Git分布式版本控制工具

# git的历史背景

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。

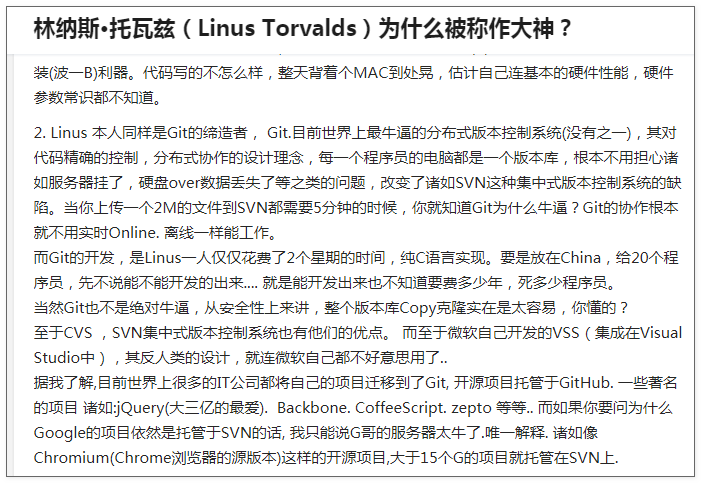
到了 2005 年，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux 的缔造者 Linus Torvalds ）不得不吸取教训，只有开发一套属于自己的版本控制系统才不至于重蹈覆辙。他们对新的系统制订了若干目标：

* 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）

自诞生于 2005 年以来，Git 日臻成熟完善，在高度易用的同时，仍然保留着初期设定的目标。它的速度飞快，极其适合管理大项目，它还有着令人难以置信的非线性分支管理系统（见第三章），可以应付各种复杂的项目开发需求。

Linus Torvalds 就是linux之父,创造了git





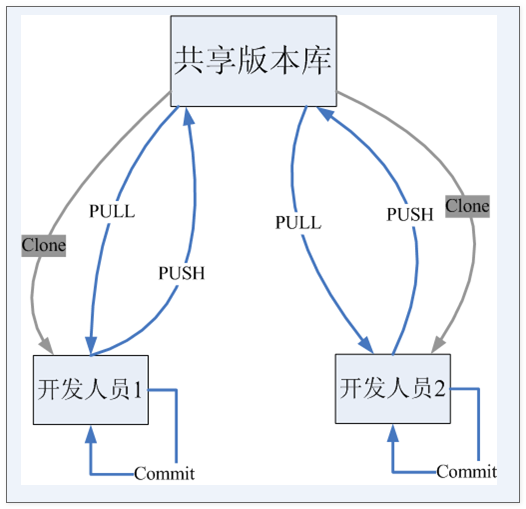
当初linux的创始人一直手工合并世界各地的linux贡献者提交的代码，为什么要手工提交呢？因为他觉得cvs svn 效率太低， 太Low ,逼格不够。而使用付费的版本控制工具，又违背Linux开源之道。

然后bitkeeper免费送给linux社区的人使用它公司的版本控制系统。然后有些有志青年，开始破解bitkeeper的协议。

结果可想而知，（劳资白给你们鞋穿，原来你们还要来搞劳资皮革厂），就收回协议，不免费了。

然后Linux创始人用两周时间开发了一个新的版本控制系统，这就是最早的Git。

# git的使用流程

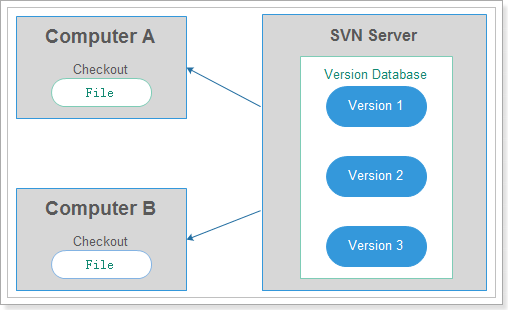


# GIT和SVN的对比

## SVN的原理分析

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。

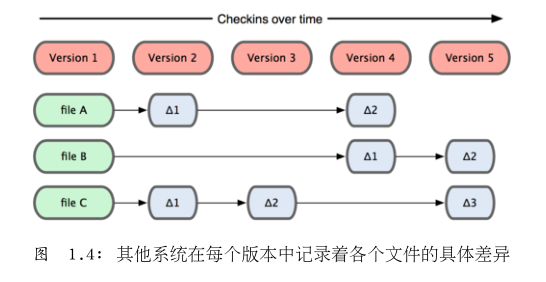
但是相较于其优点而言，集中式版本控制工具缺点很明显：

1. 服务器单点故障
2. 容错性差

## GIT原理的分析

### Svn记录具体差异

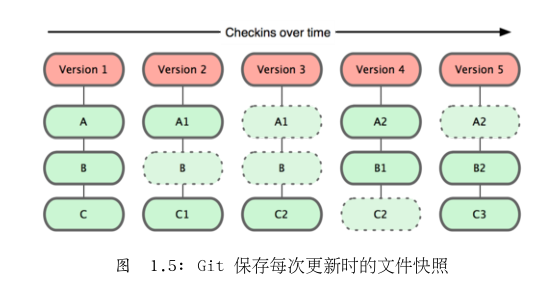
Git 和其他版本控制系统的主要差别在于，Git 只关心文件数据的整体是否发生变化，而大多数其他系统则只关心文件内容的具体差异。这类系统（CVS，Subversion，Perforce，Bazaar等等）每次记录有哪些文件作了更新，以及都更新了哪些行的什么内容，请看图 1.4。



### Git记录整体变化

Git 并不保存这些前后变化的差异数据。实际上，Git 更像是把变化的文件作快照后，记录在一个微型的文件系统中。每次提交更新时，它会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照的索引。

为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一连接。Git 的工作方式就像图 1.5 所示。



这是 Git 同其他系统的重要区别。它完全颠覆了传统版本控制的套路，并对各个环节的

实现方式作了新的设计。Git 更像是个小型的文件系统，但它同时还提供了许多以此为基础

的超强工具，而不只是一个简单的 VCS。

### 操作本地执行

使用Git 不需要从远程服务器上拉取数据，而直接从本地数据库读取后展示。所以任何时候你都可以马上翻阅，无需等待。

如果想要看当前版本的文件和一个月前的版本之间有何差异，Git 会取出一个月前的快照和当前文件作一次差异运算，而不用请求远程服务器来做这件事，或是把老版本的文件拉到本地来作比较。

### 数据完整性

在保存到 Git 之前，所有数据都要进行内容的校验和（checksum）计算，并将此结果作为数据的唯一标识和索引。换句话说，不可能在你修改了文件或目录之后，Git 一无所知。

这项特性作为 Git 的设计哲学，建在整体架构的最底层。所以如果文件在传输时变得不完整，或者磁盘损坏导致文件数据缺失，Git 都能立即察觉。

Git 使用 SHA-1 算法计算数据的校验和，通过对文件的内容或目录的结构计算出一个SHA-1 哈希值，作为指纹字符串。该字串由 40 个十六进制字符（0-9 及 a-f）组成，看起来就像是：

*24b9da6552252987aa493b52f8696cd6d3b00373*

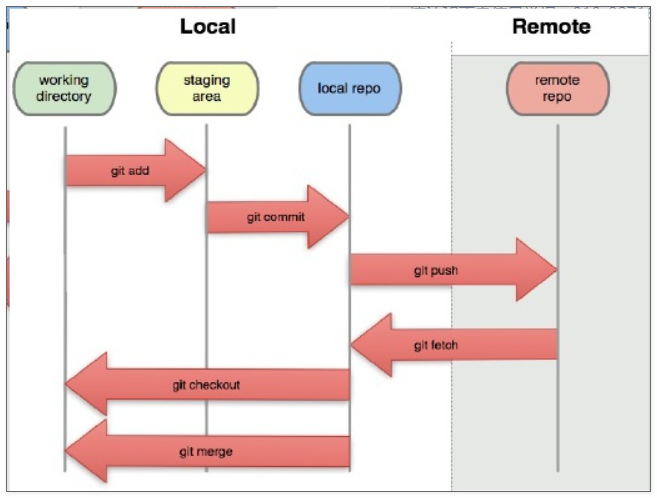
Git 的工作完全依赖于这类指纹字串，所以你会经常看到这样的哈希值。实际上，所有保存在 Git 数据库中的东西都是用此哈希值来作索引的，而不是靠文件名。

### 三种状态

以下的概念非常重要。

对于任何一个文件，在 Git 内都只有三种状态：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。



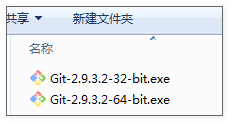
1. 如果是 git 目录中保存着的特定版本文件，就属于已提交状态；
2. 如果作了修改并已放入暂存区域，就属于已暂存状态；
3. 如果自上次取出后，作了修改但还没有放到暂存区域，就是已修改状态。

# 本地安装git服务器

在本地电脑中安装git服务器 表示本地也是一个服务器。

* 安装git服务器：参考资料中有文件。

选择一个符合的版本来安装在本地，一直点击点击下一步就可以了。



安装好了之后：右击鼠标可以看见如下，说明本地有了git的环境，可以使用git的命令来上传代码，下载代码了。



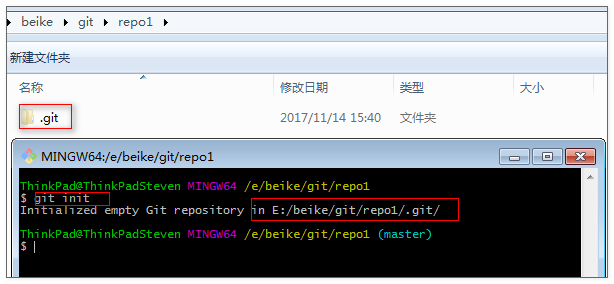
# git常用基本的命令

## 提交代码

* 初始化一个仓库

在命令行中输入命令：

|  |
| --- |
| $ git init |



* 配置全局的用户名和邮箱

git服务器得知道你是谁，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。

在命令行中输入：

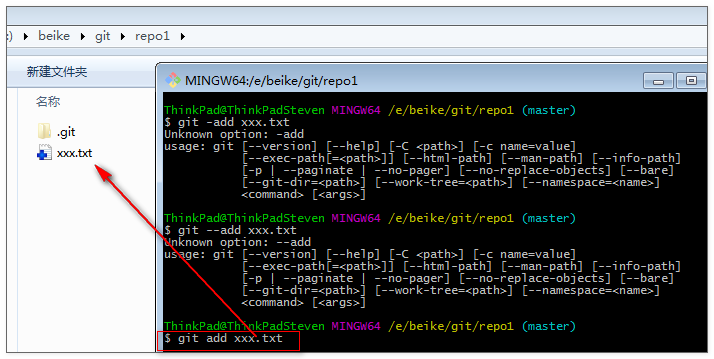
|  |
| --- |
| $ git config --global user.name "Your Name"  $ git config --global user.email "email@example.com" |

注意：如果使用--global参数：表示本台电脑里面的设置都是使用该用户及邮箱

* 添加到文件到git缓存中

在命令行中输入：如下命令

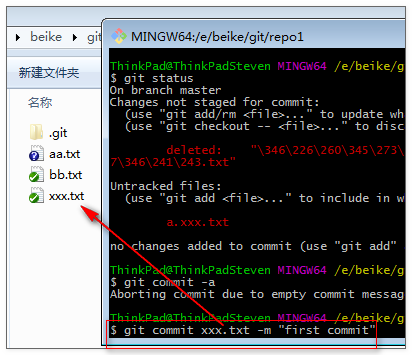
|  |
| --- |
| $ git add xxx.txt |



* 提交到本地仓库

在命令行窗口执行命令：

|  |
| --- |
| $ git commit xxx.txt -m "first commit" |



本地提交流程结束。

提交代码到远程仓库:

第一步：添加远程仓库名称到本地（这里以github为例将其作为远程仓库，如果没有远程仓库，自己先创建一个，此处不再讨论）

|  |
| --- |
| $ git remote add origin https://github.com/liu654011945/gittest01.git |

第二步：推送代码到远程仓库中

|  |
| --- |
| $ git push origin master |

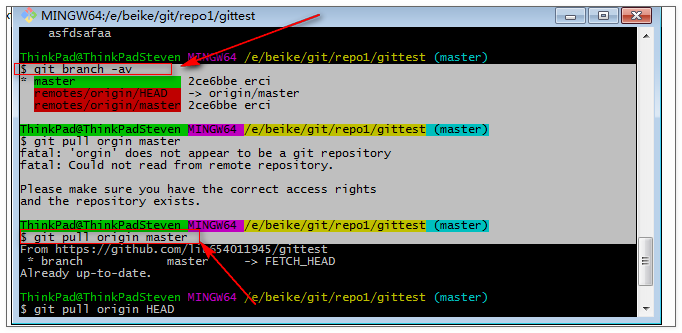
## 下载代码

* 克隆代码：

|  |
| --- |
| $ git clone https://github.com/liu654011945/gittest.git |

* 拉取代码（暂时忽略）

拉取代码，先查看版本，再选中版本拉取。



git pull origin master

# 其他的命令

* 恢复到之前的任意一个版本。

还原或者叫做回退：

在命令行执行命令如下：表示回退到最新版本。适用于还没有commit之前的操作。

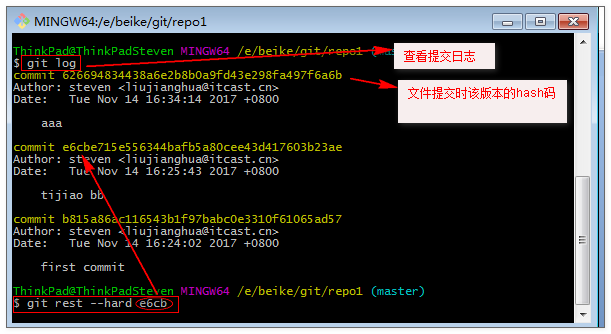
|  |
| --- |
| $ git reset --hard head |

如果想回退到前一个版本使用如下的命令：

|  |
| --- |
| $ git reset --hard head^ |

如果想回退到指定的版本，使用如下命令

|  |
| --- |
| $ git reset --hard e6cb |

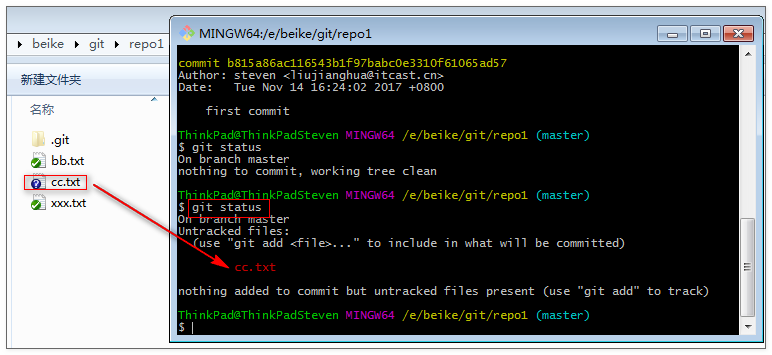


* 查看git状态：

git status命令可以列出当前目录所有还没有被git管理的文件和被git管理且被修改但还未提交(git commit)的文件。

命令行：

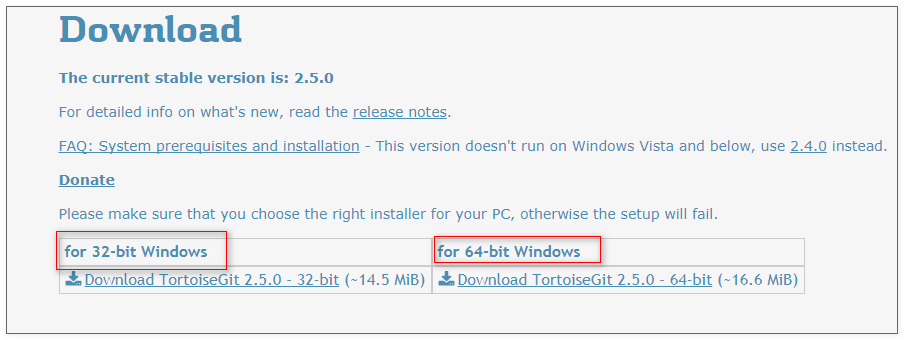
|  |
| --- |
| $ git status |



# 安装git代码上传工具tortoisegit（客户端）

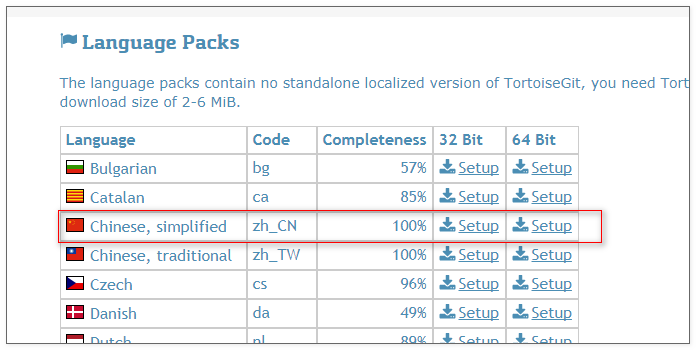
下载地址：

https://tortoisegit.org/download/



该客户端默认是英文版本的，可以使用中文语言包本地化.

下载语言包：https://tortoisegit.org/download/



根据需要下载32 和64位

注意：git服务的安装版本和语言包版本要一致才能安装成功。

操作步骤：

第一步：先安装tortoisegit。

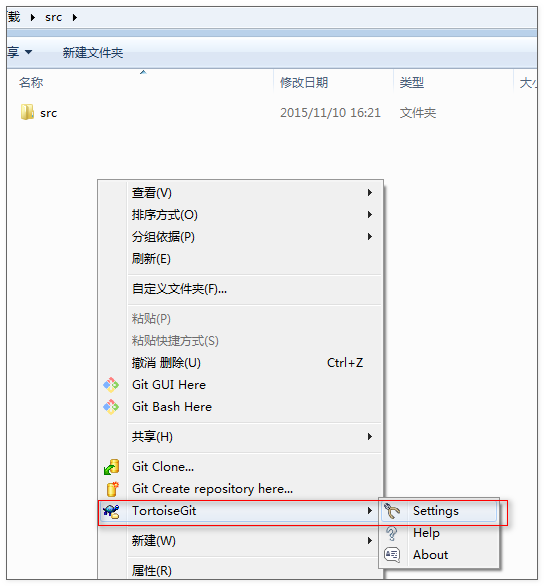


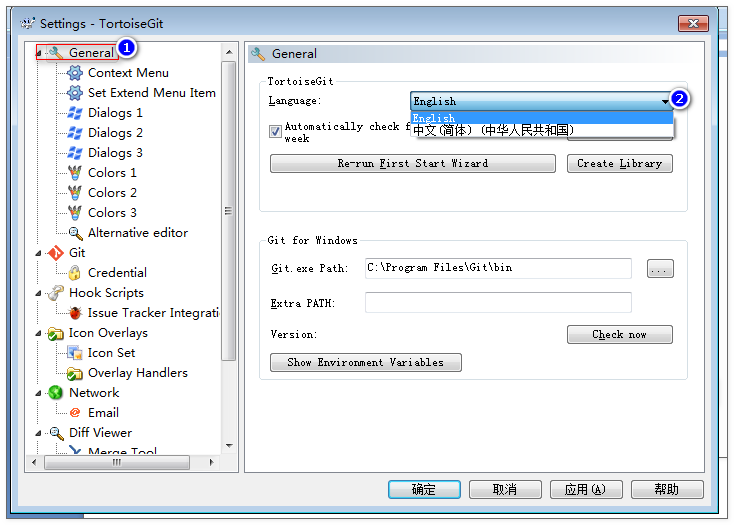
第二步：再安装语言包。



重启 电脑。

第三步：切换成中文版：





选择中文（简体）即可。

重新启动电脑，使之生效。

# Git服务器搭建（linux）

github可以作为git服务器，但是github上边的都是公开免费的，如果想要不公开需要收费。所以我们自己也搭建git服务器。Git服务器可以搭建在windows,也可以搭建在Linux中，由于windows中copssh登录git服务器软件已经收费，免费版本有很多缺点。因此把git服务器搭建在Linux中。

## 依赖环境

Git需要很多依赖环境：因此安装git需要先安装下列软件：

[root@itcast-01 ~]# yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel perl-devel gcc-c++

## 安装git

上传git安装包：

安装包：git-2.9.3.tar.gz

tar -xvf git-2.9.3.tar.gz

安装命令：(进入git解压目录)

[root@itcast-01 git-2.9.3]# make prefix=/usr/local all

[root@itcast-01 git-2.9.3]# make prefix=/usr/local install

## 创建git用户

添加Linux的用户，用户名为git的用户：

[root@itcast-01 ~]# useradd git

设置git用户的密码

[root@itcast-01 ~]# passwd git

切换到git用户

[root@itcast-01 ~]# su - git

## 初始化仓库

初始化仓库，仓库名为taotao.git：

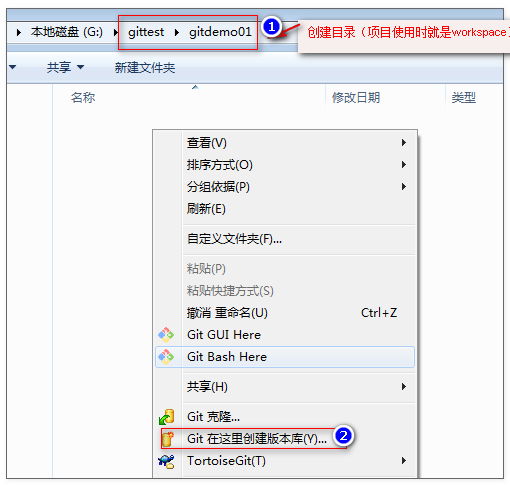
[git@itcast-01 root]$ cd

[git@itcast-01 ~]$ git init --bare taotao.git

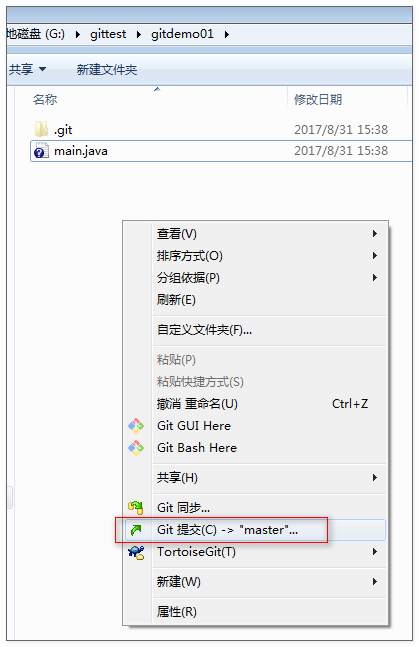
始化空的 Git 仓库于 /home/git/taotao.git/

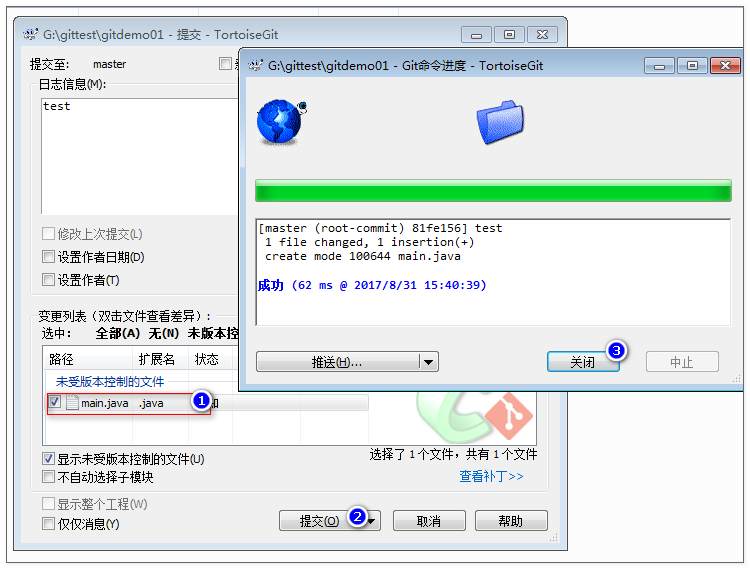
# 使用tortoisegit 提交代码(push)

## 第一步：创建目录：并初始化为本地仓库。

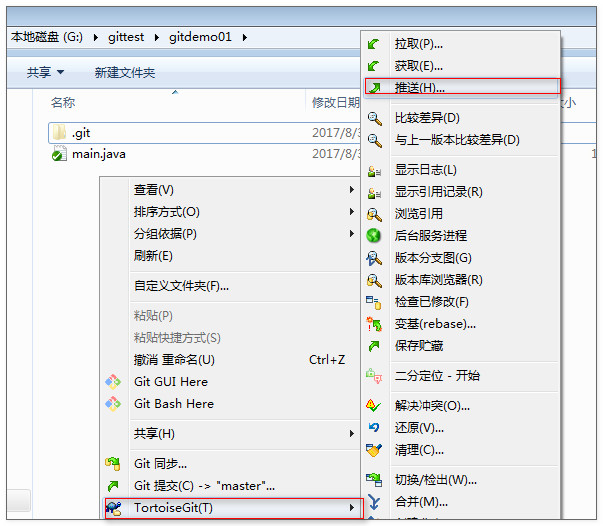


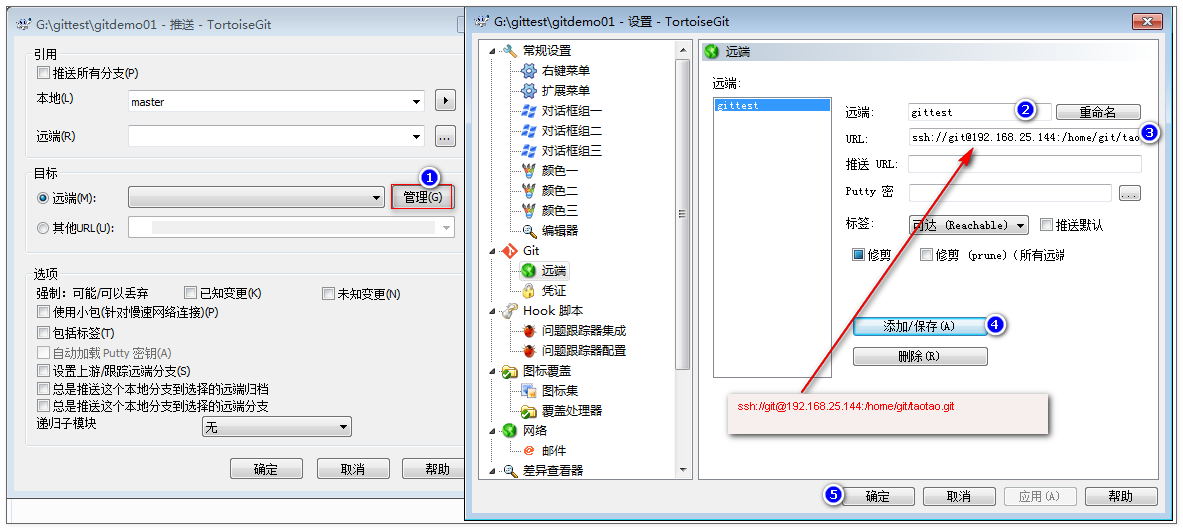
## 第二步：创建一个java文件，并提交到本地仓库





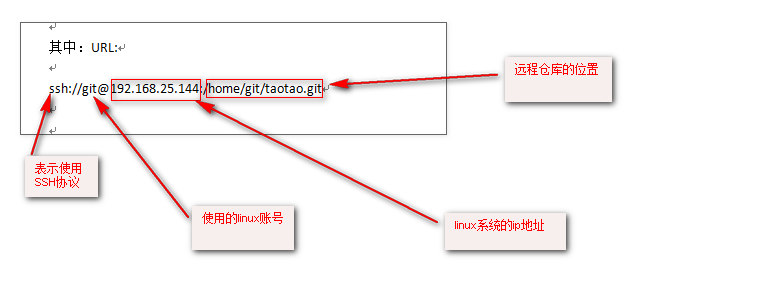
## 第三步：推送到远程git服务器中



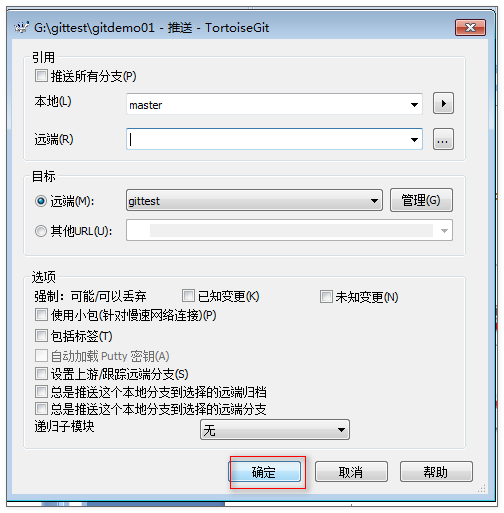


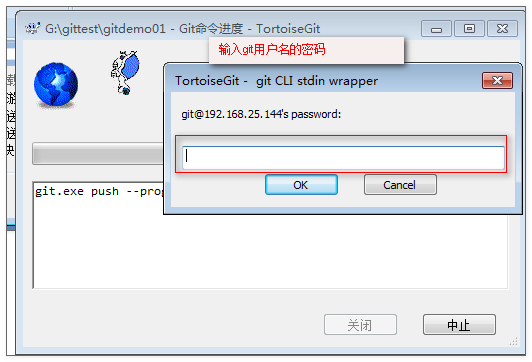
其中：URL:

ssh://git@192.168.25.146:/home/git/taotao.git

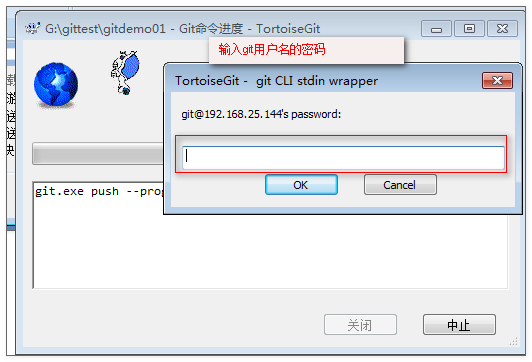


点击确定:





提示成功并关闭.

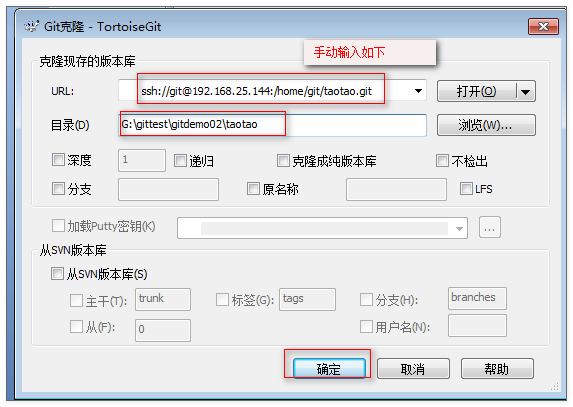


以上使用这个方式每次都需要提交用户名和密码.

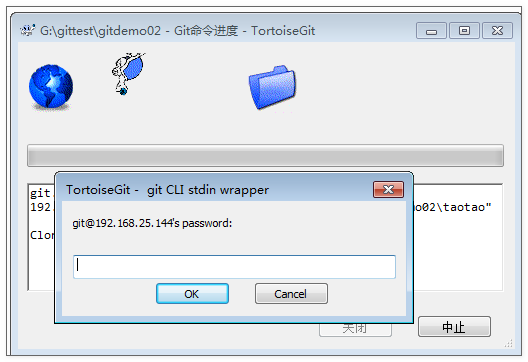
# 使用tortoisegit克隆代码

## 第一步：创建目录，并从远程服务器上clone到本地

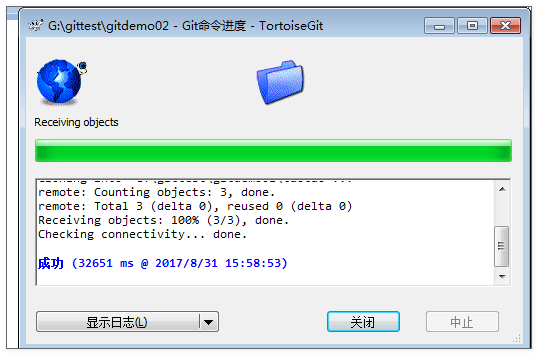


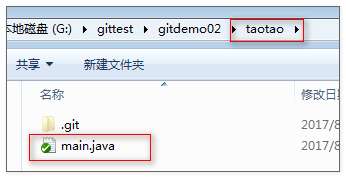


## 第二步：输入密码：



## 第三步：克隆提示成功。





下载clone代码成功。

以后如果有更新直接使用pull即可

# 更新拉取代码

拉取代码也叫做：pull

