**GIT的使用**

本文简单介绍如何使用git工具，在开发过程中的一些简单操作和基本原理，不作深入探讨。如果想进行详细了解可自行百度，网上资料非常多。

这是官方中文资料：<https://git-scm.com/book/zh/v2>

1. Git简介

Git是一个分布式版本控制系统，从技术原理上来说，它无需中央服务器，每个人电脑里都有完整的版本库，没有网络也可以工作，在有网络时再同步或“交互”大家的修改。不过，在实际应用中，很少直接在个人工作电脑之间直接交换修改，为了方便“交换”大家的修改，也为了便于统一管理，通常还是有一台电脑来充当“中央服务器”。

大多数情况下，Git中的操作都只需要访问本地文件和资源，一些分支操作和提交都在本地仓库进行，当在需要的时候再与服务器仓库同步。

Git的一些重要概念：

1. 工作目录(Working Directory)

实际存放文件的目录，在初始化Git的时候设置的目录

1. 版本仓库（Repository）

工作区有一个隐藏目录.git，这就是Git的版本库。

stage暂存区（也叫index）, 类似一个缓存区域，临时保存你做的改动。

自动创建的第一个分支master，以及指向当前分支的一个指针叫HEAD

1. 工作目录下面的文件两种状态：已跟踪（Tracked）或未跟踪（Untracked）

未跟踪的文件的改动不受到Git的版本控制。

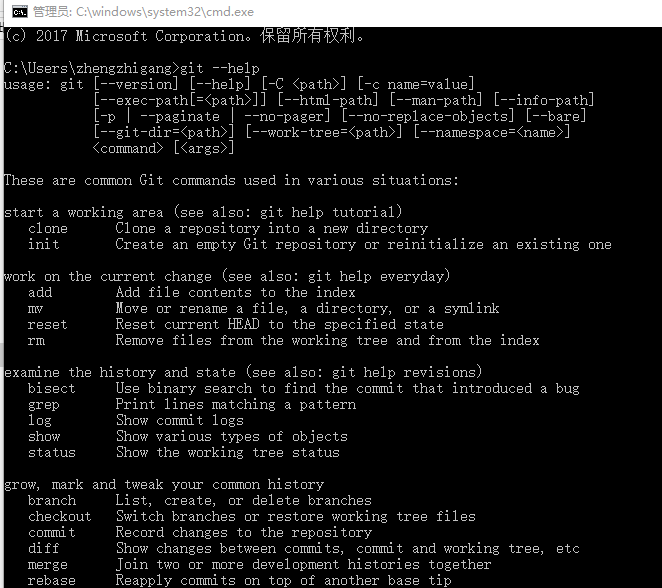
已跟踪文件包括状态：未修改（unmodified）、已修改（modified）、已暂存（staged）三种状态。未修改（unmodified）状态其实也就是已提交（committed）状态。

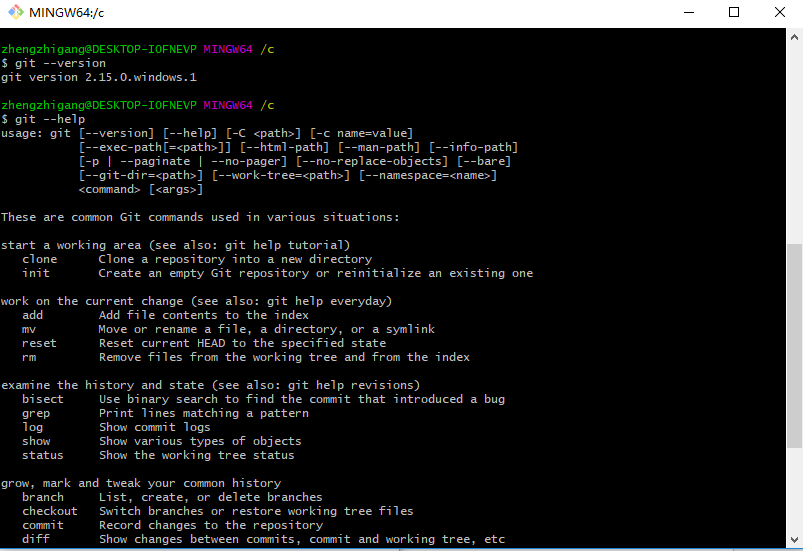
1. 工具安装
2. git window平台安装程序

<https://github.com/git-for-windows/git/releases/download/v2.15.0.windows.1/Git-2.15.0-64-bit.exe>

下载后安装，选择缺省安装，一路next即可。

安装完成后，　可以使用cmd命令窗口或git bash中执行git各种命令





Git Bash可在windows平台模拟linux命令操作

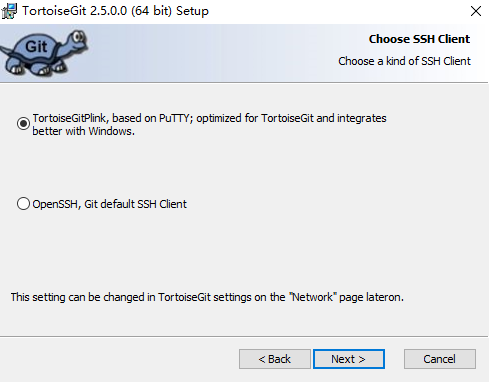
1. Windows平台下的GUI工具（Tortoisegit 64位）

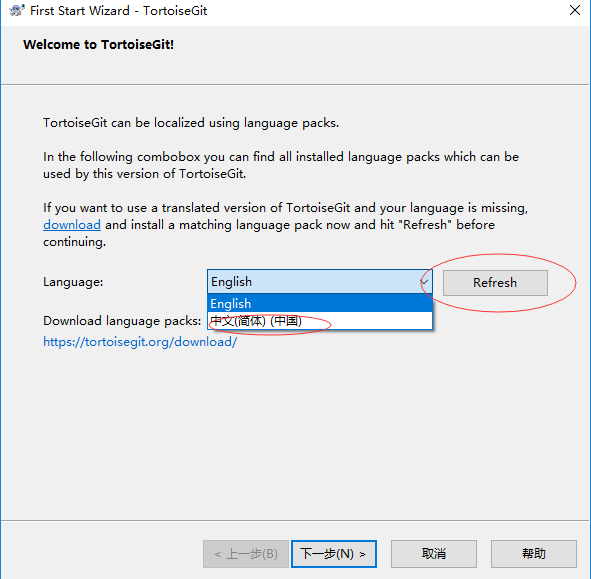
下载地址：

<https://download.tortoisegit.org/tgit/2.5.0.0/TortoiseGit-2.5.0.0-64bit.msi>

中文语言包：

<https://download.tortoisegit.org/tgit/2.5.0.0/TortoiseGit-LanguagePack-2.5.0.0-64bit-zh_CN.msi>



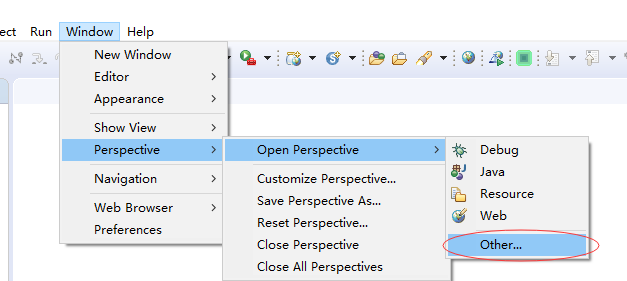


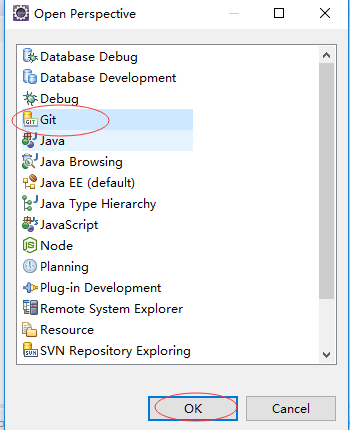
在这一步， 可选择语言，如果语言包还没有安装， 可以在此时去安装语言包，等语言包安装完成后，再在此处点击Refresh按钮，可以看到中文选项。

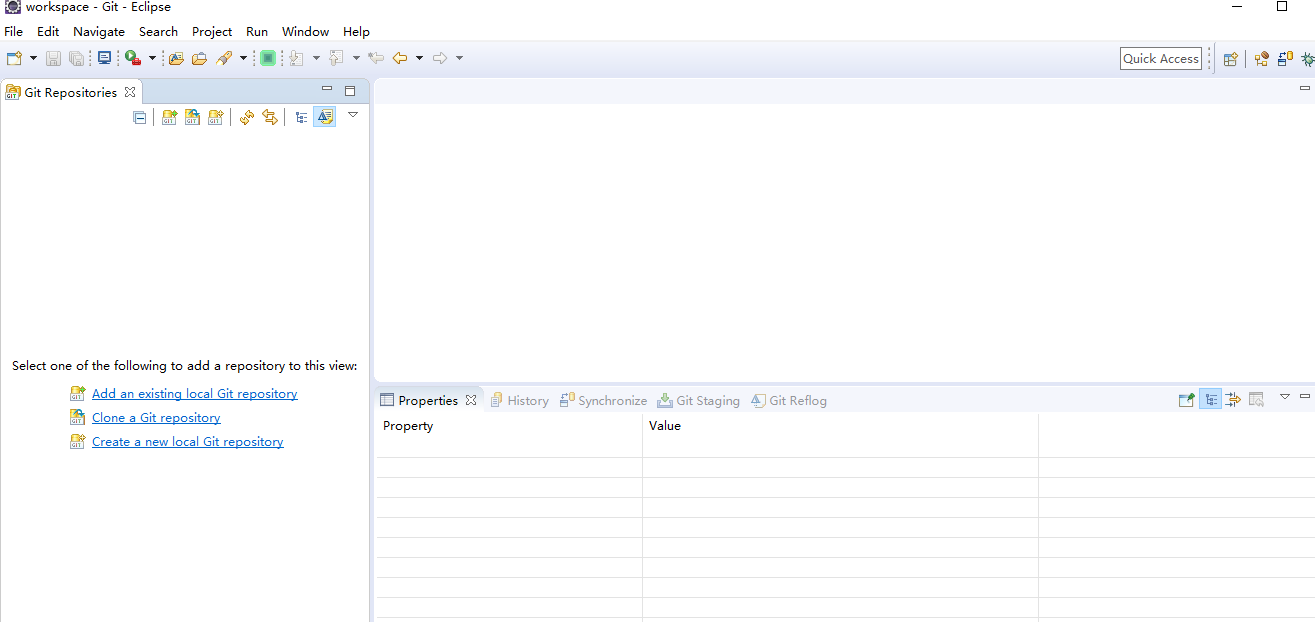
安装完成后，可以在windows资源管理器中使用。

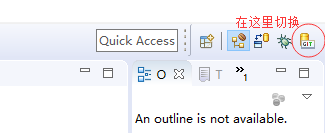


1. eclipse已经自带git插件

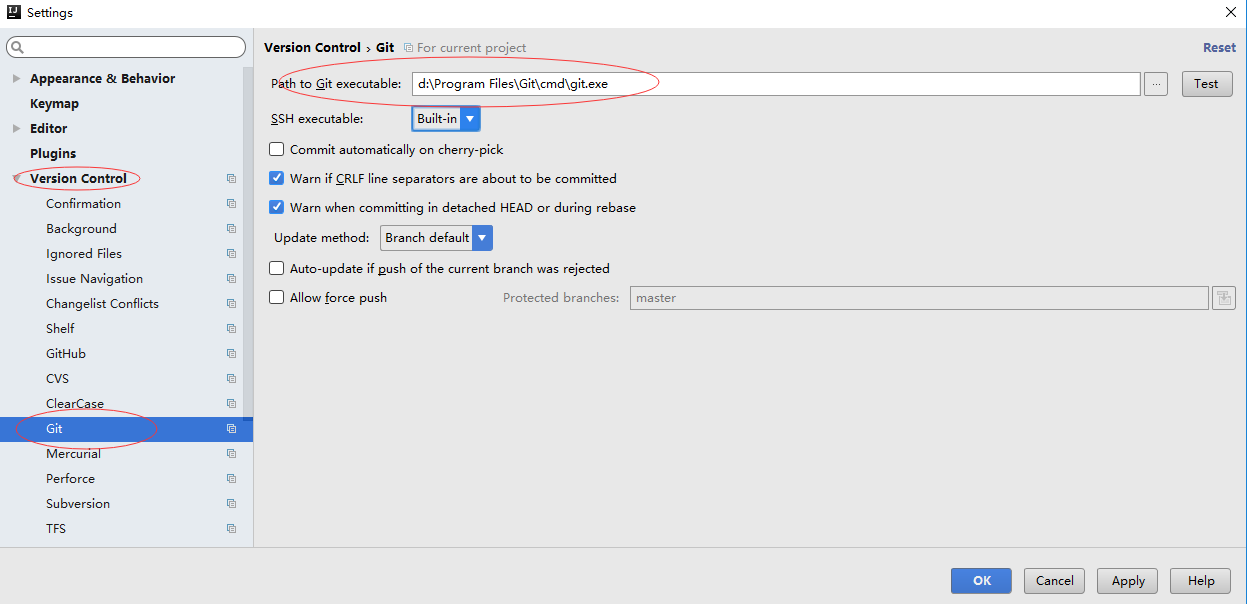








1. Idea的git配置



1. git基本操作
2. 配置用户名密码和生成ssh密钥

Git同步如push代码，需要身份认证，如用户名密码（http/https）或ssl密钥 (ssh登录方式)

为避免每次输入用户名和密码，可执行配置命令, 如：

git config --global user.name "zhengzhigang"

git config --global user.email " zhengzhigang@ultrapower.com.cn"

也可在IDE的git设置界面添加配置项

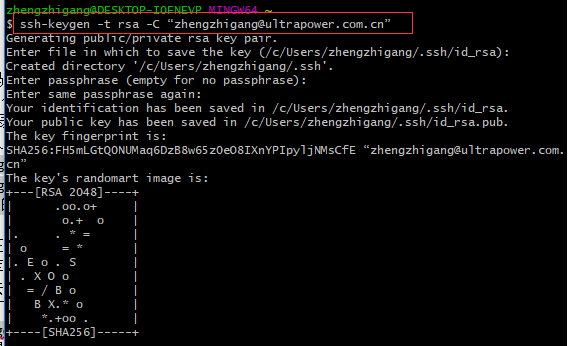
如果是SSH认证，需要事先创建好SSL密钥对， 然后将公钥给对方，比如中央服务器，登录到服务器管理界面提交公钥文件内容。

创建密钥的方法：

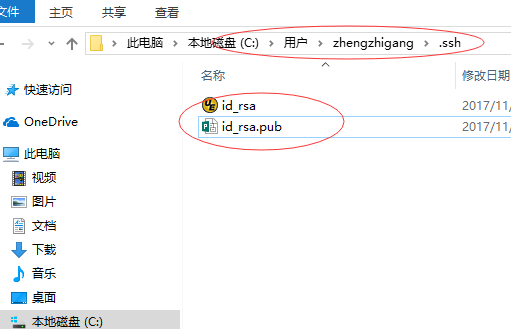
1. git-bash下输入命令：

ssh-keygen -t rsa -C “zhengzhigang@ultrapower.com.cn”

如果希望简单点，可以设置密码为空，直接回车(3次)即可。

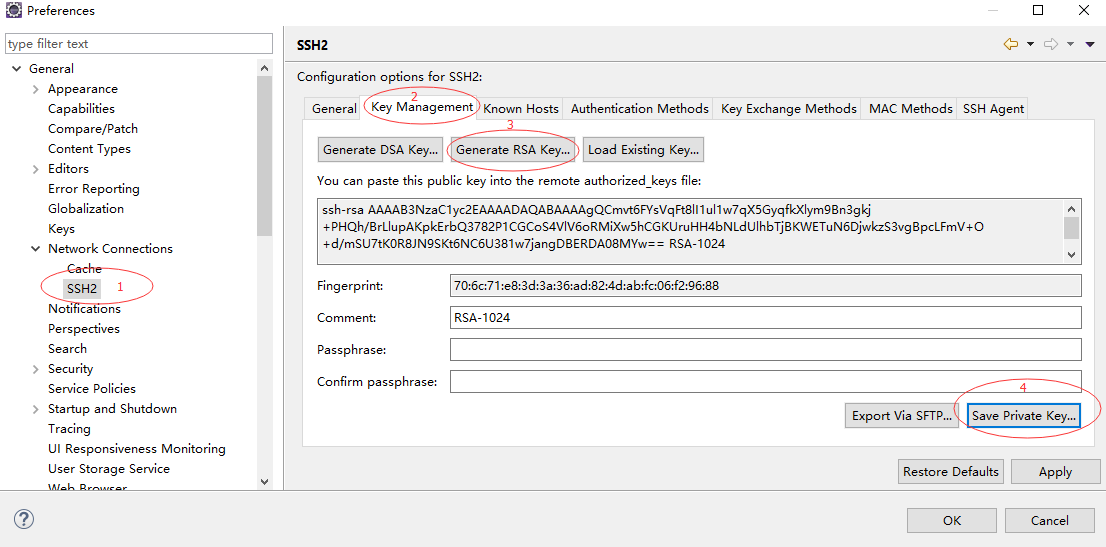


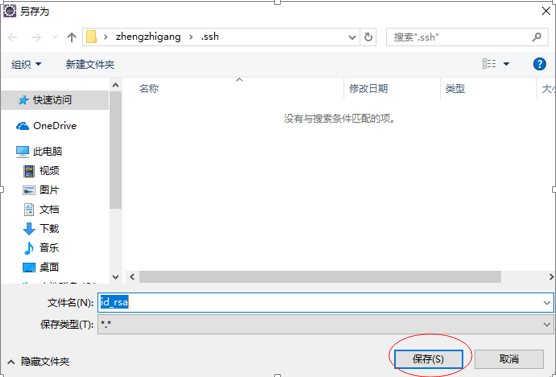
生成的文件如下：



将id\_rsa.pub文件内容提交到服务器

1. 也可以通过eclipse创建ssh-key





1. 还可以使用其他任何创建SSH密钥的工具
2. git init

在本机创建一个repo，初始化仓库。一般我们比较少在本地工作机开始创建并初始化一个仓库，大多数情况是从远程将一个已有的repo代码拉到本机上。

1. git clone从远程机器获取源码

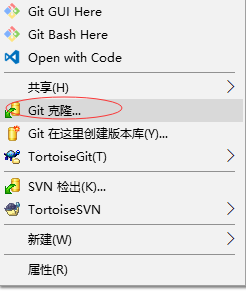
开始团队合作开发项目，第一步，从服务器获取源码。

创建一个本地空目录， 进入此目录

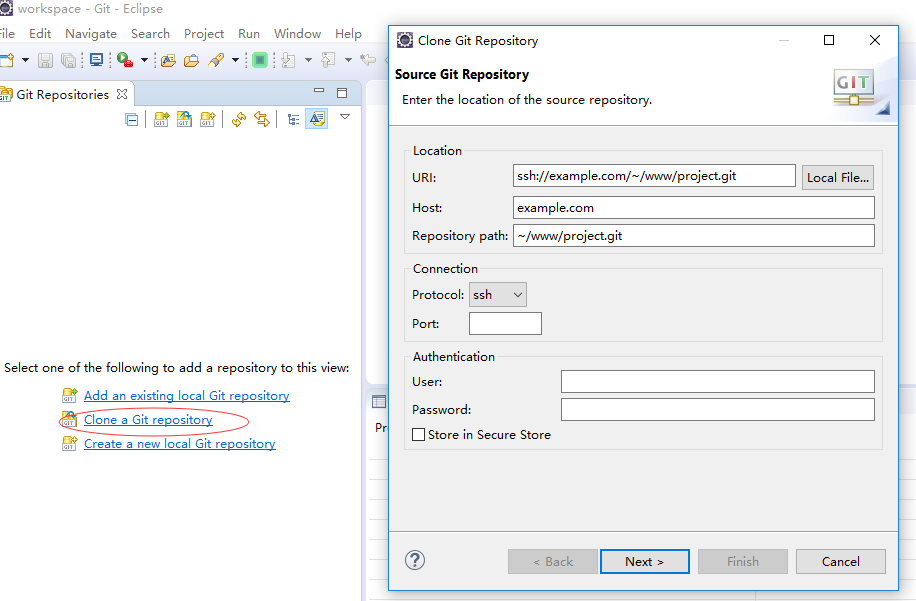
1. 命令行：git clone [url]

例如：git clone ssh://example.com/~/www/project.git

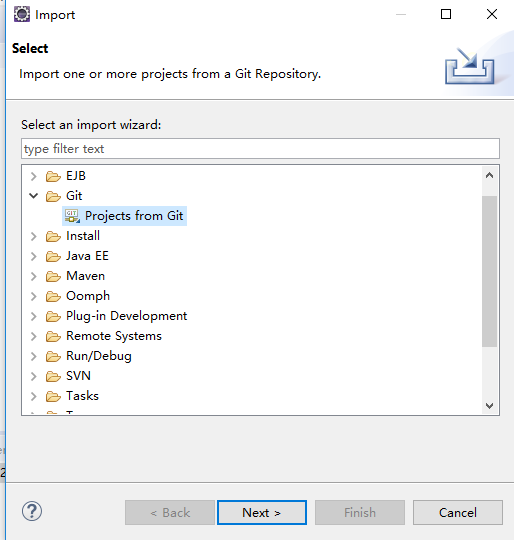
1. Tortoisegit：在windows资源管理器中点击右键

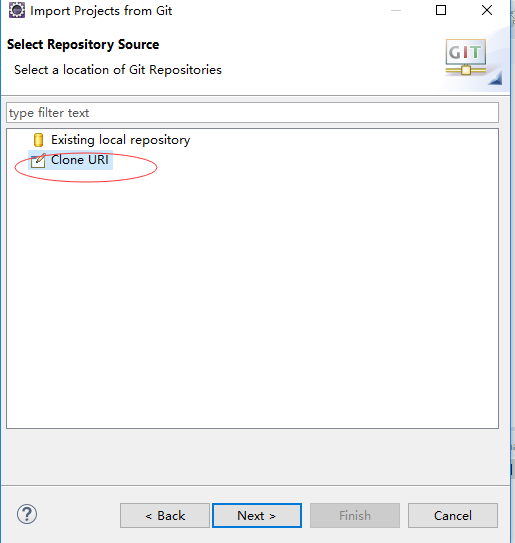


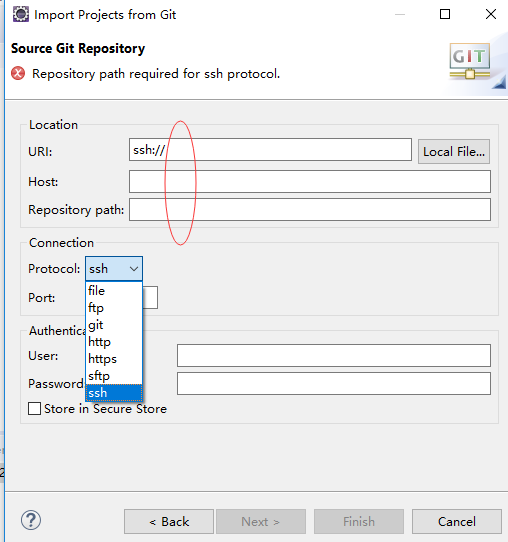
1. Eclipse ：Clone远程项目源码



或者import一个远程url的方式导入项目



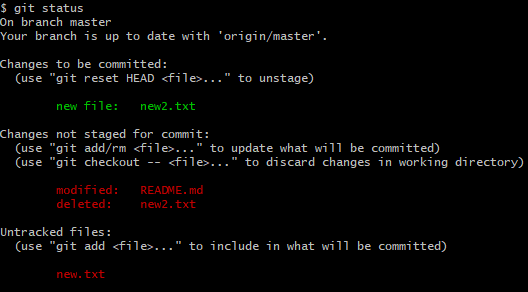




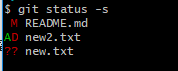
当克隆完某个项目后，可以看到一个名为origin的远程库（origin只相当于一个别名，使用git remote命令查看），Git 默认使用origin来标识所克隆的原始仓库。建立一个指向它的master分支的指针，origin/master表示远程master分支。同时还会建立一个属于自己的本地master分支。

1. 常用命令
2. git status

git status 查看仓库当前的状态， 可以看到当前是在哪个分支中，以及哪些文件被改动



git status –s 简单的查看有哪些文件发生变化（添加删除修改）

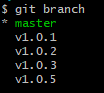


Git的状态有

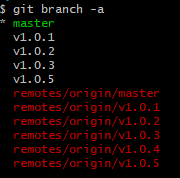
如果在当前分支上做了增删改查

1. git分支

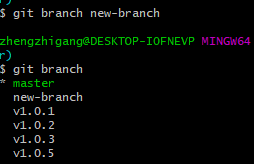
* git branch 查看本地分支， 当前所在的分支名前会有一个\*号



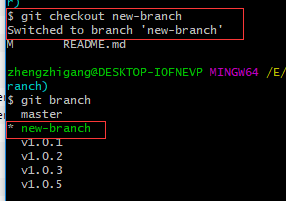
* git branch -a 查看所有分支，包括远程的。



* git branch new-branch 在本地创建了一个名为new-branch的新分支。如果加上 -b参数则可在创建后切换到新分支



* git checkout new-branch 切换到new-branch分支。



文件状态 M 表示这个文件是原分支带过来的修改（在这里new-branch是从master创建出来的）

* 创建远程origin的分支到本地。

git checkout -b mydev origin/develop在本地创建了一个mydev分支并和远程develop分支关联起来，同时切换到mydev分支。

这里本地和远程分支不同名，但是建议本地分支和关联的远程分支的名字保持相同。

* 合并分支

假设现在要把new-branch分支合并到master分支

git checkout master 先切换到master分支

git merge new-branch 把new-branch合并到当前分支

* git branch -d new-branch 删除分支new-branch，删除前需切换到其他分支



1. git add添加文件到仓库中

当修改完了文件或创建了一个新文件， 使用add命令将其加入到仓库

git add test.java将test.java文件添加到仓库

git add file1.txt file2.txt 同时将这2个文件添加到仓库

1. git commit 提交版本

git commit -m "本次版本提交的说明文字"

git的commit操作，你就理解是对一些操作进行保存的快照，一次状态存盘。

1. git log 查看提交的历史记录

git log --pretty=oneline 使用 –pretty=oneline参数只按行显示commit id和comments

1. 使用reset取消文件暂存

git reset HEAD mytest.txt 将已经暂存的文件mytest.txt取消暂存状态

1. 撤消文件修改

git checkout -- mytest.txt 取消最近一次add或commit后的修改，也就是用暂存区中mytest.txt文件来覆盖了工作区中的mytest.txt文件。

注意这个命令中间有 -- 符号分割, 没有 -- 符号则变成了切换分支命令

1. git rm filename 将文件从版本库移除

git rm test.txt 将test.txt从版本库移除

1. git push

将当前分支的所有commit推送到服务器

1. git pull

将远端分支自动合并到本地仓库中当前分支, 它实际包含2个动作， 先执行fetch命令将远程数据拉到本地仓库, 再执行merge命令将当前分支对应的远程分支数据合并到当前分支。

如果加上 - -rebase参数(或-r) ，则是git fetch后执行git rebase命令。

1. git fetch 从远程仓库抓取数据到本地

git fetch [remote-name] 将远程仓库最新提交的数据拉取到本地。如git fetch origin命令会将远程服务器的最新数据拉取到origin下。

1. 本地分支和远程分支建立关联

如果本地新建的分支和远程分支没有关联， 可以使用下面的命令建立关联

git branch --set-upstream-to origin/devtest devtest 当本地分支和远程分支建立关联后执行pull, push等命令操作就不需要指定远程的分支。

1. 解决冲突

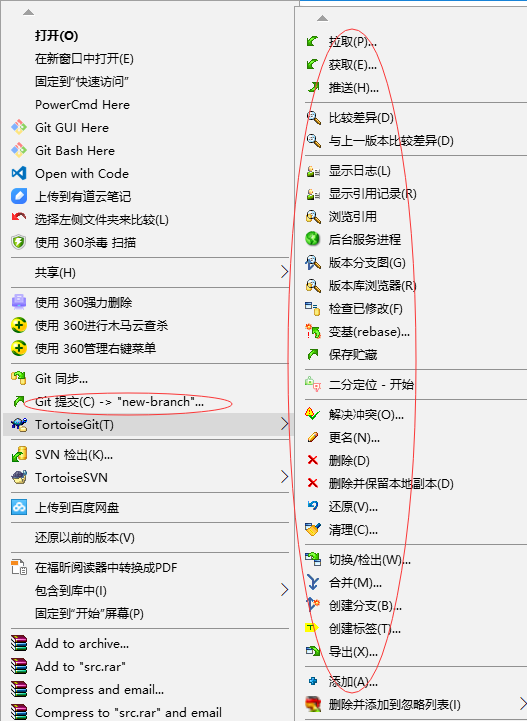
当执行merge或rebase做合并操作（执行pull也会有合并）时，如果多个提交是修改文件的相同位置，git不能自动合并，则有可能产生文件冲突，解决冲突的步骤：

1. 编辑冲突的文件解决冲突
2. git add file
3. git rebase --continue

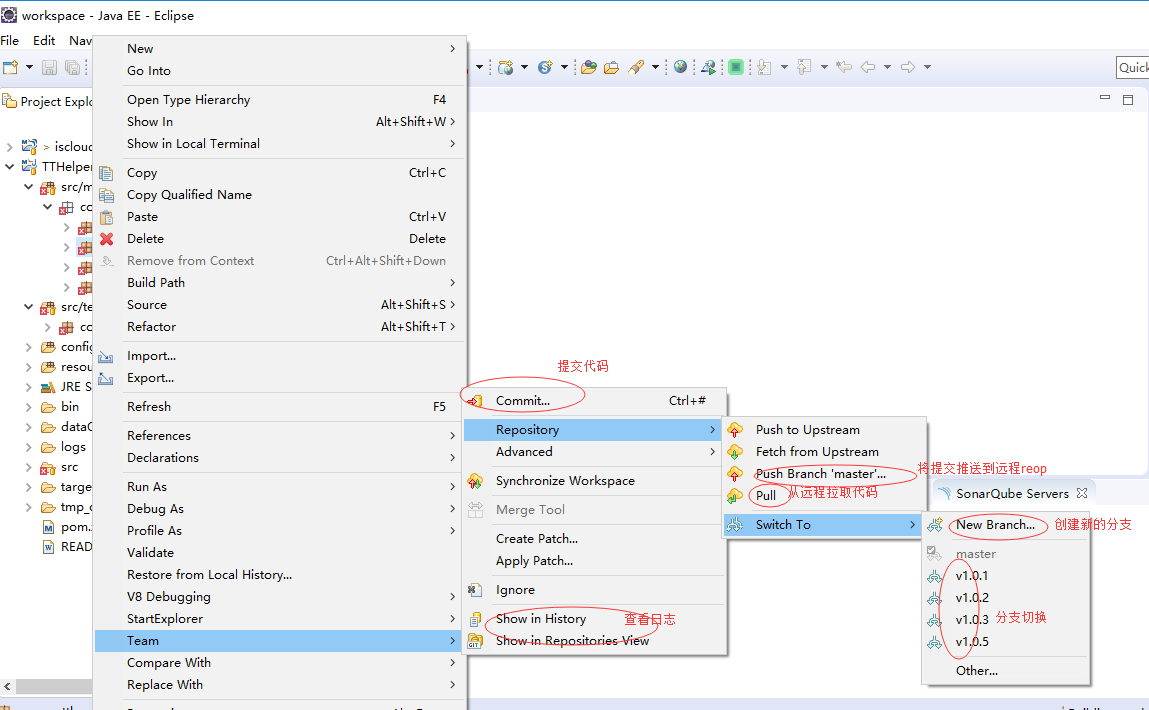
如果解决冲突执行了add后想放弃所有操作则可执行git rebase --abort

1. 在windows资源管理器中操作git

在工作目录中使用右键菜单选择相应的操作

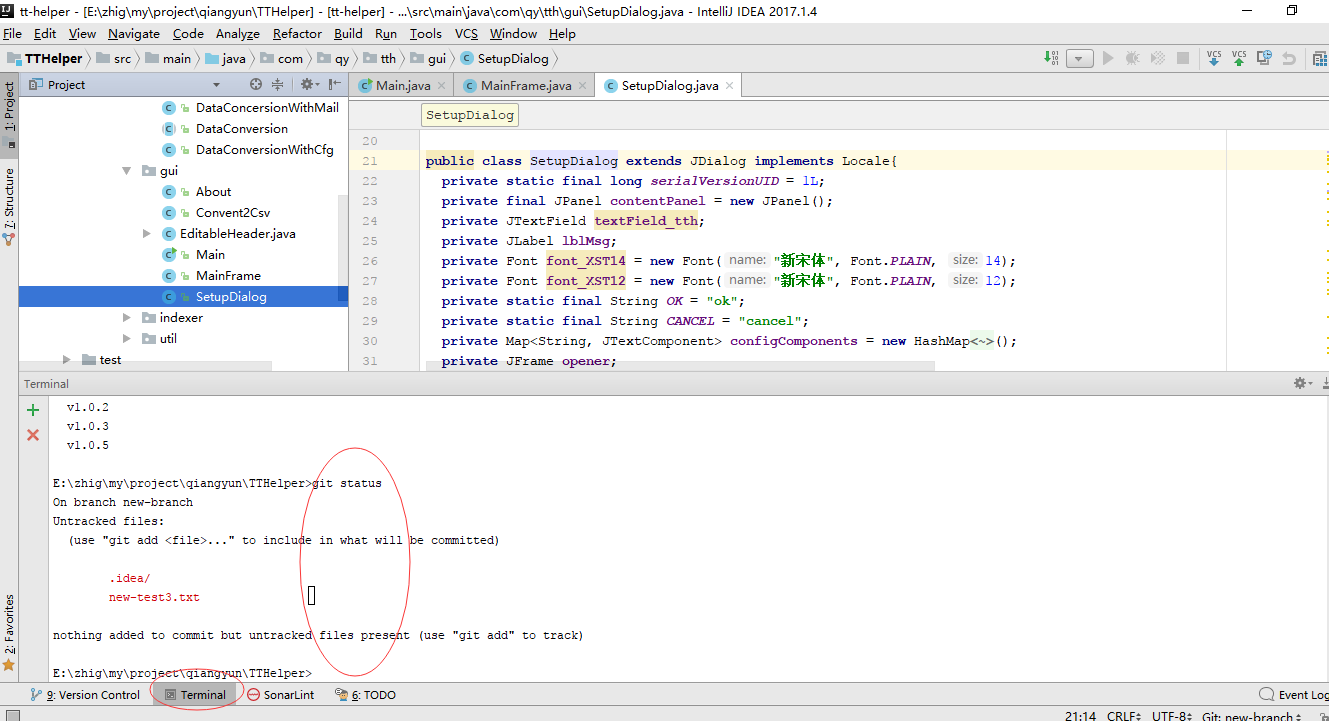


1. 在Eclipse中操作git

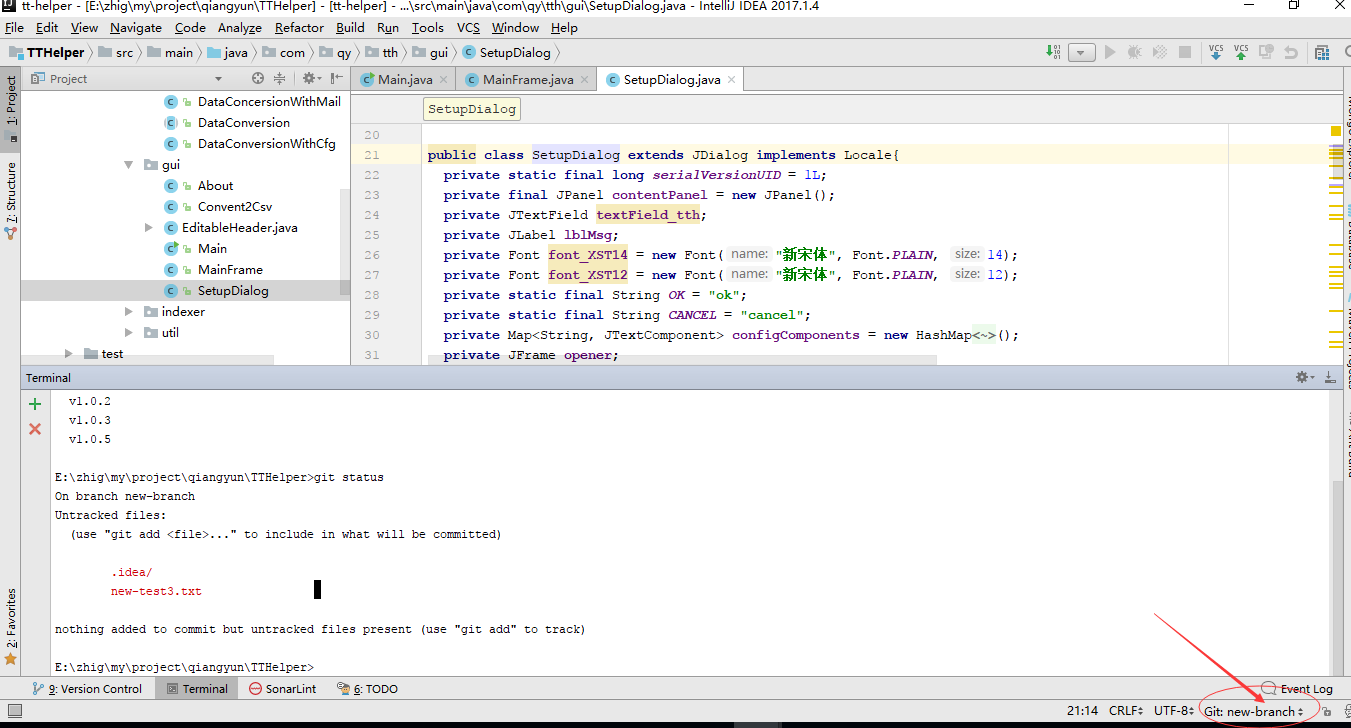


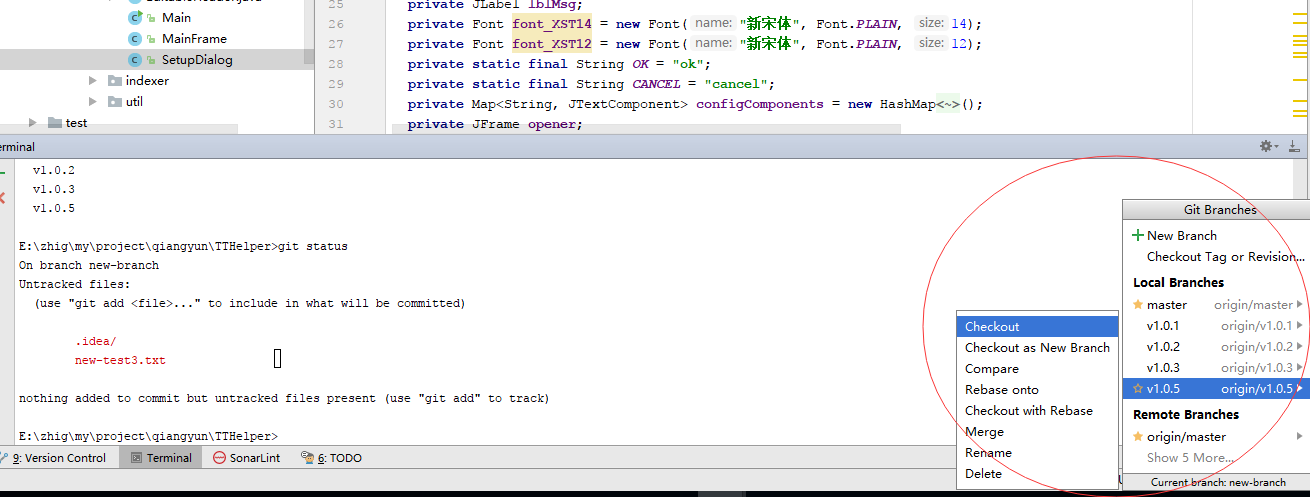
1. 在IDEA中使用git

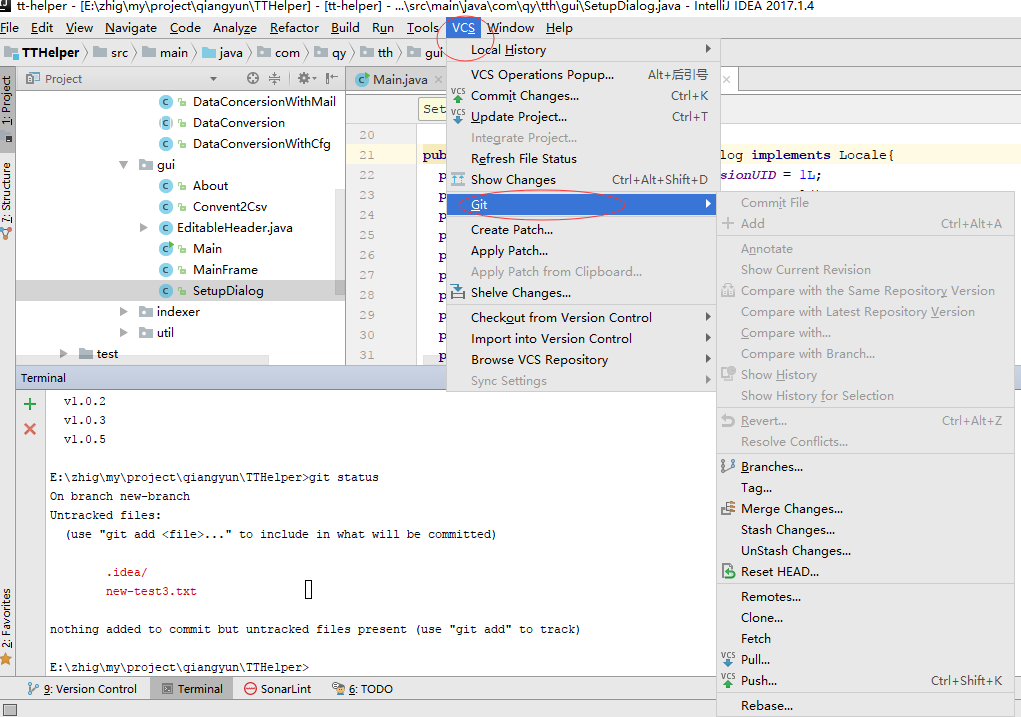
* 在Terminal窗口执行git命令



* 使用git功能菜单







1. 使用git的开发流程

Git鼓励多分支开发，如果你足够理解了git，那么可以随心所欲地利用git的分支来完成工作，可以采用很多种方式和工作流程。这里有一种简单的方式可供参考：

假设远程服务器有master和develop分支， 其中master分支是用来做版本发布用，我们平常的开发工作是在develop分支上进行，当通过git克隆项目到本地后，接下来

1. 根据远程develop分支创建一个本地develop分支作为本地的主开发分支

把本地develop分支用来与远程develop分支同步用

git checkout -b develop origin/develop

1. 创建一个工作分支

git checkout –b mydev

1. 当前分支在mydev，开始你的开发工作
2. 准备将当前的工作放入暂存区

git status 或git status -s 查看一下状态

根据情况执行

git add . 或git file1 file2 …

1. 提交工作

git commit –m “xxxxxxxx”

1. 准备推送到服务器

git checkout develop 先切换到develop

git pull 获取远程最新的修改，此时不会产生冲突

git checkout mydev 回到工作分支

git rebase develop 用rebase合并主干的修改，如果有冲突在此时解决

git checkout develop回到主分支

git merge mydev合并工作分支的修改，此时不会产生冲突

1. git push 完成推送
2. git checkout mydev切回工作分支或git branch -d mydel删除工作分支