Git教程

文档说明

写本教程的目的：

1. 领导安排的任务，新来的同事好多没用过git，写一个教程帮他们快速入门

2.

本教程试用人群：

没接触过git，想学习git的新手

分享一下我在学习时主要参考的资料

<http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000> git廖雪峰教程 （一个写的比较好的git教程）

<https://www.git-scm.com/doc> git官网 英文版资料

<https://www.git-scm.com/book/zh/v2> git官网 中文版资料

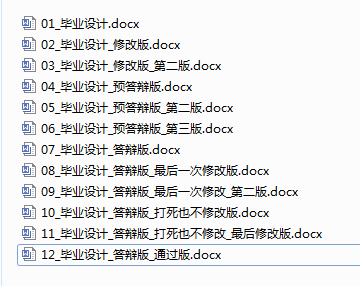
由于时间匆忙，编写时可能存在错误或者不合理的地方，欢迎大家批评指正。

1 Git是什么？

Git是目前世界上最先进的分布式版本控制系统（没有之一）。

如果你写过毕业设计，那你一定有过这样的经历，写一版毕业设计，给老师，老师指出你要修改的地方，然后再给一版，再修改，然后一直修改，到最后不知道修改了多少版。你想找到某一版的其中的一段，完了，有那么多的版本，怎么找。而且有的版本可能丢了或者删除了。

如果你是计算机、软件有关专业的学生，还要写程序。为了防止毕业答辩的时候代码丢失，我们往往要给代码备份。我刚开始的做法是每次改完代码把整个程序压缩，放到云上。但是，时间长了，逐渐发现这种做法的坏处，即使你只改了几十行代码，你也要把程序打包一遍放到云上。有没有好的办法呢。



我来告诉你，有这么一个软件，能记录你一共有多少个版本，每次修改了什么，是谁什么时间修改该的。而且可以很方便的切换到历史版本。这个软件就是git。

这个软件用起来就应该像这个样子，能记录每次文件的改动：

版本 用户 说明 日期

1 张三 项目初始化。 2017-01-01 10:38

2 张三 添加用户登陆功能 2017-01-03 18:09

3 李四 添加用户管理功能 2017-01-04 09:51

4 张三 添加加密功能 2017-01-05 09:00

1.1 Git的诞生

很多人都知道，Linus在1991年创建了开源的Linux，从此，Linux系统不断发展，已经成为最大的服务器系统软件了。

Linus虽然创建了Linux，但Linux的壮大是靠全世界热心的志愿者参与的，这么多人在世界各地为Linux编写代码，那Linux的代码是如何管理的呢？

事实是，在2002年以前，世界各地的志愿者把源代码文件通过diff的方式发给Linus，然后由Linus本人通过手工方式合并代码！

你也许会想，为什么Linus不把Linux代码放到版本控制系统里呢？不是有CVS、SVN这些免费的版本控制系统吗？因为Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不但速度慢，而且必须联网才能使用。有一些商用的版本控制系统，虽然比CVS、SVN好用，但那是付费的，和Linux的开源精神不符。

不过，到了2002年，Linux系统已经发展了十年了，代码库之大让Linus很难继续通过手工方式管理了，社区的弟兄们也对这种方式表达了强烈不满，于是Linus选择了一个商业的版本控制系统BitKeeper，BitKeeper的东家BitMover公司出于人道主义精神，授权Linux社区免费使用这个版本控制系统。

安定团结的大好局面在2005年就被打破了，原因是Linux社区牛人聚集，不免沾染了一些梁山好汉的江湖习气。开发Samba的Andrew试图破解BitKeeper的协议（这么干的其实也不只他一个），被BitMover公司发现了（监控工作做得不错！），于是BitMover公司怒了，要收回Linux社区的免费使用权。

Linus可以向BitMover公司道个歉，保证以后严格管教弟兄们，嗯，这是不可能的。实际情况是这样的：

Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！一个月之内，Linux系统的源码已经由Git管理了！牛是怎么定义的呢？大家可以体会一下。

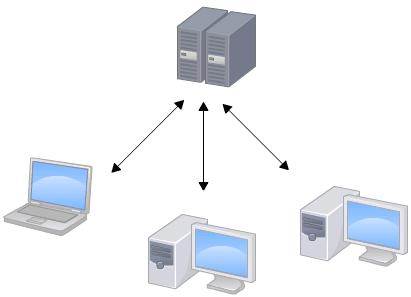
Git迅速成为最流行的分布式版本控制系统，尤其是2008年，GitHub网站上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等。

历史就是这么偶然，如果不是当年BitMover公司威胁Linux社区，可能现在我们就没有免费而超级好用的Git了。

1.2 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。

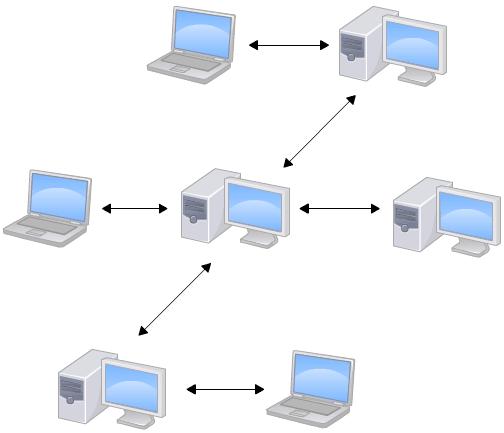


集中式版本控制系统最大的毛病就是必须联网才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

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。

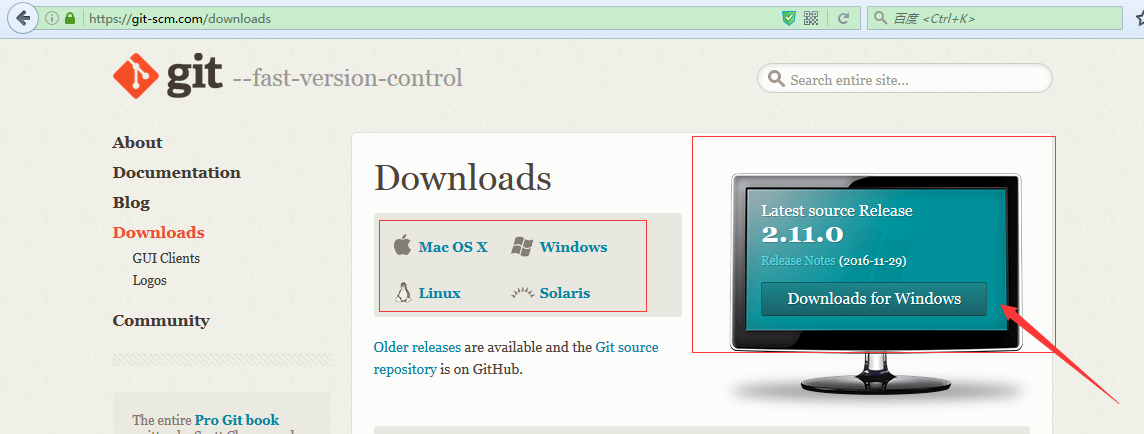
微软自己也有一个集中式版本控制系统叫VSS，集成在Visual Studio中。由于其反人类的设计，连微软自己都不好意思用了。

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！

2 安装Git

2.1 在windows上安装Git

在git官网 <https://git-scm.com/downloads> 下载最新的git安装包(注意有32位安装白和 64位安装包),下载完成以后，带点击exe文件进行安装，一直点下一步直到完成安装。





2.2 在Linux上安装Git

首先，你可以试着输入git，看看系统有没有安装Git：

$ git

The program 'git' is currently not installed. You can install it by typing:

sudo apt-get install git

像上面的命令，有很多Linux会友好地告诉你Git没有安装，还会告诉你如何安装Git。

如果你碰巧用Debian或Ubuntu Linux，通过一条sudo apt-get install git就可以直接完成Git的安装，非常简单。

老一点的Debian或Ubuntu Linux，要把命令改为sudo apt-get install git-core，因为以前有个软件也叫GIT（GNU Interactive Tools），结果Git就只能叫git-core了。由于Git名气实在太大，后来就把GNU Interactive Tools改成gnuit，git-core正式改为git。

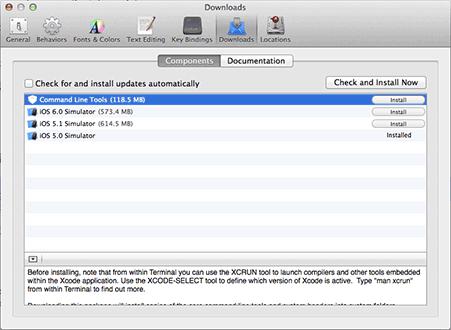
如果是其他Linux版本，可以直接通过源码安装。先从Git官网下载源码，然后解压，依次输入：./config，make，sudo make install这几个命令安装就好了。

2.3 在Mac OS X上安装Git

如果你正在使用Mac做开发，有两种安装Git的方法。

一是安装homebrew，然后通过homebrew安装Git，具体方法请参考homebrew的文档：[http://brew.sh/](http://brew.sh/" \t "http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)。

第二种方法更简单，也是推荐的方法，就是直接从AppStore安装Xcode，Xcode集成了Git，不过默认没有安装，你需要运行Xcode，选择菜单“Xcode”->“Preferences”，在弹出窗口中找到“Downloads”，选择“Command Line Tools”，点“Install”就可以完成安装了。



Xcode是Apple官方IDE，功能非常强大，是开发Mac和iOS App的必选装备，而且是免费的。

2.4 在eclipse里安装git插件

2.4.1 检查是否安装git插件

先检查eclipse里有没有安装git插件，

1. 如果安装了git，直接配置就可以使用。

2. 如果没有安装，则先安装git插件，然后配置使用。

检查方法：(如下图)

在 windows - Preferences - Team 下查看，如果有git选项，证明eclipse已经安装git插件了，如果没有，则没有安装git插件。

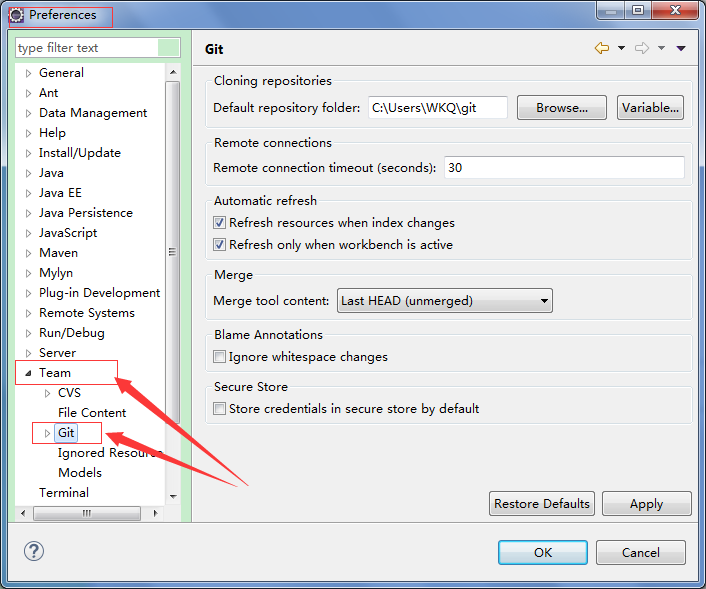


图2.4.1\_1 检查eclipse是否安装git插件

2.4.2 eclipse里安装git插件

如果没有安装git插件，可以在eclipse里下载，Help - Eclipse - Markettaplace ，搜索git，找到EGit，点击install，安装即可。如下图 2.4.1\_1 2.4.1\_2 2.4.1\_3

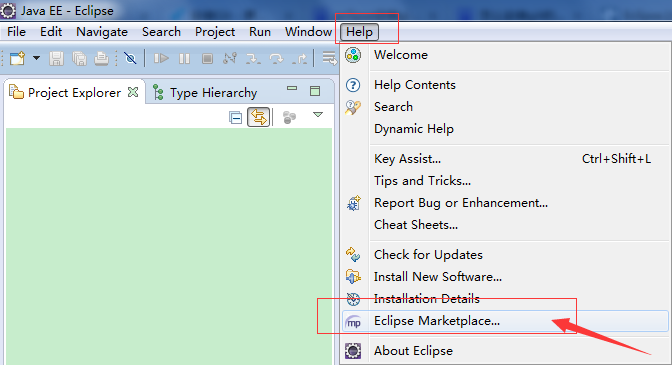


图2.4.1\_2 打开eclipse市场

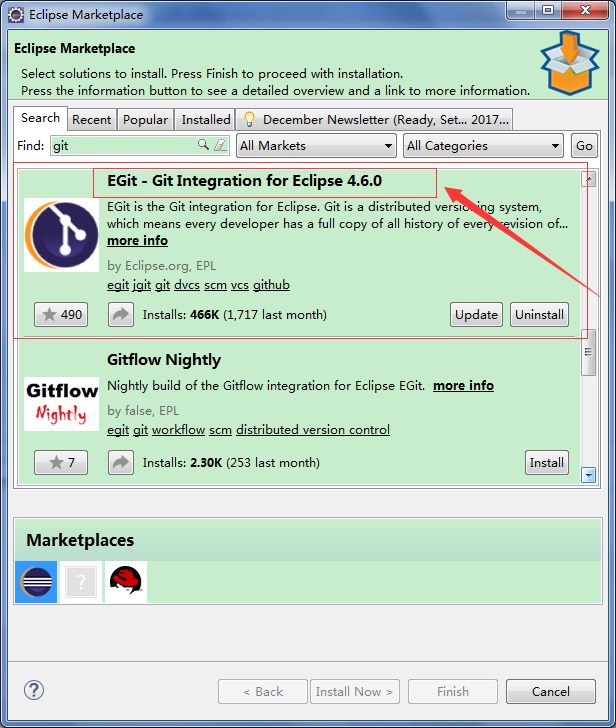


图2.4.1\_3 搜索git插件，找到EGit

2.5 配置git

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。

配置git，就相当于给git一个身份证号，用来区分git仓库。

2.5.1 在命令行配置git

Linux、 unix、 windows 都可以通过命令行来配置git，方法如下：

安装完成后，还需要最后一步设置，在命令行输入：

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

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

例如：git config --global user.name "Your Name"

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

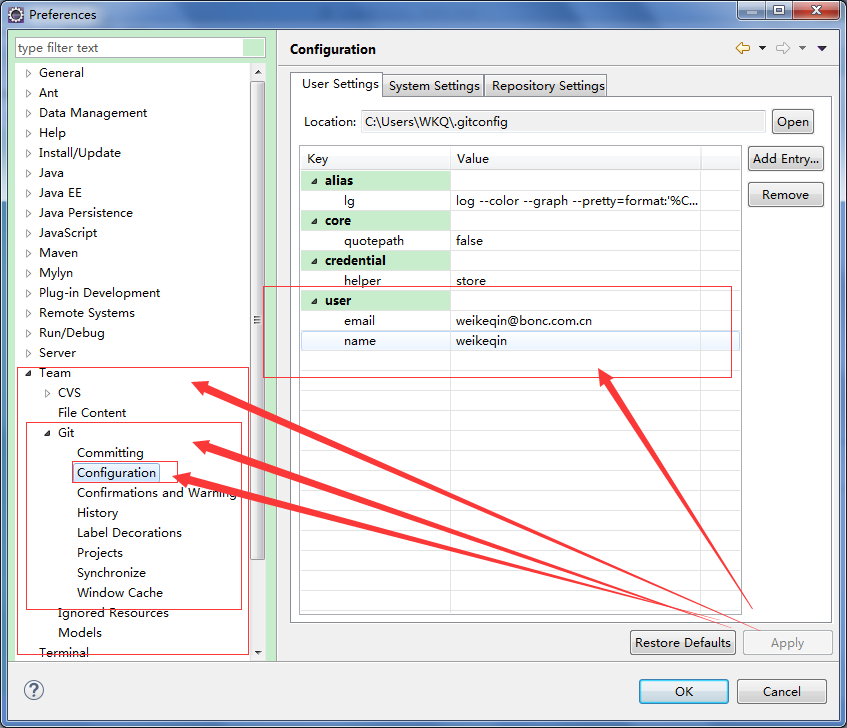


图2.5\_1 命令行配置git

注意git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然也可以对某个仓库指定不同的用户名和Email地址。

2.5.2 Eclipse里配置git

Preferences - Team - Git - Configuration



3 使用git

3.1 从远程仓库克隆

在公司，我们经常会遇到有一个项目，使用git进行分布式管理，领导让你写某个模块，然后告诉你项目的git地址，然后让你干活。

是不是一脸茫然。项目没给我怎么让我干活。别急。我来告诉你怎么做。

首先，什么是git地址

<https://git.oschina.net/wkq278276130/git_test.git>

[git@git.oschina.net:wkq278276130/git\_test.git](mailto:git@git.oschina.net:wkq278276130/git_test.git)

上面的地址就是2个git地址，他们是同一个项目的两个地址。你一定在奇怪，一个项目有2个地址，怎么来区分。他们其实是一个地址，只是使用不同的协议而已。第一个使用https协议，安全，第二个使用ssh协议，速度快，不用输密码，(如果使用ssh协议，需要配置SSH KEY)。

温馨提示，咱们公司的gitlab，使用的是http协议和ssh协议。

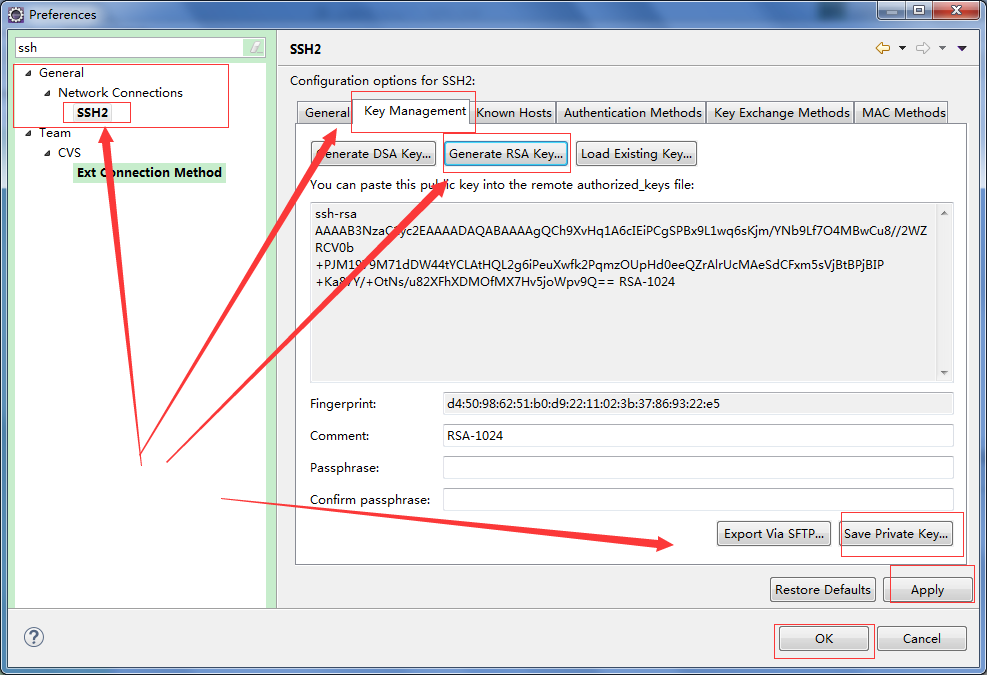
3.1.0 生成并配置SSH KEY

通过命令生成SSH Key

ssh-keygen -t rsa -C ["weikeqin@bonc.com.cn"](mailto:\"weikeqin@bonc.com.cn\") 用的时候记得邮箱写自己的

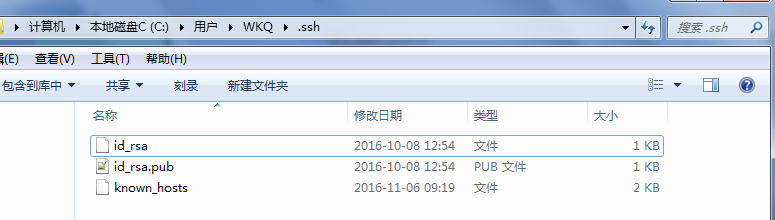
按3个回车，密码为空

通过eclipse生成SSH Key

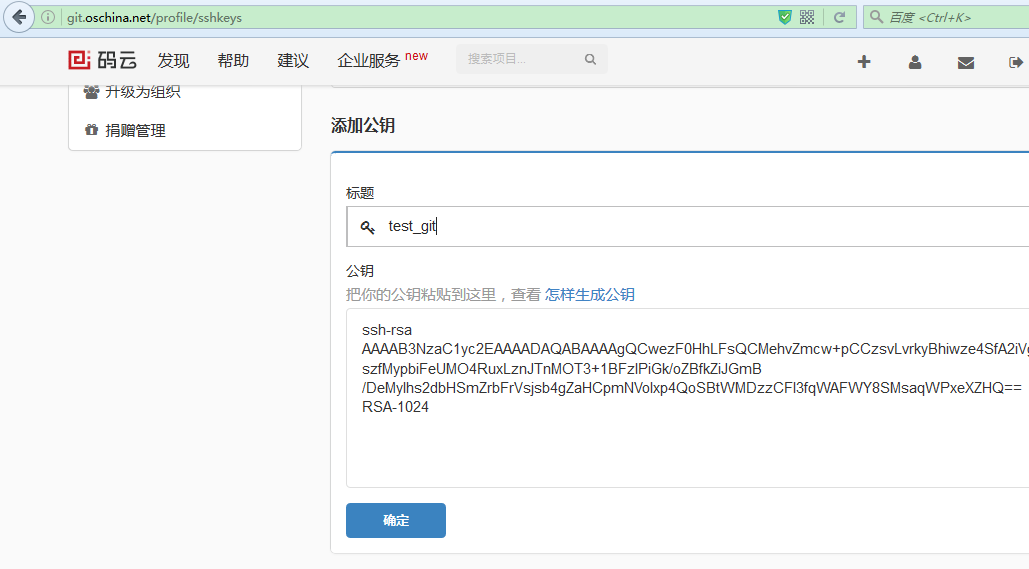


生成SSH KEY 以后，会保存在系统盘当前用户的.ssh目录下。

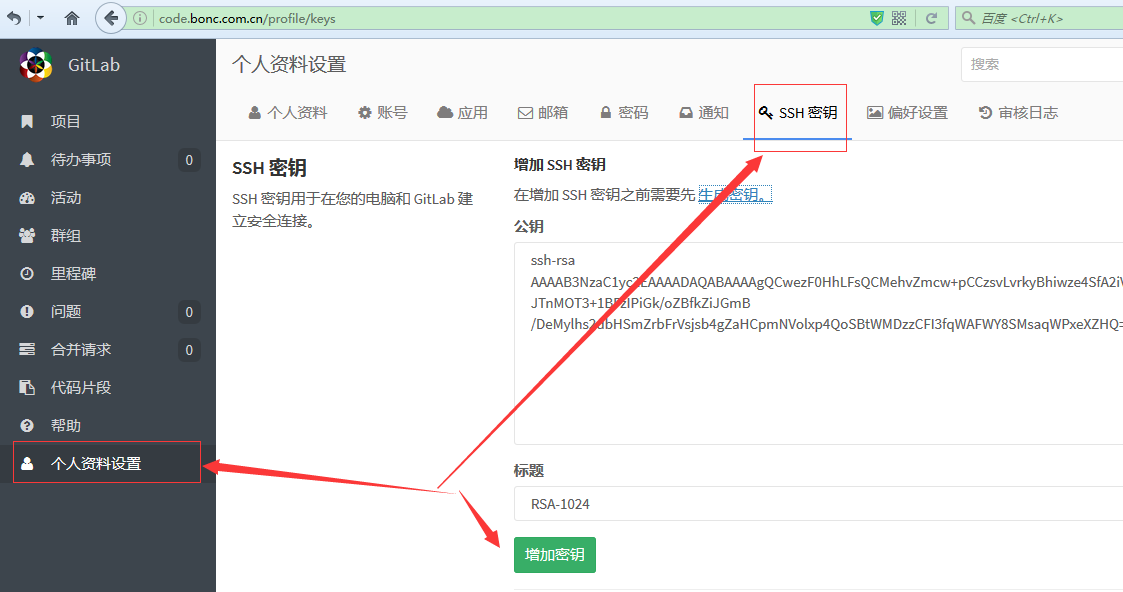
Id\_rsa 是密钥 is\_rsa.pub是公钥，我们需要把公钥添加到git服务器上



Git.oschina 添加SSH KEY 把is\_rsa.pub里的内容复制到公钥框里就可以了

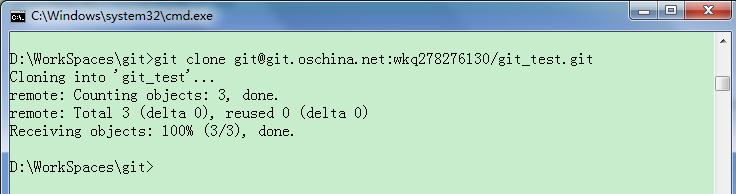


Gitlab 添加SSH KEY 把is\_rsa.pub里的内容复制到公钥框里就可以了



3.1.1 使用命令行克隆

git clone [git@git.oschina.net:wkq278276130/git\_test.git](mailto:git@git.oschina.net:wkq278276130/git_test.git)



克隆完，就可以在我本地的 D:/WorkSpaces/git/ 下找见 git\_test目录

3.1.2 使用eclipse克隆

3.1.2.1 只克隆到本地

先在eclipse里找到git视图，然后通过这个来下载git仓库，这个例子中我是用ssh协议来克隆，因为使用ssh协议速度快，而且不用输密码。

Windows - show views - other - git repositories

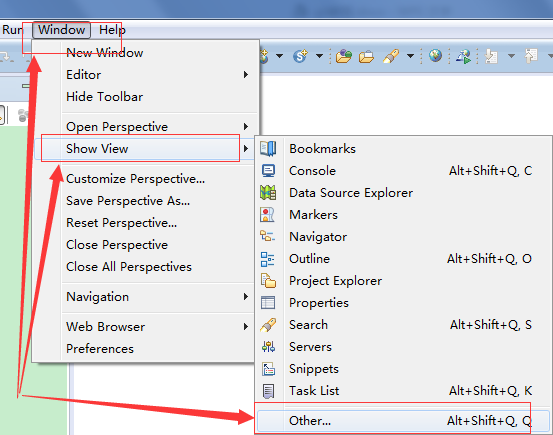


图3.1.2.1\_1 在eclipse里找git视图

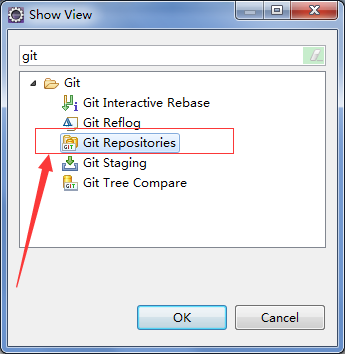


图3.1.2.1\_2 在eclipse里找git视图

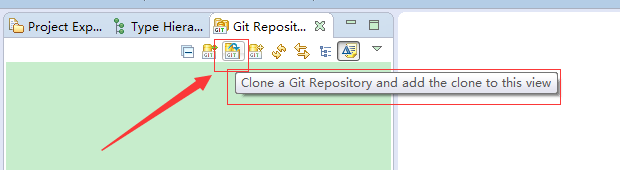


图3.1.2.1\_3 通过git 视图来下载git仓库

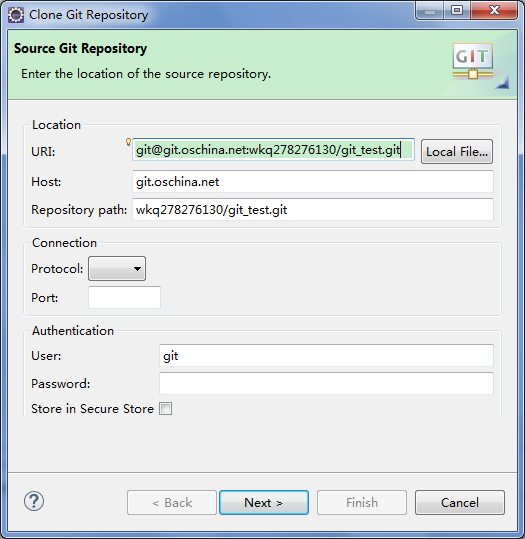


图3.1.2.1\_4 通过git 视图来下载git仓库

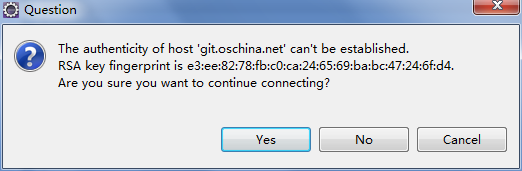


图3.1.2.1\_4 把地址添加到本地

这个是ssh协议里的内容，意思和git.oschina.net这个地址建立联系了，RSA key是XXXX，询问你是否通过这个连接。点YES就可以。

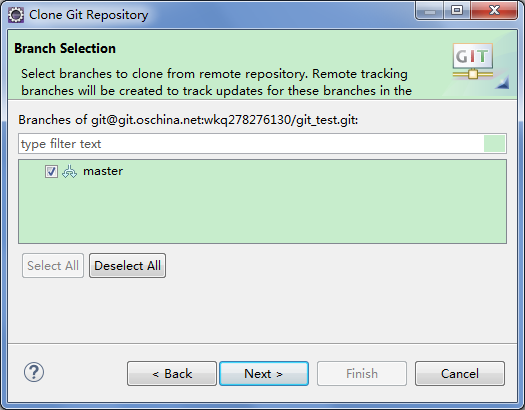


图3.1.2.1\_5 远程仓库里的内容

选择master分支(主分支)

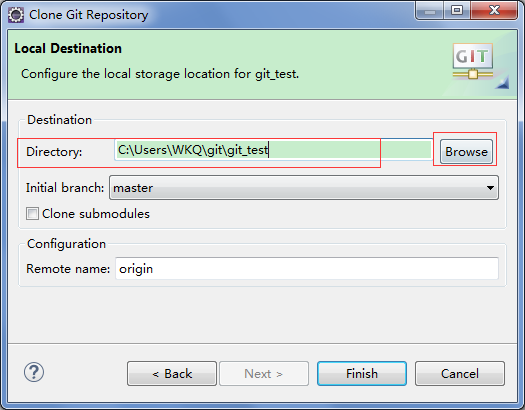


图3.1.2.1\_6 选择仓库保存到本地的位置

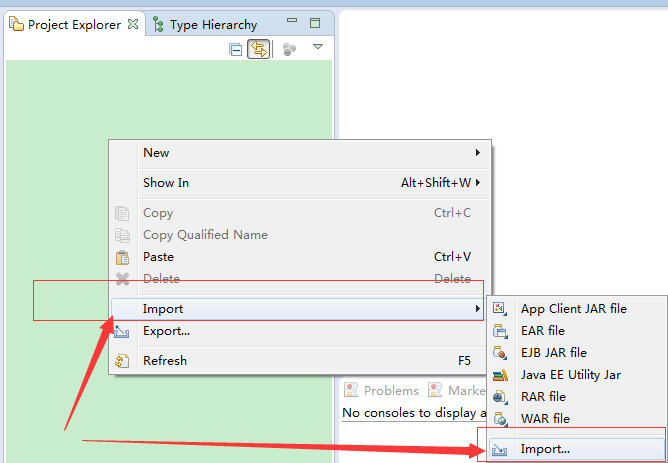
在这个页面你可以选择仓库(项目)保存的位置，默认是保存到系统盘当前用户的git文件夹下。

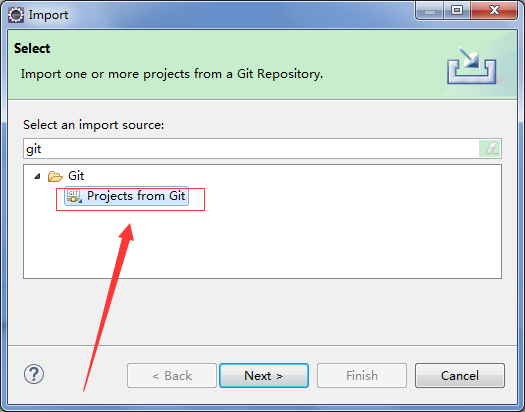
点击finish后就把仓库（项目）复制到本地了。

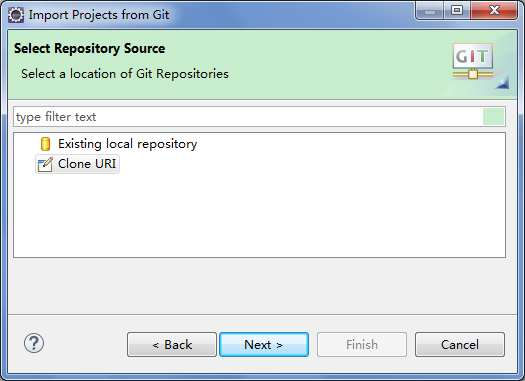
把仓库导入到本地就算完成了。

3.1.2.2 克隆到本地并且导入eclipse

这个步骤相当于把 克隆 和 导入项目到eclipse 这两个步骤合二为一了，其实原理都一样。







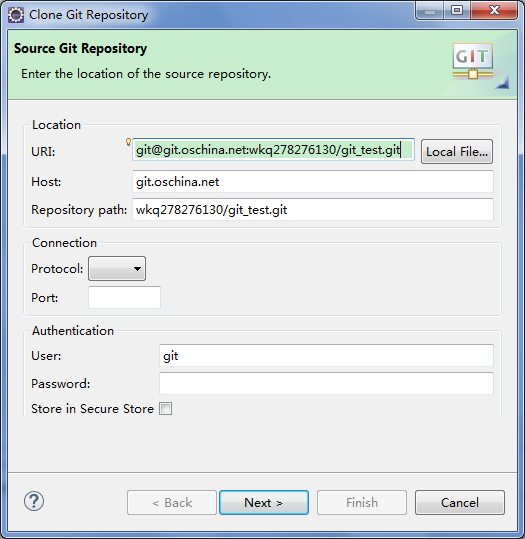


图3.1.2.1\_4 通过git 视图来下载git仓库

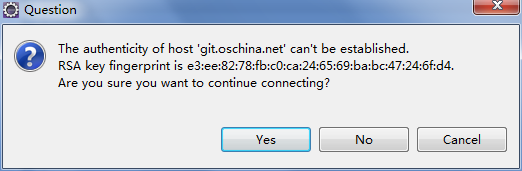


图3.1.2.1\_4 把地址添加到本地

这个是ssh协议里的内容，意思和git.oschina.net这个地址建立联系了，RSA key是XXXX，询问你是否通过这个连接。点YES就可以。

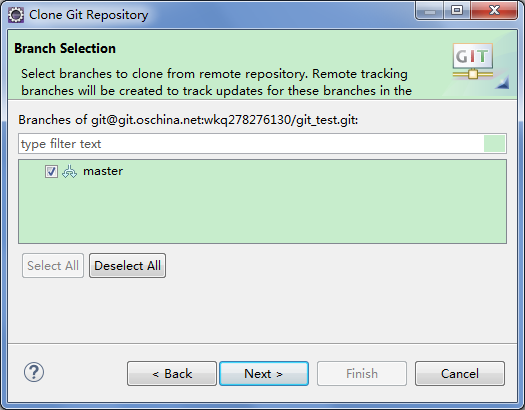


图3.1.2.1\_5 远程仓库里的内容

选择master分支(主分支)

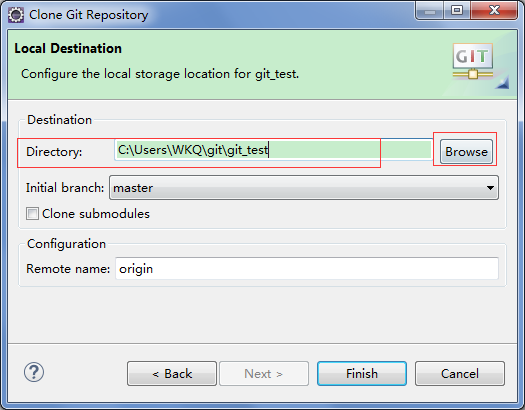
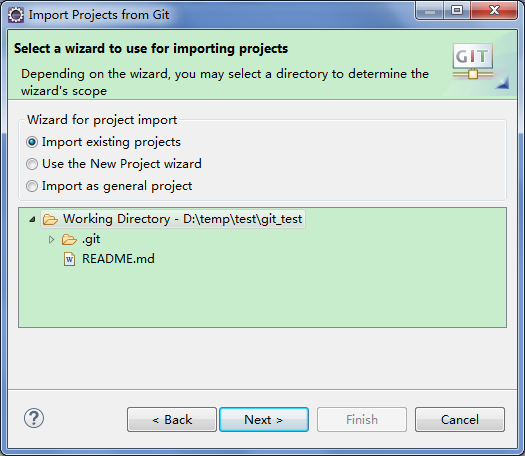
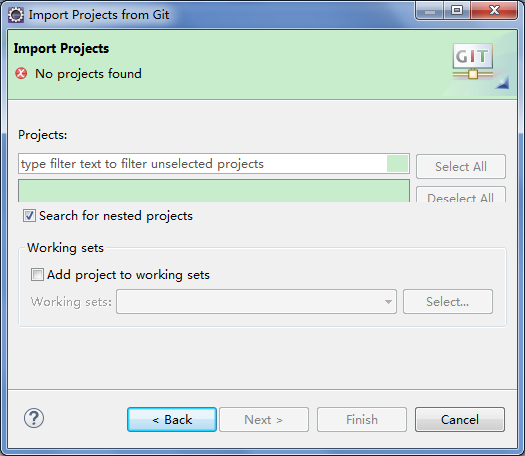


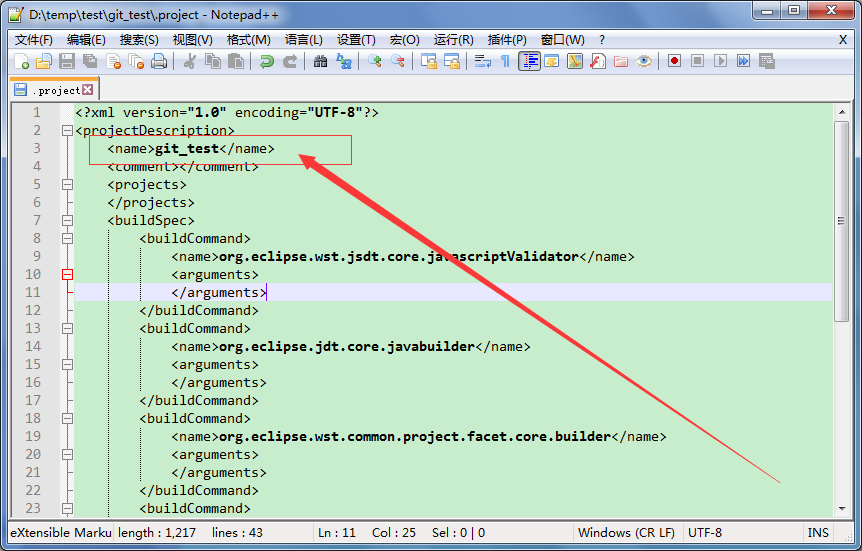
图3.1.2.1\_6 选择仓库保存到本地的位置

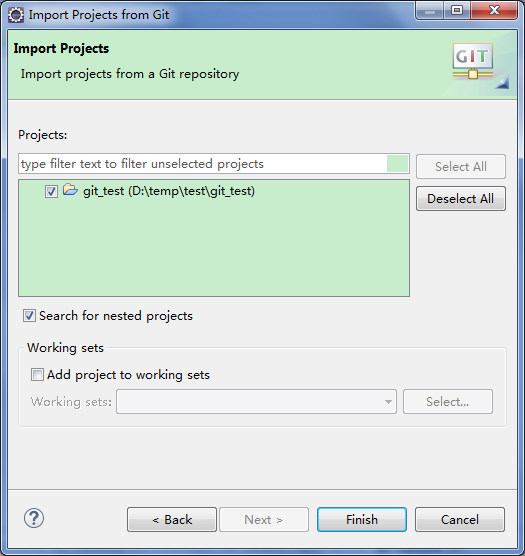
在这个页面你可以选择仓库(项目)保存的位置，默认是保存到系统盘当前用户的git文件夹下。





这个时候eclipse可能会提示 no prjects found，不要着急，复制一个.project文件过来骗骗eclipse就可以了。记得把.project文件里的项目名改了。





点击finish后就把仓库（项目）复制到本地了。

把仓库导入到eclipse就算完成了。

3.2 对文件修改后提交

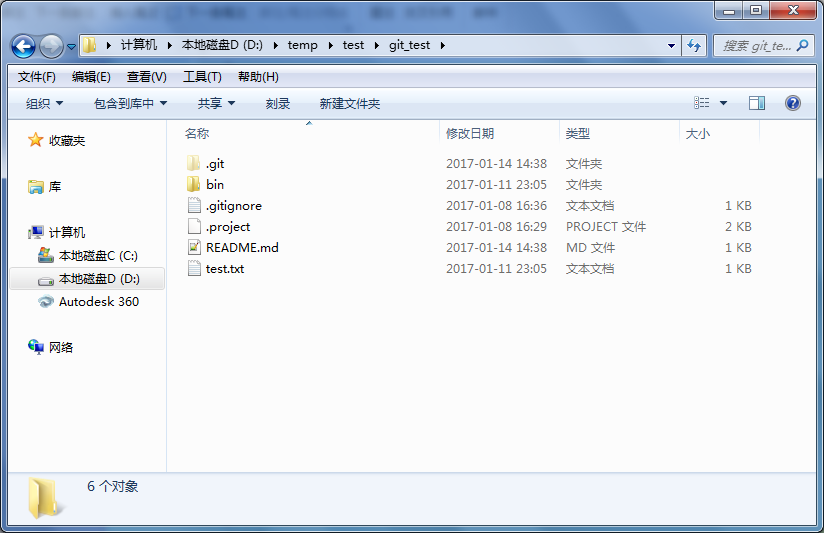
在公司里经常会遇到修改代码后提交的情况，修改代码后怎么提交呢。看下面的内容。

要使用git，还得理解工作区和暂存区的概念。

3.2.0 工作区和暂存区

工作区（Working Directory）

就是你在电脑里能看到的目录，比如我的git\_test文件夹就是一个工作区：



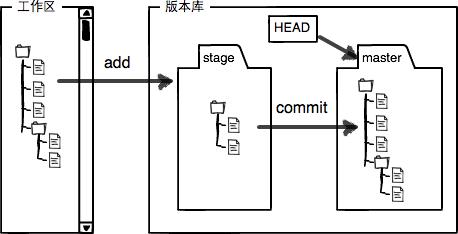
版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。

第一步是用git add把文件添加进去，实际上就是把文件修改添加到暂存区；

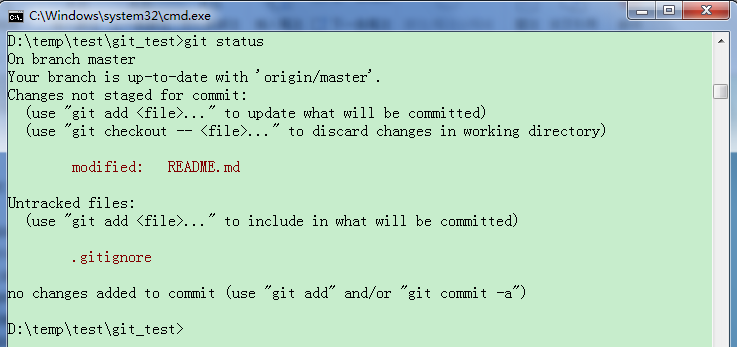
第二步是用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支。



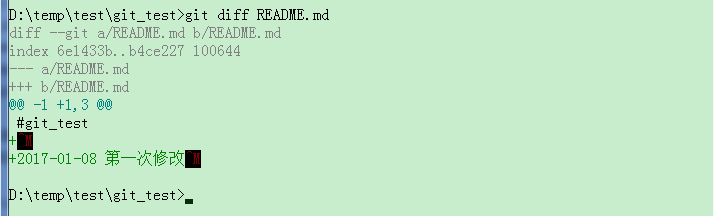
现在，我修改了README.md文件，在文件里添加了一句话，怎么提交修改呢

3.2.1 使用命令行提交修改

使用git status 查看仓库的状态，可以看到修改了README.md文件，添加了.gitignore文件

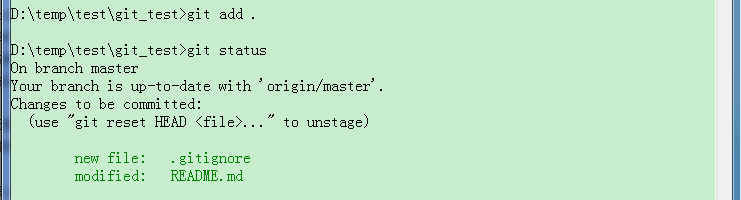


使用git diff 命令查看修改了什么内容



可以看到我添加了一行，图中绿色部分

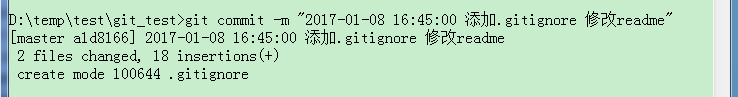
使用git add命令把文件标记为添加状态，意思是告诉git，这个文件我修改了

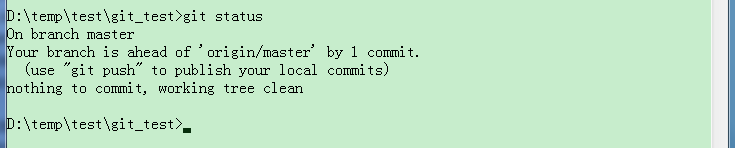


可以看到文件变为添加状态

用git commit命令把修改的提交。

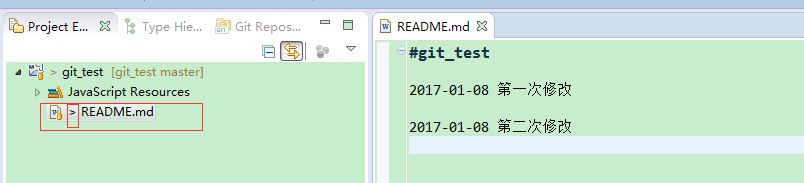
git commit是什么意思呢？ 你对文件进行修改，然后提交修改到版本库里，就好比玩RPG游戏时，每通过一关就会自动把游戏状态存盘，如果某一关没过去，你还可以选择读取前一关的状态。有些时候，在打Boss之前，你会手动存盘，以便万一打Boss失败了，可以从最近的地方重新开始。Git也是一样，每当你觉得文件修改到一定程度的时候，就可以“保存一个快照”，这个快照在Git中被称为commit。一旦你把文件改乱了，或者误删了文件，还可以从最近的一个commit恢复，然后继续工作，而不是把几个月的工作成果全部丢失。

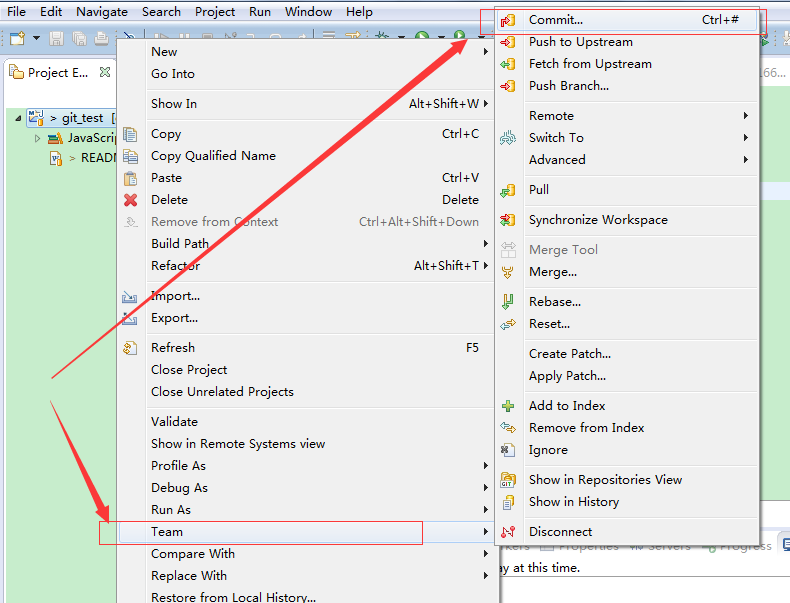


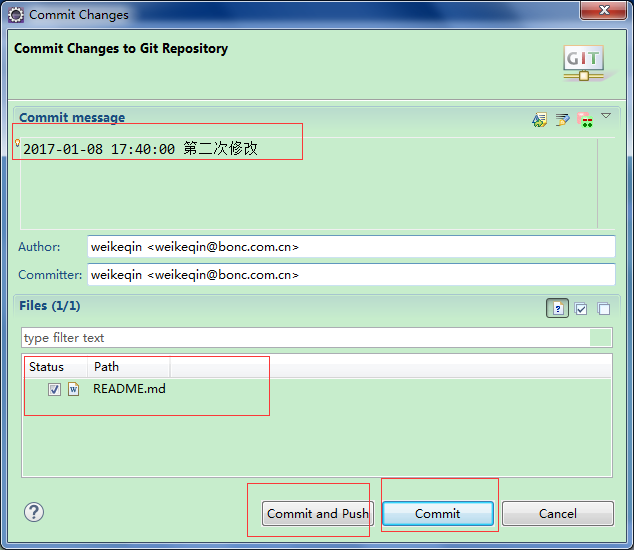


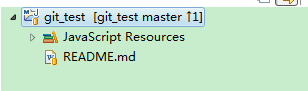
提交后可以看到比远程仓库领先一个版本。

3.2.2 使用eclipse提交修改





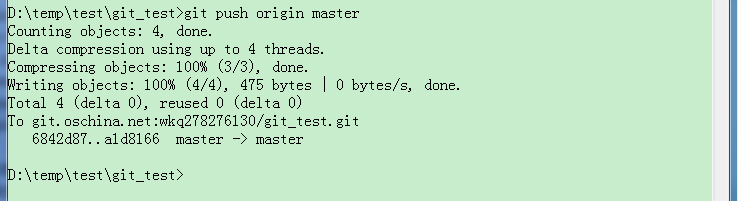




3.3 把修改提交到远程仓库

3.3.1 使用命令行把修改提交到远程仓库

使用 git push 命令把修改提交到远程仓库

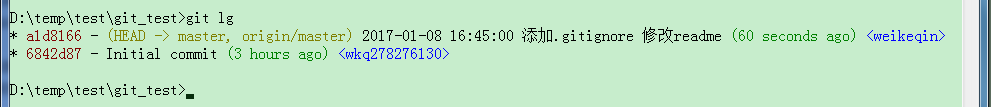


可以看到，把本地的修改提交到远程仓库了。

使用git lg 命令查看修改历史

lg是配置的别名，具体配置如下

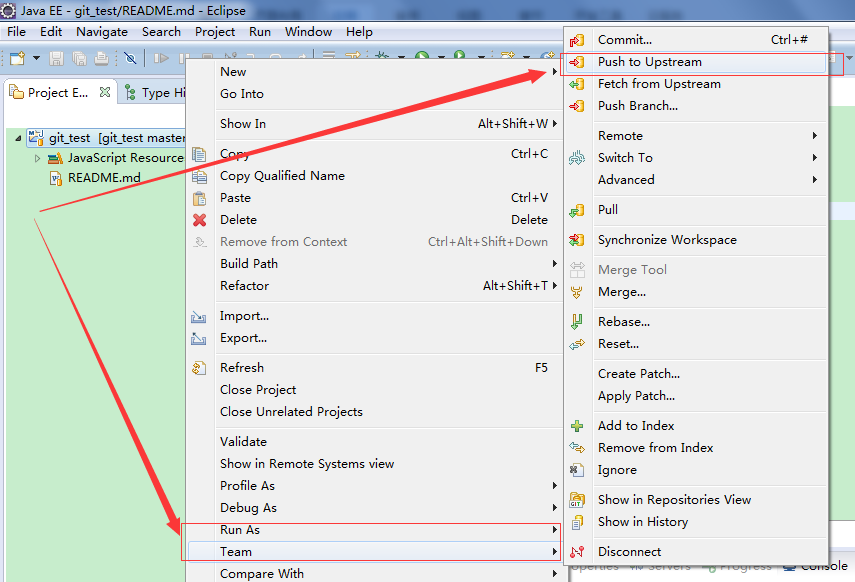
git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

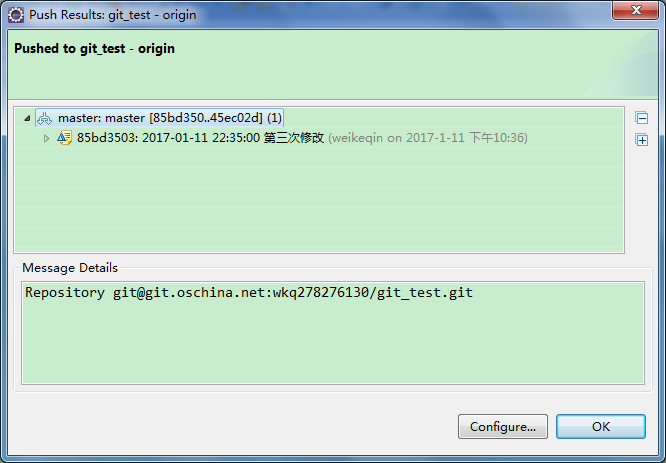


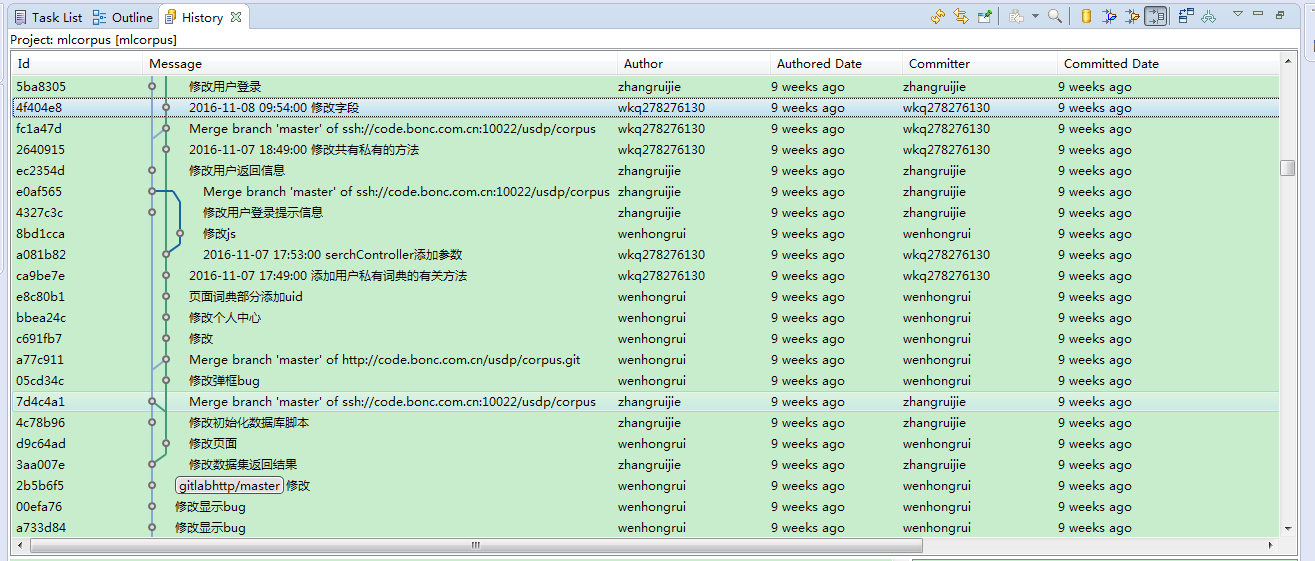




3.3.2 使用eclipse把修改提交到远程仓库



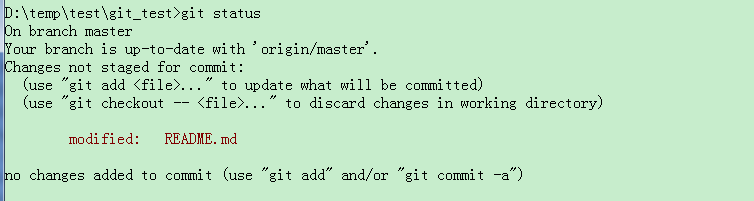




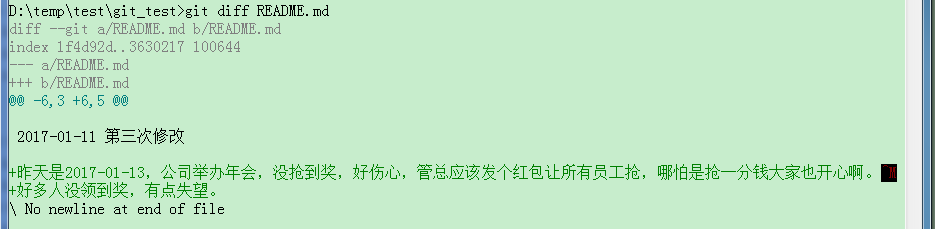
3.4 想撤销修改怎么办

3.4.1 用命令行撤销修改

用git status 查看文件状态



用 git diff 命令查看修改了什么内容



可以看到添加了两行内容（图中绿色的部分）

这次的修改不能提交啊，要是让领导或者管总看到，不给我发年终奖了。怎么撤销这次的修改。

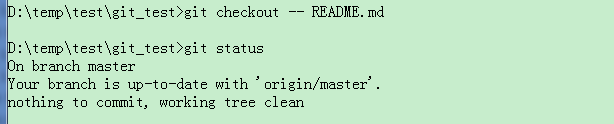
有2个办法

1. 自己手动删了。（适用于修改内容少的时候）

如果一次修改了多个地方，而且修改的地方不连续，就比较麻烦了

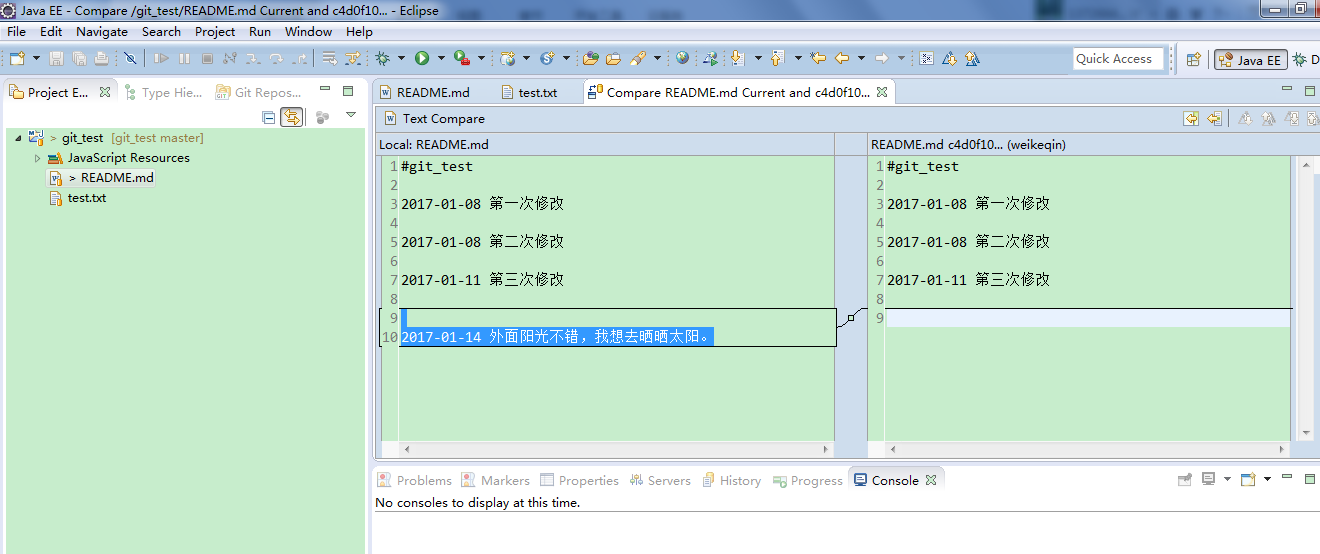
1. 用git撤销。（适用于任何时候）

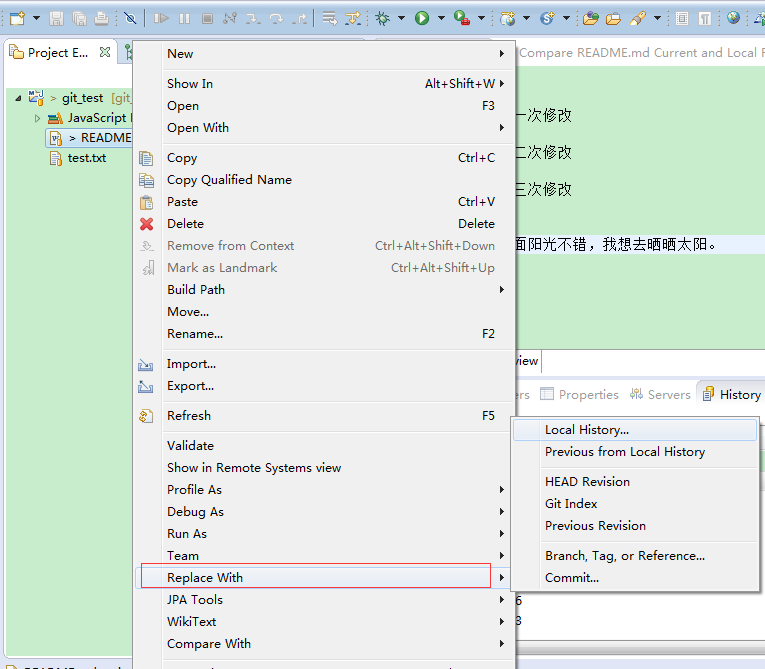
git checkout -- README.md



用git status命令查看，是不是刚才添加的内容撤销了

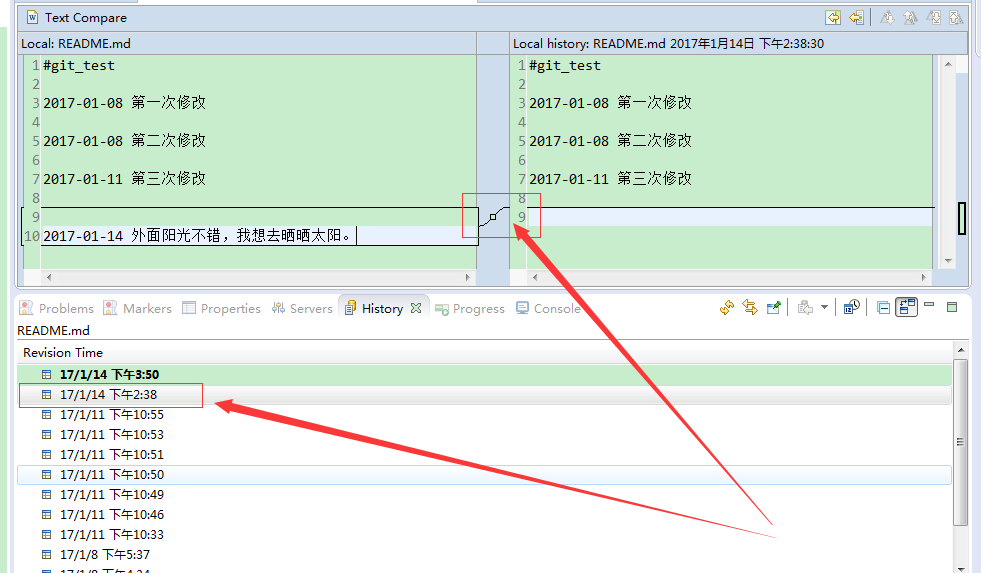
3.4.2 用eclipse撤销修改



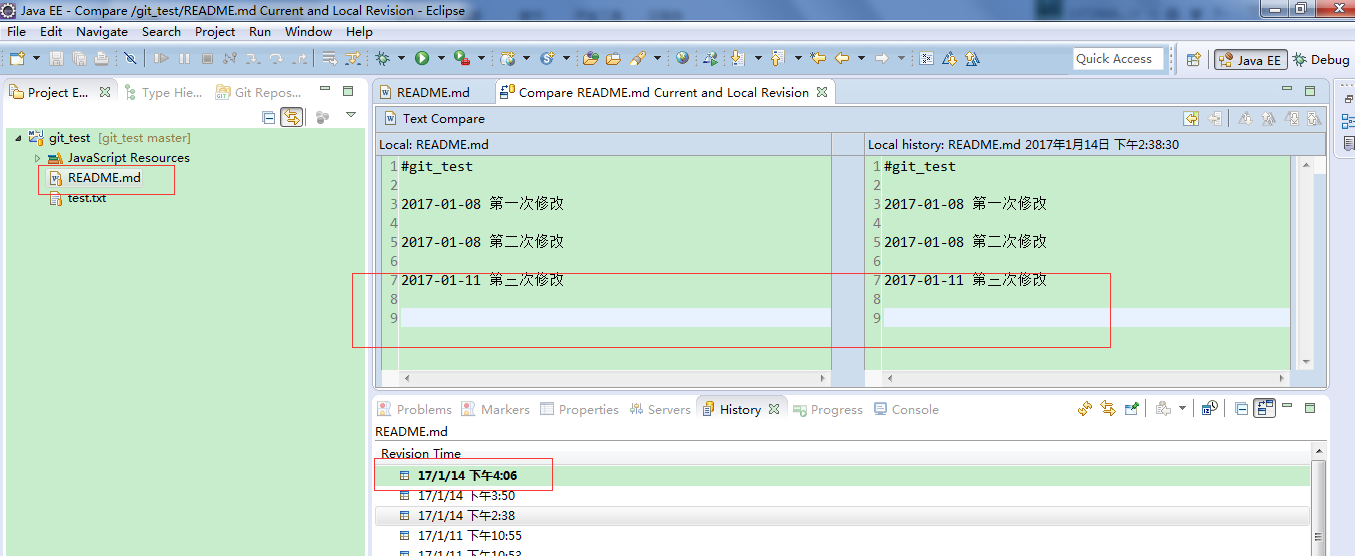


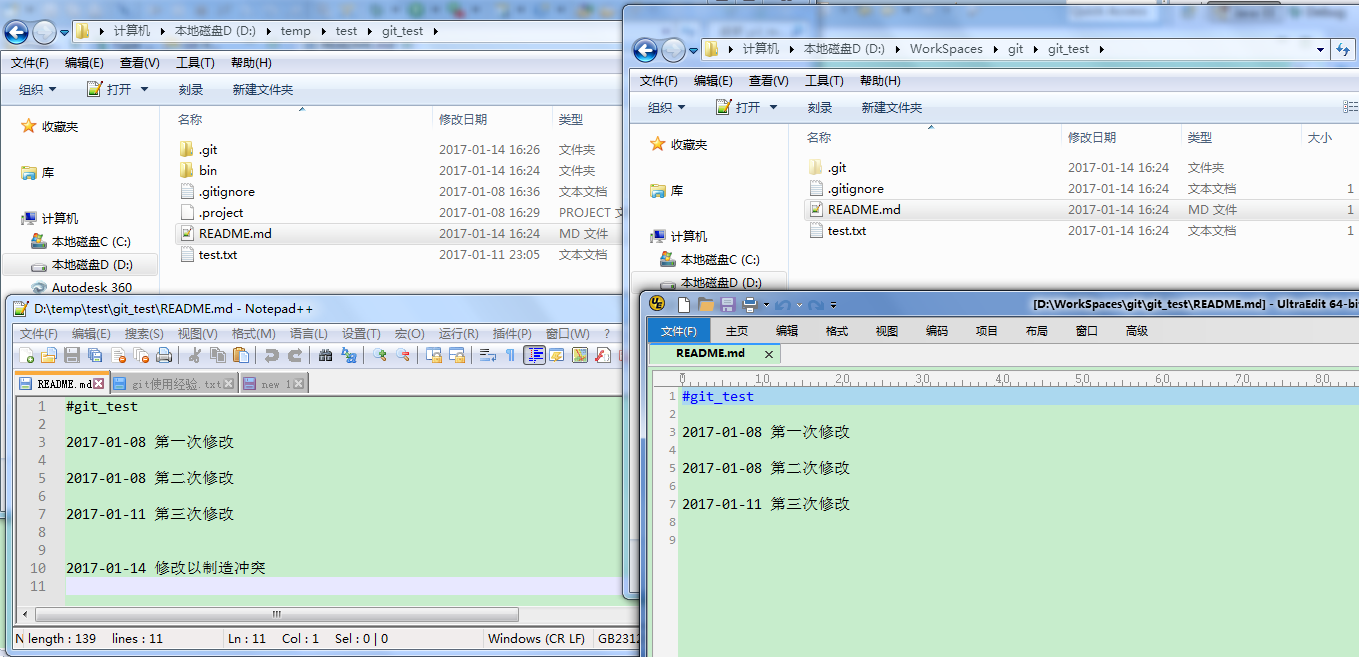
这儿有多种撤销方法，我介绍2种：

1. 在本次修改之前本地没有提交，可以使用 Replace With Local History 或 HEAD Revision
2. 在本次修改前已经提交过，使用Replace With Local History



撤消后，保存，如下图，可以看到，README.md里的修改撤销了，如下图，History里多出一条记录，左边的README.md也没有 > 的标志了。





3.5 冲突了怎么办

首先我告诉你怎么制造冲突，假如你和张三负责同一个模块，你们两个都需要对README.md文件修改，两个人都修改了README.md文件，然后张三先提交并且推送到远程仓库(远程服务器)，然后你提交的时候会发现冲突。

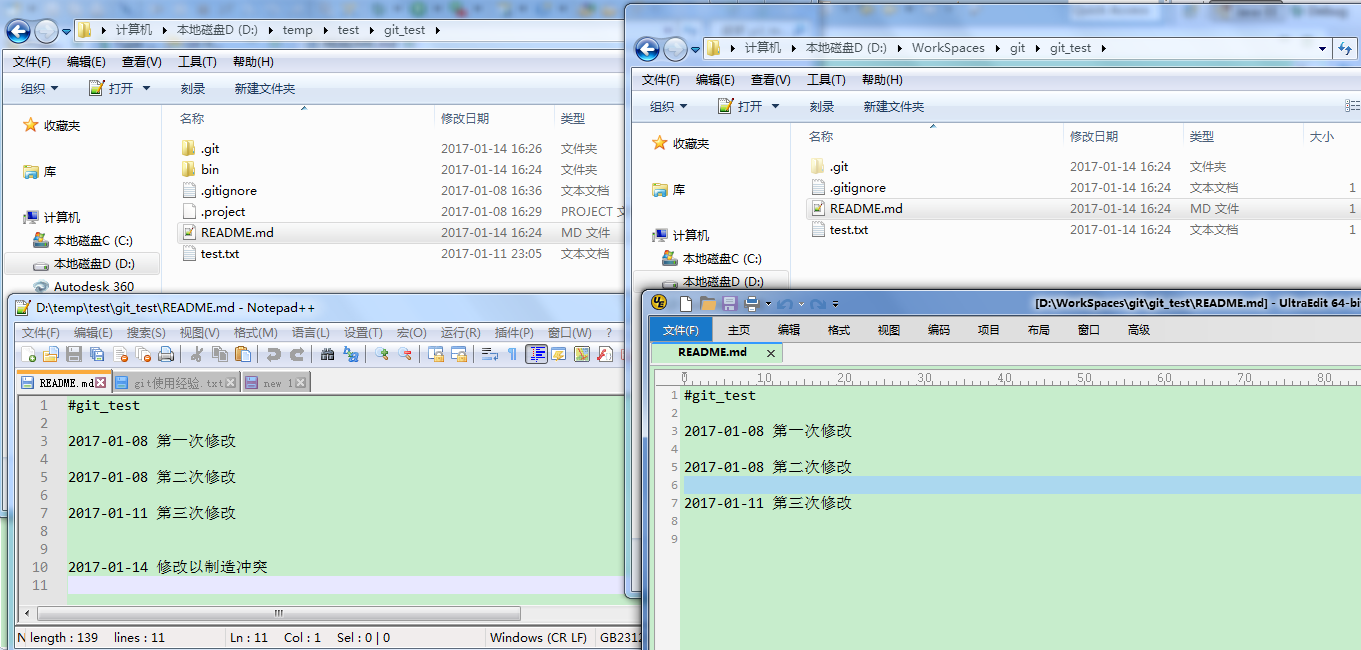
简单地说，两个人”同时”对同一个文件修改时，会导致冲突。 （这里的同时相对于上一次提交）

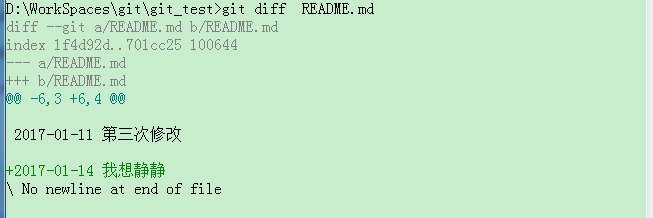
同时要告诫大家，每天上班了先更新代码，修改代码前更新代码，每天下班后及时提交代码。这么做是为了避免冲突。（因为解决冲突是一件没有成果，浪费工作时间，降低工作效率的事情）

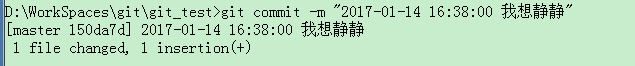
知道怎么导致冲突了，然后来告诉你遇到冲突了怎么办。

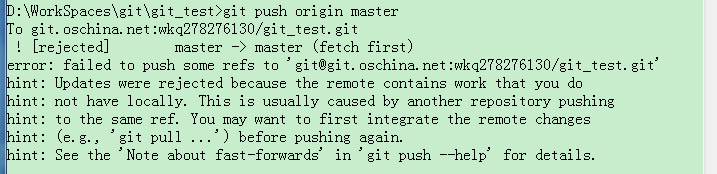
看下面这张图，同一个仓库(项目)，我在电脑上存了两份，会用git的人一般没人这么干，也就我这么无聊，为了一个人制造冲突，然后在电脑上存了两份。

可以看到左边的是比较新的，按照上面举的例子，是张三的，右边这个是比较旧的，然后我在旧的里面修改一些东西后提交。



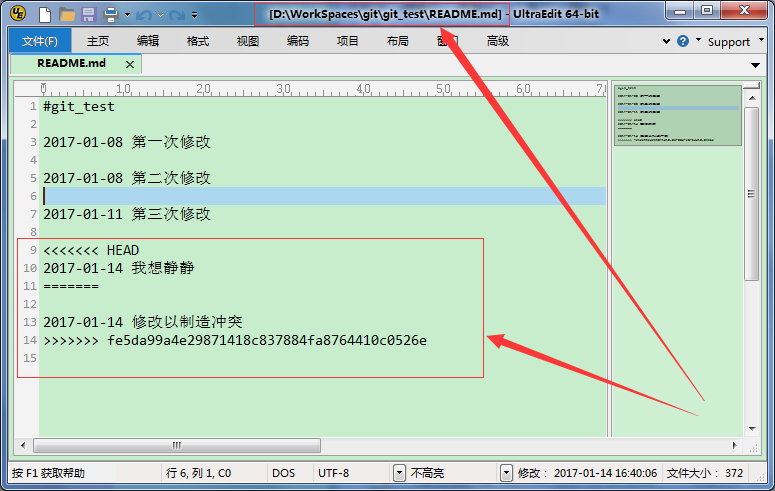






好了，制造下冲突了，这儿有两个办法，两个办法都可以解决，但是有个小小的不同

1. 使用git rease
2. 使用git merge



然后会发现README.md变成这个样子了

<<<<<< HEAD 和 ====== 之间是一个人修改的内容

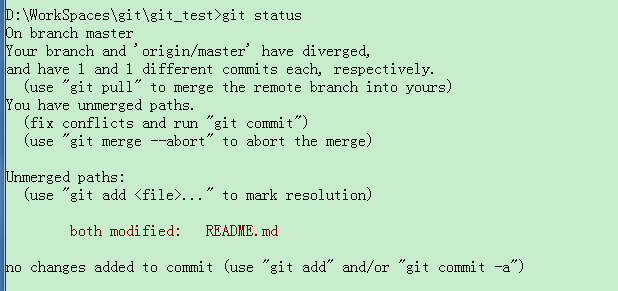
====== 和 >>>>>> 之间是另一个人修改的内容

咱们要把两个人修改的合并起来，也就是解决冲突。



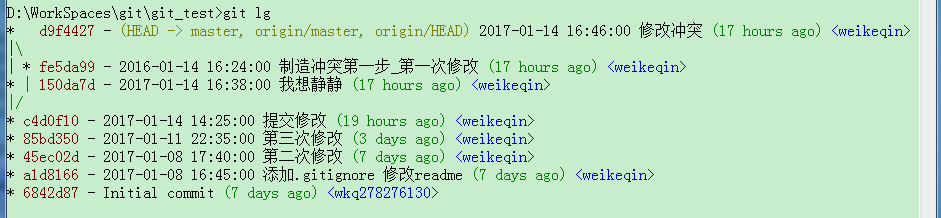
然后手动修改，上图是修改后的。可以看到冲突解决了。

PS：这次的冲突比较简单，在实际的项目里，冲突可能不止一个地方。



解决冲突后，然后保存修改，提交修改，推送到远程仓库（远程服务器）

顺便说一下，使用git lg命令查看解决冲突后的信息，如下图，可以看到fe5da99和150da7d曾经有冲突



3.6 本地提交完想撤销这次提交怎么办

我们可能会遇到这种情况，领导让加一个功能，然后你开始写代码，写完以后在本地提交（也就是add commit 还没有push），领导对你说，这个功能不需要了，然后，我明白你的心情。（此刻，你需要的不是菜刀，而是镇定和技术）。

3.6.1 用命令行撤销这次提交

这个时候，你首先要知道你提交的版本和你要恢复的版本。

使用git lg命令来查看你提交过的版本，也可以使用git reflog命令来查看提交过的版本



假如f8229b3是我新添加的功能，领导说不要了，那么，我们要恢复到d9f4427这次提交。

使用git reset --hard commit\_id

备注：git reset 有多个选项，git reset [--hard|soft|mixed|merge|keep] [<commit>或HEAD]，

这儿我使用git reset --hard



如图，此时版本恢复到d9f4427这次提交了，也就是版本回退到d9f4427了，看一看自己的文件/代码，是不是刚才新加的功能没有了。

如果领导又变卦了，说，刚才添加的功能还是加上吧，你是不是要疯了，不要急，

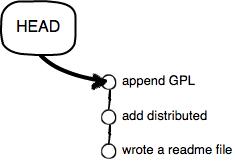
使用git reflog命令查看提交的版本，再次使用git reset命令回到你想要的版本。

git reset --hard f8229b3，自己试一试，是不是刚才添加的功能又回来了。

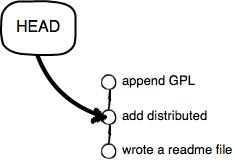


偷个懒，图片拷自网上git廖雪峰教程

Git的版本回退速度非常快，因为Git在内部有个指向当前版本的HEAD指针，当你回退版本的时候，Git仅仅是把HEAD从指向append GPL：



改为指向add distributed：



然后顺便把工作区的文件更新了。所以你让HEAD指向哪个版本号，你就把当前版本定位在哪。

3.6.2 用eclipse撤销这次提交

和3.4.2 用eclipse撤销修改 的方法一样

3.7 把服务器的版本回退

如果哪一天，领导告诉你需要把服务器上的版本回退了，我觉得要做的第一件事不是立刻把服务器上的版本回退，而是等一等，想一想，商量商量，是否真的要回退。如果真的要把服务器上的版本回退。说明你们白干了XX天。这个是要在工作中极力避免的事情。

想把服务器的版本回退，有多种办法，

1. 先用在服务器上找到旧版本的代码，用旧版本的代码覆盖现在的代码，然后提交，推送到服务器，OK。 这个办法比较笨，但是对不熟悉git的人特别管用。

2. 使用git reset方法回退，并且强推。 不推荐

3. 使用git revert方法回滚，提交。 推荐

3.7.1

上一个是笨办法，这个当然简单了，首先按照3.6的使用git reset回退到你想回退的那个版本，然后使用 git push origin HEAD --force （把本地的强制推送到远程服务器），然后服务器的版本就会回退。 但是这种做法有缺点

git reset 和 git revert 的区别

reset是指将当前head的内容重置，不会留任何痕迹。

git reset --hard HEAD~3 会将最新的3次提交全部重置，就像没有提交过一样。

根据--soft --mixed --hard，会对working tree和index和HEAD进行重置。

revert是撤销某次提交，但是这次撤销也会作为一次提交进行保存(这样就不会丢失原来修改过，但是没有提交的内容)

3.7 分支管理

现在有一个项目由你负责，这个项目现在在2.0版本，有6个模块，现在领导让你先做模块1，然后你开始写代码。然后，过了一会，领导让你写模块2，你模块1还没有写完，还没有完成，也没有测试，不能提价，然后过了一会，忽然领导又说，1.0版本发现一个bug要赶快改。完了，这怎么办。

不要慌，git已经把办法给你想好了。

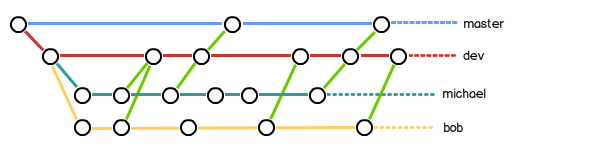
在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活。

那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

所以，团队合作的分支看起来就像这样：

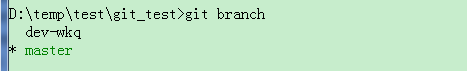


3.7.1 在命令行使用分支

git branch dev-wkq 创建新分支dev-wkq



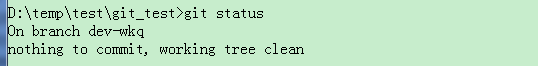
git branch 查看分支



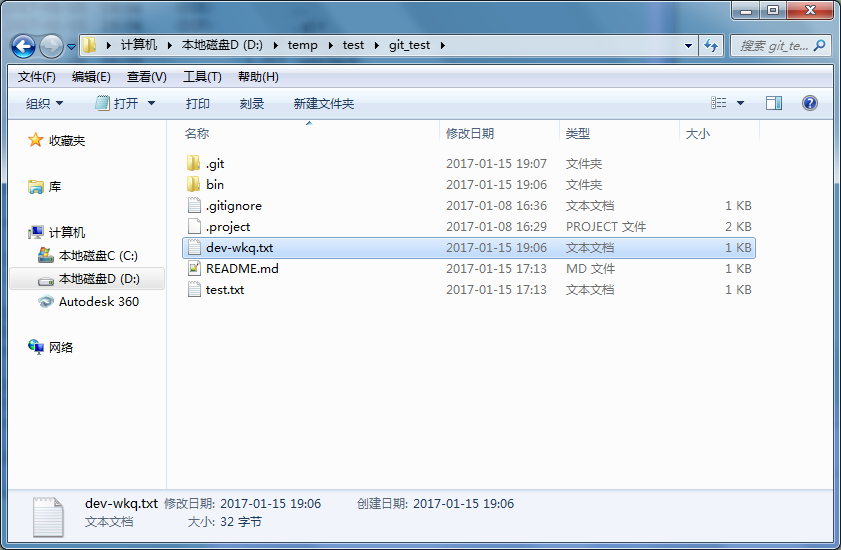
git checkout dev-wkq 切换到分支dev-wkq

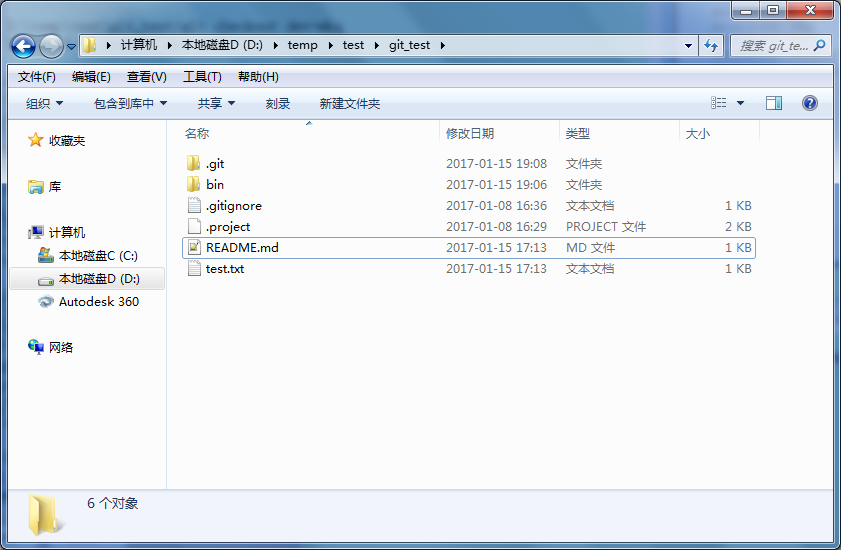


git status 查看仓库状态

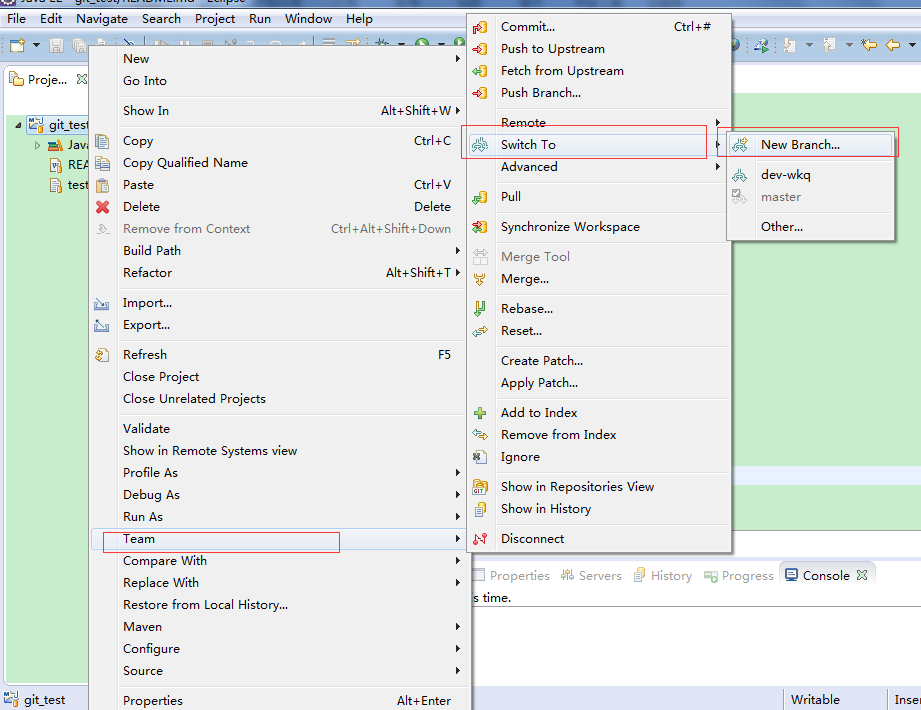


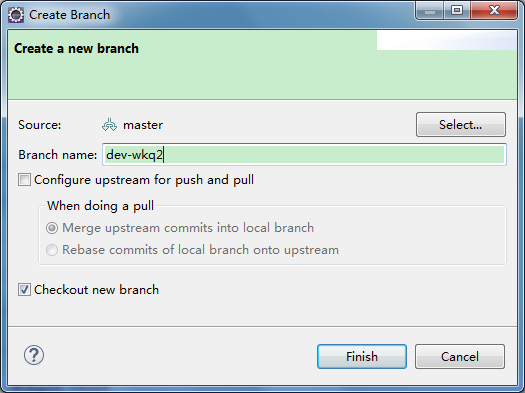
我在dev-wkq分支上添加一个dev-wkq.txt文件，添加内容并提交。



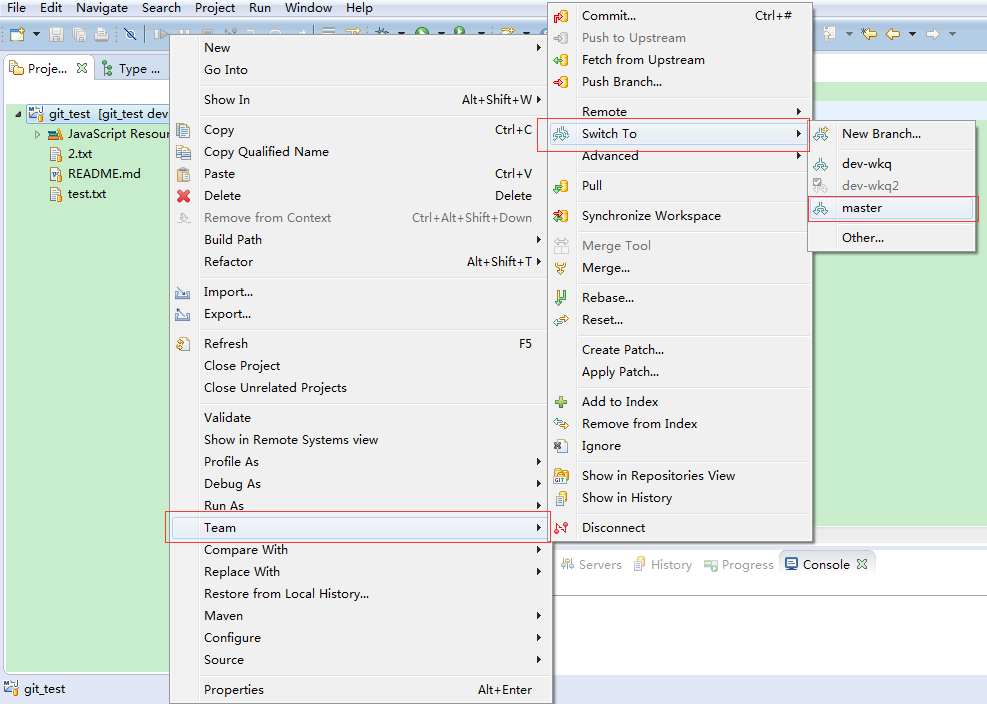


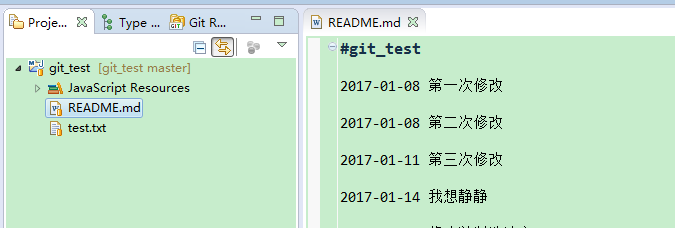
3.7.2 在eclipse里使用分支

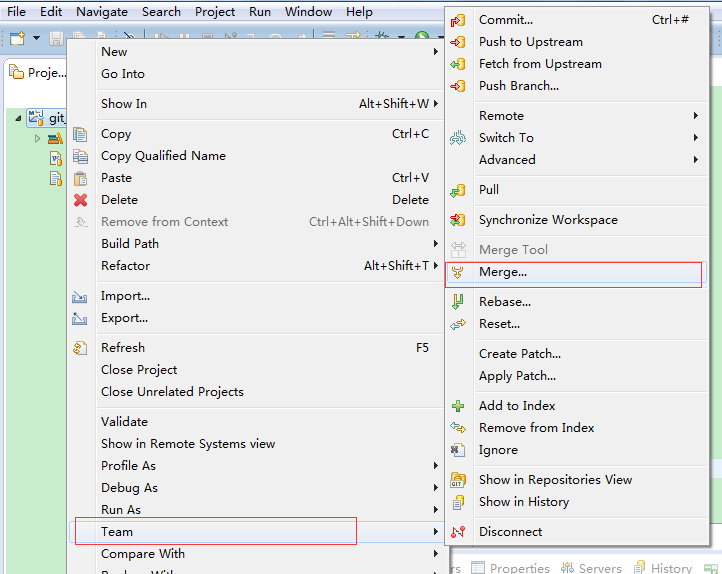


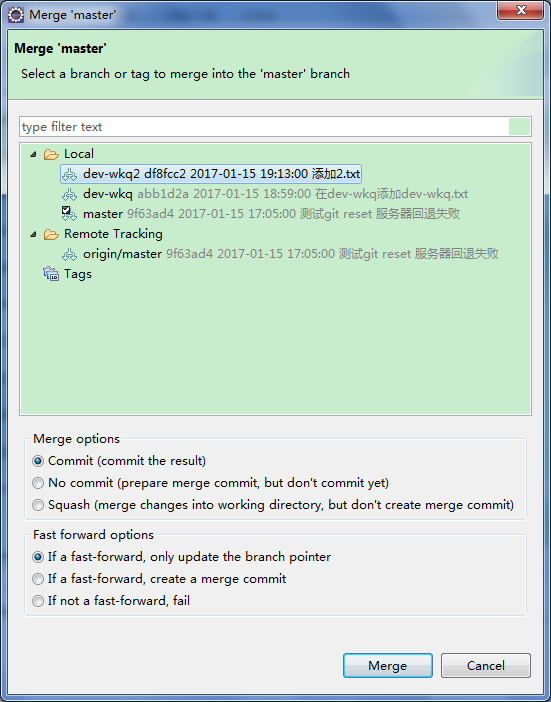




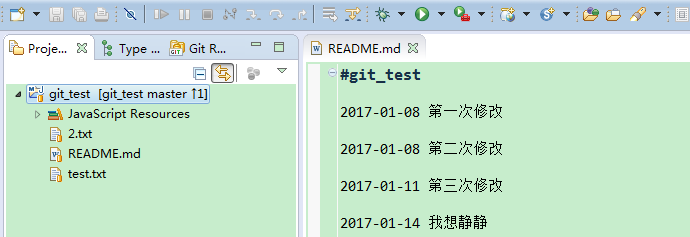










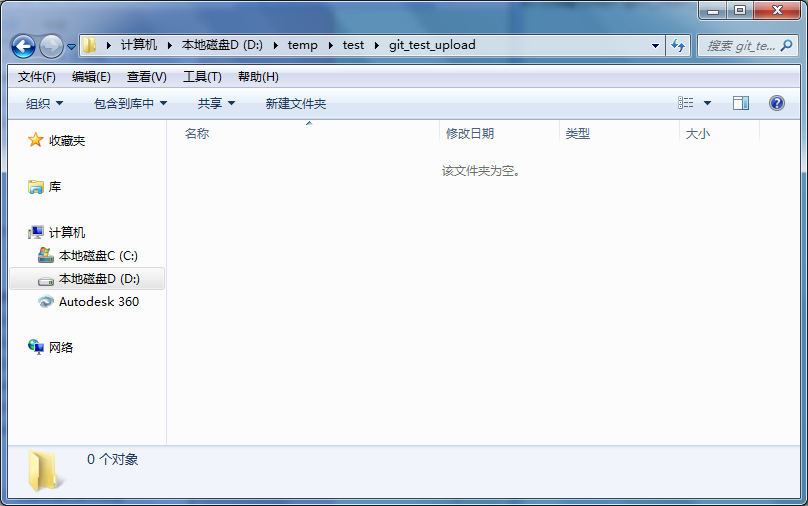


3.8 上传新项目到gitoschina或gitlab

3.8.1 在命令行上传

git是对文件操作的，所以上传项目可以认为是上传文件。

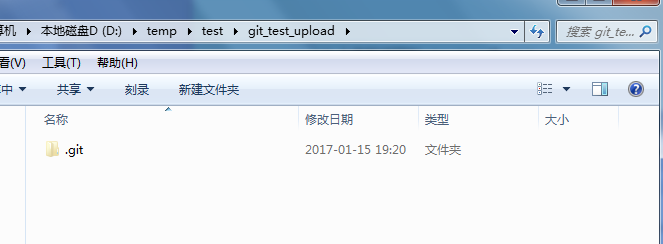
首先来看，D:\temp\test\git\_test\_upload 这个是一个空目录。



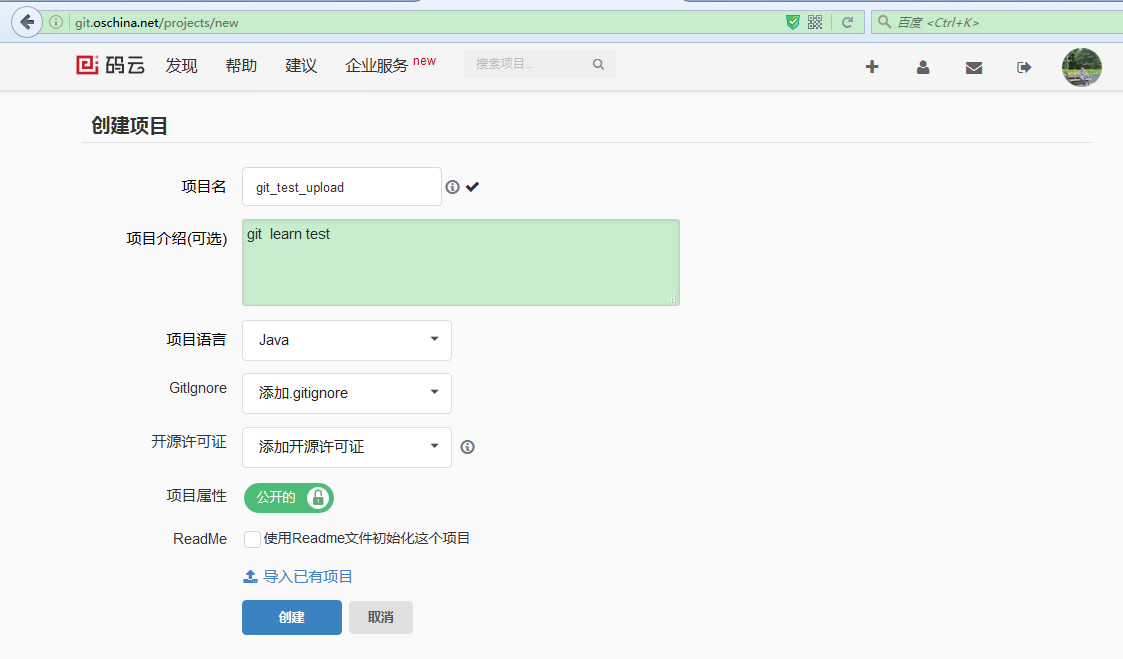
git init 是把该目录变为git仓库



可以看到多了一个隐藏的.git目录，这个目录非常重要，里面保存的是修改的快照及操作等。



我在gitoschina创建一个项目，名字可以随便起，但是一般用你要上传的项目名来命名。



复制gitoschina（服务器）上该项目的地址



git remote add origin [git@git.oschina.net:wkq278276130/git\_test\_upload.git](mailto:git@git.oschina.net:wkq278276130/git_test_upload.git) 把本地的git仓库和服务器上的建立关联

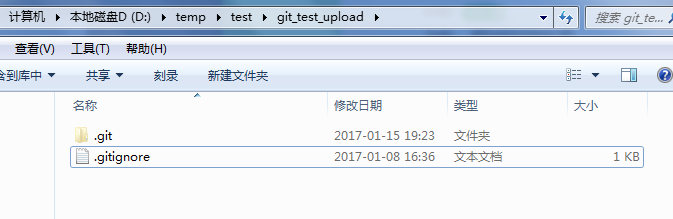


使用git pull来更新项目，（即使远程仓库没有提交，也要养成这个好习惯）

然后添加.gitignore文件（这个文件非常重要，如果不添加以后会遇到很多麻烦事）

这个文件的作用是 规定符合一些要求的文件或者目录不上传。

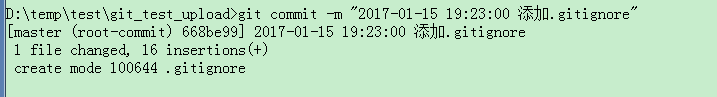
可以先从别的地方拷贝一份



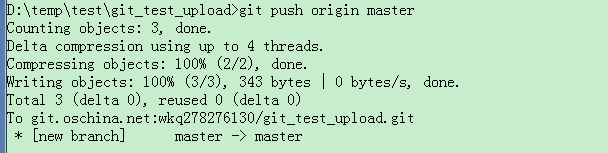
git add . （或者git add \*） 把所有文件标记为暂存状态



git commit 提交修改



git push -u origin master 推送到远程服务器 (-u 可以不写)

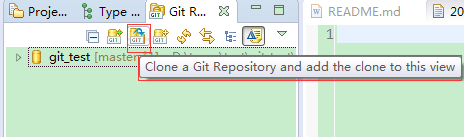


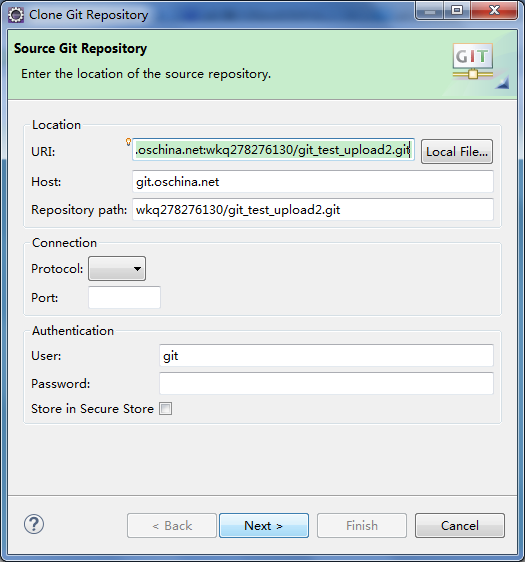
好了，新建的项目就放上传gitoschina(服务器)了。

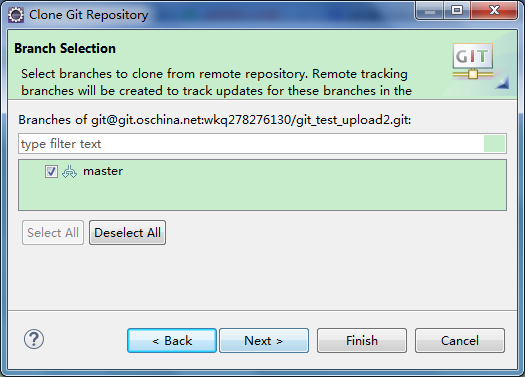
3.8.2 在eclipse里操作

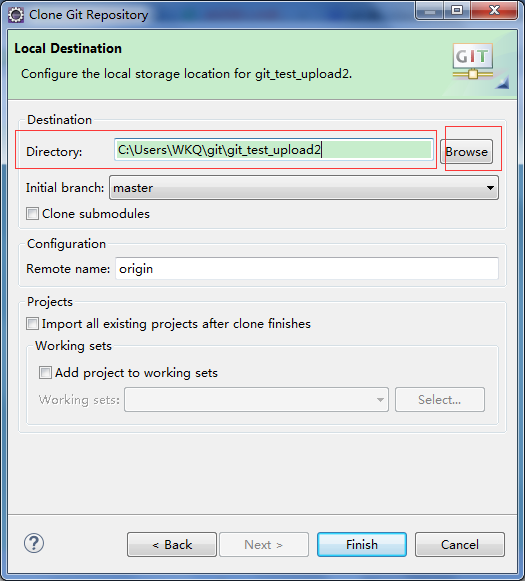
先在gitoschina新建一个项目，在项目里添加一个文件，比如README.md，否则用eclipse克隆的时候会提示你仓库为空。

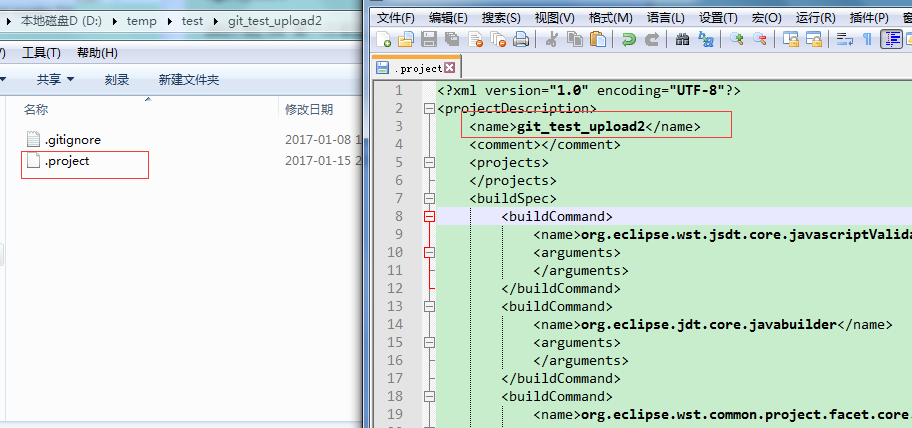


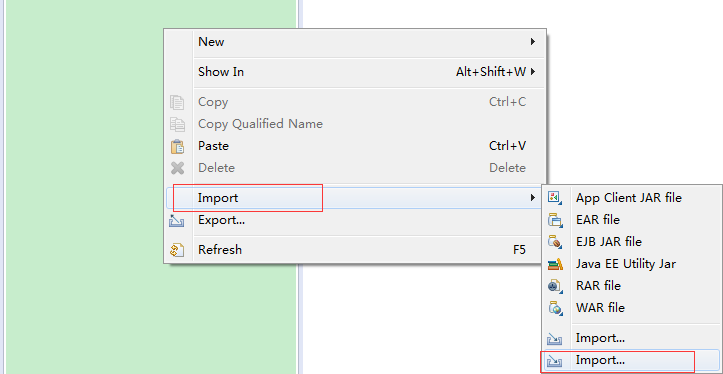




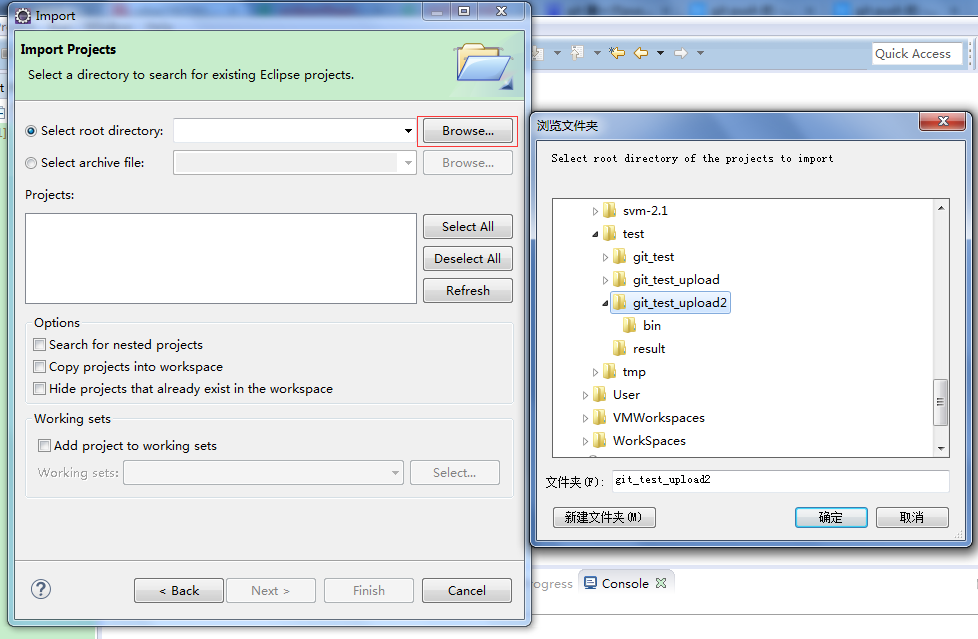


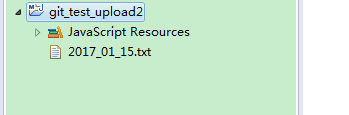


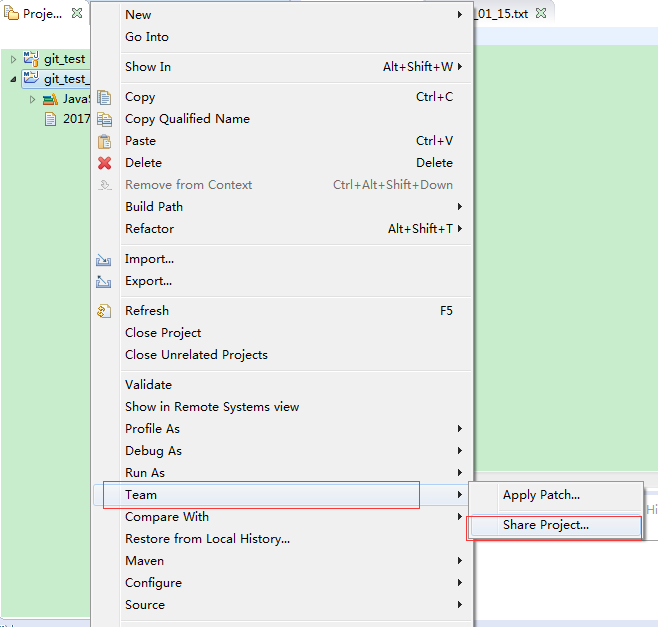


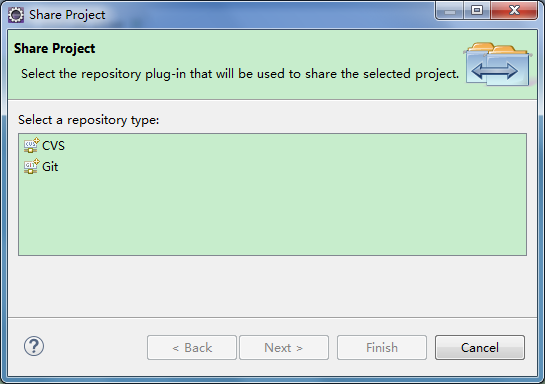


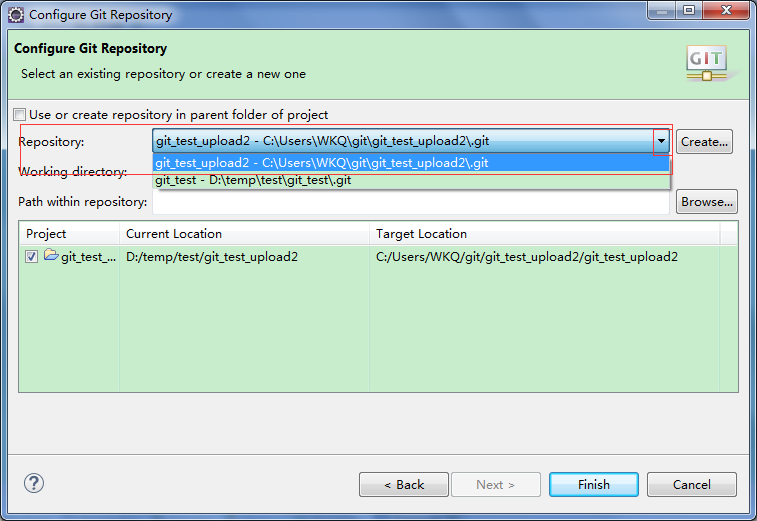




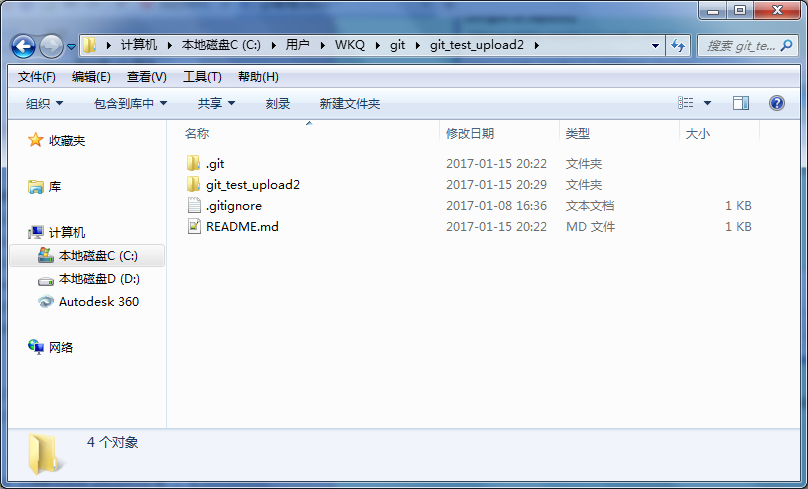




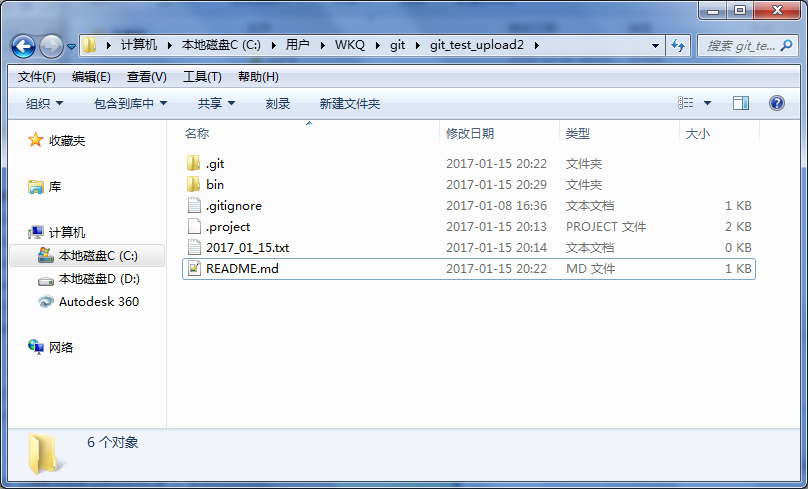




好了，这个就把本地的项目（D:/temp/test/git\_test\_upload2）转移到git本地仓库（C:/Users/WKQ/git/git\_test\_upload2）了



如果觉得这种目录不习惯，可以把git\_test\_upload2目录里的东西拷出来，如下图。git管理的是文件，更换目录不会产生影响。



然后 git pull，git commit，git push。

最后你发现，把项目上传到服务器了。



4 搭建Git服务器

自己上网搜。

我没有搭建过git服务器，因为gitoschina和github已经足够我日常使用。

gitoschina和github是互联网上比较流行的开源代码托管平台（git仓库/服务器）

还有，我就不告诉你公司的linux运维会搭建git服务器。

<http://blog.csdn.net/pandakong/article/details/7214268> windows上搭建git服务器

5. 常用git术语解释

5.1 版本库(repository)

什么是版本库呢？版本库又名仓库，英文名repository，你可以简单理解成一个目录，这个目录里面的所有文件都可以被Git管理起来，每个文件的修改、删除，Git都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻可以“还原”。

5.2 把文件添加到版本库

首先这里再明确一下，所有的版本控制系统，其实只能跟踪文本文件的改动，比如TXT文件，网页，所有的程序代码等等，Git也不例外。版本控制系统可以告诉你每次的改动，比如在第5行加了一个单词“Linux”，在第8行删了一个单词“Windows”。而图片、视频这些二进制文件，虽然也能由版本控制系统管理，但没法跟踪文件的变化，只能把二进制文件每次改动串起来，也就是只知道图片从100KB改成了120KB，但到底改了啥，版本控制系统不知道，也没法知道。

不幸的是，Microsoft的Word格式是二进制格式，因此，版本控制系统是没法跟踪Word文件的改动的，前面我们举的例子只是为了演示，如果要真正使用版本控制系统，就要以纯文本方式编写文件。

因为文本是有编码的，比如中文有常用的GBK编码，日文有Shift\_JIS编码，如果没有历史遗留问题，强烈建议使用标准的UTF-8编码，所有语言使用同一种编码，既没有冲突，又被所有平台所支持。

5.3 git add

git add告诉git，把文件添加到仓库

5.4 git commit

git commit -m “2017-01-08 10:00 修改README.txt”

用命令git commit告诉Git，把文件提交到仓库

简单解释一下git commit命令，-m后面输入的是本次提交的说明，可以输入任意内容，当然最好是有意义的，这样你就能从历史记录里方便地找到改动记录。

你对文件进行修改，然后提交修改到版本库里，就好比玩RPG游戏时，每通过一关就会自动把游戏状态存盘，如果某一关没过去，你还可以选择读取前一关的状态。有些时候，在打Boss之前，你会手动存盘，以便万一打Boss失败了，可以从最近的地方重新开始。Git也是一样，每当你觉得文件修改到一定程度的时候，就可以“保存一个快照”，这个快照在Git中被称为commit。一旦你把文件改乱了，或者误删了文件，还可以从最近的一个commit恢复，然后继续工作，而不是把几个月的工作成果全部丢失。

5.5 git status

git status命令可以让我们时刻掌握仓库当前的状态

5.6 git lg

lg其实是个别名，详细配置如下：

git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置

5.7 git diff

git diff顾名思义就是查看difference

5.8 git log

git log命令显示从最近到最远的提交日志

需要友情提示的是，你看到的一大串类似3628164...882e1e0的是commit id（版本号），和SVN不一样，Git的commit id不是1，2，3……递增的数字，而是一个SHA1计算出来的一个非常大的数字，用十六进制表示。

而且你看到的commit id和我的肯定不一样，以你自己的为准。为什么commit id需要用这么一大串数字表示呢？因为Git是分布式的版本控制系统，后面我们还要研究多人在同一个版本库里工作，如果大家都用1，2，3……作为版本号，那肯定就冲突了。

5.9 git reflog

5.10 git checkout

5.11 git reset

git reset HEAD readme.txt

1. 常见问题

6.1 在cmd命令行无法使用git

因为你的环境变量没配置，系统识别不了git命令，所以会报错，在环境变量path里添加git软件的bin路径就可以了（类似于JDK）

我的git安装在F:\Program Files\Git，所以我在path里添加F:\Program Files\Git\bin就可以在命令行使用了

6.2 中文不能正确地显示

在默认设置下，中文文件名在工作区状态输出，文件名的中文不能正确地显示，而是显示为八进制的字符编码。如：

$ git status -s

?? "\350\257\264\346\230\216.txt\n

通过将Git配置变量 core.quotepath 设置为false，就可以解决中文文件名称在这些Git命令输出中的显示问题。

$ git config --global core.quotepath false （在命令行输入这个配置）

6.3 Your console font probably doesn't support Unicode. If you experience strange characters in the output, consider switching to a TrueType font such as Consolas!

解决办法：

1. 修改命令行的字体，使用新宋体。

2. 设置git的参数，按照如下设置

git config [--global] core.quotepath off // [--global是参数，可以写上，也可以不写]

example：1. git config --global core.quotepath false 全局(一个电脑上)设定

2. git config core.quotepath false 只在某一个项目里设定

git config --global --unset i18n.logoutputencoding

git config --global --unset i18n.commitencoding

6.4 每次都要输入用户名密码

git命令行每次都要输入用户名密码，能不能只输入一次永久使用

git config --global credential.helper store 保存输入的密码 “store” 模式会将凭证用明文的形式存放在磁盘中，并且永不过期。

git config --global credential.helper cache 保存输入的密码 “cache” 模式会将凭证存放在内存中一段时间。 密码永远不会被存储在磁盘中，并且在15分钟后从内存中清除。

<https://my.oschina.net/songxinqiang/blog/192567>

<http://blog.csdn.net/hudashi/article/details/7664460>

<http://www.oschina.net/translate/10-tips-git-next-level> 10 个迅速提升你 Git 水平的提示

<http://blog.csdn.net/wh_19910525/article/details/7554430> git cherry-pick

<http://blog.csdn.net/pandakong/article/details/7233721> git reset soft mixed hard