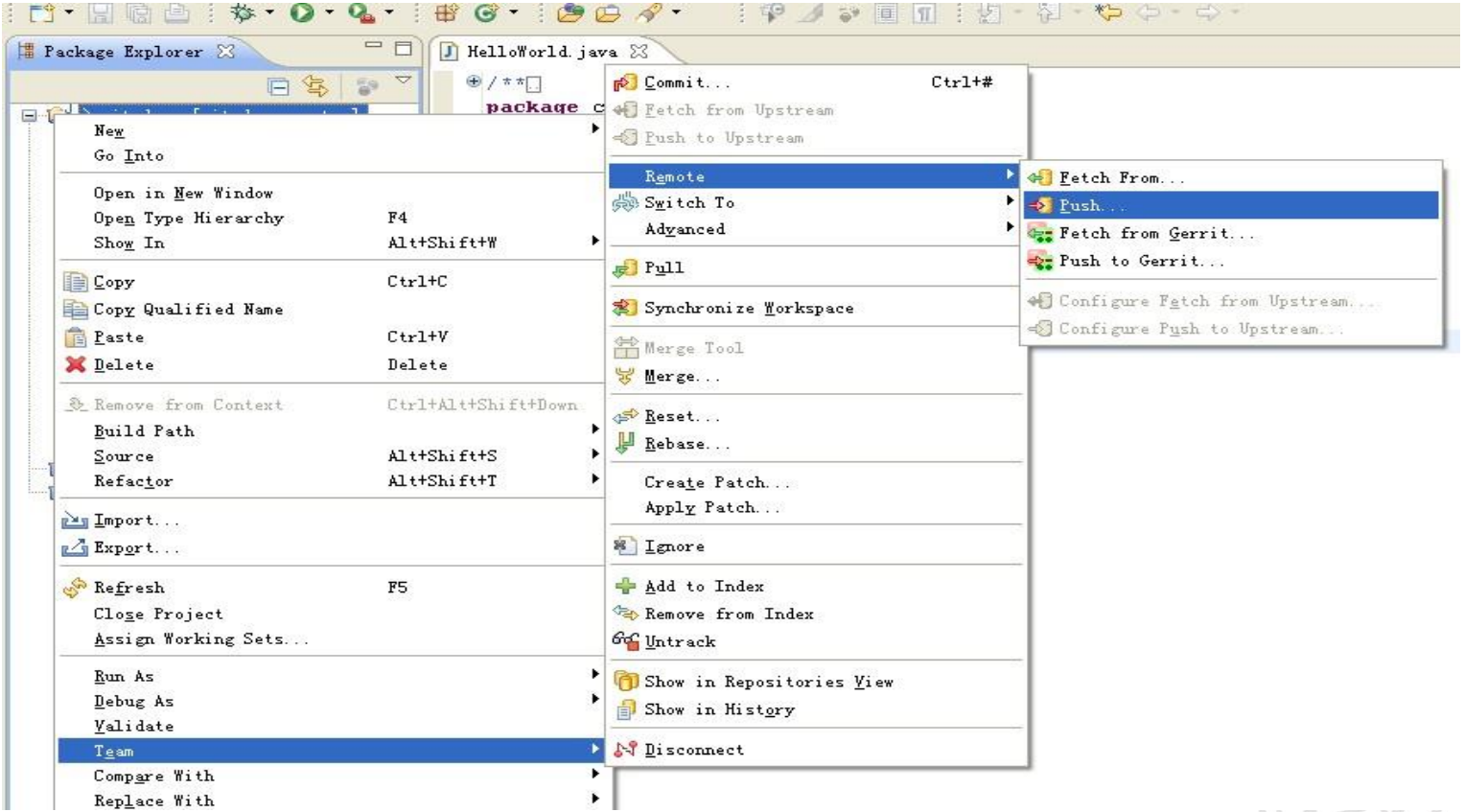


# 用与远程库的操作交互

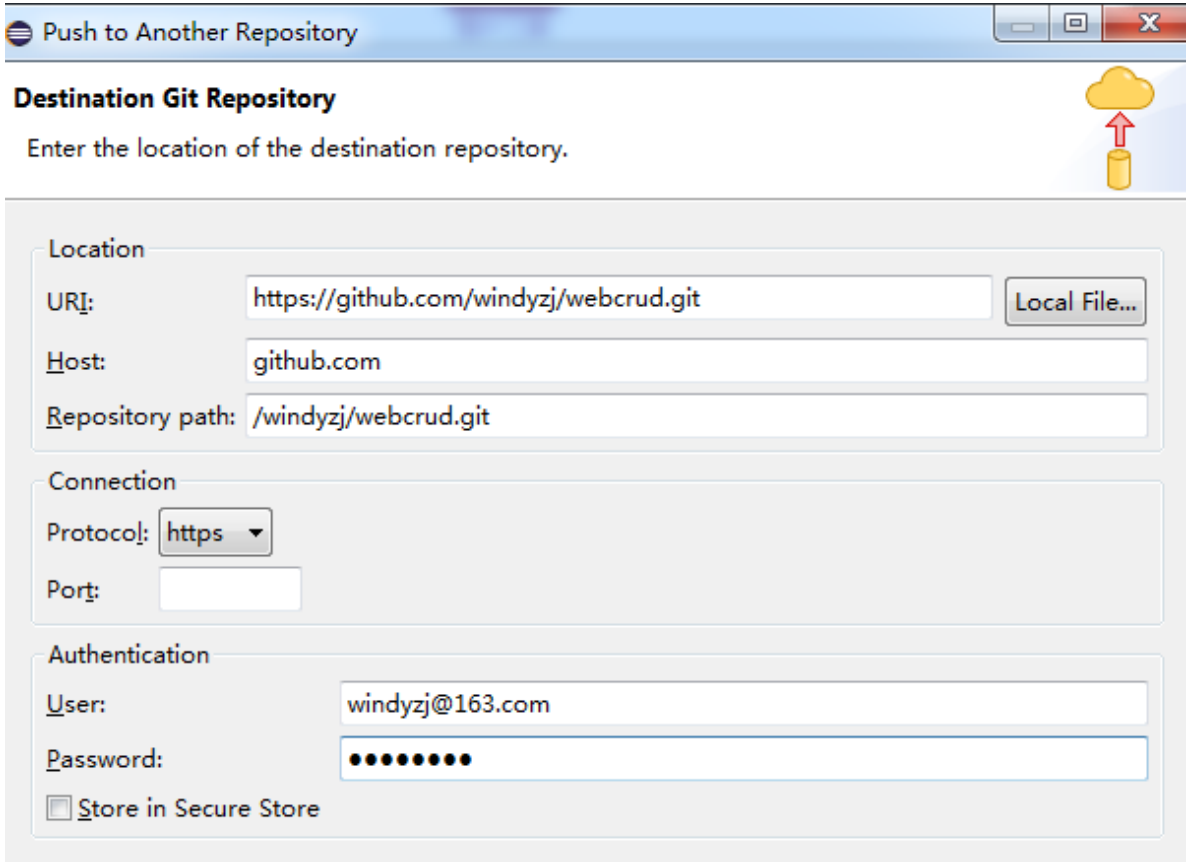
## 1、 Github上新建一个同名的空仓库



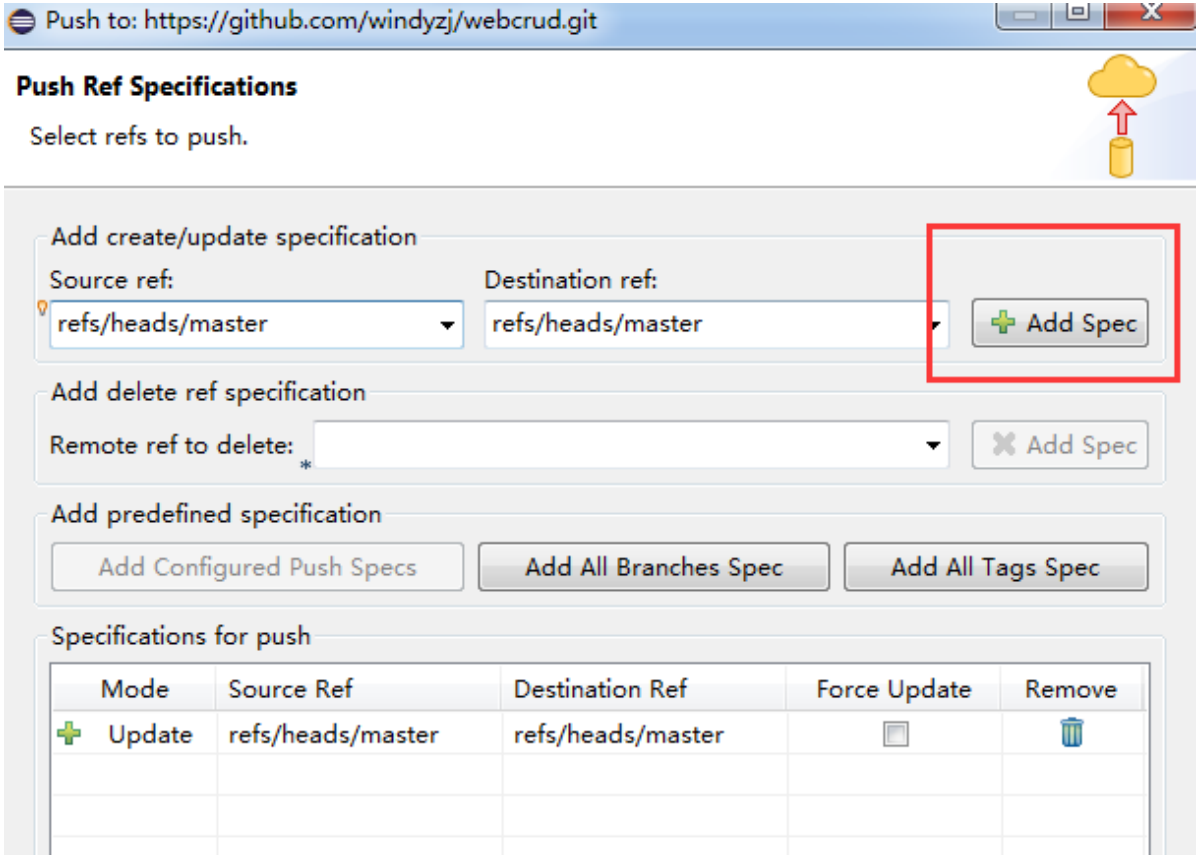
## 2、 本地可以Push到Remote



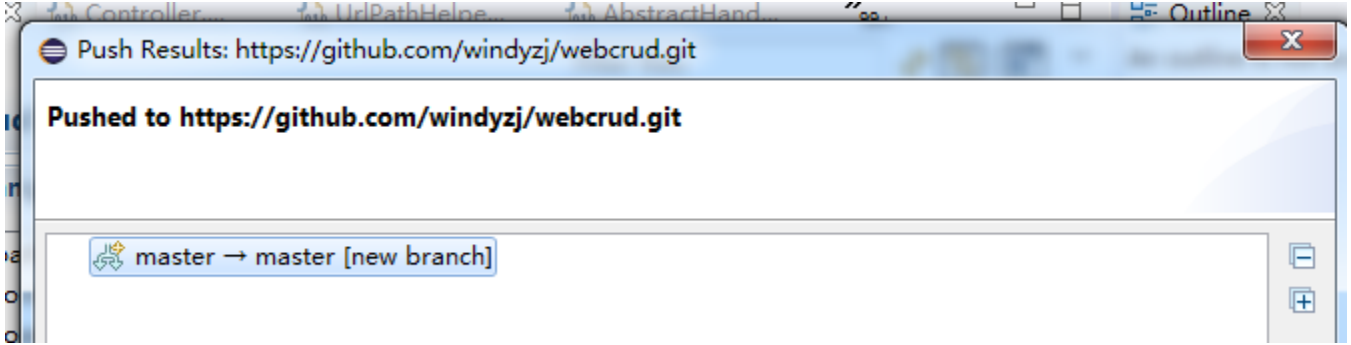
3、 填写push信息，将远程仓库的地址复制到URI中，然后在下方填写Github的用户名密码。



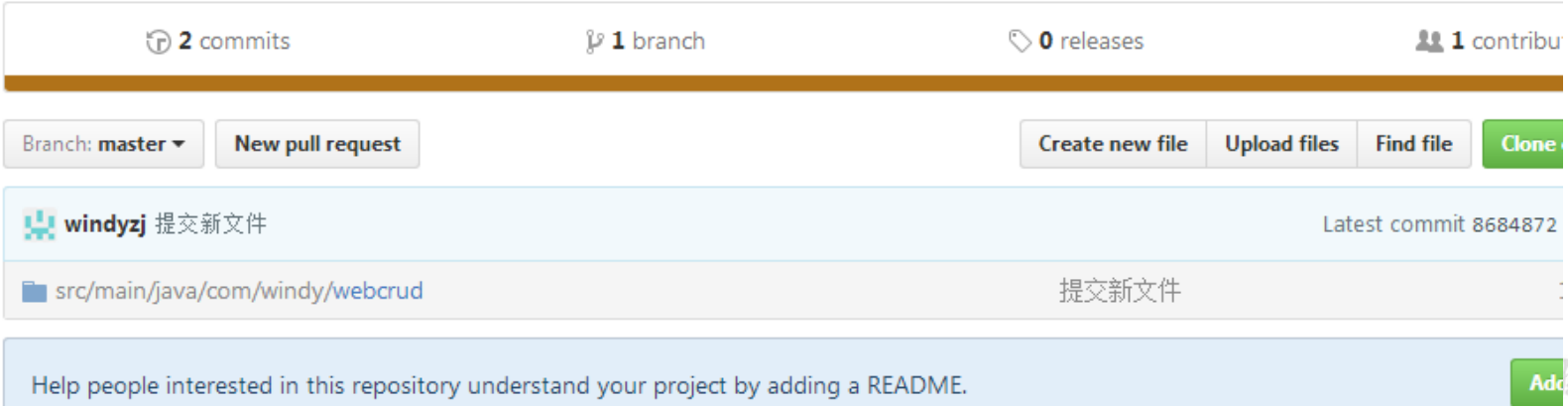
## 4、指定push的本地分支和远程分支



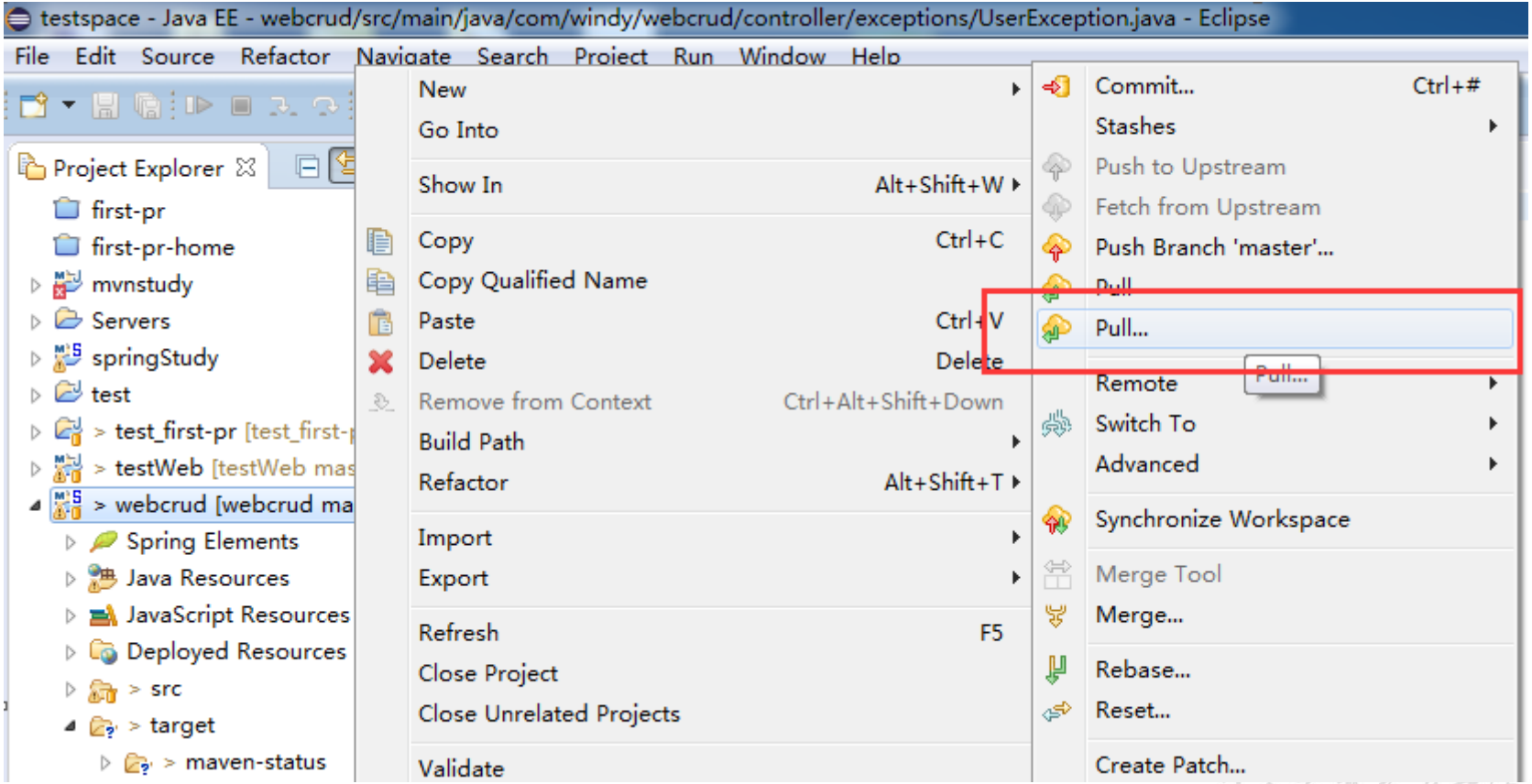
## 5、 Eclispe最后一步点击Finish 即可。



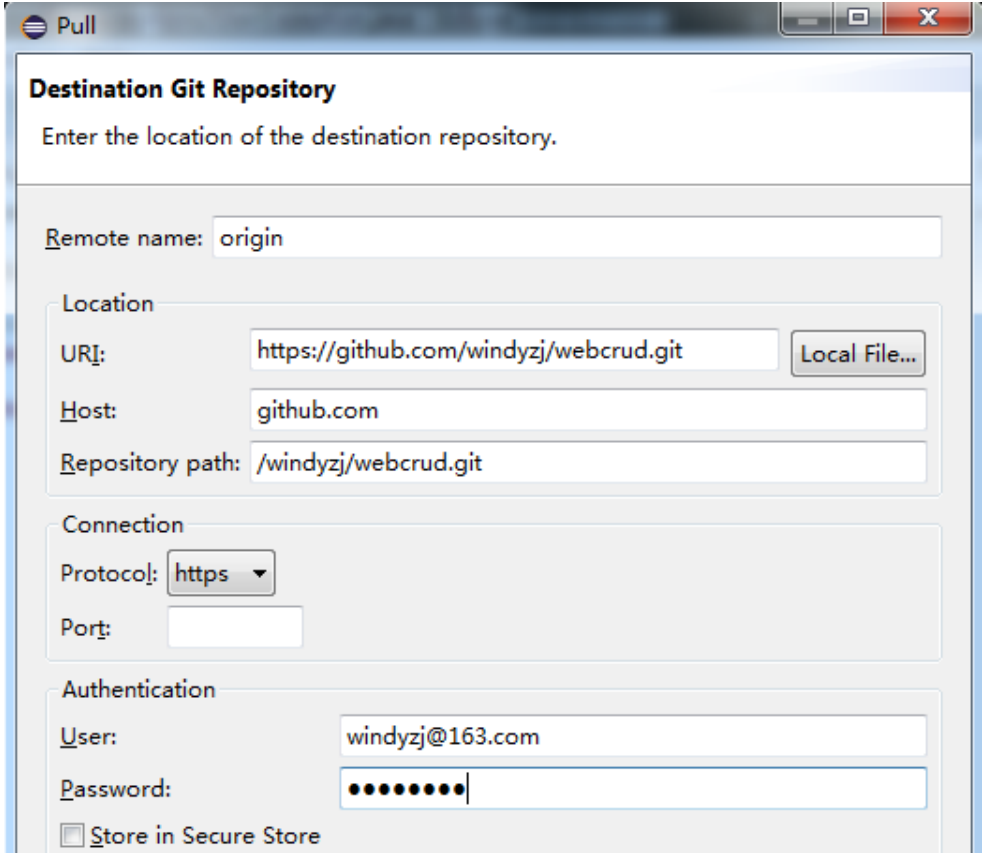

## 6、 Eclipse上传成功后，可以去GitHub上查看上传的代码



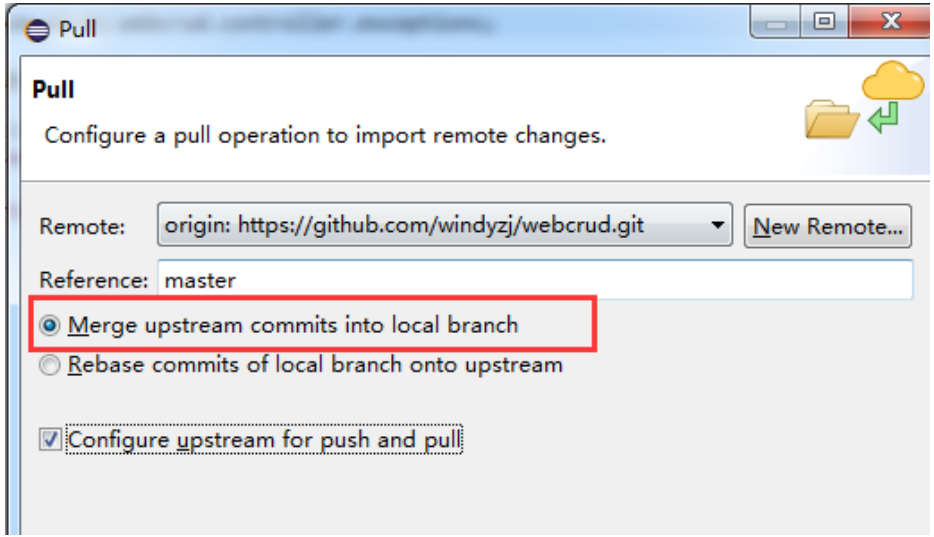
## 7、Github服务器上面更新了，pull到本地



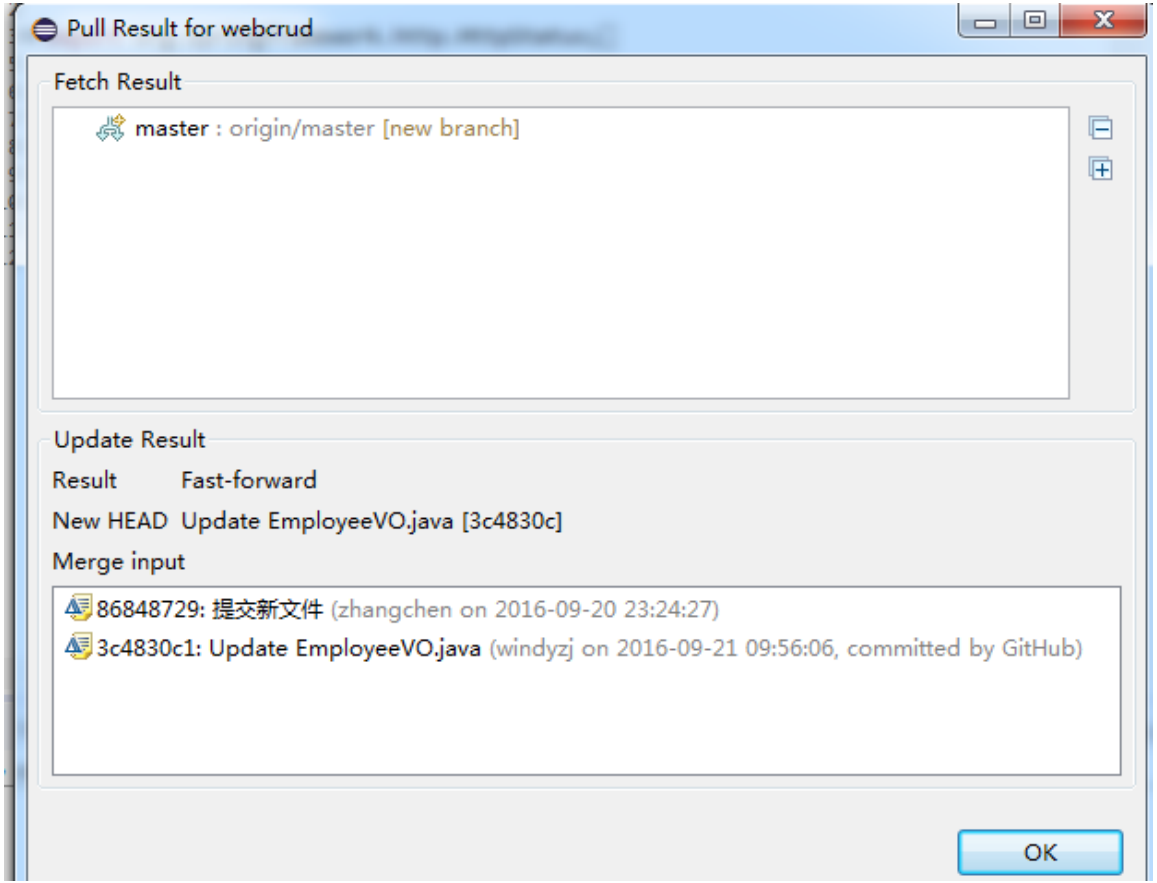
## 8、同push类似，这里要填写远程仓库地址，和登录用户名密码



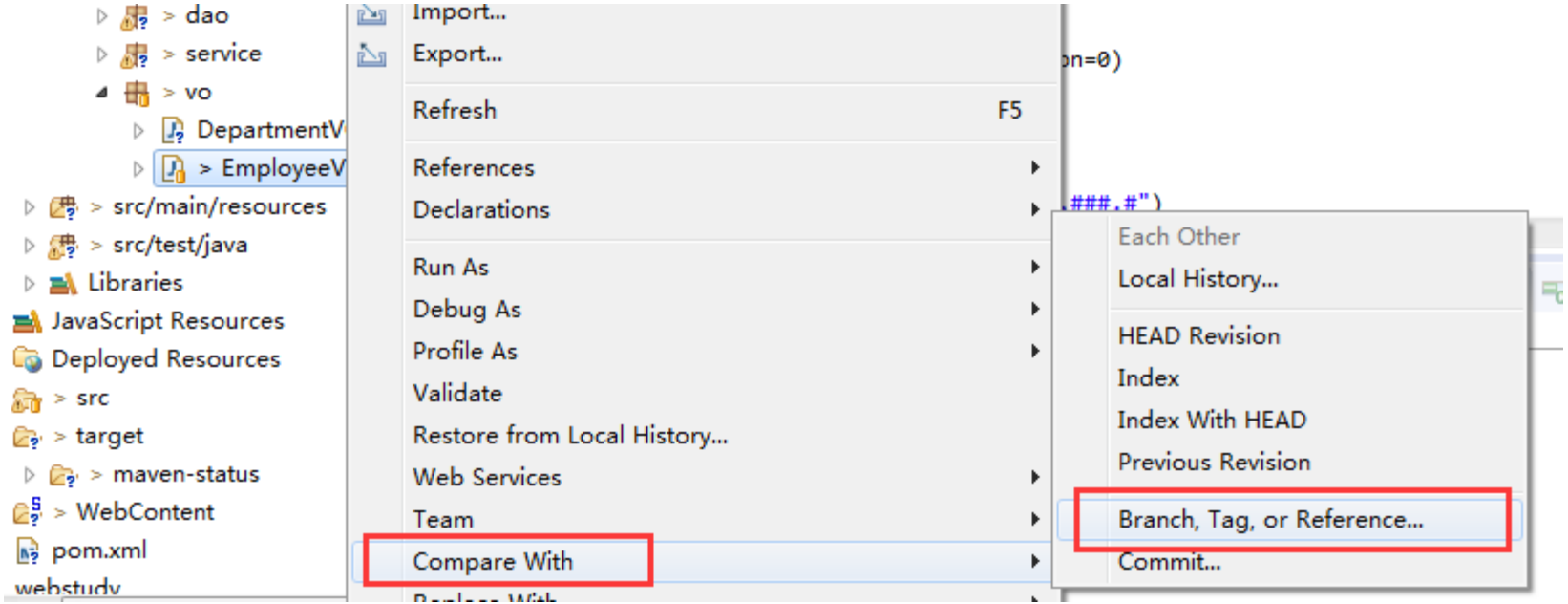
- 这里如果Remote的下拉列表里没带过来，请在New Remote里再填一遍。
- 单选项选择Merge模式
- 下方勾选Configue upsteam for push and pull 后，以后可以不同每次pull、push 都这么配置了，以后以此次的配置为默认值。
- 点击Finish



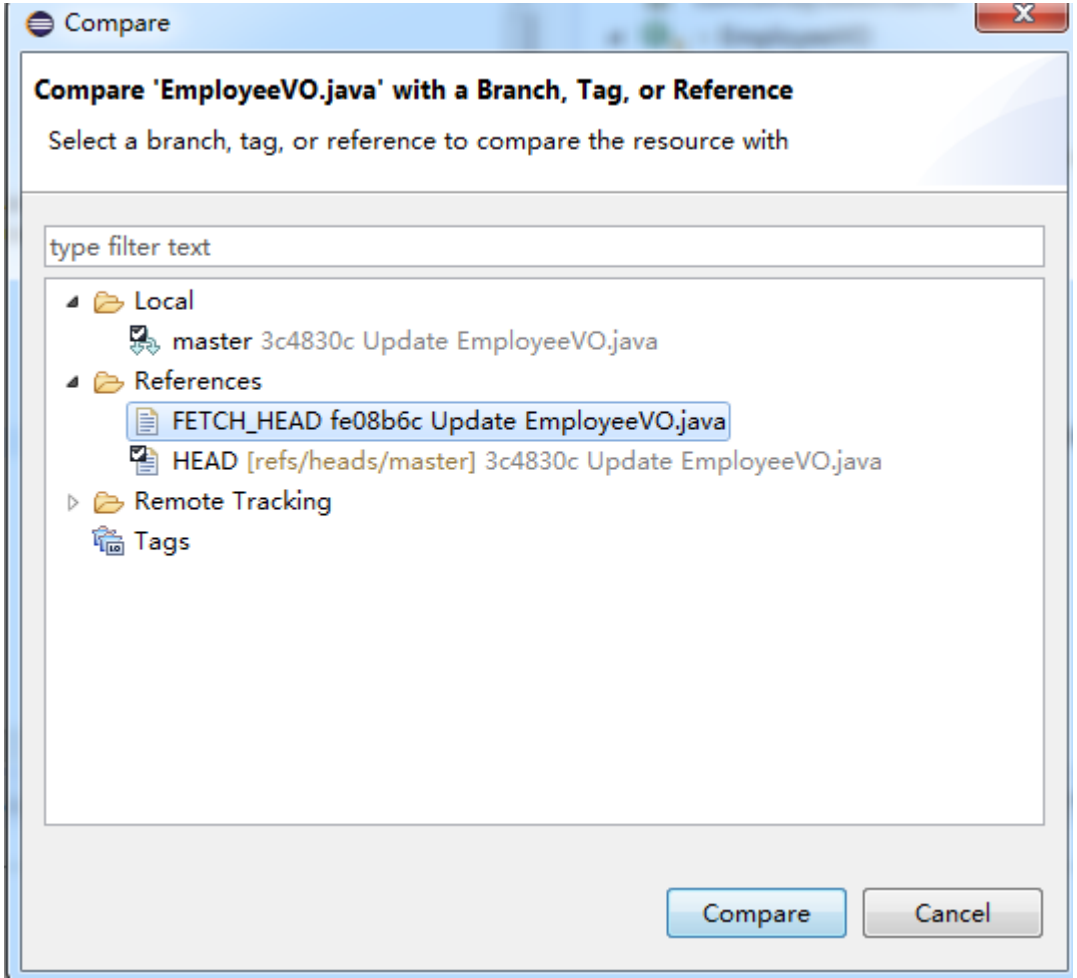
**更新完成后会弹出提示，然后去查看一下代码。**



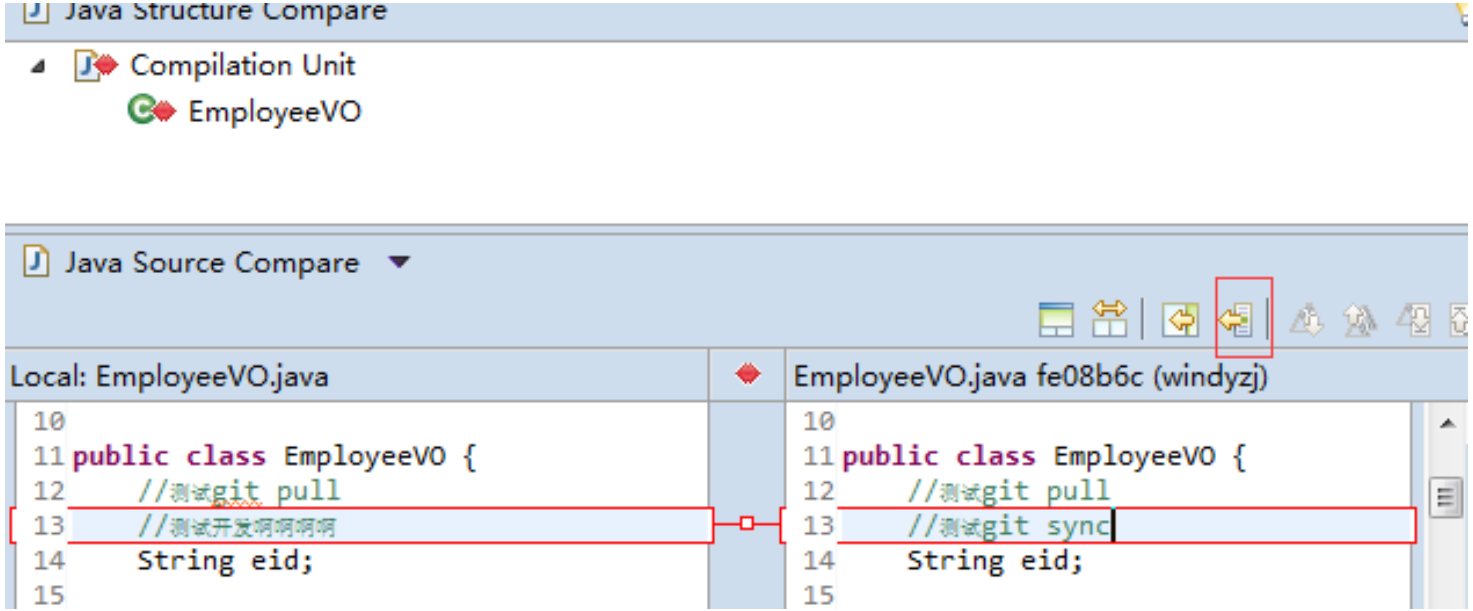
## 学会运用比较工具



# Egit操作—Github

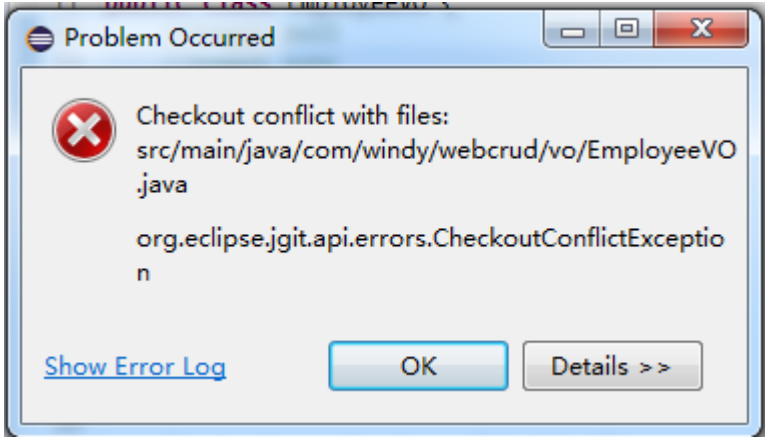


## 将服务器代码同步到本地



## ➢ 解决冲突

- 先pull下来以后，会发生冲突报错，其实并没有pull成功，因为你的修改并没有提交成为本地版本，Git无法进行merge。



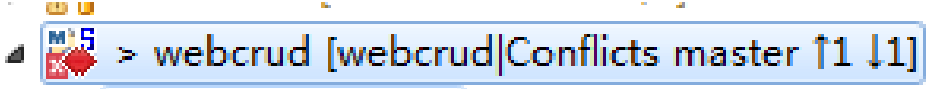

- 所以解决冲突之前先要把你自己的程序提交到版本库
- add index→commit
- 完成提交后,再次pull，Git会自动帮你合并版本，如果是同一文件同一位置的代码，Git会让你手工合并。

## ➢ 手工合并

- 右侧就是合并时同一处代码发生了冲突，需要手工合并。

```
11 public class EmployeeVO {
12     //测试git pull
13 <<<<<<< HEAD
14     //测试开发啊啊啊啊
15 =======
16     //测试git sync
17 >>>>>>> branch 'master' of https://github.com/windyzj/webc
```

- 项目会出现右侧的状态



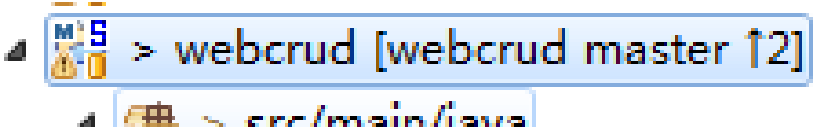

- 那么三步：

  1、编辑代码

  2、add index

  3、commit

  合并完成后，项目状态会如右侧图示。




## ➢ 合并完成后，再Push，则成功完成提交。

## 关于老版本EGit没有[Pull...] 只有[Pull]的解决方案

windows-->Preference-->Repository Settings中选择自己的仓库点击OPEN,添加如下配置:

```
[branch "master"]
    remote = origin
    merge = refs/heads/master

[remote "origin"]
    url = https://xxxxxxxx/xxxxx.git
    fetch = +refs/heads/*:refs/remotes/origin/*
    push = refs/heads/master:refs/heads/master
```

# 目录



尚硅谷

## Git工作流

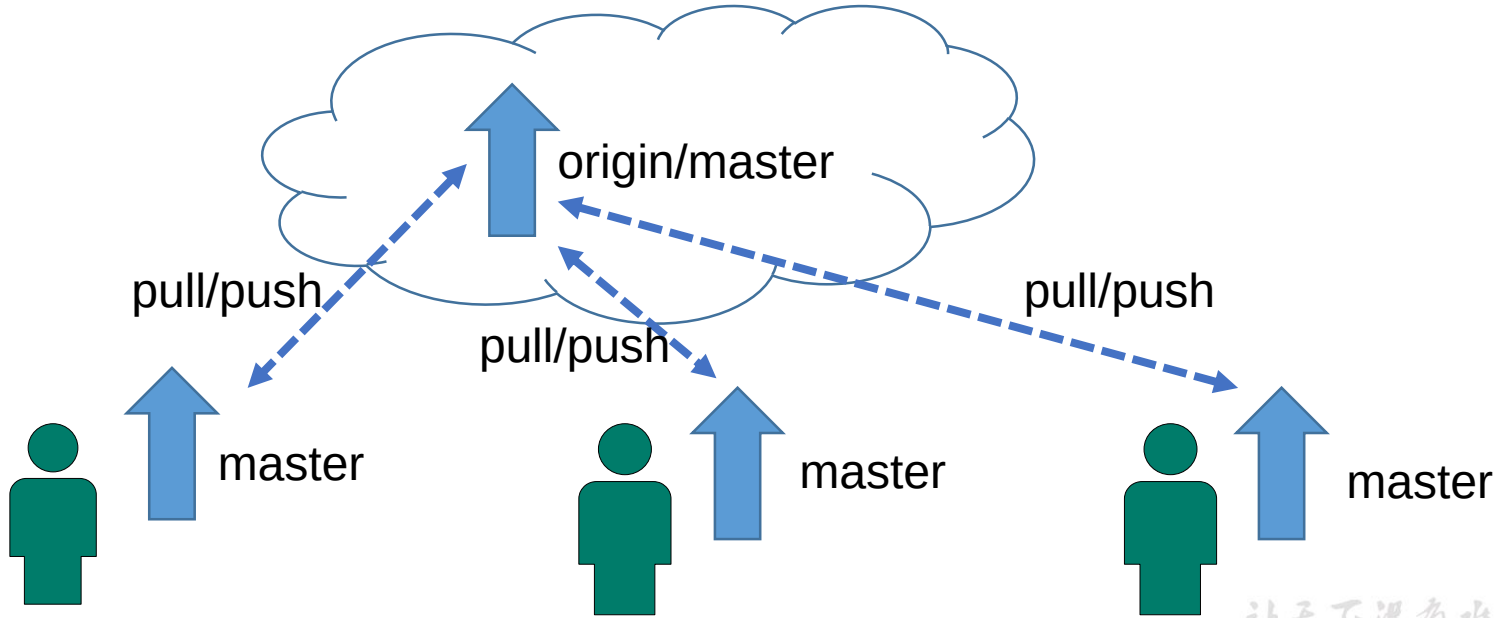
- 简单来说就是，一个项目的成员们在工作中统一使用Git的工作方式。

## 集中式工作流

## GitFlow工作流

# 集中式工作流

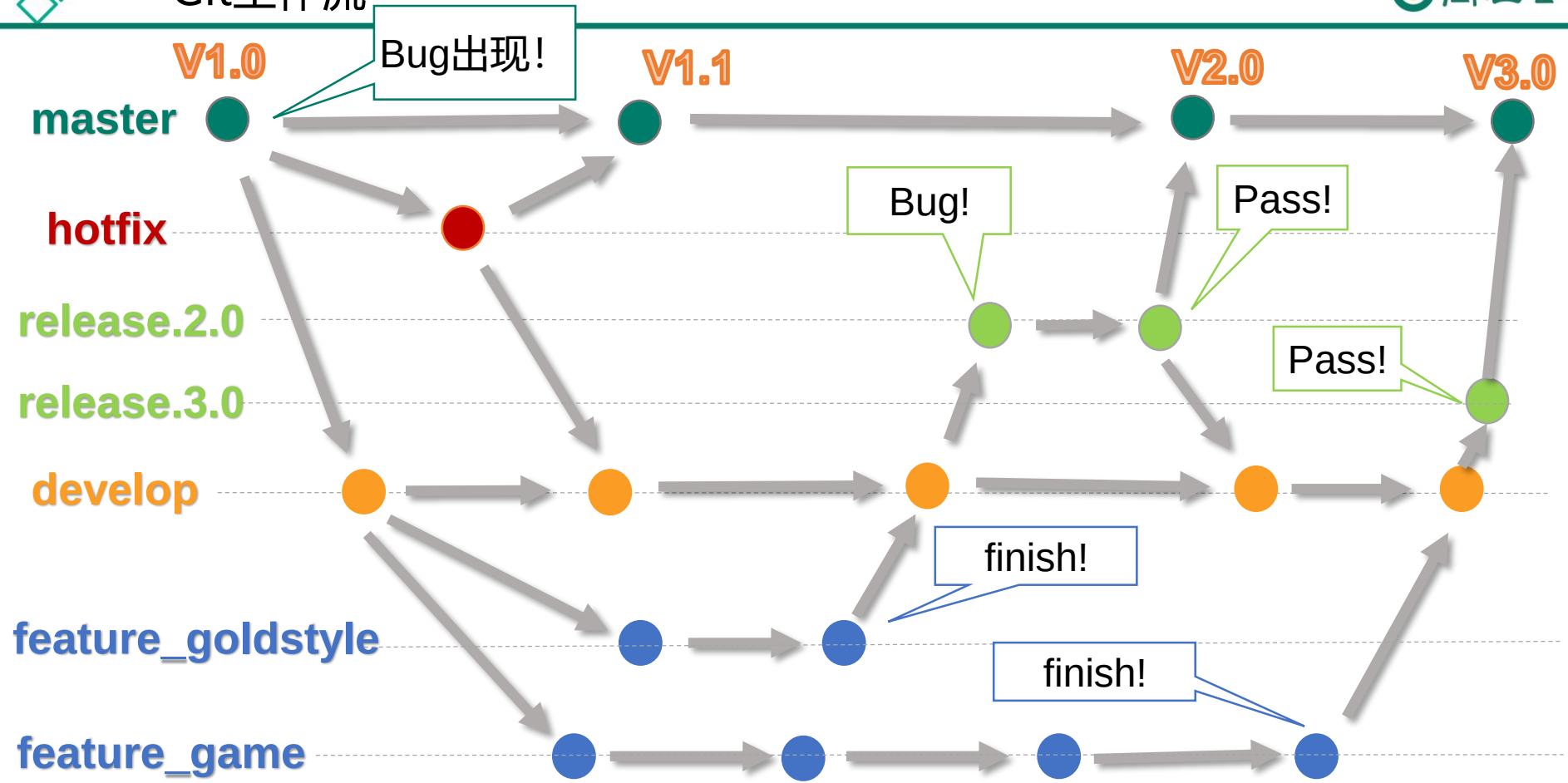
- 像SVN一样，集中式工作流以中央仓库作为项目所有修改的单点实体。所有修改都提交到Master这个分支上。
- 这种方式与SVN的主要区别就是开发人员有本地库。Git很多特性并没有用到。



## GitFlow工作流

- Gitflow工作流通过为功能开发、发布准备和维护设立了**独立的分支**，让发布迭代过程更流畅。严格的分支模型也为大型项目提供了一些非常必要的结构。

# Git工作流



# 分支种类

- **主干分支 master**

  主要负责管理正在运行的生产环境代码。永远保持与正在运行的生产环境完全一致。

- **开发分支　develop**

  主要负责管理正在开发过程中的代码。一般情况下应该是最新的代码。

- **bug修理分支　hotfix**

  主要负责管理生产环境下出现的紧急修复的代码。 从主干分支分出，修理完毕并测试上线后，并回主干分支。并回后，视情况可以删除该分支。

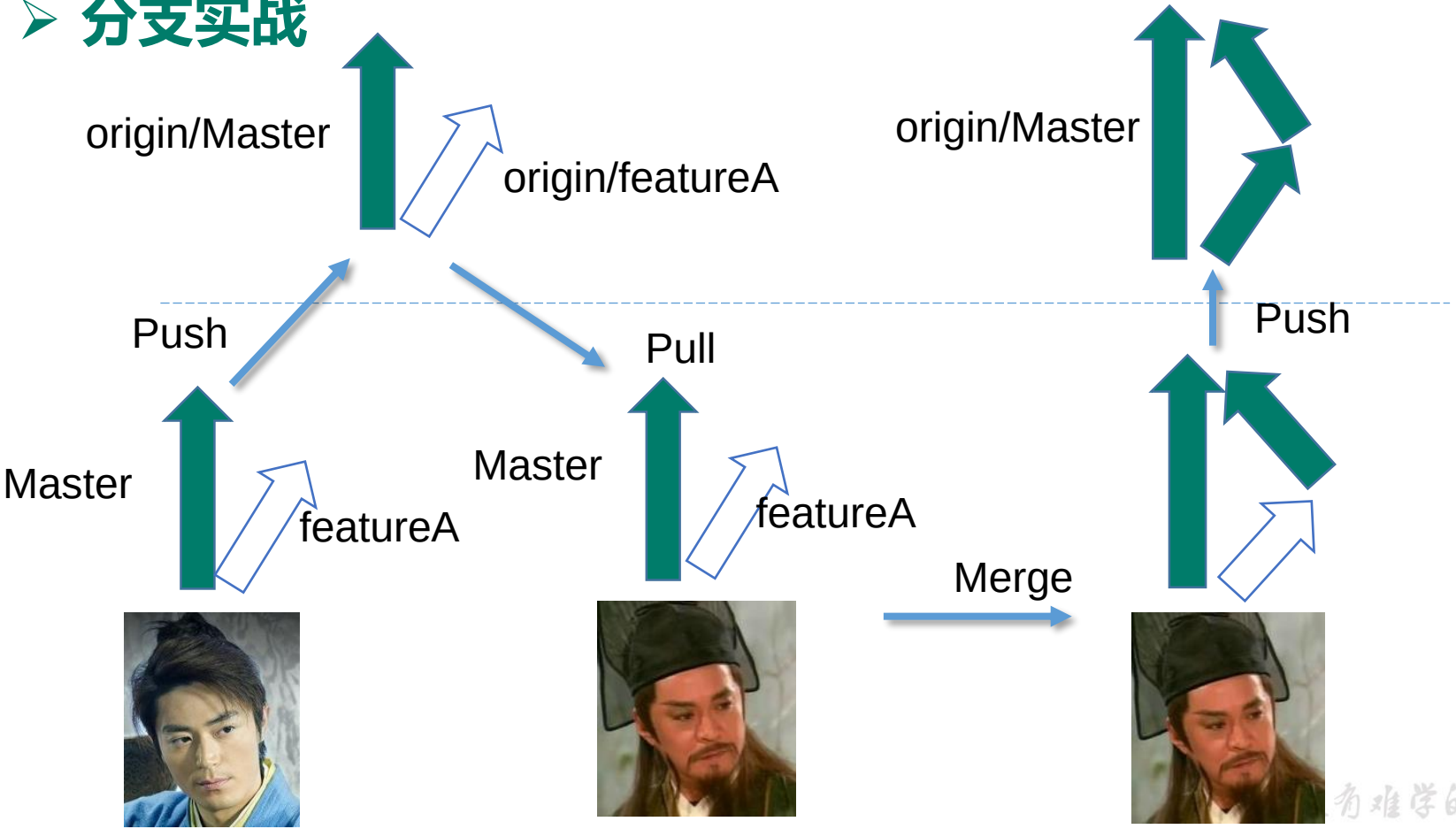
- **发布版本分支 release**

较大的版本上线前，会从**开发分支**中分出**发布版本分支**，进行最后阶段的集成测试。该版本上线后，会合并到主干分支。生产环境运行一段阶段较稳定后可以视情况删除。

- **功能分支 feature**

为了不影响较短周期的开发工作，一般把中长期开发模块，会从**开发分支**中独立出来。 开发完成后会合并到**开发分支**。

## 分支实战


origin/Master
origin/featureA
Push
Pull
Master
featureA
Master
featureA
Merge
origin/Master
Push

# 完成你的第一次pull request

- 感想留言板：https://windyzj.github.io/try_git/


尚硅谷